. . . . . Reporte Tecnico . . . . . .

Concentracion IA Avanzada para la Ciencia de Datos: Reportes Reto 2022

Gildardo Sanchez-Ante, Editor
Tecnológico de Monterrey
Campus Guadalajara

in: Reporte Tecnico CS-0001-2022. See also BibTeX entry below.

BibTeX:

```
@article{ConcentracionIA2022,
  author      = {Gildardo Sanchez-Ante, Editor},
  title       = {Concentracion IA Avanzada para la Ciencia de Datos: Reportes Reto 2022},
  journal     = {Reporte Tecnico CS-0001-2022},
  institution = {Tecnologico de Monterrey, Campus Guadalajara},
  month       = {Diciembre},
  year        = {2022}
}
```


document created on: 8 de diciembre de 2022
created from file: output.tex
cover page automatically created with CoverPage.sty
(available at your favourite CTAN mirror)

Tecnológico de Monterrey

Prefacio

Este documento resume el trabajo desarrollado por los estudiantes de la Concentración en Inteligencia Artificial Avanzada para la Ciencia de Datos al resolver el reto que correspondió al semestre Agosto-Diciembre de 2022.

En esta ocasión el reto consistió en que dadas imágenes (fotografías) de cuerpos de agua, estimar algunos parámetros con el uso de herramientas de la inteligencia artificial, más específicamente del aprendizaje automático. De hecho, el problema puntual que se planteó fue estimar dos variables: flujo y el nivel del agua en ciertos períodos de tiempo, partiendo de un conjunto de imagenes para las cuales se dispone de esos valores (ground truth).

El problema fue planteado a nosotros por el Prof. Troy Gilmore, de la Universidad de Nebraska-Lincoln, asi como su estudiante de doctorado, Ken Chapman, quienes en conjunto con otros investigadores del **GaugeCam GRIME Lab** (`http://gaugecam.org`) ya habían trabajado algo en el problema usando algunos métodos del aprendizaje computacional. Mi agradecimiento a todos ellos por la gran apertura en compartirnos sus imágenes y sus datos, asi como su experiencia y comentarios a lo largo de la concentración.

Los equipos tuvieron toda la libertad de analizar, proponer y probar ideas para resolver el problema y sabemos que al final lograron resultados interesantes pues la precisión en los modelos superó lo que el equipo del GaugeCam GRIME Lab había obtenido. Creo que la bondad de este proyecto fue que nadie tenía una respuesta esperada. Es decir, es un tema abierto a la investigación y eso es muy rico por la multitud de aristas y opciones que presenta, pero también puede ser visto como un entorno abrumador y plagado de incertidumbre. Como quiera que haya sido, en estos reportes se puede apreciar que los equipos tomaron caminos diferentes, exploraron diversas opciones y al final plantearon sus resultados ante un público internacional. Están listos para resolver problemas de esta naturaleza como verdaderos profesionales. En verdad que como alguien que los vió avanzar a lo largo del semestre, me siento emocionado y orgulloso de su trabajo.

Debo decir también que este tipo de proyectos (retos) en este nivel de las carreras profesionales (séptimo semestre) es en mi opinión el escenario más favorable para el éxito. La idea de compilar aquí los resultados de esta generación es en parte para dejar una memoria de lo que se ha ido haciendo, pero también para plantear una referencia de la calidad de los trabajos, con la intención final de que ésta siga siempre mejorando.

Mi reconocimiento a todos los estudiantes, y mi agradecimiento a las autoridades del Tec, pues nos facilitaron nuestro trabajo impresionantemente.

Diciembre 2022

Gildardo Sánchez-Ante
Profesor

Índice general

Camera based Water Stage and Discharge Prediction with Machine Learning

Ernesto Adrián Álvarez Salazar, Carlos Javier Leal Beltrán, Carlos Moisés Chávez Jiménez, and Luis Armando Salazar Lopez

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

1. Introducción

La medición y el modelado precisos del nivel (la altura del nivel del agua en el rio) y la descarga (el caudal del rio) de la corriente son importantes para la gestión diaria del agua, e l pronóstico y la gestión de inundaciones, la evaluación del cumplimiento de los acuerdos de uso del agua y el diseño de embalses, sistemas de suministro de agua, puentes y alcantarillas (Boiten, 2008). Los datos continuos de series de tiempo de medidores de flujo también son críticos para calibrar y/o validar modelos de aguas superficiales y subterráneas, mientras que las brechas en estos datos aumentan la incertidumbre en las predicciones de un modelo. El nivel de la corriente normalmente se mide con sensores flotantes, de presión, ópticos y acústicos (Turnipseed y Sauer, 2010). Estos sensores tradicionales pueden fallar y requerir un mantenimiento regular, los cuales son costosos.

Como resultado, es posible que se produzcan lagunas en los registros de flujo y descarga debido a una instalación incorrecta (p. ej., especialmente durante estudios a corto plazo, cuando las características del sitio no son bien conocidas), fallas en los equipos y/o brechas en el financiamiento de los programas de monitoreo. Por supuesto, las cámaras también pueden fallar, pero pueden proporcionar una redundancia económica con información que no está disponible en los sensores que emiten mediciones escalares individuales. Por ejemplo, las imágenes proporcionan una verificación visual de las condiciones hidrológicas, incluida la presencia de hielo, obstrucciones o cambios importantes en la geometría del canal. En este estudio, se opta por un enfoque al monitoreo pasivo que utiliza imágenes de lapso de tiempo que se pueden combinar con mediciones de sensores tradicionales que son adecuadas para llenar los vacíos en los registros de caudales.

2. Hipotesis

Utilizando la investigación previamente realizada por el equipo del socio formador, mediante optimización de las imagenes utilizables en la problemática implementando librerias de computer vision en python, a través del desarrollo

de modelos matemáticos y el análisis de datos, se espera obtener formas distintas y optimizadas de abordar la problemática, modelos de regresión que nos permitan predecir el nivel y la descarga de una corriente con más eficacia y un modelo de clasificación que permita distinguir y optimizar entre imágenes utilizables dentro del proceso de obtención de datos, mejorando así en un aspecto general, la parte práctica en la resolución de esta problemática.

Ademas, comprobar que hay una diferencia detectable entre la predicción con imagenes de diferentes temporadas y en un lapso temporal pequeño (imagenes de entrenamiento y evaluación con no más de 1 año de diferencia) y una predicción con imagenes de la misma temporada pero con un lapso temporal largo (imagenes de entrenamiento y evaluación con más de 5 año de diferencia)

3. Tratamiento Inicial de los Datos

Para comenzar a trabajar con los datos, es necesario que pasen por un proceso de preparación que nos permita obtener la mejor parte de ellos. Este proceso se divide en tres etapas: Limpieza, Transformación y Visualización. A continuación desglosaremos las fases involucradas a este proceso:

3.1. Limpieza de los datos

La limpieza es la primera y una etapa fundamental del tratamiento de la información. Aquí se busca eliminar la mayor cantidad de imperfecciones que pudieramos llegar a encontrar. Cosas como valores faltantes, datos fuera de rango, dividir la información disponible en .entrenamientoz "pruebas", eliminar columnas innecesarias para el análisis, etc.

SensorTime	CaptureTime	Filename	hlineAgency
SiteNumber	TimeZone	Stage	Discharge
CalcTimestamp	fNumber	isoSpeed	shutterSpeed
grayMean	graySigma	entropyMean	entropySigma
hMean	hSigma	sMean	sSigma
vMean	vSigma	areaFeatCount	grayMean 0
graySigma 0	entropyMean 0	entropySigma 0	hMean 0
hSigma 0	sMean 0	sSigma 0	vMean 0
vSigma 0	grayMean 0	graySigma 0	entropyMean 0
entropySigma 0	hMean 0	hSigma 0	sMean 0
sSigma 0	vMean 0	vSigma 0	WeirAngle
WeirPt1X	WeirPt1Y	WeirPt2X	WeirPt2Y
WwRawLineMin	WwRawLineMax	WwRawLineMean	WwRawLineSigma
WwCurveLineMin	WwCurveLineMax	WwCurveLineMean	WwCurveLineSigma

Variables disponibles en nuestro archivo de datos. (figura 3.1.1)

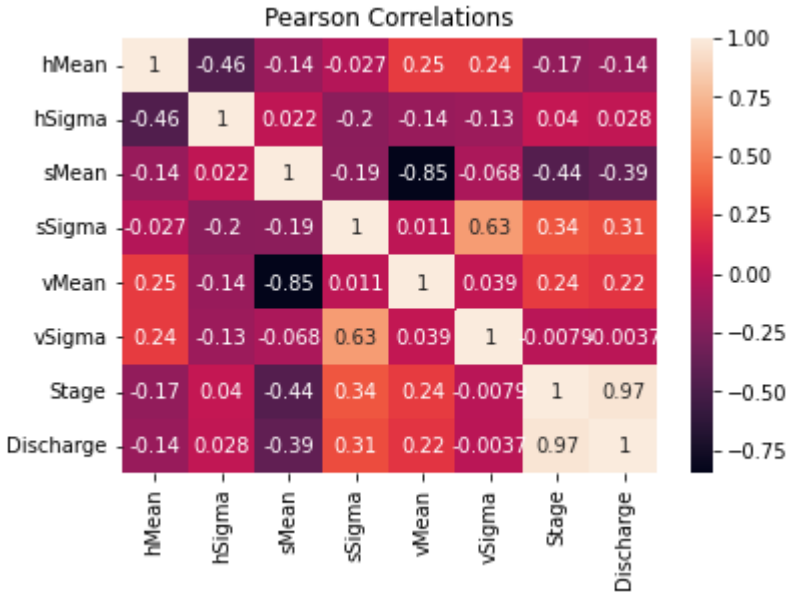
3.2. Transformación de los datos

Una vez revisados los datos, entendemos que debemos transformarlos para generar gráficos y diagramas que faciliten la búsqueda de patrones y la determinación de los parametros necesarios para un modelo predictor de regresión y uno de clasificación.

Para esto, revisamos el conjunto de datos con la información proporcionada. Encontramos que primero debemos ordenar los datos con base en el timestamp. Esta variables que indican el tiempo con el que fueron realizadas las capturas de datos. Dentro del conjunto de datos se proporcionan tres variables para indicar el tiempo de captura: ÇaptureTime","SensorTimez ÇalcTimestamp", y cada una de estas contiene un formato distinto y a primera instancia nada fácil de trabajar, lo que indica que tenemos que separarlas para que queden enumeradas de una en una. Desarrollamos un método para tratar las fechas consiguiendo así, un orden en el formato de la captura de datos. En cuanto a lo demás, los datos venían en buenas condiciones, por lo que no tuvimos que arreglar valores nulos.

3.3. Visualización de los datos

Siendo las variables "Stagez "Discharge"las que nos iteresan predecir con un modelo, las comparamos con el resto de las variables para ir descartando las de menor correlacion.



Ejemplo de las correlaciones de pearson realizadas. (figura 3.3.1)

Tras comparar con y analizar cada una de las variables dentro del conjunto de datos con las variables stage y discharge escogimos las siguientes variables, ya

que cuentan con una correlación menor a -30 ó mayor a 30, por lo tanto tienen una correlación considerable para incluirse en un modelo: sSigma, grayMean 0, hMean 0, grayMean 1, hMean 1, sMean s, Sigma 1, WeirPt1Y, WeirPt2X, WeirPt2Y.

4. Generación y evaluación de los modelos de predicción para Discharge y Stage.

Tras identificar las variables que deseamos predecir y las variables que podemos utilizar para lograrlo, comenzamos a realizar modelos con el fin de obtener resultados similares a la investigación pasada, lo anterior con el fin de contar con un punto de partida.

Con la finalidad de mantener nuestros datos más limpios y prevenir que los modelos implementados caigan en un .ºverfitting"separamos nuestros datos en tres categorías: "Trainingçuyo proposito es entrenar al modelo, "Testingçuyo proposito es probar que tan bien entrenado se encuentra el modelo conforme a los datos del training y por último "Validationçuyo proposito es presentar imagenes nuevas que validen el modelo una vez probado, con la finalidad de comprobar que el modelo reacciona bien ante datos nuevos.

4.1. Stage

Inicialmente generamos un modelo regresor con Random Forest sin limpiar los datos para comprobar la fiabilidad mostrada en la investigacipon inicial. Los resultados obtenidos fueron los siguientes:

R^2: 0.8337732202287705
MSE: 0.10366011037446508
RSMSE: 0.32196290217114315
MAE: 0.1533417855444603
Error estandar: 0.3219642351177667

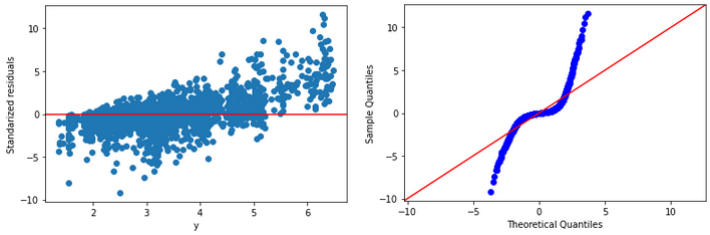


Gráfico de residuos y cuantíl. (figuras 4.1.1 y 4.1.2)

Utilizaremos también los modelos Multi Layer Perceptron y Support Vector Regression para contrastarlos con los del paper.

Resultados del MLP:

R^2: 0.8452408041827444
MSE: 0.09650884978917687
RSMSE: 0.31065873525329507
MAE: 0.153458677001836
Error estandar: 0.31070539268642416

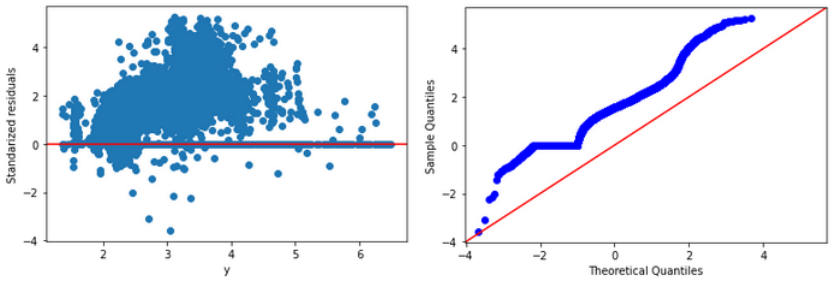


Gráfico de residuos y cuantíl. (figuras 4.1.3 y 4.1.4)

Resultados del SVR:

R^2: 0.8068711443896812
MSE: 0.12043642135528387
RSMSE: 0.34703950979000053
MAE: 0.18047332110724765
Error estandar: 0.3467160676989866

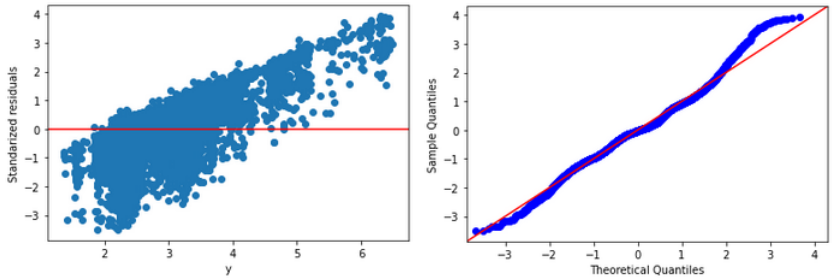


Gráfico de residuos y cuantíl. (figuras 4.1.5 y 4.1.6)

Generaremos ahora un modelo regresión de tipo mínimos cuadrados ordinarios. Esto nos permitirá comparar este modelo con el anterior de Random Forest.

Resultados del OLS:
R^2: 0.5771135725862011
MSE: 0.26371475042654247

RSMSE: 0.5135316450098694
MAE: 0.34110993914908105
Error estandar: 0.5133338519175923

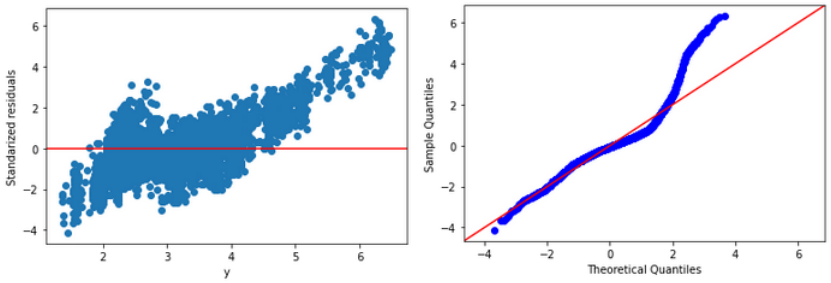


Gráfico de residuos y cuantíl. (figuras 4.1.7 y 4.1.8)

4.2. Lasso y factor de inflación de varianza

Utilizaremos dos metodos para conocer saber si las variables independientes que elegimos tienen la suficiente influencia para quedarse en el modelo.

El primero será Lasso. Este método nos otorga los coeficientes de cada variable para ver cuál tiene más influencia que otras.

Los resultados obtenidos fueron los siguientes:

```
[-1.25239704e-02  1.17108446e-02 -2.25235500e-03  8.22206662e-03
  1.42008591e-03  2.56597630e-03  1.54785389e-02 -1.22254590e-02
  2.03009591e-03  5.13341240e-03 -1.94424549e-03 -2.41175383e-03
 -1.52971933e-06  5.48523134e-07  2.47408363e-05 -6.51908366e-05
 -3.14883927e-06 -3.04336756e-05  7.14533728e-05]
2.170640546638908
```

Coeficientes obtenidos mediante LASSO. (figura 4.2.1)

El segundo método es con el factor de inflación de varianza. Este también nos muestra la influencia que tiene cada variable con la que queremos proyectar. En dado caso que una salga con un factor de inflación de varianza de más de 10, qué en general, un VIF superior a 10 indica una alta correlación y es motivo de preocupación, la descataremos por tener una relación directa de la variable a proyectar. Lo cual generaría una desviación del modelo.

Los resultados obtenidos fueron los siguientes:

	features	vif_Factor
0	graySigma	3.057105e+01
1	hMean	6.263629e+01
2	vSigma	1.328840e+01
3	grayMean 0	2.819411e+01
4	vMean 0	1.174884e+02
5	hMean 1	7.661614e+01
6	hSigma 1	2.448502e+01
7	WeirPt1X	1.932004e+06
8	WeirPt1Y	8.425170e+05
9	WeirPt2X	9.543357e+06
10	WeirPt2Y	6.260092e+05
11	WwRawLineMin	4.169160e+00
12	WwRawLineMax	8.595448e+02
13	WwRawLineMean	2.169796e+04
14	WwRawLineSigma	5.015167e+03
15	WwCurveLineMax	7.891915e+02
16	WwCurveLineMean	2.138986e+04
17	WwCurveLineSigma	4.889393e+03

VIFs obtenidos. (figura 4.2.2)

Obtuvimos valores atípicos conforme a los siguientes métodos:

- Valores atípicos de residuos
- Valores altos de apalancamiento
- Valores altos de distancia de Cook
- Valores altos de DFFIT
- Valores altos de DFBetas

Una vez que obtuvimos estos valores, los eliminamos de nuestros datos de entrenamiento con el fin de ver si logramos generar mejores modelos. Habiendo eliminado los valores atípicos, generaremos los mismo modelos para ver si obtenemos una mejor efectividad.

Empezaremos generando primero un modelo con Random Forest por ser el que se utilizó en el Paper. Verémos si podemos generar una efectividad mayor a la mencionada por los investigadores.

Resultados del Random Forest trás eliminar datos atípicos :

R^2: 0.8272486821654673
MSE: 0.1077288551141227
RSMSE: 0.3282207414441121
MAE: 0.15628777936281502
Error estandar: 0.32824741648132333

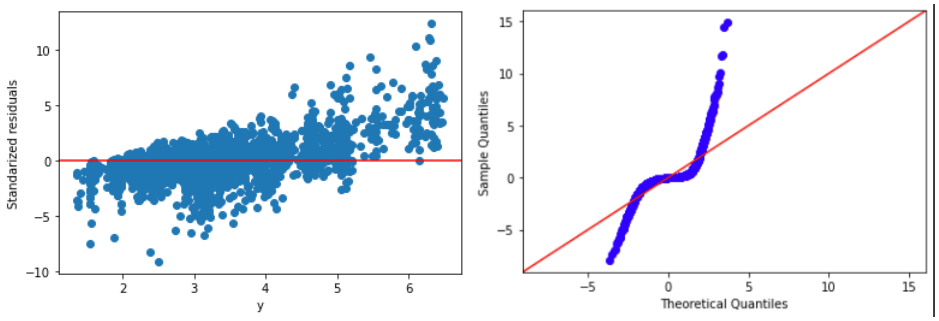


Gráfico de residuos y cuantíl. (figuras 4.2.3 y 4.2.4)

Ahora, generaremos uno con MLP. Esto para contrastar su efectividad con la de Random Forest y comparar cuál es mejor.

Resultados del MLP trás eliminar datos atípicos :

R^2: 0.8366963392403801
MSE: 0.10183723418208397
RSMSE: 0.3191194669431559
MAE: 0.1608734256261398
Error estandar: 0.31601780927805584

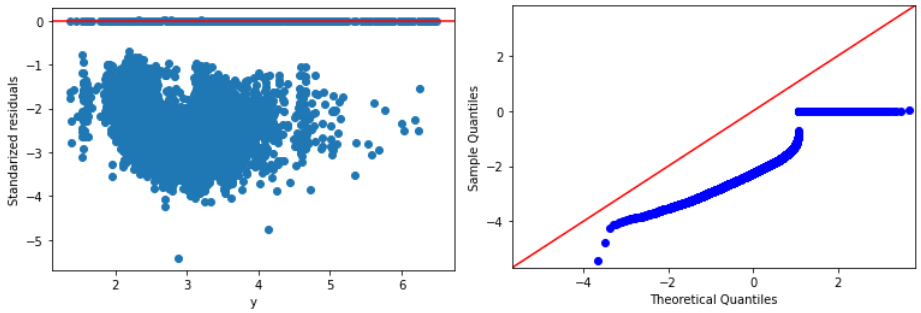


Gráfico de residuos y cuantíl. (figuras 4.2.5 y 4.2.6)

Lo mismo con SVR:

R^2: 0.8068711443896812
MSE: 0.12043642135528387
RSMSE: 0.3470350979000053
MAE: 0.18047332110724765
Error estandar: 0.3467160676989866

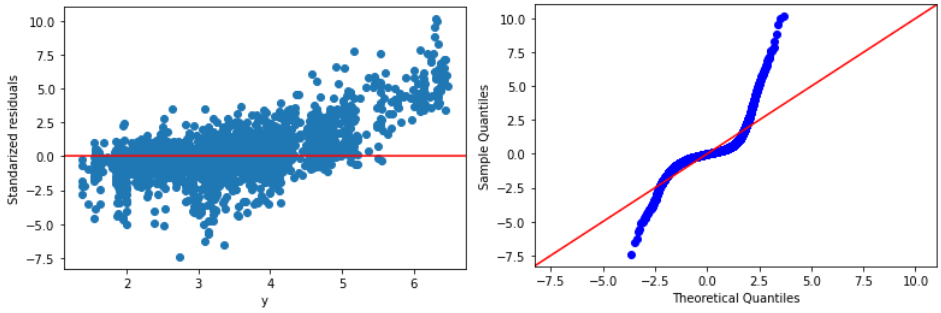


Gráfico de residuos y cuantíl. (figuras 4.2.7 y 4.2.8)

Podemos comparar los resultados antes y después de retirar residuos observando la figura 4.2.9, en la cual indicamos el método y el método sin residuos con la expresión "s/r":

Method	R-squared	MSE	RSMSE	MAE	Std Error
RFR	0.83377	0.10366	0.32196	0.15334	0.32196
RFR - s/r	0.82724	0.10772	0.32822	0.15628	0.32824
MLP	0.84524	0.09650	0.31065	0.15345	0.31070
MLP - s/r	0.83669	0.10183	0.31911	0.16087	0.31601
SVR	0.80687	0.12043	0.34703	0.18047	0.34671
SVR - s/r	0.80687	0.12043	0.34703	0.18047	0.34671
OLS	0.57711	0.26371	0.51353	0.34110	0.51333

Tabla de comparación de métricas. (figura 4.2.9)

4.3. Discharge

Inicialmente generamos un modelo regresor con Random Forest sin limpiar los datos para comprobar la fiabilidad mostrada en la investigacipon inicial. Los resultados obtenidos fueron los siguientes:

R^2: 0.8247287083095377
MSE: 233956.12149014932
RSMSE: 483.690108943887
MAE: 200.22409922729435
Error estandar: 483.5868563381082

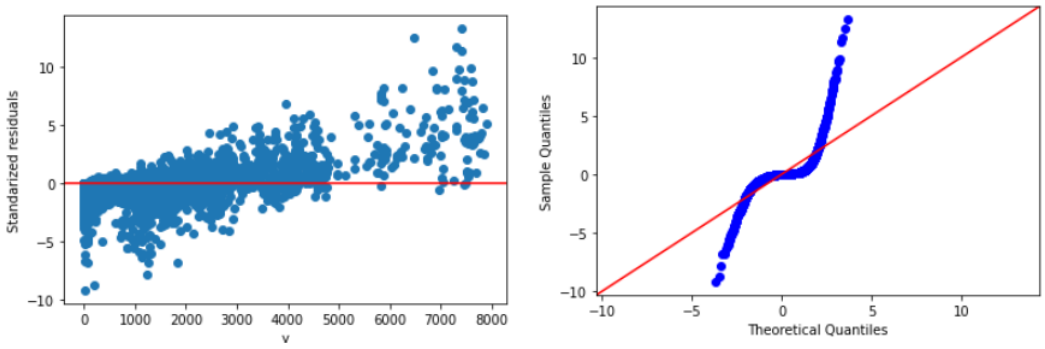


Gráfico de residuos y cuantíl. (figuras 4.3.1 y 4.3.2)

Utilizaremos también los modelos MLP y SVR para contrastarlos con los del paper.
Resultados del MLP:
R^2: 0.9106606310053236
MSE: 119252.22929996347
RSMSE: 345.3291608016379
MAE: 157.9595011798805
Error estandar: 345.3897105087386

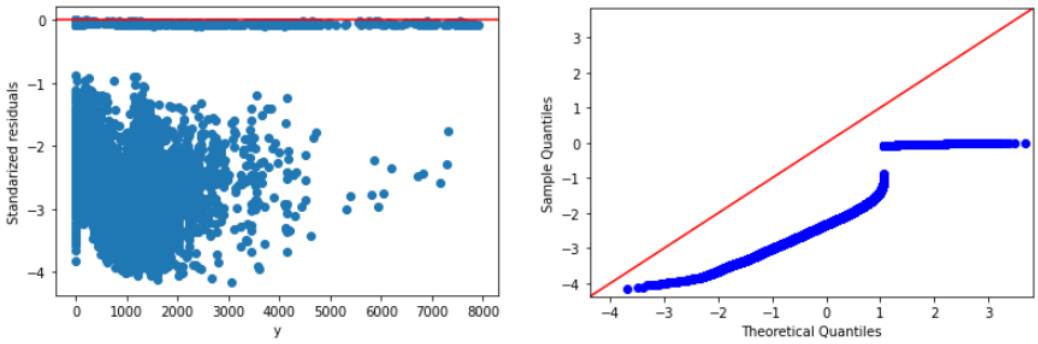


Gráfico de residuos y cuantíl. (figuras 4.3.3 y 4.3.4)

Resultados del SVR:
R^2: 0.47833182204956115
MSE: 696334.5933095505
RSMSE: 834.4666520056692
MAE: 370.20593703858617
Error estandar: 818.5175314021255

Los gráficos de residuos y cuantíl lucen así:

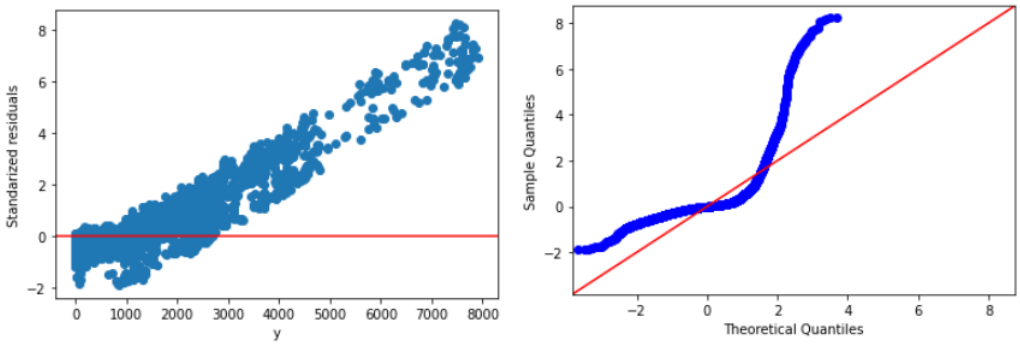


Gráfico de residuos y cuantíl. (figuras 4.3.5 y 4.3.6)

Generaremos ahora un modelo regresión de tipo mínimos cuadrados ordinarios. Esto nos permitirá comparar este modelo con el anterior de Random Forest.

Resultados del OLS:
R^2: 0.5825741299373127
MSE: 557189.5809496772
RSMSE: 746.4513252380741
MAE: 471.07220350721855
Error estandar: 746.029710625578

Los gráficos de residuos y cuantíl lucen así:

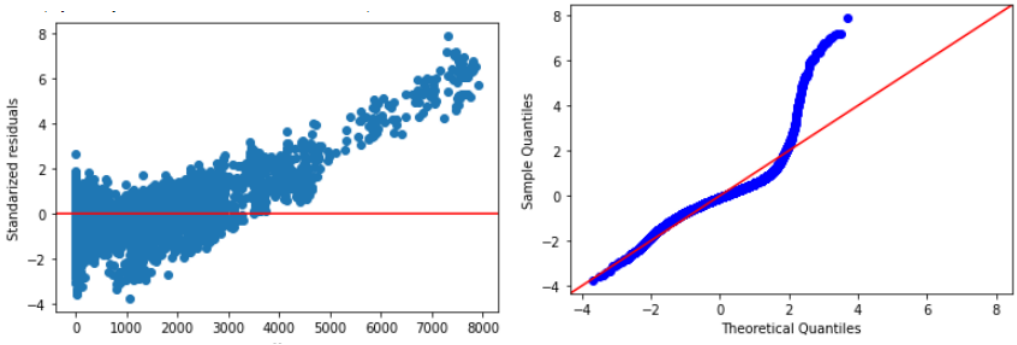


Gráfico de residuos y cuantíl. (figuras 4.3.7 y 4.3.8)

4.4. Lasso y factor de inflación de varianza

Una vez más, utilizaremos LASSO y VIF para conocer saber si las variables independientes que elegimos tienen la suficiente influencia para quedarse en el modelo.

Los resultados obtenidos fueron los siguientes:

```
[-9.46134860e+01  1.73567409e+02 -1.30637753e+04  7.62791997e+01
 -1.48959685e+02 -4.23588021e+01  7.89276654e+01 -2.44886993e+01
  5.90886936e+01  1.64691606e+01  3.26945918e+01  1.84438576e+03
 -1.72173355e+01 -4.88554572e+01  8.88958750e+01  1.84314888e+00
  5.10904651e+01  5.76515441e+01 -1.82703479e+00  1.01521820e+02
 -7.56246079e+03  2.24644243e+03  1.57402750e+01 -1.08667974e+02
 -4.33387073e+01 -1.46288750e+02 -2.33075285e+01 -7.41032630e+01
  2.40466609e+00  4.68145566e+00 -2.18273840e+00 -1.97971625e+00
 -6.64474554e-03 -7.28889608e-03  3.82315972e-02 -9.88586799e-02
 -5.04423832e-03 -4.67612282e-02  1.17679214e-01]
777.4131998292414
```

Coeficientes obtenidos mediante LASSO. (figura 4.4.1)

	features	vif_Factor		features	vif_Factor
0	grayMean	8.058535e+04			
1	graySigma	2.066673e+04	21	graySigma 1	8.005780e+03
2	entropyMean	4.847876e+03	22	entropyMean 1	4.428803e+03
3	entropySigma	3.962144e+03	23	entropySigma 1	5.098589e+03
4	hMean	7.942564e+04	24	hMean 1	3.299515e+04
5	hSigma	2.076683e+04	25	hSigma 1	8.263393e+03
6	sMean	3.304433e+04	26	sMean 1	4.266018e+04
7	sSigma	4.865507e+01	27	sSigma 1	3.499547e+01
8	vMean	5.181360e+03	28	vMean 1	4.746214e+03
9	vSigma	6.192783e+01	29	vSigma 1	1.503577e+01
10	grayMean 0	1.492708e+04	30	WeirPt1X	2.181726e+06
11	graySigma 0	2.083634e+03	31	WeirPt1Y	9.441640e+05
12	entropyMean 0	1.475363e+03	32	WeirPt2X	1.077333e+07
13	entropySigma 0	1.380218e+03	33	WeirPt2Y	7.118787e+05
14	hMean 0	1.551499e+04	34	WwRawLineMin	4.248188e+00
15	hSigma 0	2.228781e+03	35	WwRawLineMax	9.295935e+02
16	sMean 0	3.072351e+04	36	WwRawLineMean	2.532234e+04
17	sSigma 0	3.787064e+01	37	WwRawLineSigma	5.185550e+03
18	vMean 0	4.037329e+03	38	WwCurveLineMax	8.463212e+02
19	vSigma 0	1.520417e+01	39	WwCurveLineMean	2.494834e+04
20	grayMean 1	2.985415e+04	40	WwCurveLineSigma	5.068000e+03

VIFs obtenidos. (figura 4.4.2)

Una vez que obtuvimos estos valores, los eliminamos de nuestros datos de entrenamiento con el fin de ver si logramos generar mejores modelos. Habiendo eliminado los valores atípicos, generaremos los mismo modelos para ver si obtenemos una mejor efectividad.

Empezaremos generando primero un modelo con Random Forest por ser el que se utilizó en el Paper. Verémos si podemos generar una efectividad mayor a la mencionada por los investigadores.

Resultados del Random Forest trás eliminar datos atípicos :

R^2: 0.823540678021367
MSE: 235541.9314408406
RSMSE: 485.32662346180905
MAE: 200.50039029957205
Error estandar: 485.2362009818441

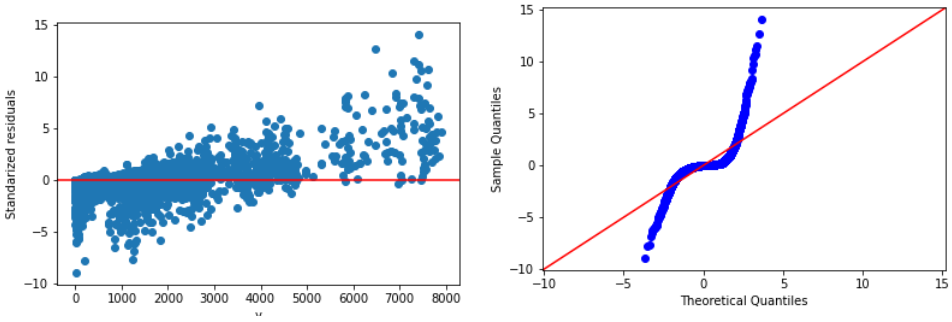


Gráfico de residuos y cuantíl. (figuras 4.4.3 y 4.4.4)

Ahora, generaremos uno con MLP. Esto para contrastar su efectividad con la de Random Forest y comparar cuál es mejor.

Resultados del MLP trás eliminar datos atípicos :

R^2: 0.9059160163932675
MSE: 125585.44920092999
RSMSE: 354.38037361136406
MAE: 165.7657409533583
Error estandar: 354.0705546392842

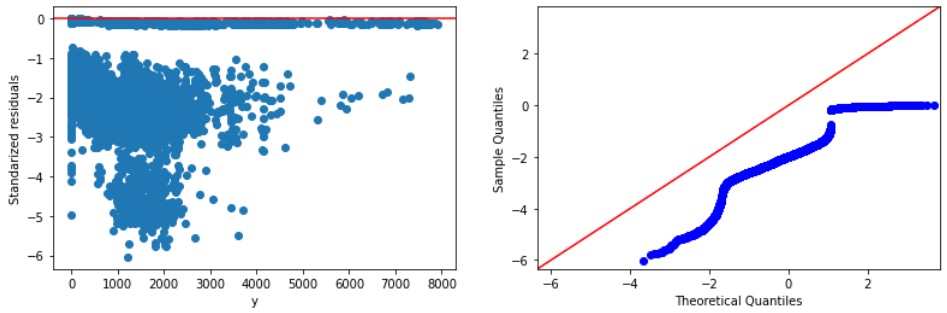


Gráfico de residuos y cuantíl. (figuras 4.4.5 y 4.4.6)

Lo mismo con SVR:

R^2: 0.47833182204956115
MSE: 696334.5933095505
RSMSE: 834.4666520056692
MAE: 370.20593703858617
Error estandar: 818.5175314021255

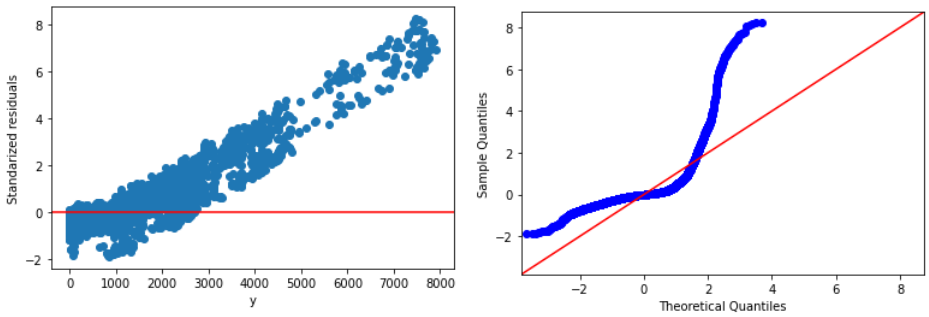


Gráfico de residuos y cuantíl. (figuras 4.4.7 y 4.4.8)

Podemos comparar los resultados antes y después de retirar residuos observando la figura 4.2.9, en la cual indicamos el método y el método sin residuos con la expresión "s/r":

Method	R-squared	MSE	RSMSE	MAE	Std Error
RFR	0.82472	233956.1214	483.6901	200.2240	483.5868
RFR - s/r	0.82354	235541.9314	485.3266	200.5003	485.2362
MLP	0.91066	119252.2292	345.3291	157.9595	345.3897
MLP - s/r	0.90591	25585.4492	354.3803	165.7657	354.0705
SVR	0.47833	696334.5933	834.4666	370.2059	818.5175
SVR - s/r	0.47833	696334.5933	834.4666	370.2059	818.5175
OLS	0.58257	557189.5809	746.4513	471.0722	746.0297

Tabla de comparación de métricas. (figura 4.4.9)

4.5. Acercamiento

Tras analizar las figuras 4.2.9, 4.4.9 y observar con detalle nuestros resultados llegamos a dos puntos de interés:

Es más sencillo predecir el discharge de un cuerpo de agua que su stage, esto es una observación de utilidad para investigaciones futúras.

Si bien tenemos modelos que logran predecir de manera convincente tanto stage como discharge, el aporte que deseamos proveer con nuestro trabajo a la investigación previa tiene como intención no solo proveer modelos alternativos para la predicción sino también optimizar el proceso de entrenamiento e introducción de imágenes válidas a redes comvolucionales.

5. Deep Learning aplicado a la problemática

Con la finalidad de realizar análisis óptimo de la alta cantidad de imagenes proporcionadas por el socio formador, como equipo optamos por un modelo

de clasificación binario aplicado a las imagenes,el modelo tiene como proposito principal, separar las imágenes optimas para análisis, de las imágenes con altas cantidades de ruido, ya sea obscuras, congeladas o inundadas.

Ejemplo de cuatro clases de imagenes encontrables (figura 5.0.1)

5.1. Entrenamiento y evaluación del modelo

Al igual que en la sección anterior del documento, dividimos nuestras imagenes en training, validation y testing, todo esto a la par que importamos las fotografías, también se definió el tamaño inicial y el batch size para el entrenamiento del modelo. La distribución de imágenes fué la siguiente:

group	images	classes
training	571	2
testing	580	2
validation	571	2

Distribución de las imágenes (figura 5.1.1)

Despues de haber importado las imagenes en estos grupos y con dos clases, vamos a empezar el desarrollo y entrenamiento de la red convolucional. Para este caso, probamos diferentes tipo de redes pre-entrenadas, por ejemplo: VGG16, VGG50, MobileNetV2, ResNet101V2, etc. Con la que obtuvimos un mejor accuracy fue con la última; pero no es la única capa que contiene la red. A continuación mostramos todas las capas y parámetros entrenables con las que cuenta:

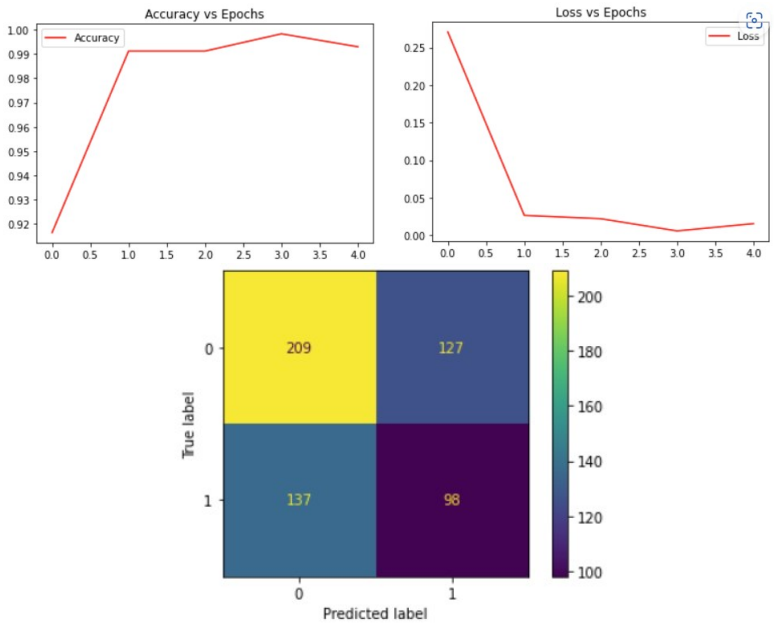
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet101v2 (Functional)    (None, 16, 16, 2048)      42626560

 conv2d (Conv2D)             (None, 14, 14, 64)        1179712

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 64)                802880

 activation (Activation)     (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

 activation_1 (Activation)   (None, 2)                 0

=================================================================
Total params: 44,609,282
Trainable params: 44,511,618
Non-trainable params: 97,664
_________________________________________________________________
```

Descripción de la red convolucional (figura 5.1.1)

Esta red generó que para su entrenamiento tuvieramos que probar diferentes entornos, entre ellos: Collab, una instancia de AWS con el plan gratuito y nuestra propia computadora personal. El objetivo siempre fue que pudieramos probar el mejor modelo que encontraramos sin el obstaculo que pueden llegar a ser los recursos.

Con AWS definimos un tipo de instanciaEl tipo de instancia t2.micro. Nos ofrece una unidad SSD de 30 Gb de uso general de EBS, una sola CPU y una instalación limpia de Ubuntu 18.04 LTS. Seleccionamos este sistema operativo dado que es el entorno con el que más hemos trabajado a lo largo de la carrera. Esta instancia tiene un costo por servicio de 0.0116 USD por hora, lo cual es lo suficientemente bajo para poder trabajar con los 100 USD disponibles. Para pasar las imágenes de nuestra computadora personal al servidor utilizamos la herramienta FTPS que ofrece Linux. A este servicio solamente se le define la carpeta de origen y de destino, el par de llaves de acceso a la instancia y la URL con puerto para hacer en enlace en la conexión; y con eso pudimos hacer la transferencia del total de imágenes. Aunque, después de hacer toda la transferencia, nos acabamos el almacenamiento debido al peso de las imágenes y la cantidad de estas que utilizamos para el desarrollo.

Dado que nos terminamos el almacenamiento en AWS, recurrimos a utilizar Collab con el uso limitado de la GPU. Esto nos sirvió para disminuir considerablemente los tiempos de carga y entrenamiento. Para evaluar nuestro modelo de manera más precisa, generamos una matriz de consufión que nos indicara lo correcto o no que fue la predicción; siendo 1 las imagenes correctas y 0 las incorrectas. Al final pudimos obtener los siguientes resultados:



Graficos de resultados de la red (figura 5.1.3)

Train Loss: 0.0156 - Train Accuracy: 0.9929
Validation Loss: 0.5579 - Validation Accuracy: 0.9507
Test Loss: 0.003451 - Test Accuracy: 0.998248

5.2. Predicción con el modelo

Como vimos anteriormente, la matriz de confusión nos mostró que los resultados no fueron los esperados, pero es considerablemente eficiente encontrando imagenes incorrectas. Nos aprovechamos de eso y vamos a utilizar este modelo de clasificación para solamente filtrar las imagenes incorrectas. Habiendo dicho lo anterior, generamos dos tipos de predicciones. Primero, buscaremos clasificar imagenes que sean de otra temporada diferente a las imagenes con las que entrenamos. Además, buscaremos después predecir imagenes que sean de la misma temporada, pero con 5 años de diferencia. De esta forma podremos comprobar nuestra hipotesis de ver la posibilidad de predecir algo en un lapso temporal pequeño y algo con un lapso temporal largo.

El dataset que generamos contienen las imagenes que nuestro modelo de clasificación puso como incorrectas:

	A	B	C
1	20170303StateLineWeir_20170303_Farrell_008.jpg		
2	20170303StateLineWeir_20170303_Farrell_010.jpg		
3	20170303StateLineWeir_20170304_Farrell_019.jpg		
4	20170303StateLineWeir_20170304_Farrell_033.jpg		
5	20170303StateLineWeir_20170304_Farrell_035.jpg		
6	20170303StateLineWeir_20170304_Farrell_036.jpg		
7	20170303StateLineWeir_20170304_Farrell_042.jpg		
8	20170303StateLineWeir_20170305_Farrell_045.jpg		
9	20170303StateLineWeir_20170305_Farrell_047.jpg		
10	20170303StateLineWeir_20170305_Farrell_065.jpg		
11	20170303StateLineWeir_20170306_Farrell_079.jpg		
12	20170303StateLineWeir_20170306_Farrell_083.jpg		
13	20170303StateLineWeir_20170306_Farrell_087.jpg		
14	20170303StateLineWeir_20170306_Farrell_089.jpg		
15	20170303StateLineWeir_20170306_Farrell_094.jpg		
16	20170303StateLineWeir_20170307_Farrell_109.jpg		
17	20170303StateLineWeir_20170307_Farrell_112.jpg		
18	20170303StateLineWeir_20170307_Farrell_113.jpg		
19	20170303StateLineWeir_20170308_Farrell_128.jpg		
20	20170303StateLineWeir_20170308_Farrell_130.jpg		
21	20170303StateLineWeir_20170308_Farrell_131.jpg		
22	20170303StateLineWeir_20170308_Farrell_140.jpg		
23	20170303StateLineWeir_20170308_Farrell_146.jpg		

Dataset resultante (figura 5.2.1)

Así, podremos leer las imagenes que queremos quitar y las removeremos del dataset de regresión. Para así verificar si el accuracy disminuyó o aumentó.

6. Resultados regresión con las imagenes clasificadas.

Enfocandonos de nuevo en el modelo de regresión, quitamos un total de 300 imagenes con la predicción de diferentes epocas. Habiendo retirado estas imagenes, repetimos los procedimientos anteriores para verificar si nuestra hipotesis inicial era correcta o no. Para este nuevo caso vamos a probar un procedimiento diferente. Dado que vimos con las correlaciones que el stage y el discharge tienen una correlación de 0.97, vamos a intentar predecir uno y utilizar el resultado para predecir el otro. Vimos que tuvimos más efectividad prediciendo el discharge por lo que predeciremos este primero, y luego utilizarémos la predicción para predecir el stage.

6.1. Discharge

Dos temporadas diferentes, mismo año Resultados del Random Forest:

R^2: 0.8428585325677589
MSE: 230830.60878807204
RSMSE: 480.4483414354472
MAE: 199.334044234427
Error estandar: 480.519141246104

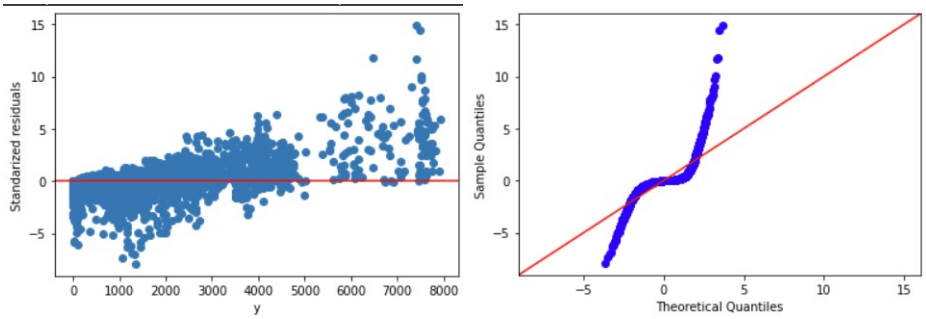


Gráfico de residuos y cuantíl. (figuras 6.1.1 y 6.1.2)

Ahora, generaremos uno con MLP. Esto para contrastar su efectividad con la de Random Forest y comparar cuál es mejor.

Resultados del MLP:

R^2: 0.9157598224645278
MSE: 123743.27771192083
RSMSE: 351.77162721277114
MAE: 160.9001891478277
Error estandar: 350.27568388863943

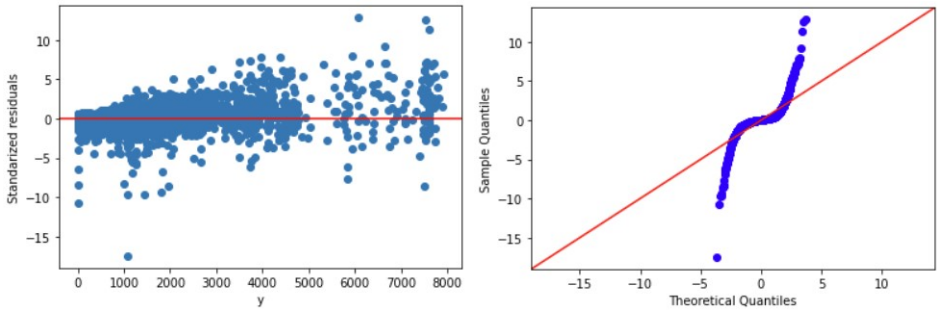


Gráfico de residuos y cuantíl. (figuras 6.1.3 y 6.1.4)

Resultados del SVR:

R^2: 0.4464374257558764
MSE: 813147.4658028209
RSMSE: 901.7468967525316
MAE: 390.64895141446806
Error estandar: 881.2995623089982

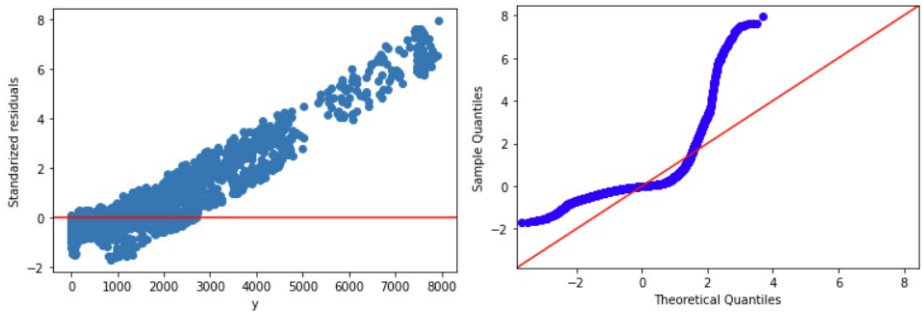


Gráfico de residuos y cuantíl. (figuras 4.4.7 y 4.4.8)

Misma temporada, 5 años de diferencia Resultados del Random Forest:

R^2: 0.8399852532461007
MSE: 235051.26948237605
RSMSE: 484.8208632911501
MAE: 200.801479362815
Error estandar: 484.8879655345423

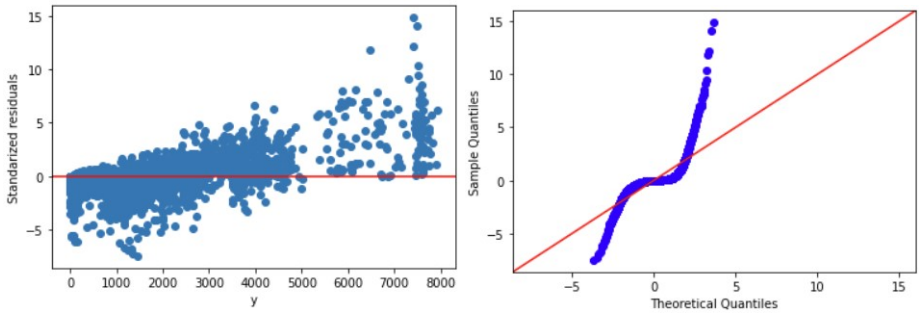


Gráfico de residuos y cuantíl. (figuras 6.1.1 y 6.1.2)

Resultados del MLP:

R^2: 0.9110624489392322
MSE: 130643.48571346734
RSMSE: 361.4463801360685
MAE: 162.61324691171185
Error estandar: 361.17446747102395

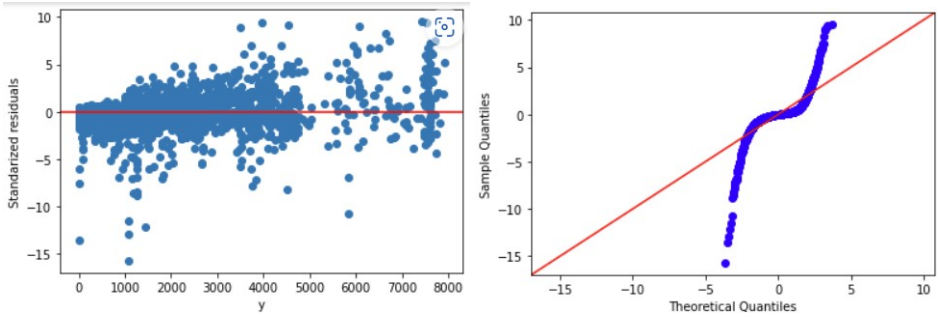


Gráfico de residuos y cuantíl. (figuras 6.1.3 y 6.1.4)

Resultados del SVR:

R^2: 0.4464374257558764
MSE: 813147.4658028209
RSMSE: 901.7468967525316
MAE: 390.64895141446806
Error estandar: 881.2995623089982

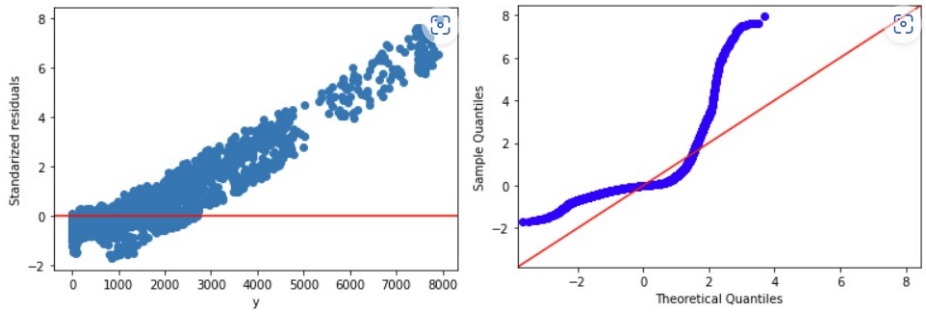


Gráfico de residuos y cuantíl. (figuras 6.1.5 y 6.1.6)

6.2. Stage

Habiendo predicho el discharge, vamos a pasar las predicciones como parámetro a Stage. De esta forma deberíamos obtener una mejor predicción.

Dos temporadas diferentes, mismo año Resultados del Random Forest:

R^2: 0.9348233685274212
MSE: 0.043348343158582964
RSMSE: 0.20820264925928048
MAE: 0.10233146695197334

Error estandar: 0.20823741404732551

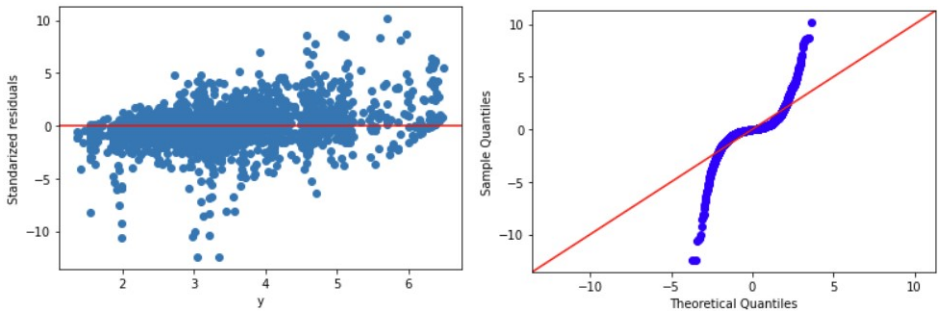


Gráfico de residuos y cuantíl. (figuras 6.2.1 y 6.2.2)

Resultados del MLP:

R^2: 0.9289439205480987
MSE: 0.04725870677867795
RSMSE: 0.21739067776396934
MAE: 0.12115135168475165
Error estandar: 0.21460197784806434

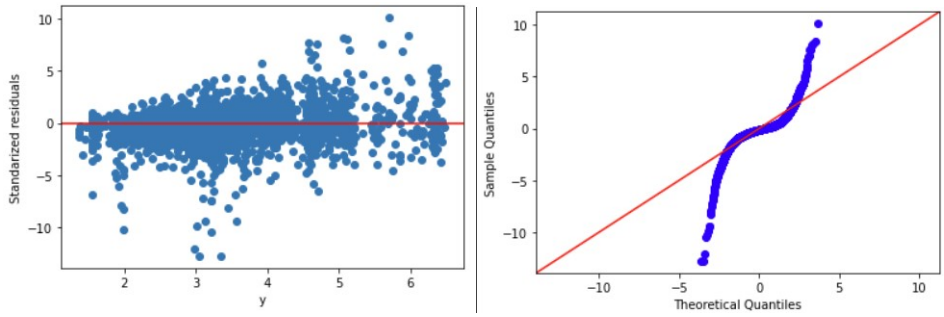


Gráfico de residuos y cuantíl. (figuras 6.2.3 y 6.2.4)

Resultados del SVR:

R^2: 0.9340041135981467
MSE: 0.04389322163735938
RSMSE: 0.20950709209322577
MAE: 0.11508667649868728
Error estandar: 0.20952219080878662

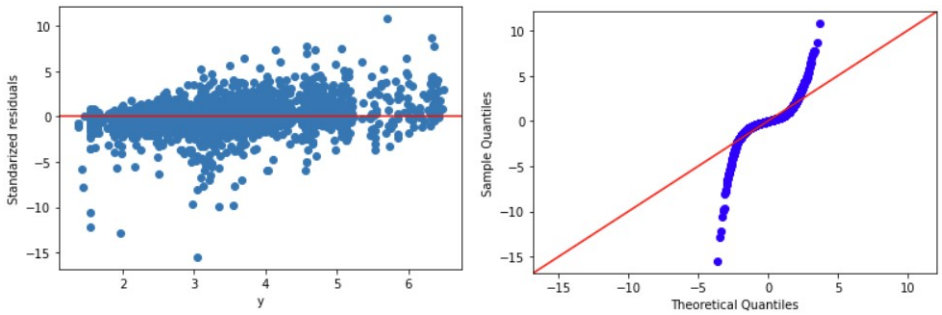


Gráfico de residuos y cuantíl. (figuras 6.2.5 y 6.2.6)

Misma temporada, 5 años de diferencia Resultados del Random Forest:

R^2: 0.9347624280044261
MSE: 0.043388874107227755
RSMSE: 0.2082999618512393
MAE: 0.10250343556823586
Error estandar: 0.20833654038438645

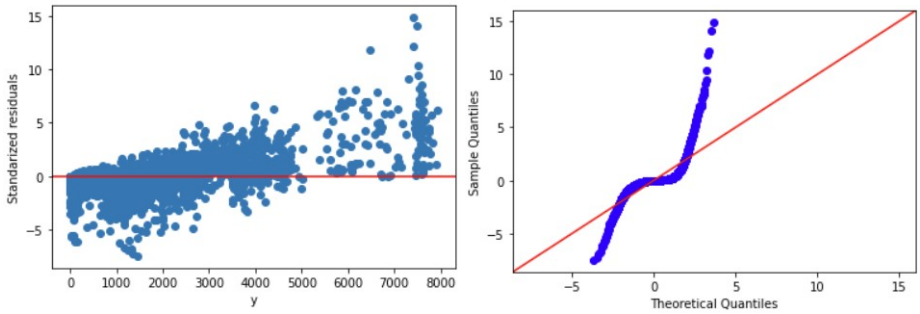


Gráfico de residuos y cuantíl. (figuras 6.1.1 y 6.1.2)

Resultados del MLP:

R^2: 0.9307376890618712
MSE: 0.046065688800834406
RSMSE: 0.21462918906997344
MAE: 0.11710421955192502
Error estandar: 0.2143773137454902

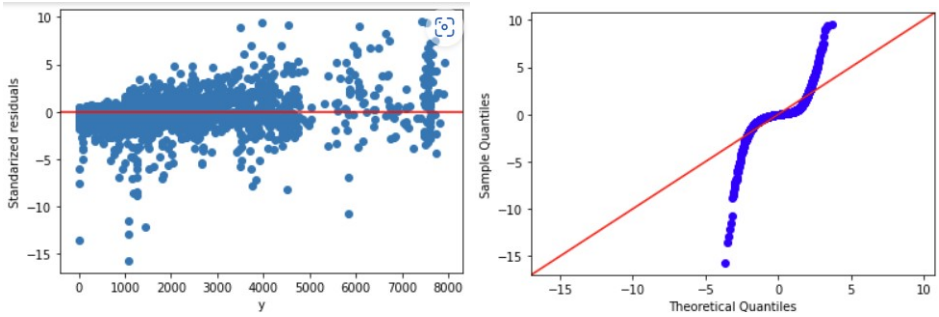


Gráfico de residuos y cuantíl. (figuras 6.1.3 y 6.1.4)

Resultados del SVR:

R^2: 0.9340041135981467
MSE: 0.04389322163735938
RSMSE: 0.20950709209322577
MAE: 0.11508667649868728
Error estandar: 0.20952219080878662

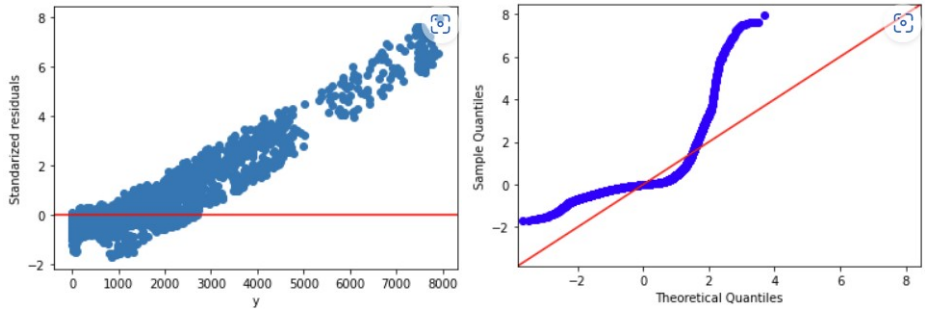


Gráfico de residuos y cuantíl. (figuras 4.4.7 y 4.4.8)

7. Conclusiones Finales

Despues de ver los resultados, creemos que la hipotesis inicial se cumplió en uno de los ambitos especificados.

Primero, pudimos hacer un modelo de predicción con 0.934 de efectividad para el stage y 0.911 para el discharge. Este no se cumplió de la forma que queríamos ya que no llegamos al 0.95 buscado para poder tomar la hipotesis como verdadera. Además, notamos una muy pequeña diferencia en efectividad al hacer la eliminación de las imagenes. Aunque también cabe destacar que solo hicimos eliminación de imagenes por temporadas, no clasificamos el dataset entero.

Segundo, sí hay una diferencia detectable entre la predicción de imagenes en temporadas diferentes y temporadas iguales. Esto nos dice que sí hay una afectación o cambio en las imagenes que se clasifican conforme se cambia de estación. Esto puede ser muy útil para generar investigación a futuro que nos ayude a predecir el cambio en las estaciones por el cambio climático.

Al final, a pesar de no poder generar un modelo que alcanzara 0.95 de exito, creemos que hemos aportado un avance importante para la investigación y el análisis de este río. Se puede tomar el desarrollo previamente realizado para enriquecer las pruebas y el estudio de las diferentes variables que envuelven al río. En general, confiamos y esperámos que esta investigación nos acerque al desarrollo de un sistema que envuelva más objetos de agua, no solo ríos. Para que así, podamos tener un análisis profundo de las condiciones y caracteristicas de los cuerpos de agua que se extienden por México.

Measuring water stage level using machine learning and neural networks

Rodrigo Morales, Jessica Nicole Copado Leal, Carlos Estrada Ceballos, and Andrés Olvera Rodríguez

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

Resumen When site visits are biased toward base flow or fair weather circumstances, time-lapse images of streams and rivers can offer additional qualitative information about hydrologic conditions at stream gauges. Quantitative data from still images from ground cameras are also abundant and can improve monitoring of current flow. Considering a time lapse photograph. It can be useful to fill data gaps when sensors malfunction and/or when funding for monitoring projects is interrupted. Machine learning models that employ scalar image features that could be programmatically computed to fill data gaps in flow meter records were built and evaluated. Automated analysis of time-lapse images from a single camera at a single location. A ground-based fixed camera was used to capture time-lapse photos as part of a watershed documentary imaging project.

Keywords: rivers · hydrology · images · computer models

1. Introduction

Being able to measure, predict and understand how a body of water behaves is a very important task in both hydrology and engineering. This way we can better manage water consumption in industry and cities, we can predict whether there will be a flood or even a drought, we can build better bridges taking into account the change of water throughout the year, and much more. With this objective in mind, work began on a project to create a model capable of predicting both the height and flow of different rivers.

The information with which we started working with comes directly from sensors placed in the water, and data captured by cameras placed in strategic locations so that they could capture the change in the water and the surroundings. The database obtained by these means is a combination of numerical measurements and images, both of which were used to create a deeper model. All the data used was taken from a study conducted by a research group from the University of Nebraska dedicated to solve hydrology issues.

1.1. Problem description

The problematic here is to find a way to predict the state of water (stage and discharge) at any time in the future using ML, sensor data and photographs of water bodies. By creating multiple ML models with various results, we can decide which is more effective to forecast how water behaves in other locations so that this model can be replicated to be able to solve distinct hydrological questions such as water distribution, flood and erosion management, etc.

1.2. Research question

Numerous crucial hydrological concerns, including those regarding floods and droughts, agricultural sustainability, global climate change, and water distribution, must be addressed. These questions depend on water level (stage) and streamflow (discharge). When sensors malfunction or are unable to operate due to a lack of funds, methods have been devised to fill in the gaps in sensor data. Some of these methods include cameras that cost little money and offer a variety of data that is not possible with regularly used sensors.

The research question here would be: Is a huge data set of photos sufficiently detailed to replace significant (or minor) gaps in data on the discharge and stage without manually indicating the data?

2. Work development

In order to begin the stage of analysis and characterization of the data, we employed the ELT (Extract, Load, and Transform) method after downloading the data. To build the most fitting model, we experimented with different variables, and tasks were carried out to clean, visualize, analyze, and transform data. The steps taken were to: create models, separate data by seasons, and to use images to create different models.

- Understand the research paper by reading the information explored by the research group from the University of Nebraska mentioned in their study.
- Analyze the dataset which included features that were combined with tage and discharge data from the United States Geological Survey (USGS) from more than 40,000 daylight pictures acquired at hourly intervals between 2012 and 2019.
- Data preprocessing.
- Modelation of ML and time series models.
- Create a Convolutional Neural Network (CNN) cropping some parts of the images like the edges or focusing on the white water. Also splitting data by years and seasons.
- Results and interpretations.

2.1. Dataset

Two sources provided information regarding the North Platte River State Line Weir site. From 2012 to 2019, the USGS retrieved the real-time stream stage sensor measurements and discharge computations for every fifteen minutes. The Platte Basin Timelapse gave daylight photographs for the same time period on an hourly basis. The total number of daytime photos in the dataset was around 40,000 recordings with 59 colums.

2.2. Methodology

In order to understand the problem in greater depth, it was decided to analyze the data to show its behavior over time and how the different seasons of the year affect our dependent variable. Since the data was measured throughout the year for several years, and therefore has a temporal order, we decided besides the regression model, add a a time series analysis.

A time series analysis is specifically good in these cases as it would allow us to visualize the information in a different way, being able to draw new conclusions and even observe characteristics of our dataset that are quite important for the development of the project.

For the time series analysis the only columns that were vital for this, were CaptureTime and Stage 1. CaptureTime tells us in what date the record was made, an important variable for the analysis since it marks the data in time. Stage is our dependent variable that shows the height of the water in meters. In addition to Stage we have the Discharge variable, the other dependent variable, this one we will not be used since in the analysis of the data, determined that the correlation between both is very strong.

	CaptureTime	Stage
0	2012-06-09	2.99
1	2012-06-09	2.99
2	2012-06-09	2.96
3	2012-06-09	2.94
4	2012-06-09	2.94

Figura 1: Multiple records of the variable Stage in a single day.

The data preprocessing for both the regression and time series analysis was necessary but relatively simple. First the data type of the CaptureTime variable was changed from object to datetime. Next, was to check if there were nan values in the dataset. Then, the creation of the new dataframe with only the CaptureTime and the Stage was created. It is important to mention that the sensors take several measurements in a single day, as can be seen in figure 1, so we have several records of the variable Stage and in order to perform the analysis the relationship between the variables has to be 1:1, that is, for each day there can only be one record of Stage. One way to achieve this is to group all the records per day, although it is great way to build the time series, it's not trustable since we are eliminating important information and there is no real way to determine which of the records in a day is more important than the others. Taking this into account, we decided to obtain an average of all the records in a single day 2, although this also deprives us of relevant data, it is a way to take into account all the data and their weight, without losing them. In addition, columns for month and year were added for further analysis of the data.

	Stage
Capture Time	
2012-06-09	2.965455
2012-06-10	2.931176
2012-06-11	2.931875
2012-06-12	2.965625
2012-06-13	3.042500

Figura 2: Dataframe with the dates grouped.

With our new dataset ready, we started with the analysis. The first thing we did was to make a graph where we could see in a more visual way the change of the river height along the years. Looking at the figure 3 we can see that the data has some seasonality, this is due to the change of seasons and temperatures, which directly affects the water body. For example, in the middle between years we can see large peaks, this represent a higher water level, which is due to high temperatures and possible summer rains. On the other hand, we can observe that, at the change of year, during winter, the height of the water decreases and remains constant for a while, due to the low temperatures and possibly the water freezing at this time.

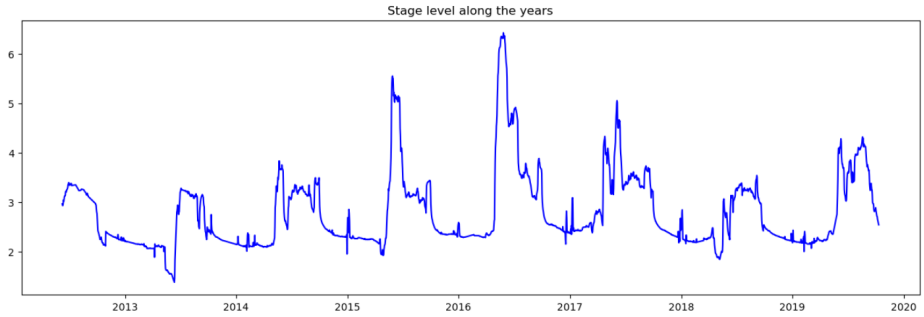


Figura 3: Seasonality of the data graphically represented.

Season	Mean
Spring	2.6861
Summer	2.6342
Autumn	3.5145
Winter	2.2852

Cuadro 1: Stage mean values per season.

Having observed the seasonal behavior, we decided to assign each month a season of the year, February to May would be spring, June to August summer, September to November autumn and December to January winter. This was done in order to be able to divide our data by seasons and see the characteristics of each one. We determined that the most constant season of the year in terms of water level is winter, although the level is approximately 2.2 meters, in summer we also found some constancy with levels of 3.5 even though they are less. These data seem to coincide with the average levels calculated by season shown in the table 1.

After the data analysis, we went on to perform a KPSS test to determine if the information followed a stationary pattern, which means that the data may change, but its change is constant, it is not affected. We performed this test using the kpss function within the statsmodels library, which gave us results greater than the critical value, so we can deduce that it develops in a stationary manner.

With all this information, the next step was to check what type of prediction model was this; MA, ARMA, ARIMA, etc. It would be necessary to test and plot the autocovariance, autocorrelation and partial autocorrelation. These graphs and metrics allow us to determine how the data of a variable is related within itself. By observing the graphs in figure 4, we were able to determine that the model that best fit our data would be an ARIMA model. This model has

both the auto-regression of the AR model and the moving average of the MA model, which makes it deeper and capable of working with more complex data.

3. Results

3.1. Regression model

We decided to make two approaches for the data with the same procedure. The first approach consisted in splitting the data using the train test split function: 70 % of the data for training and 30 % for testing, this data split was made randomly. For the second approach, the dataframe was splitted into two new frames, the training dataframe that contained the records from year 2012 to 2017 and the testing dataframe from year 2018 to 2019. Once the data was splitted, the next step was to remove the residuals with high influence using three procedures:

- Studentized residuals
- High leverage
- Cook's distance 5

For each residual elimination, there was an OLS (ordinary least squares) model built to compare them with each other, also there was one model built deleting all three procedures at the same time. For the first approach we have table 2 and for the second approach we have table 3.

Procedure	$r^2 score$
Studentized residuals	0.788
High Leverage	0.699
Cook's distance	0.663
All procedures	0.802

Cuadro 2: First approach OLS $r^2 tablescores$

Procedure	$r^2 score$
Studentized residuals	0.801
High Leverage	0.711
Cook's distance	0.797
All procedures	0.741

Cuadro 3: Second approach OLS $r^2 tablescores$

For the first approach, the model with the best r^2score was the with all the procedures deleted at the same time with a value of 0.802, as for the second approach, the best model was only deleting the studentized residuals with a value of 0.801. Once this was notices, the final step was to build different ML regression models to see which one will throw the best score. For both approaches, were implemented Decision Tree Regressor, Random Forest Regressor, Support Vecotr Regressor, MLP Regressor, and of course the OLS. As seen on tables 4 and 5, we have the r^2score, the mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE) for each ML model.

ML Regressor	r^2score	MSE	MAE	RMSE
OLS	0.802	0.084	0.217	0.290
Decision Tree	0.826	0.074	0.159	0.272
Random Forest	0.912	0.037	0.109	0.193
SVR	0.305	0.295	0.397	0.543
MLP	-1091.109	464.041	19.270	21.542

Cuadro 4: First approach ML accuracy and errors.

ML Regressor	r^2score	MSE	MAE	RMSE
OLS	0.801	0.206	0.341	0.454
Decision Tree	0.344	0.253	0.341	0.503
Random Forest	0.552	0.173	0.290	0.416
SVR	0.059	0.363	0.460	0.602
MLP	-1637.818	632.177	16.035	25.143

Cuadro 5: Second approach ML accuracy and errors.

As seen on the tables, the ML model with the highest r^2score was the Random Forest Regressor with a value of 0.912 and low error values for the first approach. On the other hand, for the second approach, the best model was the OLS with a value of 0.802 also with low value errors. So, the approach that best predicts the water stage level was the first one.

3.2. Time series model

In order to check the data interpolation, an ARIMA (Autoregressive integrated moving average) model was built. It is an statistical model that uses data variations and regressions with the final purpose to find patterns for it to make

a future prediction. The model has certain parameters that change the behavior of the model itself:

- **p:** Number of lag observations included in the model.
- **q:** Size of the moving average window
- **d:** Number of times that the raw observations are differenced.

It is important to choose the right value for each hyperparameter so thay can throw good predictions, so the auto-arima function was implemented, resulting in p = 3, q = 1 and d = 2.

Once the values of the hyperparameters are set, the ARIMA model can be build. Looking at figure 6 you can see that both lines are practically on top of each other. The model threw coefficients close to 0.05 which is the confidence level the model sets automatically, this means the model is trustworthy.

For a final step with the time series analysis, a function was applied to try predicting the year 2020. Prophet, is a library for forecasting time series based on an additive model where non-linear trends are fitted yearly, weekly, and daily seasonality. In order to predict the year 2020, inside the function you can set the amount of periods you want to predict in this case 365. Figure 7, might be a little hard to comprehend, but will explain it. The black dots are the original recordings from the dataframe, the blue line is the model's prediction and the blue shaded area is the confidence level the prediction has. The image shows the 2020 prediction, but Prophet can predict the time series weekly, monthly, or even all together. You might think, it is a really great approach to predict the stage level unfortunately, due to the fact there are missing values, the model had a score of 0.61. These was really disappoitning, but it is a great to predict the stage level over the years. If the dataframe had all the dates, maybe the value will rise.

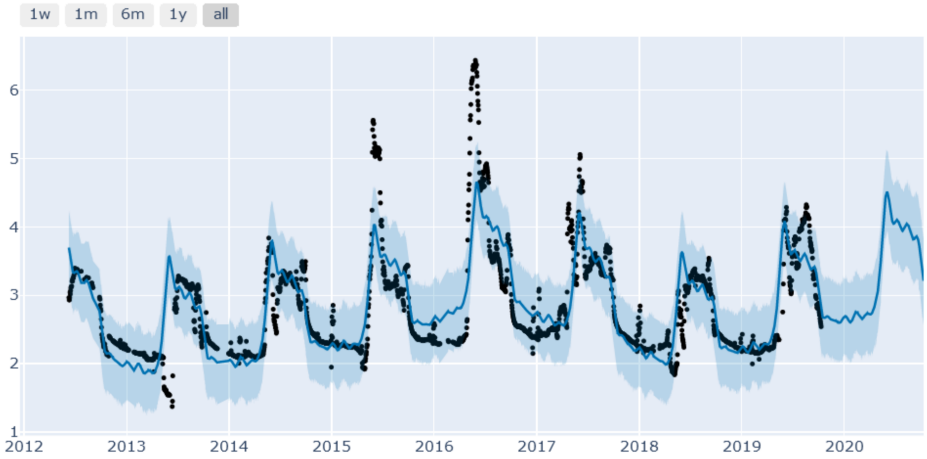


Figura 7: Prophet model accuracy plotted.

3.3. Convolutional Neural Networks

These are composed of neurons with teachable biases and weights. Each neuron receives numerous inputs, weights them, passes the total through an activation function, and then produces an output. They were created specifically to work with images and videos. They use photographs as inputs, extract and learn the image's attributes, then categorize the images using the learnt features. These neurons build the foundation for automatic recognition by learning how to translate input impulses into equivalent output signals.

For the neural networks we tried different ways to clean the variables. First we tried with the original images without removing variables or editing the pictures and then we divided the data set into three parts which was the training, testing and validation. The training part had the most data, then the test part had half of the training data and the validation part had the least.

Train	25,235
Test	12,618
Validation	4,206

Cuadro 6: Dataset Separation.

Once we separated the data set we extracted the precision as well as the errors and the errors that we extracted were mean squared error, root mean

squared error, and mean absolute error.

MSE	0.123
RMSE	0.35
MAE	0.22

Cuadro 7: Error from original image.

We can see from the data that the error is not huge but it is not so low either, so this model is not as accurate as we wanted it to be.

We then proceeded to edit the images by cutting the edges of the images and separated the datasets once again into training, testing and validation, and still got the same errors as mentioned above.

MSE	0.207
RMSE	0.45
MAE	0.313

Cuadro 8: Error from cropped edges.

If we analyze the data we can see that in this way the error is almost doubled, and we obtained an even less accurate model, so removing the edges of the images was not a very good idea.

Finally we decided to crop the image as much as possible to focus on the water foam and separate the data in the same way as previously mentioned. We can see that the error of this model is smaller than cropping the edges of the image, but it is larger than keeping the original image so that this model is also not so accurate. The model that was not so bad is to keep the original images without cropping anything from them.

MSE	0.158
RMSE	0.398
MAE	0.253

Cuadro 9: Error from water foam cropping.

The next thing we did was to focus on separating the data by season and by year, instead of editing the images. We decided that this would be a good idea since the water in a river is affected by the seasons as the weather and temperature changes. Let's first focus on separating the images by season no matter what year it was.

Then we decided to separate the data by year to see how much it changed and to visualize the images. The separation of the data was the training set from 2012 to 2016, the test set from 2017 to 2018, and the validation set in the year 2019. This left approximately 21,000 data in the training, 15,000 data in the test and 5,000 data in the validation.

MSE	0.27
RMSE	0.52
MAE	0.34

Cuadro 10: Errors of separating data by year.

When running this data in the Neural Network we found that the error was not as low as we expected so we could conclude that separating the data by year is not as efficient and cropping the images is not as efficient either.

With these results we realize that working with images is not as useful because what happens with these models is that if it were a person, and if we are trying to find out the depth of a river glancing at some images, this can be difficult because it is not as accurate just by seeing a photograph. For a model, it can be an arduous task if there are not enough images.

We decided to look for alternatives to work with these images and we realized that there is the segmentation of them. Basically what we try to do with this technique is to separate the elements of the image to leave the water of a single color and make it easier to create a predictive model that can serve us in the future for certain bodies of water that are in different locations, and thus be able to work in hydrology studies to optimize water distribution, predict droughts and floods, etc.

3.4. Segmentation

As a final idea to be able to go deeper in the handling of images for the prediction of variables, we tried to perform segmentation on the images to separate, based on the colors of the image, the different parts that compose it, this includes the water zone and the shore, vital parts for the correct analysis of the photographs. To achieve this we imported a pre-trained segmentation model that is available in Github [], we only passed the training images, training

annotations and the number of epochs for which it would train.

After that it was simply a matter of passing an image as input and observing the result image. Although we got acceptable results as observed in the image 13, it is still not a model that can be applied to all our images, especially those taken on snowy days because the large amount of white color in the image does not allow the model to discern between the different elements of the scenario.

We believe that segmentation is a good approach to what could be a viable solution to the challenge, it is a matter of having more time to train the model in question, see different ways to separate the pixels, and see how to handle special cases such as snow.

4. Conclusions

For daily water management, flood forecasting and management, determining compliance with water use agreements, and for the design of reservoirs, water supply systems, and bridges, accurate measurement and modeling of stream stage and discharge are crucial. The project's goal was to fill in data gaps for stream stage by automatically extracting features from image series using machine learning models for one location with a single camera in order to determine whether it would be feasible to apply these techniques to other locations with various image formats, lighting conditions, etc.

Along with finding features that can properly predict stage and discharge in simple situations for a specific location, it's crucial to make sure that the data utilized to develop a model in the training set encompasses the full range of values found in the test set. The model's predictions become more questionable if they are used to forecast stream stage and discharge for values other than those in the trained data set. To determine whether the training set is sufficient to create a good model to close the gap, photographs from the test set can be compared to images from the training set.

Another goal of the project was to create a foundation of software libraries that are well-suited to carry out hydrology-centric research methods in a manner that enables hydrologists to think more about hydrological questions and less about the technical aspects of the research tools. Therefore, it would be ideal for these libraries to be developed further so that they could employ more than just photos and stage and discharge measurements, compute image characteristics, and combine the data for machine learning tasks.

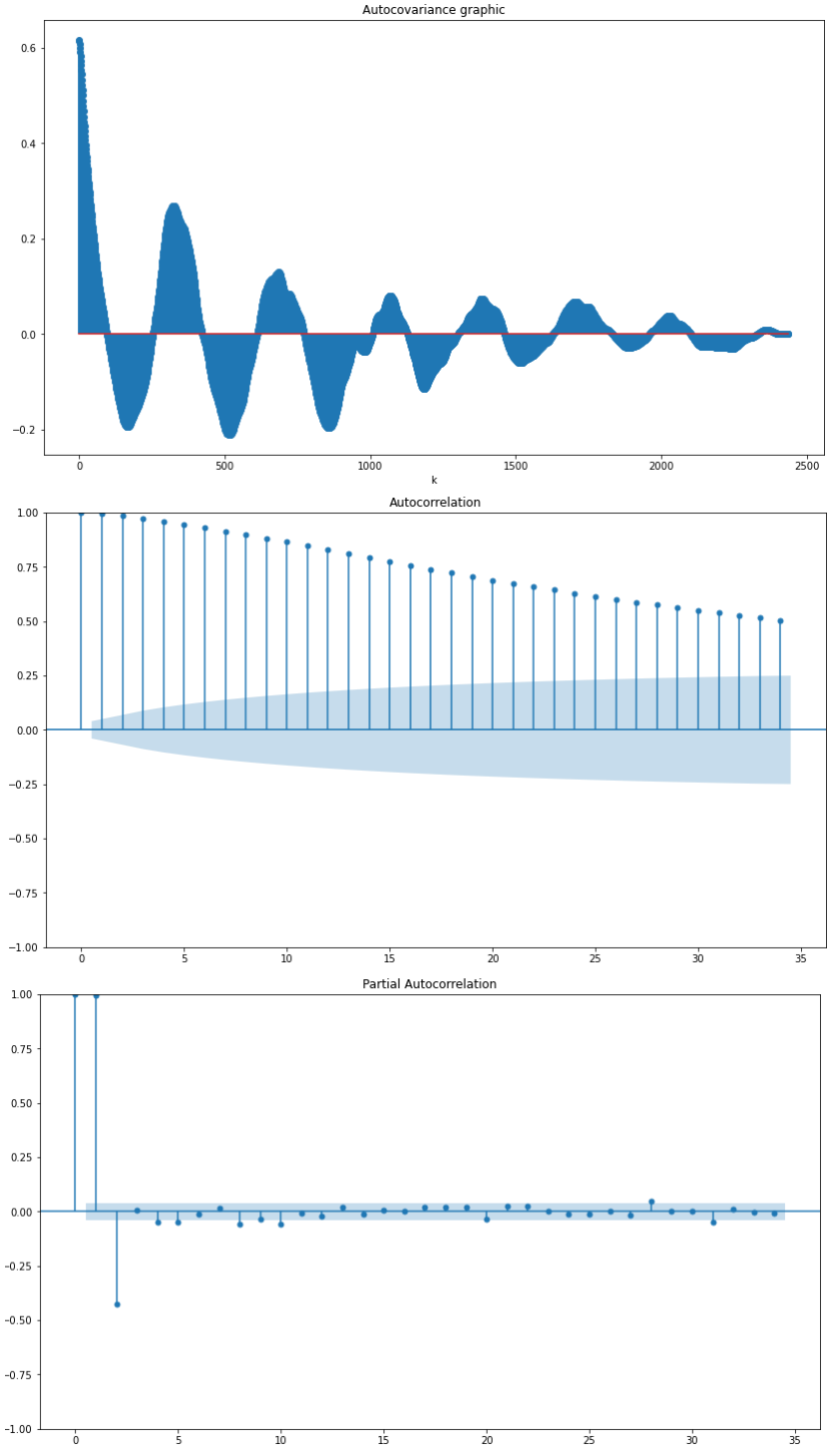


Figura 4:
1. Autocovariance plot
2. Autocorrelation plot
3. Partial autocorrelation plot

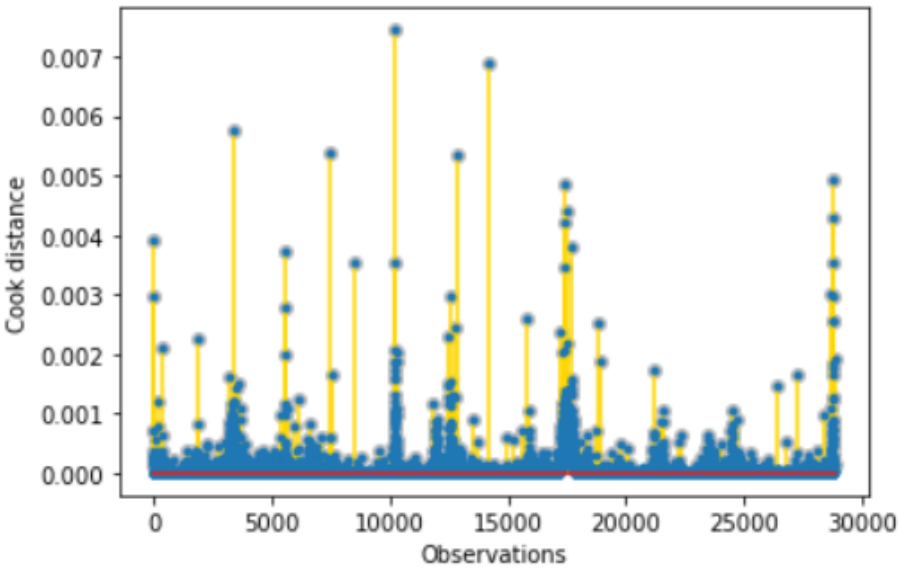


Figura 5: High cook's distance residuals in second approach.

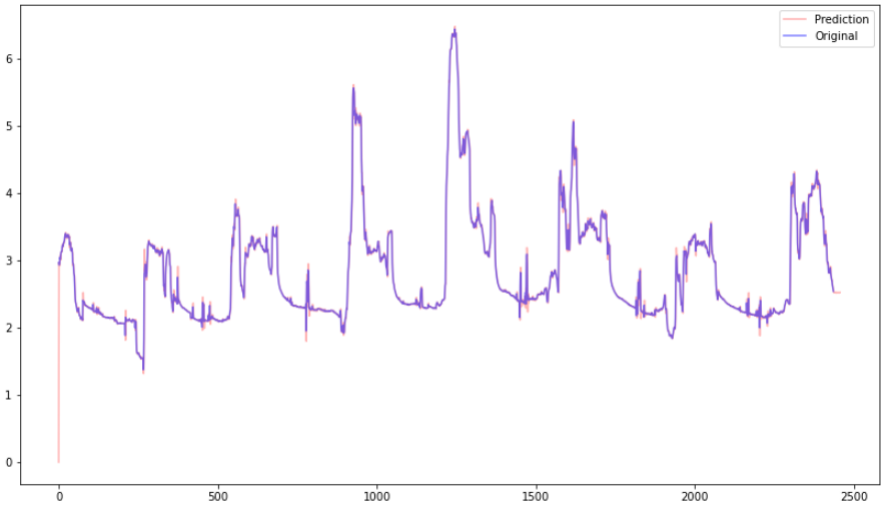


Figura 6: Overlapping graphs of prediction model and actual data.

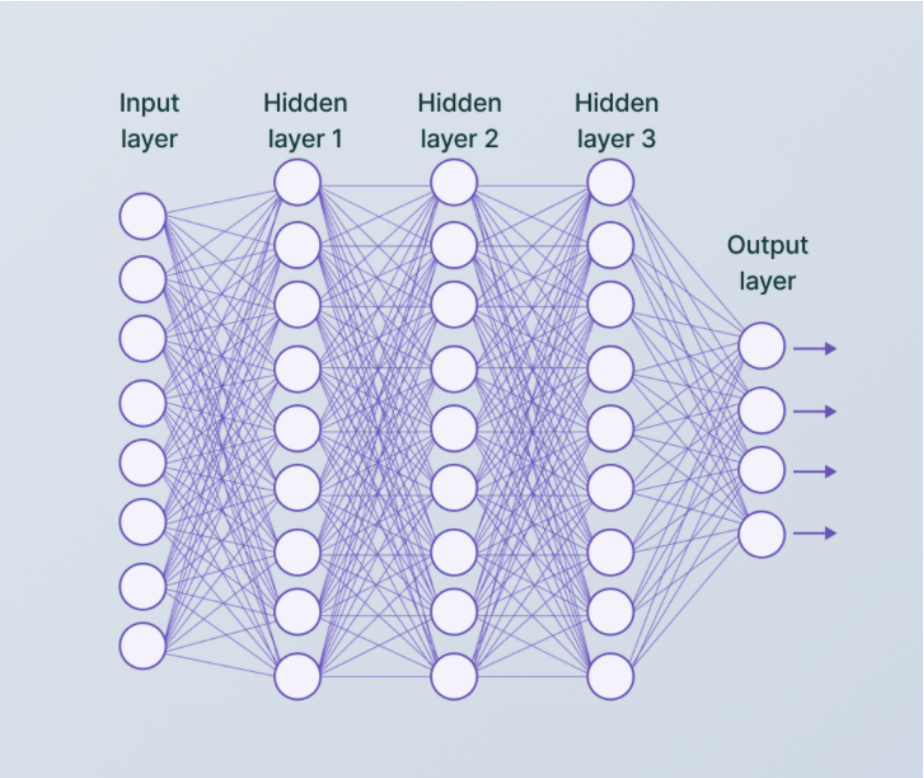


Figura 8: Convolutional Neural Networks

Figura 9: Original Images

Figura 10: Cropped Edges from Images

Figura 11: Cropped Water Foam from Images

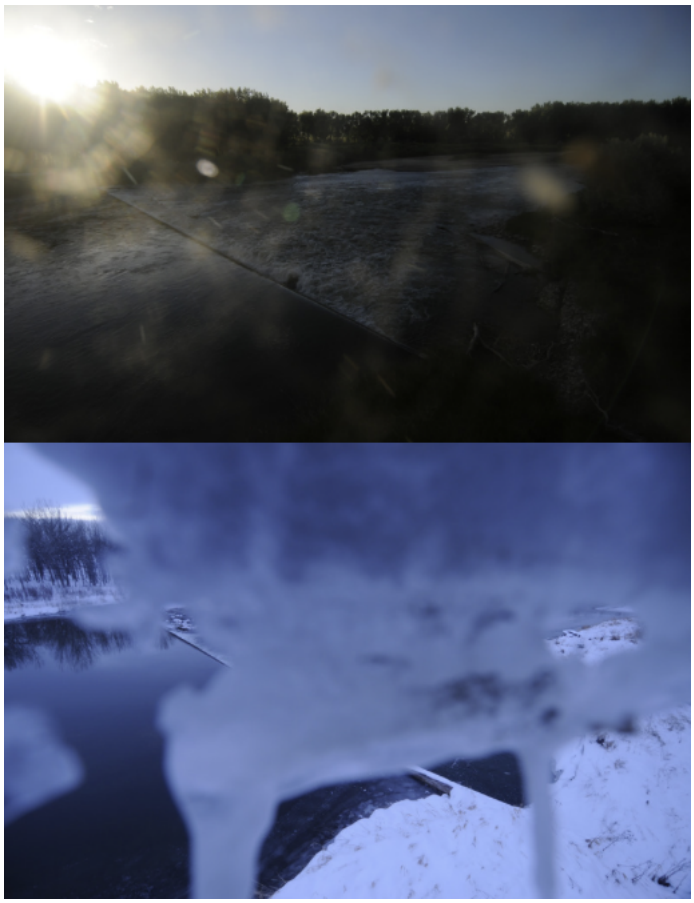


Figura 12: Images of different seasons

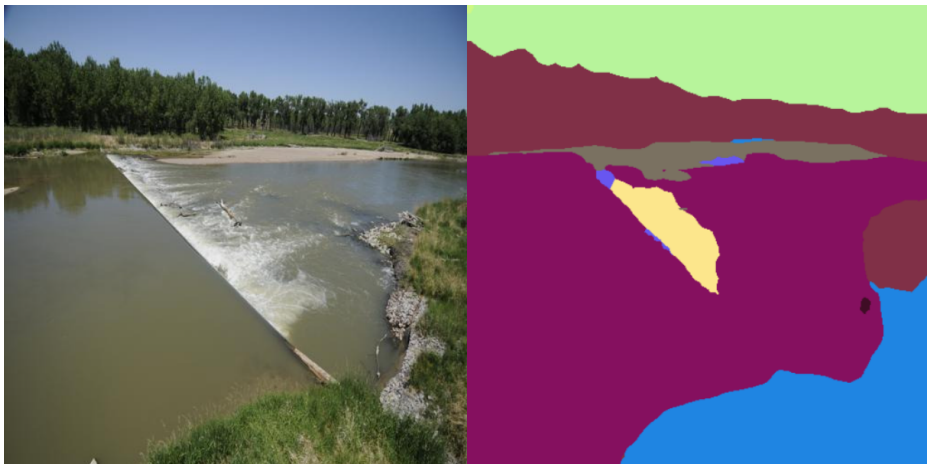


Figura 13: Resulting image of our segmentation model.

Bibliografía

[1] Agarwal, A., M.S. de los Angeles, R. Bhatia, I. Cheret, S. Davila-Poblete, M. Falkenmark, F.G. Villarreal, T. Jonch-Clausen, M.A. Kadi, J. Kindler, J. Rees, P. Roberts, P. Rogers, M. Solanes and A. Wright, 2000: Integrated water resources management, TAC Background Paper No. 4, Stockholm, Sweden, Global Water Partnership.

[2] Cohn, T.A. and H.F. Lins, 2005: Nature's style: naturally trendy, Geophysical Research Letters, v. 32, no. 23, https://doi.org/10.1029/2005GL024476.

[3] Dou, G.; Chen, R.; Han, C.; Liu, Z.; Liu, J. Research on Water-Level Recognition Method Based on Image Processing and Convolutional Neural Networks. Water 2022, 14, 1890. https://doi.org/10.3390/w14121890

[4] Koutsoyiannis, D., 2003: Climate change, the Hurst phenomenon, and hydrologic statistics, Hydrological Sciences Journal, v. 48, no. 1, pp. 3-24.

[5] Qiu, R., Cai, Z., Chang, Z. et al. A two-stage image process for water level recognition via dual-attention CornerNet and CTransformer. Vis Comput (2022). https://doi.org/10.1007/s00371-022-02501-6

Camera-based Water Stage and Discharge Prediction with Machine Learning

Andres Eduardo Nowak de Anda, Isaac Emanuel García González, Samuel Alejandro Diaz del Guante Ochoa, and Ernesto Lopez Villarreal

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

Resumen In the Advanced Artificial Intelligence for Data Science II class, a project was carried out in conjunction with the University of Nebraska-Lincoln, which had the objective of obtaining a model that predicts the accumulation and flow of water, this through the use of a database and images of photographs of a weir taken from the same angle over the years (2012-2019). The database was divided into two types of features, the generic features and hand-crafted, the generic features are those obtained through the analysis of the pixels of the images, such as colors, intensity and entropy, obtaining the average and the sum of the values of the above mentioned characteristics, the hand-crafted features, were obtained through the analysis of an expert in the area of hydrology, which identified points of interest in the photos of the weir, such as the region of the weir, the foam generated, texture, etc. There were created two types of artificial intelligence models to analyze the numerical information (MLP, KNN and random forest) and the photographs (CNN, Segmentation-MLP), the results obtained from the MLP model and CNN model were good, but they weren't convincing the predictions look more like noise of a model just trying to get the lowest error instead of understanding the problem, but the Segmentation-MLP model had results that were convincing that this model could work for majority of rivers and could be able to generalize better.

Keywords: Weir · stage · discharge · loss · accuracy · CNN · Segmentation models · MLP · river

1. Introduction

According to Oracle artificial intelligence [12] (AI) refers to those systems or machines that can replicate human intelligence and can improve iteratively from the information they collect. The application of AI is becoming more and more common in our lives, because there are more and more useful applications. According to the Harvard Business Review [11], companies use AI mainly for:

- Detect and deter security intrusions (44 %)
- Solve users' technological problems (41 %)
- Reduce production management workload (34 %)

- Measure internal compliance in the use of approved suppliers (34 %)

Within AI there is a discipline known as machine learning, whose algorithms review data and become capable of predicting future behavior. Machine learning uses algorithms to identify patterns in the data, and those patterns are then used to create a data model that can make predictions. With more experience and data, the results of machine learning become more accurate, much in the same way that humans improve with more practice.

This paper describes the development and use of different prediction models with the help of machine learning algorithms in a real problem. The problem is presented by the University of Nebraska-Lincoln where it is requested that, by analyzing images and a database with numerical values, a way to predict flow and water quantity values and find if he variables in the dataset and images given by the University of Nebraska-Lincoln are not going to be sufficient to be able to build a prediction model.

Throughout the paper, different forecasting models will be presented using various sources of information such as numerical data and images.

2. Numerical Models

2.1. Dataset analysis

The database given by the University of Nebraska-Lincoln, contains 42,059 rows and 58 columns of variables, divided into two types of data, these were obtained with collaboration from different academic areas, image engineering and hydrology, dividing the columns into generic and handmade. The generic data are values that were created based on the analysis of the images (*images are in .jpg format, and images are size of wdith: 4288, height: 2848, with color*) and pixels, like colors, intensity and entropy, the handmade features were chosen by the analysis of a hydrology expert, based on what is observed in the photos. This process of selecting characteristics was automated through the use of scripts to obtain the characteristics indicated by the hydrologist. The dependent variables to forecast with the models are stage and discharge 1, these variables describe the amount of water and flow that exists in the weir. And the time intervals in which the readings were taken by the sensors vary considerably from a couple of minutes to hours or even days because there wasn't the same amount of photos as sensor readings so only the sensor data that was taken in a range of 5 hours were saved.

The first analysis of the dataset that was done was a correlation graph of all the variables to observe if there was any relationship with the dependent variables and thus have the knowledge of which variables to use or discard for the creation of the models, eliminating columns with low correlation and lowering the complexity of the final model avoiding unnecessary data. The most popular method for calculating correlation is the Pearson's coefficient, which measures the linear dependence between two variables as long as they are continuous and quantitative. Other heatmaps were also developed using Spearman's

?? and Kendall's ?? coefficients. All three plots show similar results with the Pearson ?? correlation giving the best results. From these correlation graphs two variables were removed 'AreaFeatCount' and 'WwCurveLineMin', because the graphs showed that they didn't have any correlation with the other variables.

Then columns that were not necessary for the prediction of the dependent variables were removed, which in this case were the columns 'Filename', 'Agency', 'SiteNumber', 'TimeZone', 'CalcTimestamp', 'Width' and 'Height' since they do not affect the aforementioned analyses and are data that were obtained at the time of data storage, giving only context to the information and do not give value to the database.

The Standard deviation (SD), variance and coefficient of variation (CV) of all the data was then calculated in order to observe the dispersion of the values in relation to the dependent variables. Values equal to zero were removed to recalculate the values and compare the results of the two calculations 2. These values were also calculated by month to see how much they differ by month in all the years 3. Comparing the values between the dataset with and without values equal to zero, there is no significant difference while the values between each month is quite different from month to month, but in average the CV was very low for stage and discharge, so the error for our model has to be considered based on the standard deviation of the data.

Range	Stage	Stage without zero	Discharge	Discharge without zero
Min	0	1.37	0	6.73
Max	6.49	6.49	7,920	7,920

Cuadro 1: Min and Max range for Stage and Discharge variables

	Stage	Stage without zero	Discharge	Discharge without zero
SD	0.80	0.80	1,192.27	1,200.88
Variance	0.65	0.65	1,421,513.21	1,442,125.45
CV	0.28	0.28	1.23	1.18

Cuadro 2: Standard deviation, variance and CV for Stage and Discharge variables

For the outliers, an analysis was made of how many values had a large difference in relation to the others in the same column, as they could show significance when making the models and should not be eliminated, in the case of this database, only the values equal to 0 were eliminated because it was observed that in those same observations, the images still show a certain amount of water and in

	Stage	Discharge
Max SD	1.33	2,197.79
Mean SD	0.39	558.25
Max Variance	1.77	4,830,301.55
Mean Variance	0.29	729,272.88
Max CV	0.41	1.52
Mean CV	0.13	0.63

Cuadro 3: Mean and Max values by month of all years for standard deviation, variance and CV for Stage and Discharge variables

turn flow through the weir, so it was decided to eliminate these values as they showed discrepancy when compared with the photos.

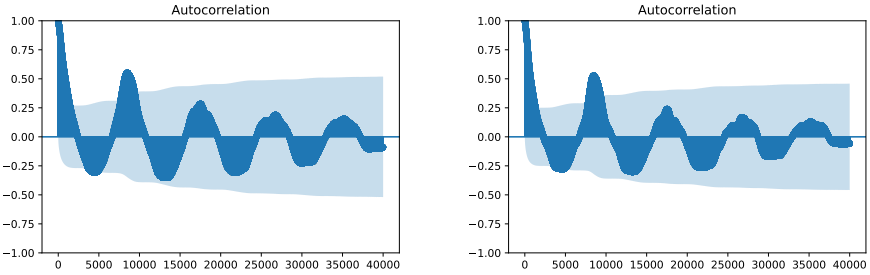
Subsequently, we checked whether the time between each observation was equidistant from each other, in order to verify whether the dataset lends itself to be treated as a time series. If this was not the case, the observations would have to be modified to comply with a time series. Coincidentally, both the observations recorded by the sensors and the camera are the same, but the time between each recording is different, on average from one and a half hours to almost two whole months. To correct this unforeseen issue, filling missing data to make it equidistant by one hour (using forward filling of data and mean for the rest of the values) and removing duplicate time values were done. This also ensures that there is no null data.

Two tests were performed to see if the data were stationary or not, Dickey-fuller and Kwiatkowski-Phillips-Schmidt-Shin (KPSS), the first method showed that the values are stationary and the second method showed that they were not stationary. With each analysis different results were obtained depending on which test will be used, demonstrating the conflict between the values are or are not stationary, but based on the graphs and previous analysis it can be concluded that the KPSS test is probably the correct one, since no stationarity is found in the data.

Autocorrelation 1 and partial autocorrelation plots 2 were created to verify that the data is correlated. The graphs show sinusoidal shapes that serve as evidence that they are not stationary. It was determined that approximately any value of the time series has a lag of approximately 5,000 observations which means it depends on the previous 5,000 data to determine its stage and discharge.

Seasonality **?? ??** plots were made, to verify that the data have a seasonal behavior, the dataset was divided by months, joining in 3 months, by four each year, thus dividing it into the seasons of the year, showing that there is seasonality in the years.

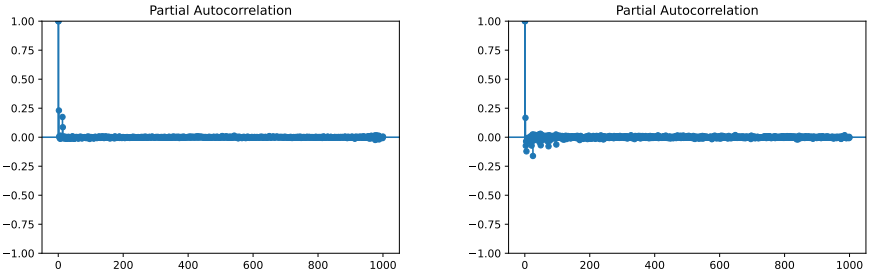
A time series analysis 3 was made in order to obtain through graphs, to observe if there is any relationship with time, these being divided by months of the year, gathering all the data by year and see if there are differences in



(a) Autocorrelation of stage values with a lag of 40,000.

(b) Autocorrelation of discharge values with a lag of 40,000.

Figura 1: Plots of autocorrelation.



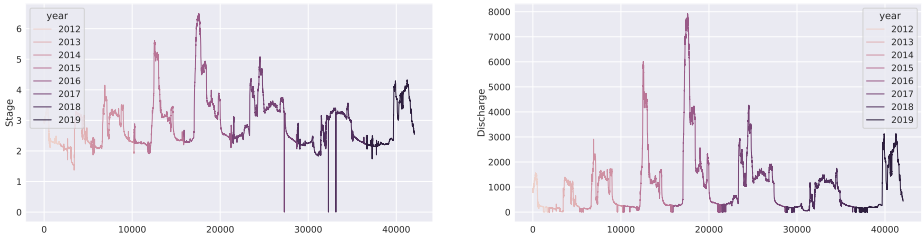
(a) Partial autocorrelation of stage values with a lag of 1,000.

(b) Partial autocorrelation of discharge values with a lag of 1,000.

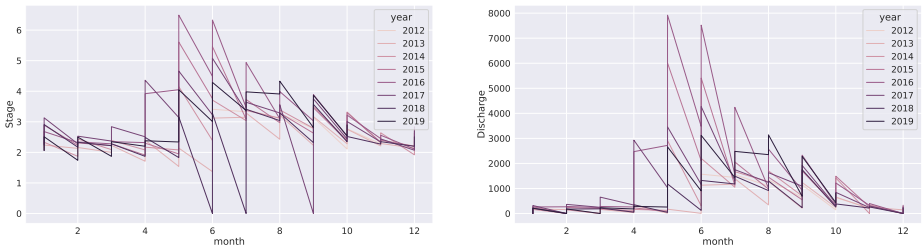
Figura 2: Plots of partial autocorrelation.

the months of each year. It was observed that each month there is an increase in the values of the dependent variables, thus demonstrating that they have a similar behavior in all years. Similarly, this same process of time series analysis was performed using only the columns containing generic values. All graphs give similar results for the data set containing both generic and hand-crafted variables. Another interesting thing found in the time series by month is that there is a very big spike at the start of each month, maybe because the weir is closed at the end of the month and then opened at the begging of the month, this gives us predictable information that can be used to train our models.

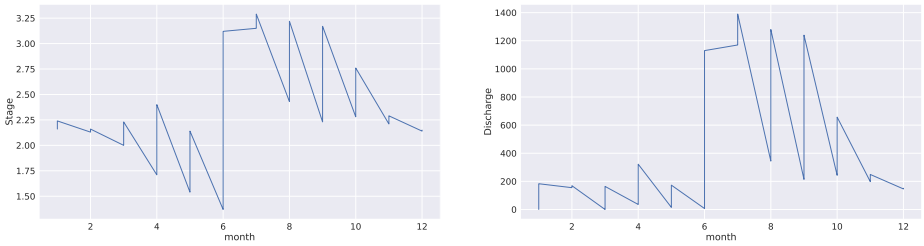
And lastly because of the standard deviation of the data, the seasonality and pattern of the data, the data was **divided by years for training and testing** (ex. train: 2012 to 2017, test: 2018 to 2019), instead of grabbing data randomly from all the dataset, because grabbing randomly would mean that the training dataset is going to see the values from the testing dataset, they are so close to the average and the pattern is similar for all the data. **The dependent variable to**



(a) Time series through the years of Stage and Discharge. X is all the variables ordered in time, Y axis is the stage and discharge variables.



(b) Time series by month of all the years of the variables Stage and Discharge.



(c) Time series by month in the year 2013 of Stage and Discharge.

Figura 3: Time series plots.

use is only the stage variable, because stage represents the height of water and discharge the volume in time. Only the stage variable was used, because with a picture it would be difficult to get the information of movement, so only the stage variable is used as output in all the models.

2.2. Models

2.3. General information and preprocessing

General information For these models the data was divided by train (2012 to 2017) and test (2018 to 2019), with (**46 input variables**, combining generic and handcrafted variables and only **1 variable as output**, the stage variable). The training of the model is possible thanks to the cross-validation method, this is used to find the best variables for each model. CV consists of repeating

and calculating the arithmetic mean obtained from the evaluation measurements over different partitions.

Preprocessing Standard scaling based on the training data was applied before passing the values to the models

MLP The first model that was used is the multilayer perceptron [2] (MLP) 4, this is an artificial neural network formed by multiple layers which serve to solve nonlinear problems, as does a simple perceptron, this model is connected in a unidirectional way, passing the information from left to right with different possible numbers of inputs and outputs, The input layer is where the information is delivered and has no processing, the output layer is the last layer of the artificial neural network, where the results of the processing of the hidden layers will be obtained and the hidden layers are the ones that will process the information, being that unlimited layers can be added for possible better results, these being found between the input and output layer.

Each neuron created in the artificial neural network [3] contains a constant value showing the relative weight which provides the importance of the input within the function of the neuron, in addition, they contain an activation function that defines the output of a node given one or more inputs. The activation function used in this model is the Rectified Linear Unit (ReLU) 1, which transforms the negative input values to a 0 and positive values leaves them as they are. ReLU function

$$ReLU, f(x) = max(0, x) \tag{1}$$

MLP model	
layer	**Number of neurons**
input	46, ReLU
1st layer	512, ReLU
2nd layer	256, ReLU
3rd layer	128, ReLU
4th layer	128, ReLU
output	1, Linear function

Cuadro 4: MLP model used.

Random Forest for regression The second tested model was the Random Forest Regressor. This type of model is based on a decision tree. The flow of

information starts at the root of the tree and follows a path according to the conditions it encounters at each node until it reaches a leaf node (a node that does not point to any other nodes). Another factor that characterizes this type of model is ensemble learning, which consists of the process of using multiple models at the same time, generally trained under the same information, but there exists an alternative that samples the information in each iteration and obtains the average of the results in order to calculate a more accurate forecast.

KNN The last implemented model that used the numerical dataset was a variation of the K Nearest Neighbor (KNN) known as KNN regression. This model works on the same principle as the classical version where a number of neighbors are chosen to form nearest neighbor clusters to an observation in the plane. However, instead of generating classification areas, the nearest neighbors will be used to calculate an average that will be the value corresponding to the dependent variable of that observation.

Distance metrics to determine the closest observations include calculations like the Euclidean distance, Manhattan distance or Minkowski distance.

3. Image Models

3.1. General information and preprocessing

General information The dataset was divided in train (2012 to 2016), validation (2017), testing (2018 to 2019), with (**2 input variables** for the two models and, **1 variable as output**, the stage variable).

Preprocessing Neither the input or output variables were scaled for the following models.

3.2. Image dataset analysis

The second part of this study consisted of performing a direct analysis on the images captured at the weir site, making the appropriate modifications and creating predictive models based on the photographs.

Images were originally set at an average resolution of 4288 x 2848 pixels and occupied an storage amount of approximately 240 gigabytes (GB), so the size of the photographs was scaled down to 512 x 512 pixels to decrease the average size of a file. After this reduction, the total dataset was approximately 2 GB.

Just as the first part of the paper, an analysis of the variables along with the images was elaborated to identify any discrepancy between the numerical values obtained by the sensors and the image that belongs to that observation. By examining the rows that have zero water level and comparing it to the corresponding image, consistently some water can be found in each observation without exception. It was decided to remove the observations from the data set that meet this condition.

3.3. Models

CNN model

Preprocessing The content of the photographs was modified so the majority of the image would only contain the river, hiding the top part of the image with a solid black shade. This was done in order to cut out unnecessary information, and hopefully reduce noise that could bias the learning ability of the models.

The first model used for image analysis was a convolutional neural network [5] (CNN), it is a type of artificial neural network and a variation of the multilayer perceptron, based on neurons of the primary visual cortex of a biological brain, using two-dimensional matrices for image classification and/or segmentation. For the extraction of segments and features of images it uses alternating layers of convolutional neurons and downsampling neurons. When processing the data, its dimensionality is reduced and the pixels in each segment are analyzed, thus identifying the color of each pixel of the image on the RGB scale from 0 to 255 and then normalizing them to a scale of 0 to 1.

In the image processing phase, a group of pixels is selected, then it is mathematically analyzed in a scalar product against a smaller matrix called kernel. The process manages to visualize all the input neurons from left to right and from top to bottom, this results in the generation of a new output matrix.

The CNN model uses a Residual Neural Network (ResNet 50 layers **??**). This model is a combination of the results of the ResNet model and the variable month when each photo was taken 4 as input to a MLP model that used the ELU 2 function as activation for the layers and a linear function for the output layer.

$$ELU = \begin{cases} x & \text{for} x >= 0 \\ \alpha(e^x - 1) & \text{for} x < 0 \end{cases} \tag{2}$$

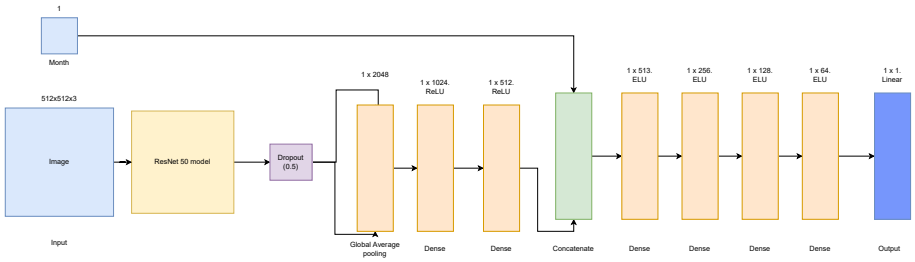


Figura 4: ResNet-MLP model architecture used.

A CNN model was used following the hypothesis statement that maybe the CNN model can find some information in the image that could explain the phenomenon of the stage variable, perhaps the model by itself could find something

in the image that could explain the stage of the river. This information would be combined with the month the data was taken, because of the phenomenon observed in the time series plots (that the data has a pattern that repeats each month).

Segmentation model and MLP

Important information The river width variable is calculated using Euclidean distance from left shore to right shore, the binary mask and the river area is calculated summing all the 1 ones in the binary mask.

preprocessing For the segmentation model, data augmentation techniques were applied as part of the image processing. Data augmentation consists of methods to artificially increase the number of elements from existing parts by slightly modifying the photograph to achieve a greater diversity of features in order to reduce the impact of obstacles when training machine learning models such as over-fitting. Filter techniques were also implemented in the image to randomly modify the color, saturation, brightness and contrast values, horizontal rotation of some images, cropping and resizing of some sections of the image without changing the image dimension and random rotation. No scaling is done for the input or output variables.

The image segmentation model is a sub-domain of computer vision and digital image processing which aims at grouping similar regions or segments of an image under their respective class labels. Image segmentation is an extension of image classification, where in addition to the classification of objects in images, the model also locates where the object is.

For this model a segmentation model is first trained to be able to do the segmentation of the river images, using a U-Net model with a seresnext101 backbone (the encoder of image for the model), and the output of the segmentation model is a binary mask, where 1 is the river and 0 is everything else. Then from the **binary mask, the river width and river area** is calculated to later be used in an MLP model.

The second model is an MLP model 5 that consists of two inputs, the river width and month and one output, the stage variable. This MLP model uses as activation the ReLU function 1.

The reason this Segmentation-MLP model was used, was because of an hypothesis, a natural river formation [10] is a V-shape figure **??** because of water erosion, so wouldn't that mean that the deeper the river gets, wouldn't the river also get a bigger width and a bigger area thanks to the V-shape of the river. Furthermore, there is a paper [1] that confirms this suspicion. In this paper they calibrate their cameras to get a conversion from pixels in the image to centimeters. Then, they would get the width of the river from the images and with the bathymetry of the river they would calculate the stage of the rivers. In this model instead the stage of the river is calculated based on an MLP model that learns the formula to calculate the stage.

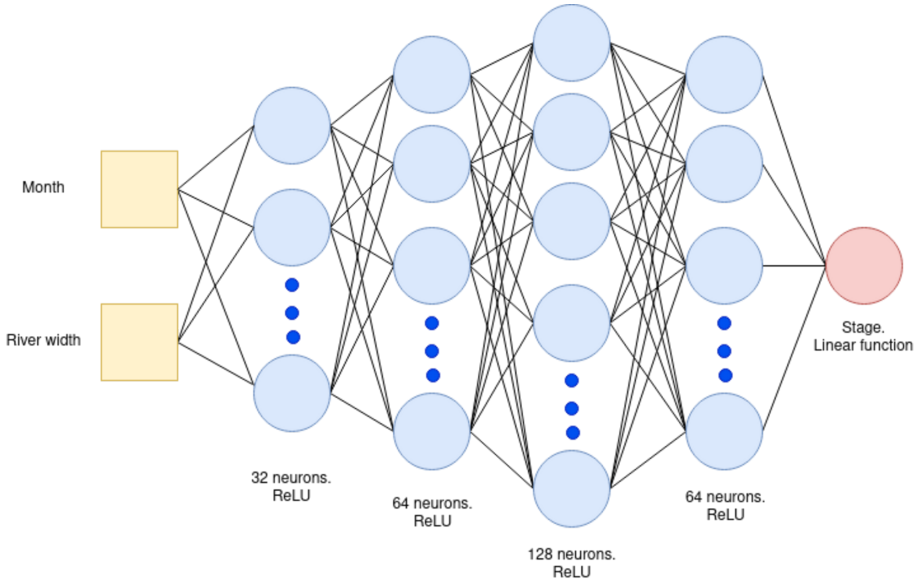


Figura 5: MLP using data from the segmentation model and month.

4. Experiments

4.1. Numerical model experiments

General information The inputs were passed with a standard scaler and the models were trained with cross validation using *random search* to find the best combination of parameters for the models. And the output of the models was the stage variable.

1st experiment For the first experiment the (MLP, KNN, random forest) models were trained with **Generic and Hand crafted features**.

2nd experiment For the second experiment the (MLP, KNN, random forest) models were trained with **Generic and Hand crafted features** and making feature selection with the **LASSO** 9 method.

3rd experiment For the third experiment the (MLP, KNN, random forest) models only **Generic features** were used.

4th experiment For the fourth experiment the (MLP, KNN, random forest) models were trained with only **Generic features** and making feature selection with the **LASSO** 9 method.

Best model from experiments From the four experiments the best model was the 1st experiment for all the models 4.1, and this is strange, because majority of the features didn't have that much information, so removing them would make it easier for the model to train and get better results. In the results part this is going to be referenced again to explain the results of the models.

4.2. Image model experiments

General information The output for the CNN and MLP model was the Stage variable

1st experiment with CNN model For this model the size of the images were reduced to a 512x512x3 size, for faster training and because it wasn't necessary the images in the best quality to get information from them. The inputs for the model were the month when the stage value was taken and the image for that stage value. The model was trained using The Adam optimizer 9 and MSE 9 as the loss for the model

2nd experiment with Segmentation-MLP model For this model the size of the images were reduced to a 320x320x3 size, for faster training and because the backbone for the segmentation model needed this specific size, the dataset used was the Riwa dataset [9] and 7 masks made by hand of the images of the river at hand. The inputs for the segmentation model was the image with data augmentation and the output was a binary mask, and with this binary mask the river width was calculated using the **euclidean distance** 3 from the left shore to the right shore and the river area was calculated summing up all the ones in the binary mask.

$$\text{Euclidean distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3}$$

After training the segmentation model a study was made to see the correlation 6 of the river width and river area with the Stage and Discharge variable. The correlation graph shows that river width and river area has a high correlation with stage and discharge, but the hypothesis is that the correlation could be bigger, but because the model couldn't do a perfect segmentation for images of the river the results weren't perfect

This images show that the correlations weren't perfect thanks to the bad segmentation and the time series graphs for the river width and stage 9 show that river width 9b has many outliers and that the segmentation weren't perfect because the results vary too much, for the model to work perfectly the segmentation results should be precise in cm, but for this images we didn't find a way to make the segmentations better.

For the second model, the MLP 5, the inputs for the model were the month when the stage value was taken and the river width of the image for that stage value. This model was trained with the Adam optimizer 9 and MSE 9 as the loss metric. And the model predicts the stage variable.

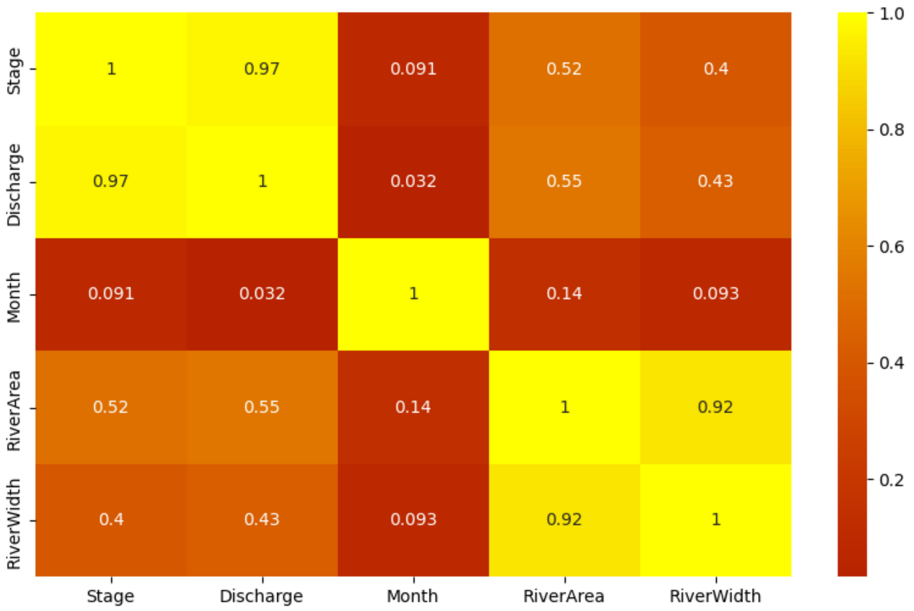
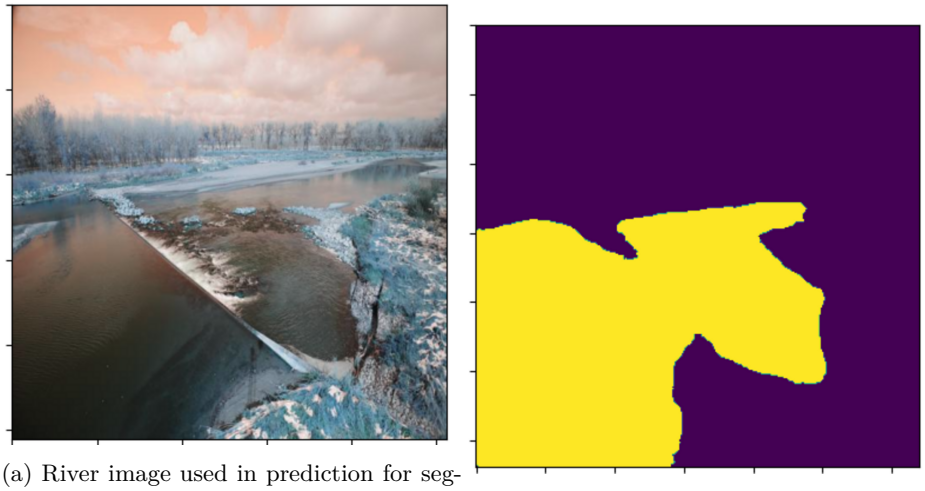


Figura 6: Pearson correlation of month, river width and area with the stage and discharge.

(a) River image used in prediction for segmentation model.

(b) Mask of river for the 1st image.

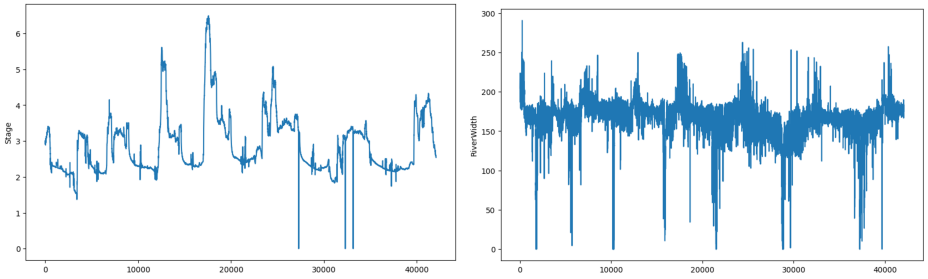
Figura 7: Image with a good segmentation.



(a) River image used in prediction for segmentation model.

(b) Mask of river for the 2nd image.

Figura 8: Image with a bad segmentation.



(a) Time series through the years of Stage. X is all the variables ordered in time, Y axis is the stage variable.

(b) Time series through the years of river width. X is all the variables ordered in time, Y axis is the river width variable.

Figura 9: Time series plots for stage and river width.

5. Results

When comparing all model results, the Convolutional Neural Network consistently gives the best results. Of these metrics, MAE was chosen to determine the effectiveness of the models 5 as it returns the expected forecast error size on average. Alternatively, MAPE was also considered as one of the primary metrics to evaluate models since in some cases the size of the error in MAE can be very large and makes it difficult to interpret the error impact on each model. MAPE scales all the results and converts them into simple to interpret percentages. That said, CNN would be the best model with a MAE of 0.1919 with a MAPE of 0.078. The next closest model to these results was the MLP Regressor with a

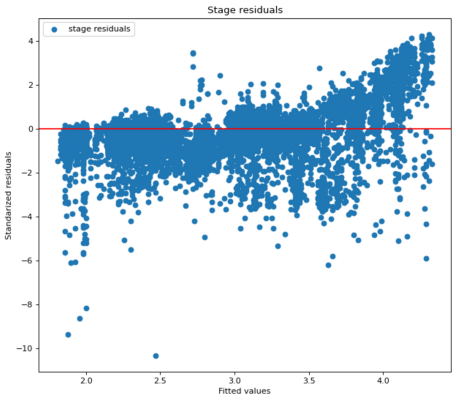
MAE 0.2223 and a MAPE 0.078. From there the next best option is the MLP Regressor with the Segmentation data (0.2790 MAE and 0.0973 MAPE), followed by the KNN Regressor (0.2821 MAE and 0.1064 MAPE) and with Random Forest Regressor at last with the worst results (0.3127 MAE and 0.1214 MAPE).

	CNN	MLP Regressor	MLP Regressor with Segmentation	KNN Regressor	Random Forest Regressor
R^2	0.8043055231	0.6911826503	0.4220925667	0.54566232	0.4919470624
MSE	0.07642631698	0.1206052058	0.2258010654	0.1774365638	0.1984144645
MAE	0.1919919813	0.2223266992	0.2790626533	0.2821863754	0.3127794728
MAPE	0.07221717524	0.0780759087	0.09730067528	0.1064906306	0.1214667585

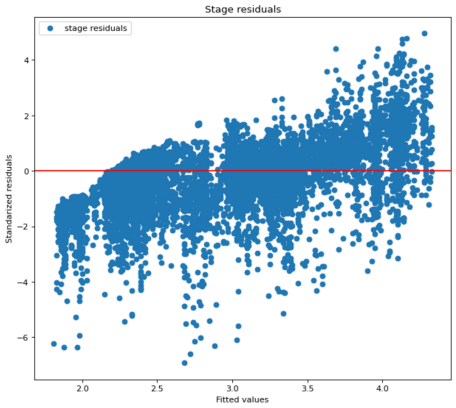
Cuadro 5: Comparison of the models results.

However, after reviewing the results of the model plots 10 11 the data seems to indicate that there is a better candidate for best model. From the top two models, CNN 11c and MLP Regressor 11b, it can be observed that the predictions generated by this two models have a high level of variability, they have a lot of noise, unlike the MLP Regressor with Segmentation 11d which in general forecasts most of its predictions under a threshold closer to the actual stage and discharge values and it does seem the model is trying to find a formula, instead of just predicting the value that would give the lowest error based on what was learn in the pattern of the training dataset that are very similar to the pattern of the testing dataset.

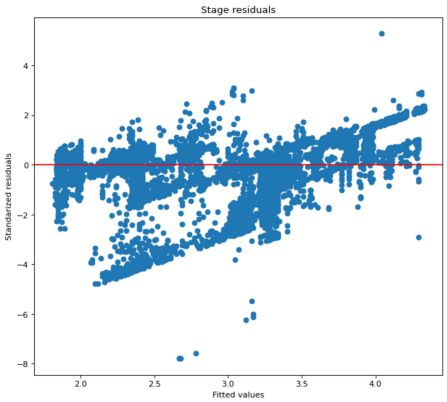
As previously said, the data has a relatively low standard deviation, and because the pattern repeats, it appears that the models were able to get good results due to the number of columns in the csv or the degree of diversity in the photos rather than truly comprehending the problem. Theoretically, the CNN model and MLP regressor model will only be capable of producing precise predictions for this particular river as a result, rather than being able to generalize and work for other rivers. The MLP Regressor with Segmentation, on the other hand, is more likely to be able to interpret general image features in a wider range of situations as a result of the manner in which the data is segmented and supplied to the model. As a result, the data suggests that MLP Regressor with Segmentation is the best model for resolving the current issue, which is predicting the stage of a river.



(a) Stage residuals for MLP regressor model.



(b) Stage residuals for CNN model.



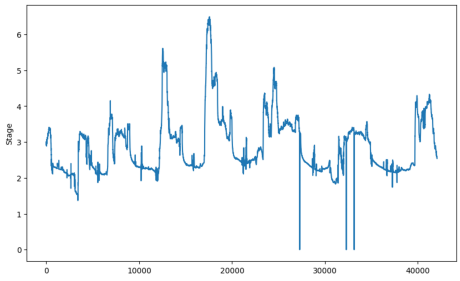
(c) Stage residuals for MLP regressor with segmentation data model.

Figura 10: Prediction residuals from the 3 best models.

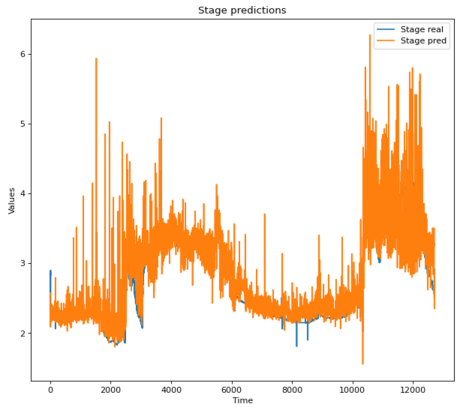
6. Future work

So it is proved in this research that it is possible to train a model that can learn to predict the stage of a river based on the width of the river and the current month. Nonetheless, in order to prove this hypothesis completely, we need a model that can provide superior segmentation for the photos in the dataset at hand; however, there may be other options beyond simply proving the hypothesis.

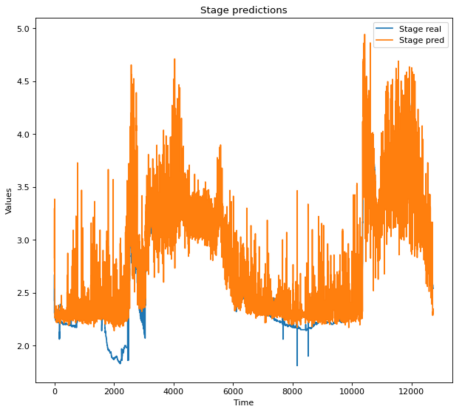
- Create 100 masks from the river dataset images to overfit the segmentation model to that river, allowing for improved segmentation precision merely to prove the hypothesis item



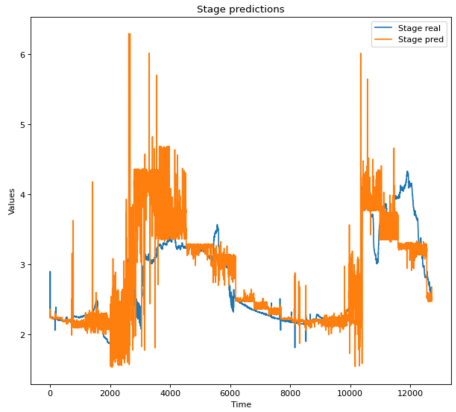
(a) Time series of the ground truth values for stage. X is Time, Y is stage



(b) Time series stage predictions and real values for MLP regressor model.



(c) Time serie stage predictions and real values for CNN model.



(d) Time serie stage predictions and real values for MLP regressor with segmentation data model.

Figura 11: Predictions from the 3 best models.

- Obtain a separate river dataset that includes photos as well as the stage value; having the images taken from a higher position will make it easier for the model to recognize the edges and segment the river, as well as eliminating reflections from the water of things near the river in favor of reflections from the sky, which will make it easier to segment. Additionally, if the images were taken of a flat section of the river rather than one that displayed the weir, the model would be simpler to divide.

If the hypothesis is confirmed in this way, the next step would be to create a *better segmentation model* that can generalize better, achieve higher precision, and work with images of rivers in the winter because segmenting rivers through

ice appears to be more challenging for the model to do. Furthermore, when using this model, the camera should be calibrated to convert pixels to centimeters or inches, which improves precision and simplifies training.

7. Conclusion

At the conclusion of this study, we presented a variety of models for predicting the river stage, and we demonstrate that a segmentation model with an MLP is most likely the most effective model for resolving this issue. By obtaining the width of the river in a picture and the month the image was taken, the Segmentation-MLP model's results demonstrate the potential for the model to be able to predict the stage of a river. But in order to fully support the theory and improve the accuracy of the river's stage, a better segmentation model or an alternative approach to segmenting the river from an image. However, this study demonstrates the prospect of reducing the requirement for sensors to determine a river's stage, and with this research, it is possible to complete any gaps in a river's stage data and receive information from a river in challenging circumstances, such as during a flood.

8. Contributions

- Andres Eduardo Nowak de Anda: Dataset analysis, dataset models (Random Forest Regressor, KNN Regressor), image analysis, image models (CNN, MLP Regressor with Segmentation)
- Isaac Emanuel García González: Dataset analysis, dataset models (MLPRegressor, KNN Regressor), image models (CNN), Report
- Samuel Alejandro Diaz del Guante Ochoa: Dataset analysis, image analysis, image models (CNN, MLP Regressor with Segmentation), Report
- Ernesto Lopez Villareal: Dataset analysis, image analysis, image models (CNN), Report

Bibliografía

[1] Eltner, A., Bressan, P., Akiyama, T., Goncalves, W., Marcato Junior, J. Using Deep Learning for Automatic Water Stage Measurements. *Water Resources Research*. **57**, e2020WR027608 (2021), https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020WR027608

[2] Ramchoun, H., Ghanou, Y., Ettaouil, M., Janati Idrissi, M. Multilayer perceptron: Architecture optimization and training. (International Journal of Interactive Multimedia,2016), https://reunir.unir.net/bitstream/handle/123456789/11569/ijimai20164_1_5_pdf_30533.pdf?sequence=1

[3] Sharma, S. Activation Functions in Neural Networks. (2017,9), https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[4] Yu, Y., Si, X., Hu, C., Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*. **31**, 1235-1270 (2019)

[5] Albawi, S., Mohammed, T., Al-Zawi, S. Understanding of a convolutional neural network. *2017 International Conference On Engineering And Technology (ICET)*. pp. 1-6 (2017)

[6] Mukherjee, S. The Annotated ResNet-50. (2018,8), https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

[7] Yan, Q., Iwasaki, T., Stumpf, A., Belmont, P., Parker, G., Kumar, P. Hydrogeomorphological differentiation between floodplains and terraces. *Earth Surface Processes And Landforms*. **43** (2017,8)

[8] Cai, S., Wu, Y., Chen, G. A Novel Elastomeric UNet for Medical Image Segmentation. *Frontiers In Aging Neuroscience*. **14** (2022), https://www.frontiersin.org/articles/10.3389/fnagi.2022.841297

[9] Blanch, X., Wagner, F. & Eltner, A. RIWA Dataset. (Kaggle,2022), https://www.kaggle.com/dsv/4289421

[10] Borneman, E. How rivers change the landscape. *Geography Realm*. (2014,4), https://www.geographyrealm.com/rivers-change-landscape/

[11] HBR How companies are already using AI. *Harvard Business Review*. (2017), https://hbr.org/2017/04/how-companies-are-already-using-ai

[12] Oracle Que es la inteligencia artificial (IA)?. *Que Es La Inteligencia Artificial (IA)? Oracle Mexico*. (2015), https://www.oracle.com/mx/artificial-intelligence/what-is-ai

9. Glossary

- R^2: Also called coefficient of determination, it is the proportion of total variance of the dependent variable, it reflects the adjustment of a model to

the variable to be explained.

$$R^2 = \frac{\sum_{t=1}^{T}(\hat{Y}_t - \overline{Y})^2}{\sum_{t=1}^{T}(Y_t - \overline{Y})^2}$$

- MSE: Mean Squared Error measures the average of the squared errors, the difference between the estimator and what is estimated.

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- MAE: Mean Absolute Error is used to quantify the accuracy of a prediction technique by comparing the predicted values against the observed ones.

$$\frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

- Stage: water level in a river or stream, units: ft.
- Discharge: volumetric flow of water, units: ft^3/s.
- LASSO: Least absolute shrinkage and selection operator is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

$$Loss = Error(Y - \widehat{Y}) + \lambda\sum_{1}^{n}|w_i|$$

- Adam: is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks.

Stream Discharge and Stage Prediction with Machine Learning

Germán A. Jaramillo Ramírez, Ana C. Munguía Romero, Carlos N. Ojeda Angulo, and Alejandro A. Bojorquez Pineda

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

Resumen Having access to streamgage imagery allows for hydrologic conditions of its location to be visually verified. The University of Nebraska extracted features of the North Platte River State Line Weir from over 40,000 daylight images taken at one-hour intervals from 2012 to 2019.

With this data, Multilayer Perceptron (MLP), Support Vector Regression (SVR) and Random Forest Regression (RFR) models for predicting stream stage and discharge were trained; with RFR yielding the best results.

Following a different approach from the original paper, the preexisting architectures *EfficientNetB0* & *ResNet101* were trained using the 40,000+ images of the river. The results initially suggest that *EffientNetB0* is the better model, but looking more closely at the resulting behaviours shows that *ResNet101*'s performance better resembles the seasonal pattern of the data.

The evaluation of the models presented in this document were done using the same loss metrics (MAE, MSE, RMSE & SE) from the original paper by the University of Nebraska.

And so, this work looks to find out if there are any improvements left to be made over the original models and work methodologies.

Keywords: Machine Learning, CNN, Regression, Stream Discharge, Stream Stage

1. Introduction

A streamgage contains instruments that measure and record the amount of water flowing in the river or stream, or its discharge [1].

Having access to continuous data from streamgages is essential when calibrating and validating ground water and surface water models. Thus, encountering gaps in the data will increase uncertainty in the model predictions.

Cameras can help in this regard, since they provide redundancy with information unavailable from sensors. This imagery allows for hydrologic conditions to be visually verified (e.g., the presence of ice, obstructions, etc.).

In the instance of an actual gap in the data, having those images allows for a qualitative assessment to confirm that the chosen gap filling method accurately represents the actual state of the stream during that certain time period.

With this, the objective of the current work is to fill stream discharge and stage data gaps using machine learning models for a location with camera availability.

The literature seems to support the notion of using cameras as a plausible way of insuring a sort of 'backup' of the data will be available in case of any technical failure of the sensors at the streamgage. Eltner, A. et al. (2021) show in their research article *Using Deep Learning for Automatic Water Stage Measurements* [2] a similar use of deep learning tools as well as photogrammetric techniques to extract information from streamgage images to achieve more reliable water stage measurements. While the objective is most definitely not the same, the study sets a precedent for machine learning and image-based feature extraction applied to water monitoring.

2. Data Description

Features were extracted from over 40,000 daylight images taken at one-hour intervals from 2012 to 2019. Dawn and dusk images that were too dark for feature extraction were removed.

The features were stored in a Comma Separated Value (CSV) file which contains the images' names as ' Filename' and, among many other features, the Stage and Discharge data from each image.

3. Methodology

3.1. Data storage

For a detailed explanation of the evolution and strategy for data management and storage, see *Appendix B. Data Storage*.

3.2. Models generated with numeric data

Data distribution The numeric data available in the CSV file is separated into three different datasets: testing, training and validation data. These datasets were used when implementing the Multilayer Perceptron (MLP), Support Vector Regression (SVR) and Random Forest Regression (RFR) models for each of the dependent variables (stage and discharge).

Recreating the original models The starting point of this work was to attempt to recreate the three models from the original research paper. Thus, the same ten features were used for the training stage: 'grayMean', 'graySigma', 'entropyMean', 'entropySigma', 'hMean', 'hSigma', 'sMean', 'sSigma', 'vMean', 'vSigma'.

Principal Component Analysis (PCA) PCA was used for feature selection and new iterations of the models were trained with these variables.

Preprocessing the data All of the above methods were alternatively trained with re-scaled data, using each of the following scalers: Standard, Robust and MinMax. Strings and non-numeric values have been dropped, except for the date, which was processed to create a new column called 'year' of numeric datatype.

3.3. Training Convolutional Neural Networks

We opted to make use of the already trained architectures *EfficientNetB0* & *ResNet101* available in the *Keras Applications* module. We concluded that it'd be a better use of our time to focus on getting and comparing results rather than building an architecture from scratch, which, in the event of failure, we wouldn't really know if it failed due to errors in the implementation or if it just wasn't the right fit for the project. Thus, we preferred to work with preexisting models that have already been proven to work in order to speed up the process.

Data distribution The images were separated into two datasets: training and testing data. The images of each dataset were selected by years (2015-17 for testing and the remaining years for training), this due to the seasonal behaviour exhibited by the data.

Preprocessing the data The images provided were of really good quality and of really big size, so resizing them was necessary in the code to ensure that the CNNs could receive them.

EfficientNet Convolutional Neural Networks (CNN) tend to have fixed resources; this is quite a limitation since, conventionally, a CNN's depth or width would be increased as much as possible within the resource budget in an attempt to achieve greater accuracy rates. These are tedious and costly processes that may not always yield the desired outcome. EfficientNet proposes a new and more efficient method of scaling that scales all dimensions (depth, width and resolution) uniformly [3].

ResNet The deeper the neural network, the harder it is to train. Due to their depth, they are able to start converging and that causes a degradation problem of training accuracy. This degradation indicates that not all systems are equally easy to optimize. As a way to ease the fitting of networks, the ResNet framework reformulates the layers of the network as learning residual functions with reference to the layer inputs, instead of having to learn unreferenced functions [4].

Scope of the model Both *ResNet101* and *EfficientNetB0* take an incredibly long time to train (6 and 8 hours, respectively), and according to Hafeez, U. U. & Gandhi, A. (2020) [5] execution times for CNN architectures are very costly and take up too many resources, even with the computational power of AWS. So it was decided to just train models to predict stream stage since the stage and discharge features are very highly correlated (making it possible to predict discharge with stage).

3.4. Loss metrics

The loss metrics (MAE, MSE, RMSE & SE) used to evaluate the models were selected from the original paper so that we would have a way to effectively compare the results of this study with those obtained previously by the University of Nebraska.

4. Results

4.1. Regression models

The regression models (MLP, RFR & SVR) trained with the numeric data extracted from the images as provided in the CSV file were overall more reliable predicting stage rather than discharge.

Tables I through VI showcase the results of each iteration of every regression model worked on in this study.

While most models were inaccurate predicting stream discharge, the Random Forest Regression using robust scaling surpassed them by a lot with an R^2 value of 0,831; error metrics, on the other hand, were pretty high all across the board and the RFR model got the lowest values of $MAE = 203{,}346$, $MSE = 232971{,}927$, $RMSE = 482{,}672$, $SE = 483{,}986$ (see *Table II*).

Cuadro 1: Discharge MLP

	MAE	MSE	RMSE	$R\hat{2}$	SE
Normal*	821.795	1383311.929	1176.143	-0.001	1178.819
Normal (+ PCA)*	678.002	1093322.927	1045.621	0.209	1046.412
Standard*	365.059	475835.195	689.808	0.656	691.74
Standard (+ PCA)	940.685	2261749.889	1503.912	-0.637	1179.035
Robust*	362.85	459946.716	678.194	0.667	679.539
Robust (+ PCA)	939.757	2259791.988	1503.26	-0.636	1179.039
MinMax*	438.588	558835.307	747.553	0.595	746.6
MinMax (+ PCA)	940.688	2261749.761	1503.911	-0.637	1179.039

*Normal: Raw data - unprocessed.
*Standard: Standardly scaled data.
*Robust: Robustly scaled data.
*MinMax: MinMax Scaling applied to data.
*(+ PCA): PCA run on the specified data.

Cuadro 2: Discharge RFR

	MAE	MSE	RMSE	$R\hat{2}$	SE
Normal*	203.481	233258.04	482.968	0.831	484.288
Normal (+ PCA)*	540.987	820620.085	905.881	0.406	908.206
Standard*	203.365	233030.754	482.733	0.831	484.048
Standard (+ PCA)	328.002	450925.772	671.51	0.674	673.313
Robust*	203.346	232971.927	482.672	0.831	483.986
Robust (+ PCA)	549.581	912066.012	955.021	0.34	957.622
MinMax*	203.43	233294.115	483.005	0.831	484.329
MinMax (+ PCA)	295.01	403544.252	635.251	0.708	637.103

*Normal: Raw data - unprocessed.
*Standard: Standardly scaled data.
*Robust: Robustly scaled data.
*MinMax: MinMax Scaling applied to data.
*(+ PCA): PCA run on the specified data.

Cuadro 3: Discharge SVR

	MAE	MSE	RMSE	R$\hat{2}$	SE
Normal*	496.4	1069794.83	1034.309	0.226	995.413
Normal (+ PCA)*	631.762	1249238.979	1117.694	0.096	1071.32
Standard*	496.4	1069794.83	1034.309	0.226	995.413
Standard (+ PCA)	489.712	1047283.417	1023.369	0.242	990.208
Robust*	496.4	1069794.83	1034.309	0.226	995.413
Robust (+ PCA)	751.152	1515365.725	1231.002	-0.097	1125.931
MinMax*	496.4	1069794.83	1034.309	0.226	995.413
MinMax (+ PCA)	476.003	999011.14	999.505	0.277	969.637

*Normal: Raw data - unprocessed.
*Standard: Standardly scaled data.
*Robust: Robustly scaled data.
*MinMax: MinMax Scaling applied to data.
*(+ PCA): PCA run on the specified data.

For predicting stream stage specifically, the RFR using standard scaling yielded the best results out of all the regression models, with an R^2 value of 0,874, and overall error metrics measuring lower than 0,3 with $MAE = 0{,}14$, $MSE = 0{,}08$, $RMSE = 0{,}284$, $SE = 0{,}284$ (see *Table V*).

Cuadro 4: Stage MLP

	MAE	MSE	RMSE	$R\hat{2}$	SE
Normal*	0.638	0.639	0.799	-0.001	0.802
Normal (+ PCA)*	0.631	0.773	0.879	-0.212	0.877
Standard*	0.211	0.112	0.335	0.824	0.336
Standard (+ PCA)	0.296	0.227	0.476	0.644	0.478
Robust*	0.271	0.22	0.469	0.655	0.464
Robust (+ PCA)	0.633	0.635	0.797	0.004	0.8
MinMax*	0.237	0.142	0.377	0.777	0.376
MinMax (+ PCA)	0.265	0.196	0.443	0.692	0.445

*Normal: Raw data - unprocessed.
*Standard: Standardly scaled data.
*Robust: Robustly scaled data.
*MinMax: MinMax Scaling applied to data.
*(+ PCA): PCA run on the specified data.

Cuadro 5: Stage RFR

	MAE	MSE	RMSE	$R\hat{2}$	SE
Normal*	0.14	0.08	0.283	0.874	0.284
Normal (+ PCA)*	0.408	0.36	0.6	0.435	0.602
Standard*	0.14	0.08	0.284	0.874	0.284
Standard (+ PCA)	0.234	0.169	0.411	0.735	0.412
Robust*	0.14	0.08	0.284	0.874	0.285
Robust (+ PCA)	0.418	0.407	0.638	0.361	0.64
MinMax*	0.14	0.08	0.283	0.874	0.284
MinMax (+ PCA)	0.21	0.141	0.375	0.779	0.376

*Normal: Raw data - unprocessed.
*Standard: Standardly scaled data.
*Robust: Robustly scaled data.
*MinMax: MinMax Scaling applied to data.
*(+ PCA): PCA run on the specified data.

Cuadro 6: Stage SVR

	MAE	MSE	RMSE	$R\hat{2}$	SE
Normal*	0.172	0.086	0.294	0.864	0.295
Normal (+ PCA)*	0.458	0.461	0.679	0.277	0.677
Standard*	0.172	0.086	0.294	0.864	0.295
Standard (+ PCA)	0.256	0.185	0.43	0.71	0.43
Robust*	0.172	0.086	0.294	0.864	0.295
Robust (+ PCA)	0.596	0.592	0.77	0.071	0.757
MinMax*	0.172	0.086	0.294	0.864	0.295
MinMax (+ PCA)	0.231	0.161	0.402	0.747	0.402

*Normal: Raw data - unprocessed.
*Standard: Standardly scaled data.
*Robust: Robustly scaled data.
*MinMax: MinMax Scaling applied to data.
*(+ PCA): PCA run on the specified data.

4.2. CNN models

The resulting values (see *Table VII*) might make it seem that *EffientNetB0* is the better model, but looking at the graph (*Figure 1*) reveals that *ResNet101*'s performance is much better (*EffientNetB0* has a lot of noise and is really just predicting an average of the samples).

Cuadro 7: Stage using CNN

	MAE	MSE	RMSE
Resnet101	0.821	0.678	0.823
EfficientNetB0	0.338	0.184	0.4290

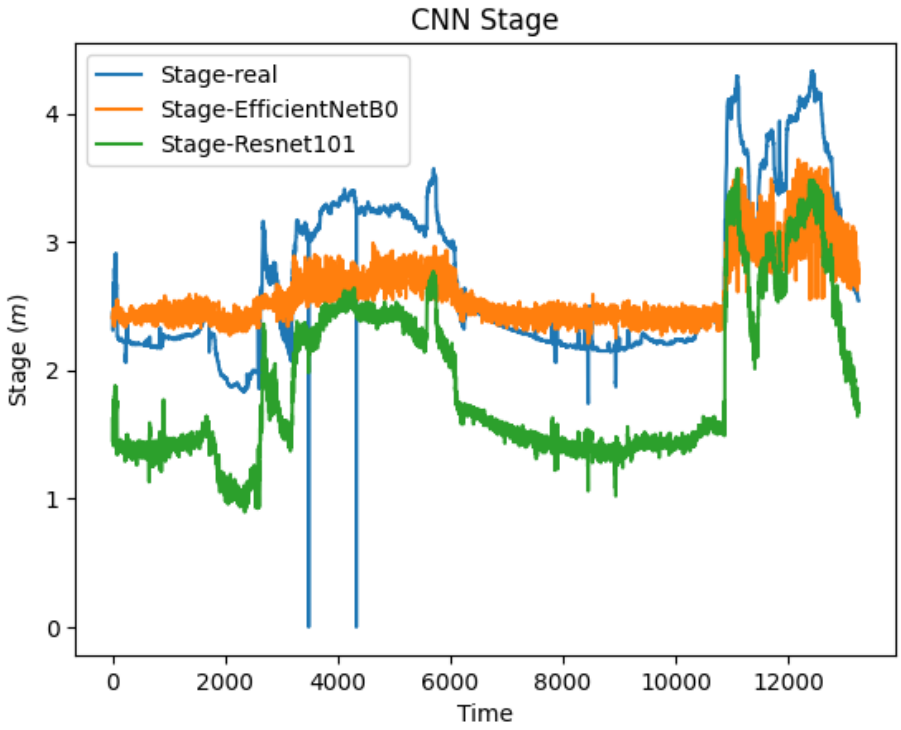


Figura 1: Comparison of models predicting stage

5. Discussion and Conclusions

In this project we proposed two different approaches: the first one using the variables from the CSV file with different scaling techniques and the second one using the 40,000+ images provided by the University of Nebraska to train preexisting CNN architectures. The best results we got were from the Random Forest Regression model using standard scaling (for predicting stage) with a MSE of 0.08 and a MAE of 0.14, which seem to be good results, however, these values are not strictly an indicator of how reliable the model is, as it could be predicting an average of the samples instead of following the seasonal pattern or trend as it should be.

We believe that better results can be obtained if we continue working on CNNs while tweaking several different aspects that were not explored in this project, for example, grouping the data by week of the year, or month, to better take into account the seasonality of the data.

6. Appendices

6.1. Feature description

Header name	**Description**
Stage or Discharge	**Response variables**
grayMean	Grey scale mean of all the pixel intensities after conversion from RGB to grey
graySigma	Grey scale sigma of all the pixel intensities after conversion from RGB to grey
entropyMean	Shannon entropy mean of all the gray scale pixel intensities after conversion from RGB to grey
entropySigma	Shannon entropy sigma of all the gray scale pixel intensities after conversion from RGB to grey
hMean	Mean of all the pixel intensities in the hue channel after conversion from RGB to HSV
hSigma	Sigma of all the pixel intensities in the hue channel after conversion from RGB to HSV
sMean	Mean of all the pixel intensities in the saturation channel after conversion from RGB to HSV
sSigma	Sigma of all the pixel intensities in the saturation channel after conversion from RGB to HSV
vMean	Mean of all the pixel intensities in the value channel after conversion from RGB to HSV
vSigma	Sigma of all the pixel intensities in the value channel after conversion from RGB to HSV
grayMean 0	Gray scale mean of all the pixel intensities in the above weir ROI
graySigma 0	Gray scale sigma of all the pixel intensities in the above weir ROI
entropyMean 0	Shannon entropy mean of all the gray scale pixel intensities in the above weir ROI
entropySigma 0	Shannon entropy sigma of all the gray scale pixel intensities in the above weir ROI
hMean 0	Mean of all the pixel intensities in the hue channel in the above weir ROI
hSigma 0	Sigma of all the pixel intensities in the hue channel in the above weir ROI
sMean 0	Mean of all the pixel intensities in the saturation channel in the above weir ROI
sSigma 0	Sigma of all the pixel intensities in the saturation channel in the above weir ROI
vMean 0	Mean of all the pixel intensities in the value channel in the above weir ROI
vSigma 0	Sigma of all the pixel intensities in the value channel in the above weir ROI

Header name	**Description**
sMean 1	Mean of all the pixel intensities in the saturation channel in the below weir ROI
sSigma 1	Sigma of all the pixel intensities in the saturation channel in the below weir ROI
vMean 1	Mean of all the pixel intensities in the value channel in the below weir ROI
vSigma 1	Sigma of all the pixel intensities in the value channel in the below weir ROI
WeirAngle	Angle of the weir
WeirPt1X	Far end of weir calculation end point X pixel position
WeirPt1Y	Far end of weir calculation end point Y pixel position
WeirPt2X	Near end of weir calculation end point X pixel position
WeirPt2Y	Near end of weir calculation end point Y pixel position
WwRaw LineMin	Minimum pixel distance from weir line to whitewater raw down stream edge
WwRaw LineMax	Maximum pixel distance from weir line to whitewater raw down stream edge
WwRaw LineMean	Mean of pixel distances from weir line to whitewater raw down stream edge
WwRaw LineSigma	Sigma of pixel distances from weir line to whitewater raw down stream edge
WwCurve LineMin	Minimum pixel distance from weir line to whitewater curve fit down stream edge
WwCurve LineMax	Maximum pixel distance from weir line to whitewater curve fit down stream edge
WwCurve LineMean	Mean of pixel distances from weir line to whitewater curve fit down stream edge
WwCurve LineSigma	Sigma of pixel distances from weir line to whitewater curve fit down stream edge

Bibliografía

[1] Utah Water Science Center What is a streamgage?. (2018), https://www.usgs.gov/centers/utah-water-science-center/science/what-streamgage

[2] Eltner, A., Bressan, P. O., Akiyama, T., Gonçalves, W. N., & Marcato Junior, J. Using Deep Learning for Automatic Water Stage Measurements. *Water Resources Research*. **3** (2021)

[3] Mingxing Tan, Quoc V. Le EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. (2020)

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition. (2015)

[5] Hafeez, U. & Gandhi, A. Empirical Analysis and Modeling of Compute Times of CNN Operations on AWS Cloud. *2020 IEEE International Symposium On Workload Characterization (IISWC)*. pp. 181-192 (2020)

Stream flow and Discharge prediction with Machine Learning & AI

Edgar Daniel Acosta Rosales and José Herón Samperio León

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

Resumen El uso de librerías para la implementación de inteligencia artificial y machine learning a incrementado en los últimos años, siendo estas herramientas capaces de facilitar procesos financieros, marketing, seguridad o áreas de la salud. En esta ocasión el tema a tratar es la predicción de valores de descarga y altura de los ríos cuando los sensores de medición fallan. El caso de estudio en especifico esta ubicado en Henry, Nebraska. El desarrollo de este proyecto se divide en dos entregas. La primera es la replicación de los datos del socio formador en un archivo CSV. La segunda es la predicción de valores por medio de imágenes obtenidas por el centro geológico de estados unidos. Los datos abarcan mas de 60,000 imágenes y 40,000 datos capturados por los sensores.

Keywords: Machine Learning · Predicciones · USGS

1. Antecedentes e introducción

El análisis del flujo y corriente de cuerpos de agua es tema de gran interés para la planeación de construcciones civiles o predicción de inundaciones.
Si bien existen métodos directos e indirectos para su calculo, como estaciones de aforo, que suelen implicar un costo elevado para países en vías de desarrollo; Disolución de trazadores, método que involucra materiales químicos, fluorescentes o radioactivos y métodos matemáticos de aproximación, ninguno de los procedimientos previos contextualiza lo que ocurre en condiciones hidrológicas, como lo puede ser la parecencia de hielo, obstrucciones del flujo o cambios importantes en la forma del caudal.

El problema que se nos presenta es justificado por dos preguntas *¿Que sucede cuando los sensores fallan? ¿Cómo se puede compensar la falta de información?*

Es por ello que empleamos Inteligencia artificial y Machine Learning para la aproximación del flujo y corriente por medio de imágenes y procedimientos de calculo que permiten obtener y complementar los vacíos en registros obtenidos por medios convencionales, además de una mayor cobertura en áreas poco accesibles y minimizar costos de implementación y recursos humanos.

Los datos recopilados se dividen en 2 *datasets*. El primero, un archivo *Comma Separated Value* (CSV) con metadata de las imágenes e información recopilada por sensores y un archivo con mas de 60,000 imágenes obtenidas al sureste de Henry, Nebraska, que datan del 9 de junio del 2012 hasta el 11 de octubre del 2019. Seleccionadas en intervalos de una hora para predecir la máxima cantidad de datos.

2. Análisis exploratorio de los datos

Los datos que nos fueron provistos por el socio formador constan de:

- `2012_2019_PlatteRiverWeir.csv` | Datos obtenidos del United States Geological Survey (USGS) capturados cada 15 minutos y datos adicionales calculados por el socio formador.
- `Imágenes` | 60,000+ imágenes obtenidas de una cámara en la frontera entre Nebraska y Wayoming.

La información recopilada en el csv contiene datos como timestamps, stage, discharge y metadata de las imagenes, además de *"Handcrafted features"*, datos que han sido calculados por el socio formador en C++ utilizando la librería OpenCV, obteniendo características escalares de las imágenes.

Para la primera parte de este proyecto, optamos visualizar de una manera más gráfica la correlación entre las variables, por medio de una matriz de Pearson, para así identificar cuáles de las variables son más relevantes en la replicación de nuestros modelos de predicción. Encontramos que las columnas:

- `Filename`
- `isoSpeed`
- `shutterSpeed`
- `Agency`
- `SiteNumber`
- `width`
- `height`
- `TimeZone`
- `CalcTimestamp`
- `exposure`
- `fNumber`

No proporcionan información relevante que beneficie al modelo y tampoco muestran una correlación con los datos restantes.

Procediendo con el análisis de variables,se emplearon las variables que representan la mayor correlación que se muestra en (Fig.1.):

- `entropyMean`
- `entropyMean 0`
- `entropyMean 1`
- `entropySigma`
- `entropySigma 0`
- `sSigma`
- `sMean`
- `hMean 1`
- `hSigma 1`
- `WeirAngle`

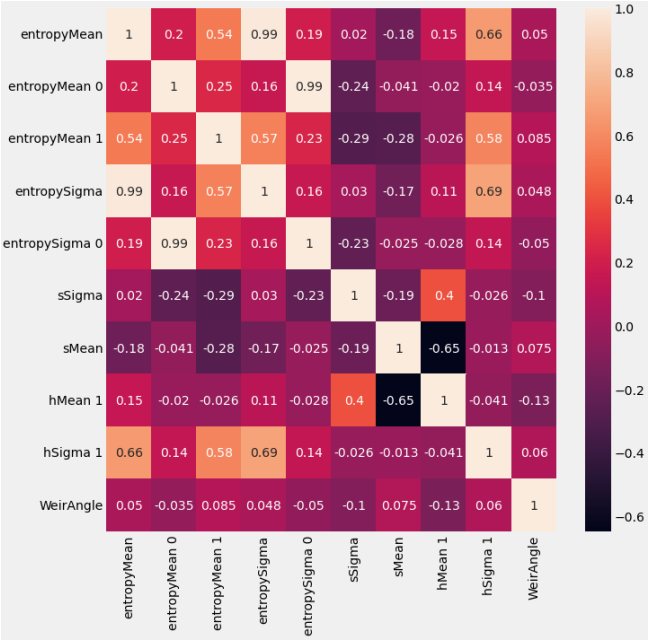


Figura 1: Correlación de Pearson de los datos de 2012_2019_PlatteRiver-Weir.csv

La condición de aceptación para las imágenes(Fig.2.) definido por le socio formador sucede si una imagen es demasiado obscura, se opta por removerla, pero si la lente de la cámara se ve bloqueada sea por neblina, condensación u otros factores, se le asigna un estado de çould not calculate". Los nombres de fichero de las imágenes fueron incluidos en el archivo CSV, donde posterior a su selección, restaron un total de 40,000+ imágenes por procesar.

El uso del filtro de Kalman forma parte del preprocesado de la información, este paso permitió limpiar la imagen de ruido que no es visible para nosotros

pero si en el análisis de la imagen.

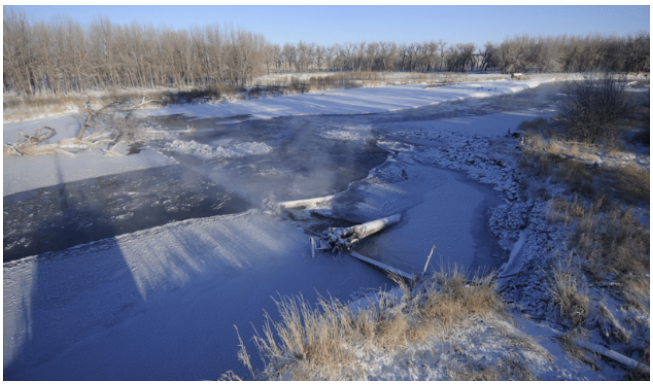


Figura 2: Henry, Nebraska (USGS, 2019)

3. Métodos de implementación - CSV data

Una vez identificamos las variables que se van a utilizar, comenzamos con el split del dataset, a diferencia de la proporción 30,70 (30 % training y 70 % test) nosotros optamos por 80,20 (80 % training y 20 % test) ya que no se sobre-entrenan los modelos de Machine Learning.

Uno de los métodos mas frecuentes para optimizar modelos supervisados es buscar la cantidad de "hidden-layers"que minimice la función de perdida con la menor cantidad de procesamiento. Realizado mediante "GridSearchCV"que consiste en ajustar los datos con diferentes parámetros y encontrar los que den mejor puntuación dentro del rango dado. La gráfica de los resultados (Fig.3.) nos permite entender como el mean-score (y) varia en función de la cantidad de hidden layers (x).

3.1. Modelos

Los modelos seleccionados para la predicción de valores con datos proporcionados en el archivo CSV fueron implementados con el motivo de recrear los valores obtenidos por el socio formador. Además de refinar los hiper-parámetros de cada modelo y obtener una predicción similar o aceptable.

- `Multi-layer Perceptron Regressor` (Deitel & Deitel, 2019; Pedregosa 2018)
- `Support Vector Regressor` (Deitel & Deitel, 2019)
- `Random Forest Regressor` (A & M, 2002)

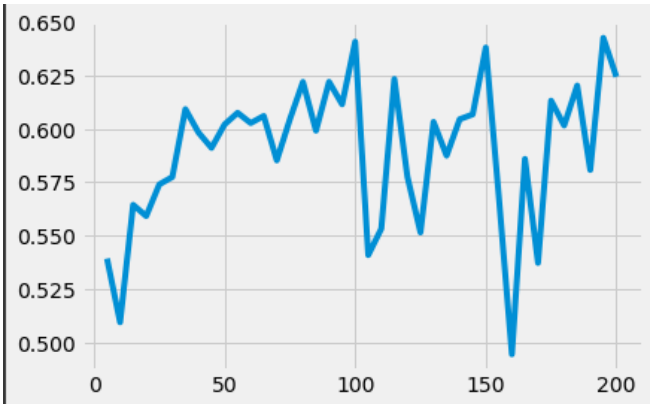


Figura 3: Gráfico de función de perdida comparada con la cantidad de hidden-layers

La implementación de cada función regresora se realizó mediante el uso de la librería Scikit-learn (Pedregosa et al., 2018).

3.2. Resultados

Train	Test	ML Model	Score
Stage			
80 %	20 %	MLP-Regressor	0.65
80 %	20 %	RFR	0.82
80 %	20 %	SVR	0.53
Discharge			
80 %	20 %	MLP-Regressor	0.63
80 %	20 %	RFR	0.78
80 %	20 %	SVR	0.20

Cuadro 1: Resultados de predicción "Stagez "DischargeÇSV

En ambos casos el Random Forest Regressor demostró dar los mejores resultados para estos casos en particular, y el SVR el peor de todos, congeniando con los resultados del socio formador en cuanto a que modelo es mejor.

4. Métodos de implementación - Análisis de imágenes

El dataset de imágenes empleadas para la predicción del *Stage* fueron divididas en la proporción 80,20 (80 % training y 20 % test) y 90,10 (90 % training y 10 % test) para la predicción de los valores de *Discharge*.

4.1. Preprocesado

Para ahorrar tiempo de procesamiento se realizaron algunos pasos previos a alimentar los modelos empleados para la predicción de variables.
El primer paso fue reducir el tamaño de las imágenes utilizando la librería PILLOW (Sweigart, 2019) para después reescalar el tamaño de las imágenes a 100x100px con una escala en blanco y negro (Fig.4.). Posterior se convirtió cada una en un arreglo unidimensional en un archivo NPY y con ello unificar todo el dataset en 3GB de información para nuestros modelos.

Figura 4: Imagen reescalada 100x100px Nebraska (USGS, 2019)

4.2. Modelos

Fueron dos los modelos implementados, específicamente para las variables que se desean predecir.

Para el modelo desarrollado encargado de la predicción de los valores *Discharge* se empleó la función de activación 'ReLu'(Fig.5.), una de las funciones mas populares en cuanto a desarrollo de redes neuronales convoluciones (Praharsha, 2022), en las capas ocultas del modelo ya que permite su implementación en

cálculos no lineales. Este modelo fue entrenado con 5 épocas y un *callback* que guarda los pesos de cada generación en un archivo temporal *hdf5*.

El segundo modelo para la predicción del *Stage* fue implementado de la misma manera, con la funcion de activación 'ReLu', una menor cantidad de hidden layers y 15 épocas por batch.

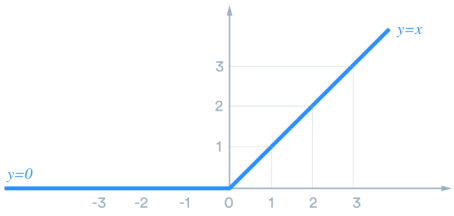


Figura 5: Representación gráfica ReLu

4.3. Resultados

Stage			mse
80 %	20 %	CNN-Regressor	0.36
Discharge			
90 %	10 %	CNN-Regressor	13,027

Cuadro 2: Resultados de predicción "Stagez "Discharge"

Para validar los resultados de los CNN se usó "mean square error"para determinar que tanto se alejan las predicciones de loa valores reales, se obtuvieron dos resultados muy extremos, para el caso del Stage se logró un MSE de 0.36 que es un error casi despreciables para el rango en el que se manejan los datos, sin embargo para el caso de Discharge el resultado da a entender que no predice acertadamente los datos por lo que es un modelo descartable y no se aconseja su uso.

5. Discusión

Los resultados en las dos entregas de este proyecto de predicción de variables, fueron aceptables, a excepción del calculo de valores *Discharge* (Table.2). Esto se debe a la variabilidad en la información recopilada por el USGS.
Algunos de los trabajos a futuro para mejorar los resultados predictores son:

- Detección de valores atípicos en el dataset - CSV
- Pasos extra en el preprocesado de imágenes que consta de:
  - Detección y recorte de los bordes del río para eliminar el ruido generado por la flora alrededor
  - Eliminación de reflejos en el agua mediante el uso de la técnica "Mean color gradientz obtener colores mas uniformes que no distorsionen la perspectiva.
- Dividir ambos datasets en series de tiempo.

6. Conclusión

El desarrollo de técnicas de Machine Learning tiene múltiples beneficios no solo para la predicción de valores y detección de datos atípicos. Estos ayudan a países en vías de desarrollo a la investigación y planeación de futuras construcciones, formas en caudales y posiblemente la implementación para el estudio de la calidad del agua, cambios climáticos, cambios en la fauna y flora que rodea el caudal además de tener una capacidad muy superior al momento de obtener métricas en lugares poco accesibles.

Disponibilidad del código
El código generado y sus requisitos puede ser encontrado en:

- https://github.com/DonKatsun/Inteligencia-artificial-avanzada-para-la-ciencia-de-datos

Bibliografía

[1] Pedregosa, F. et al. (2018) Scikit-Learn: Machine learning in Python, https://scikit-learn.org/stable/about.html. Available at: https://scikit-learn.org/stable/about.html (Accessed: November 20, 2022).

[2] Deitel, P.J. and Deitel, H.M. (2019) Python for programmers: With introductory AI case studies. Boston: Pearson.

[3] A, L. and M, W. (2002) Randomforest. Available at: https://cran.r-project.org/web/packages/randomForest/citation.html (Accessed: December 22, 2022).

[4] USGS 06674500 NPRSLW data: 06674500 NORTH PLATTE RIVER AT WYOMING-NEBRASKA STATE LINE Stage and Discharge data. Available at: https://nwis.waterdata.usgs.gov (Accessed: 29-November-2022, 2022).

[5] Hotz, N. (2022) What is CRISP DM?, Data Science Process Alliance. Available at: https://www.datascience-pm.com/crisp-dm-2/ (Accessed: December 5, 2022).

[6] Sweigart, A. (2019) Automate the boring stuff with python: Practical programming for total beginners. San Francisco, California: No Starch Press.

[7] Praharsha, V. (2022) Relu (rectified linear unit) activation function, OpenGenus IQ: Computing Expertise; Legacy. OpenGenus IQ: Computing Expertise; Legacy. Available at: https://iq.opengenus.org/relu-activation/ (Accessed: December 5, 2022).

River stage gauge by deep learning image recognition

Gerardo Villegas Contreras, Paola Naomi García Reyes, Renata Uribe Sánchez, and Ivan Emmanuel Gutiérrez Y López

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

Resumen The focus of this study is the image recognition with deep learning methods that combine feature extraction and convolutional neural networks in order to predict the river stage in different seasons of the year. The inherent advantage of having an algorithm capable of predicting the river stage lies in the circumstances where the gauging sensors could fail, so these gaps are filled with the imagery information. Therefore, an initial machine learning approach was implemented by applying a statistical analysis over the extracted features data set in an attempt to understand the yearly behavior of the stage measurements. In order to address this issue, two convolutional neural networks (CNN) were developed: a VGG16 and a second one accomplished by our research group. In both approaches, the images were previously filtered with a CNN classification algorithm to identify if the picture was suitable for the model prediction. As an additional study, there was also incorporated an hydrology expert chat-bot to get familiarized with some water terms relevant within the study and a user interface.

Keywords: Machine learning · Image processing · Deep Learning · CNN · Hydrology · River Stage · Resnet model · VGG model

1. Introduction

The era of accelerationism in which we live, leads to the inevitable scarce of resources, specifically water reserves, which can result in various negative scenarios for humanity such as famine, cold wars, economic crises and water shortages. As this persistent financial crisis continues to progress, the governments adopt an austerity policy, privatization of public services, an unemployment wave but also an unstoppable capitalism production and consumerism. [5]

Hence the importance of managing and monitoring water bodies that provide resources to populations, that, if applied correctly, could prevent both floods and future water crises. In addition to these important factors, the proper management of water bodies is capable of generating intelligent supply systems and thus avoiding water shortages, a phenomenon that is already a reality in various parts of the world.

That is the main reason on going deep into hydrology phenomena and get familiar with some water related terms such as river flows, floods and measure

techniques to analyze the main characteristics of a river: its stage and discharge, the main subjects of this study. Consequently, one of the most significant interests to hydrologists is to gauge the amount of water within a stream because of what it represents: the pure end-product of all the previous processes in a hydrological cycle. [2]

However, it is key to start defining these two main concepts of hydrology which in fact are strongly related. Based on the stream elemental concept as a body of flowing water always in a natural channel, its primary characteristic, the stream discharge or stream flow is defined as the volume of water that runs through a specific location at a specific time, normally expressed in cubic meters per second. Even though the discharge could be measured continuously, it is rather calculated as an averaged flow over a time period on a particular river. On the other hand, the stream stage refers to the water level above some arbitrary point in a stream. [3]

As mentioned previously, these two concepts are highly related, and it is the usefulness of the so-called rating curve where multiple discharge gauges can lead to interlocking both characteristics. In addition, the inherent advantage of this rating curve is the allowance of continuous stream stage measurements that can lead to the future discovery of the real discharge. Thus, the stage-discharge relationship is obtained by making a series of velocity and area evaluations at a given location while recording the level with a stilling well. [2]

It is through deep learning algorithms and some concepts from an hydrology domain expertise that this study provides an approach focused on water level prediction based on image recognition and subsequent feature extraction of a stream.

The University of Nebraska has conducted an initial case study and it is our intention to continue its development(see *Camera-based Water Stage and Discharge Prediction with Machine Learning. Hydrology and Earth System Sciences Discussions.* [1]) In the initial case study, the authors used the same imagery dataset from the North Platte River over a six-year period. Feature extraction was implemented in two different ways: by computational algorithms and manually by hydrology experts. Image features extracted by the algorithms include pixel intensity, image entropy, saturation, etc. In the other hand, the domain expertise (hydrology) features were hand-crafted recognized by and hydrologist: and they were streams characteristics such as shape and texture of white foam, and the color of the water above and below the weir.

2. Methodology

Before addressing solutions for predicting the river stage based on image recognition, a work plan was implemented following most of the important aspects of the CRISP-DM methodology as a standard process for data mining. This workflow is exemplified in Fig. 1 [4]. In addition to following up on previously extracted features (see *Camera-based Water Stage and Discharge Prediction with Machine Learning. Hydrology and Earth System Sciences Discussions*

[1]), image processing was implemented from scratch to extract new features from the images in a different approach. Subsequently, deep learning models were implemented through convolutional networks after constructing a supervised classification algorithm to identify the most relevant images that are supposed to be usable, without fog and snow and removing the darker ones because of the dawn and dusk instants. For all image and data processing we used a *Colab Notebook* with an AWS instance called *AWS Deep Learning Notebook*. This will permit a faster analysis as there is a larger capacity in infrastructure.

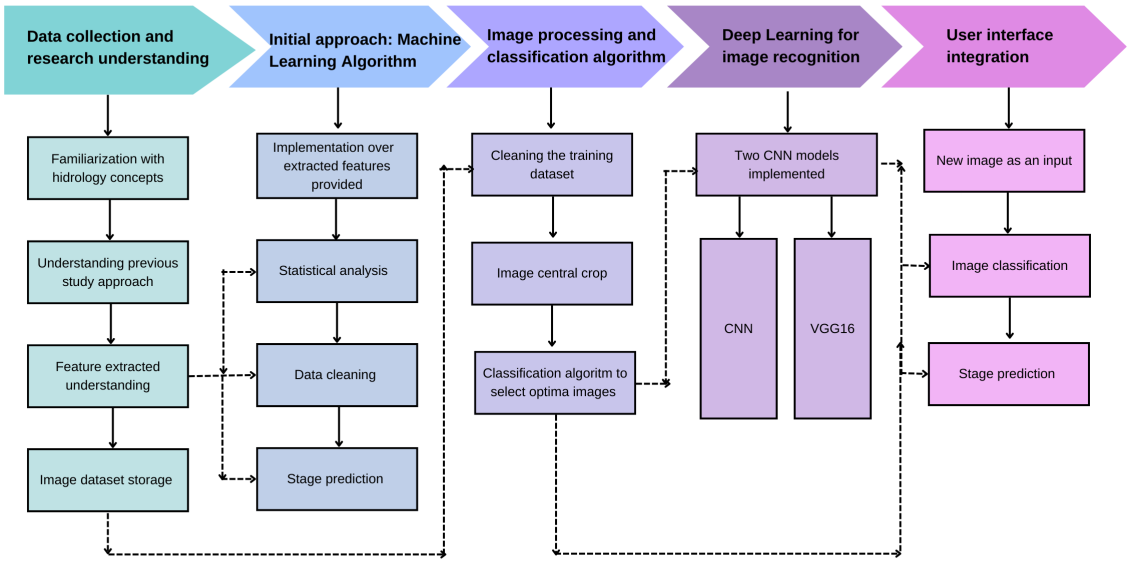


Figura 1: Study Workflow

2.1. Data collection and research understanding

As mentioned above, the data set used in this research was gathered by the University of Nebraska as a previous study approach of the river discharge and

stage. Thus, the primary study extracted features from the images, counting 55 variables including the dependent variables (stage and discharge) and predictors, such as exposure, gray scale features, width and height.For more detailed variable description see Appendix section, Table 7. Although there are also embraced some hand-crafted features created by the hydrologist experts like the measurement of the discharge-stage relationship as the rating curve, the shape and texture of the white water and the colour of the water above and below the weir. For a more detailed variable description see Appendix section, Table 7.

The whole raw imagery data was taken by a single angle corresponding to the North Platte River over a six year period (2012, 2013, 2016, 2017, 2018, 2019) through different seasons and hours, including unclear images (darker ones) from dusk/dawn moments and fog/snow (extremely white images). A sample of a suitable and clear imagery is shown in Fig. 2.

Figura 2: North Platte River imagery sample

2.2. Initial Approach: Machine Learning Algorithm

In order to start a machine learning approach to predict both stage and discharge, given the extracted features data set, there was a statistical analysis performed initially to filter the variables with more relevance within the model. After computing a matrix correlation heat map it was confirmed the expected strong relationship between the river stage and discharge. The resulting correlation coefficient correspond to 0.97, therefore, the decision to focus only on the

prediction of the river stage was made. Due to the stage-discharge correlation, all the analysis and models presented below focuses exclusively on the river stage.

As a machine learning model, an implementation with the whole data set was computed in order to identify the insignificant variables. After applying a standard scaler, the predictors that presented a p-value lower than the significance level of 0.05 and the empty columns were removed. At the end of this procedure there were 50 remaining predictors and there were deleted approximately 2000 outliers through an influence technique for each data set observation as well. As a standard statistic practice, there was applied a deeper data cleaning by identifying and subsequently removing the higher points on leverage, cook distance, betas and DFFIT. Since there was no significant cook distance, betas and DFFIT values, only outliers and higher leverage points were removed. At the same time, the preceding models implemented by the University of Nebraska were replicated to analyse their results for ourselves.

Within the statistical analysis, a calendar visualization was made in order to graphically represent the annual fluctuations of the stage in this particular stream, being its minimum representation 2.87 m during winter and the higher point 6.49 m corresponding to summer season. The calendar is shown in Fig. 3, each square represents a day where the rows are the respective years of the study. It is clearly visible an increase of the stage levels during summer, mainly represented by the months from may to september with minimum variations. This tool was usefully for understanding the seasonality, the missing data (see purple squares in Fig. 3) and perhaps the increasing global warming.

2.3. Image Preprocessing and classification algorithm

As there is interest in the exclusive use of images where enough information can be extracted in order to predict the desirable measurements, the entirety of the data samples can't be used as there are images where there's not enough light to distinguish any shape (Fig. 4a) or images that were too blurry due to weather conditions (dirt, snow, fog, rain, etc.) (Fig. 4b). Taking this into account, and given the vast amount of images, an efficient way to distinguish between useful and not useful images needed to be found. Due to the fact that the given feature extracted data set had already been cleaned by the initial study, it was concluded that a classification algorithm could be implemented to identify the optimal images, the clear ones, without blurry and dark pixels.

Besides, we explored an image pre-processing to focus mainly on the area of interest in the image, where the discharge is located, ignoring external elements such as trees and the sky. In order to accomplish this, we applied a central crop to the whole imagery data set (see a crop example in Fig. 5), a pixel reduction to 150 x 150 and a class balancing to equally train the model.

Therefore, we defined the training data set of the algorithm as the preceding extracted features observations, considering them all as optimal images and detecting those that had not been included in the cleanup to be considered unsuitable for the model. Some of the parameters considered in this section include binary cross entropy as the loss function, the *Adam* optimizer and a total

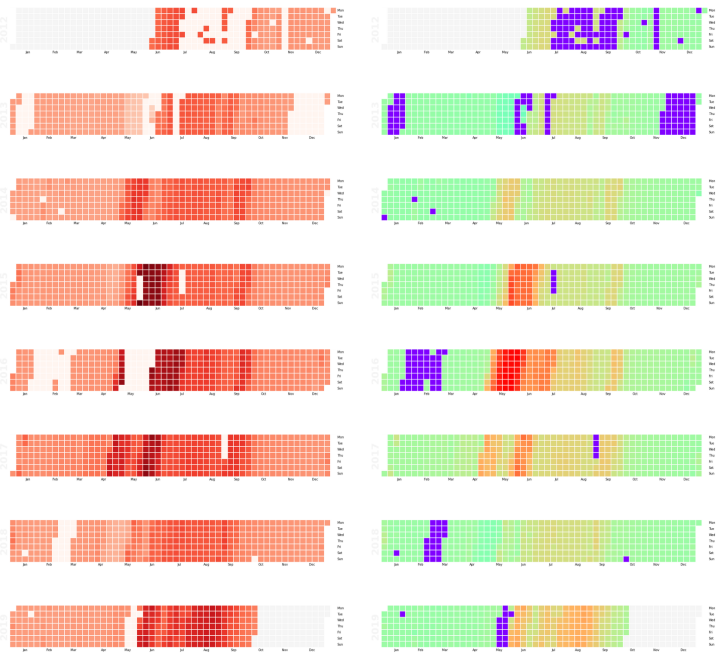


Figura 3: Stage fluctuations calendar from 2012 to 2019 and calendar missing data

of 10 epochs for training. Our CNN (Convolutional Neural Network) implemented returns 1 if the image is suitable to predict stage and 0 if it is not. This algorithm methodology is explained in the Fig. 6

2.4. Deep Learning for image recognition

In this work, we choose three different CNN algorithms to be implemented: the Resnet, VGG16 and a third model that was structured by ourselves.

As a CNN that contains 16 convolutional layers, VGG16 is a suitable model for the current work involving image feature extraction by deep image analysis just like the Resnet for the deep feedback capability even with a large number of layers. [6]

Firstly, the three models were implemented without prior image processing where only 1140 images were used for training. Subsequently, an attempt was made to implement the three models but the Resnet failed due to computational power issues, however the other two models performed very well, perhaps due to the 26,918 images used for training.

In Fig. 7 and Fig.8 it is briefly presented as a diagram, the architecture of the CNN implemented by our research group and the VGG16, both models

(a) Dark Image (b) Blurry Image

Figura 4: Images where useful information can't be extracted

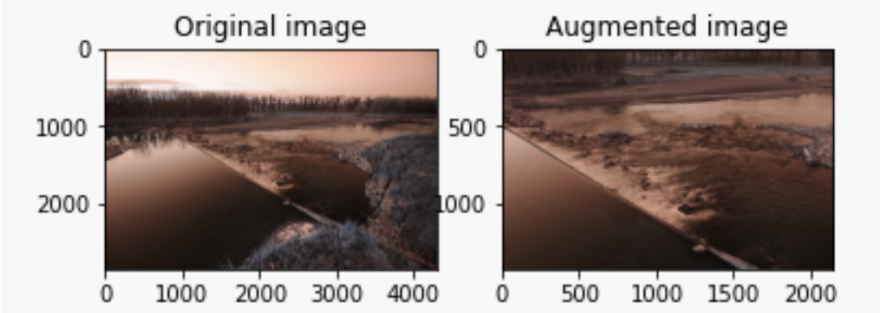


Figura 5: Centralized crop explained

developed after image pre-processing. It is important to mention that the models were trained with 30 epochs using the MSE as loss function and R^2 as the evaluation metric but an early stopping was applied to reduce resources when the model was no longer improving, so both algorithms stopped at epoch 20.

2.5. User interface integration

In an attempt to facilitate the use of our algorithm, i.e. generate stage rate predictions from new images, we conducted a user interface through the streamlit framework for ML and data science to easily create web apps open-source app framework for Machine Learning and Data Science teams. Within the web platform the user most upload an image that is previously filtered by the classification algorithm and then the stage is predicted. If there is an unsuitable picture for the model, a message is displayed with the option of introducing the desirable date if known. On the other hand, if the image is optimal, then it is introduced in the CNN for the stage prediction. On the screen there are three different stage levels: the real one, and the ones calculated with both models (VGG16 and our developed model). (This methodology is shown in Fig. 9)

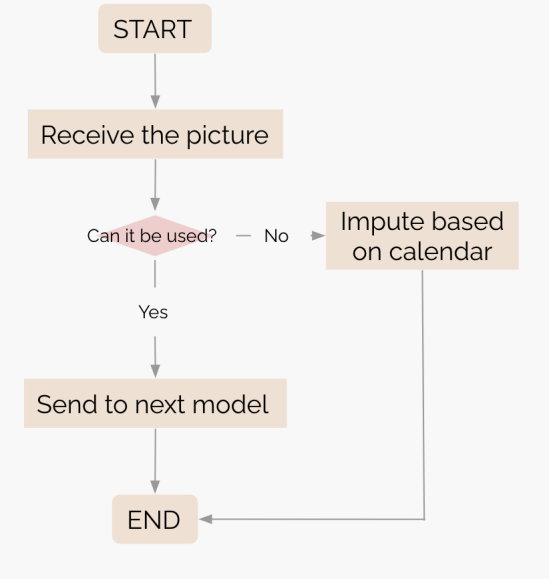


Figura 6: Classification algorithm methodology

2.6. Water chatbot

Starting from a premise of hydrological content in text form (the excerpt was extracted from the book Fundamentals of Hydrology, 2008)[2], the following natural language processing methodology (see Fig. 10) has been followed including a first tokenization, text cleaning (removing punctuation and , frequency diagrams as word clouds and the subsequent implementation of a transformer: RoBERTA to give answers to possible questions raised in relation to the topic.

3. Results

3.1. Initial Approach Machine Learning Algorithms

As a further comparison experiment, we replicate the algorithms (with no data cleaning included) in the previous study of the University of Nebraska [1]to better understand the problem and try to improve the results. In Table 1 we show the corresponding coefficient of determination of the tree regression models to predict stage: MLP(Multi Layer Perceptron), SVM(Support Vector Machine) and RFR(Random Forest Regression). As can be seen, the MLP and RFR models show a very poor performance since the R^2 value is beneath zero. Regarding to the SVM model, it is shown a better performance reaching a 0.637 coefficient but it is not the optimal.

As mentioned earlier, the intuitively strong correlation between stage and discharge led us to predict exclusively a stage measurement. A preliminary model was computed with no data cleaning resulting in a 0.665 R^2.

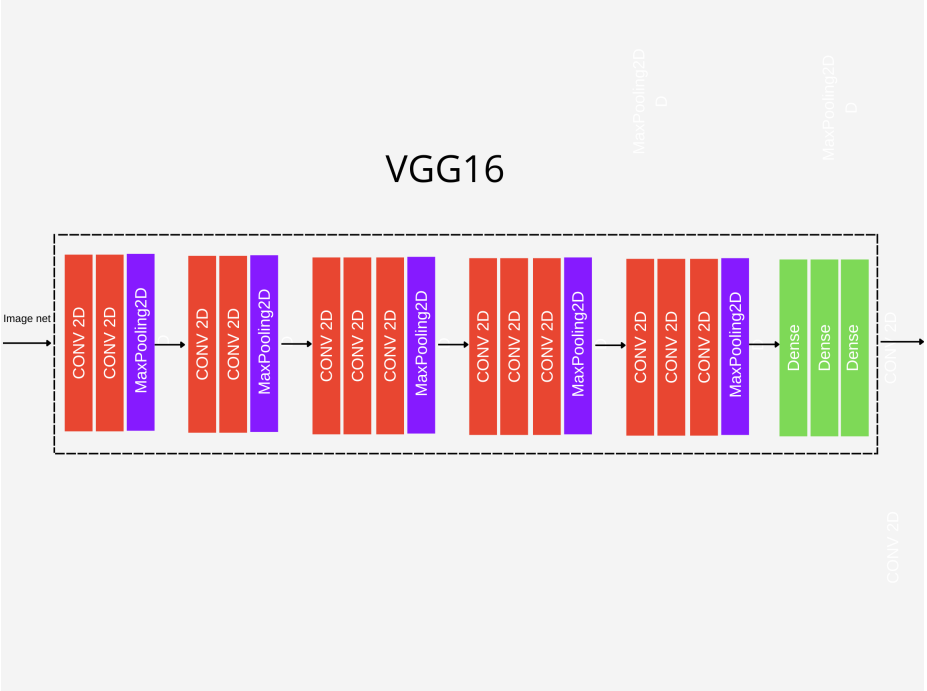


Figura 7: VGG16 architecture

University of Nebraska stage predictio	
Model	R^2
Multi Layer Perceptron	-1169.12
Support Vector Machine	0.637
Random Forest Regression	-0.0068

Cuadro 1: University of Nebraska initial study R^2 stage prediction

After the data cleaning (removal of 2,000 records of outliers, columns filled with zeros and predictors statistically non significant) there were implemented two models: the simplest linear regression and a MLP. A high leverage points was attached to the data cleaning slightly improving the R^2 of the regression model from 78.4 to 78.6, as shown in Table 2

As can be shown in Tables 1 and 2, there was a notable improvement between the coefficient of determination of the initial algorithms in contrast to the ones we implemented due to data cleaning.

However, two different models could be considered for each season, since in summer there are enough higher flood records.

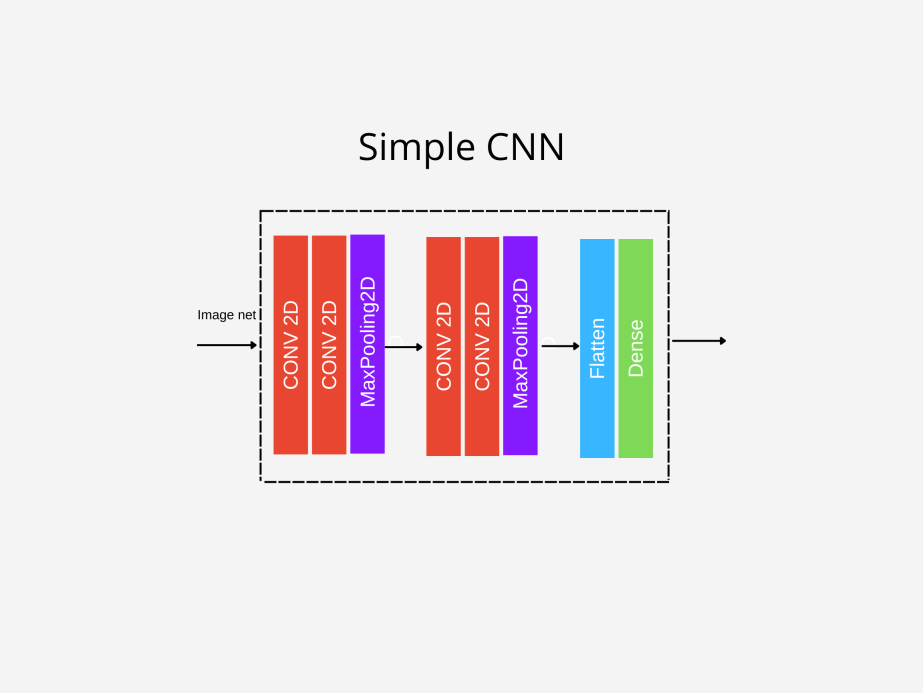


Figura 8: Simple CNN architecture

Machine learning models	
Model	R^2
LR without outliers	0.784
MLP without outliers	0.918
LR without outliers and high leverage points	0.784
MLP without outliers and high leverage points	0.928

Cuadro 2: Machine Learning R^2 results after data cleaning

3.2. Image classification algorithm

A deep learning convolutional neural network was selected as a filtering step to process images before entering in the stage prediction algorithm. The model is focused in recovering clear images, without extremely white and dark pixels. Due to the outstanding good performance of this classifier after only 10 epochs (see Table 3 and Fig. 11) nearly reaching a 100 % on accuracy (98 %) and an extremely low value of the loss function as binary cross entropy (0.004) we did not hesitate on implementing it over the main prediction algorithm.

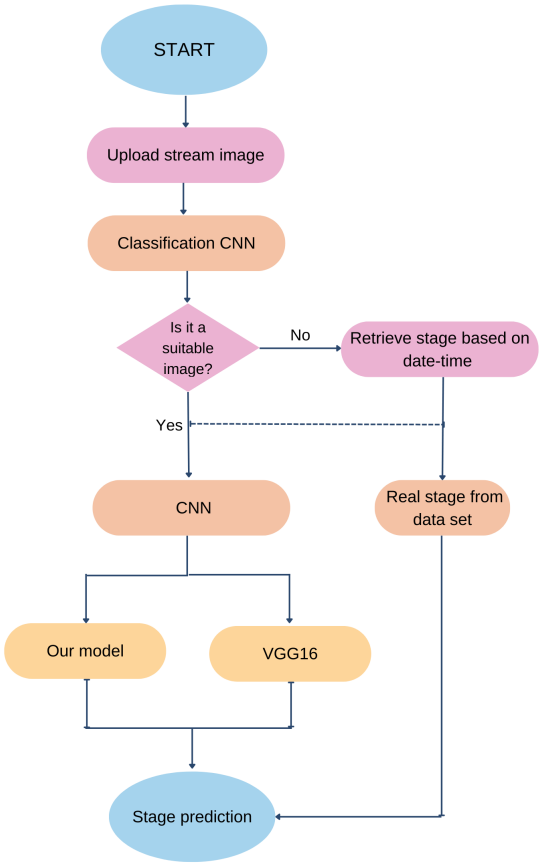


Figura 9: User interface flow

CNN classifier results	
Accuracy	Loss Function value
0.98	0.004

Cuadro 3: CNN classifier accuracy and loss function

As part of the results, the confusion matrix related to the classification model is also included (see Fig. 12), where the high performance of the algorithm can be analyzed.

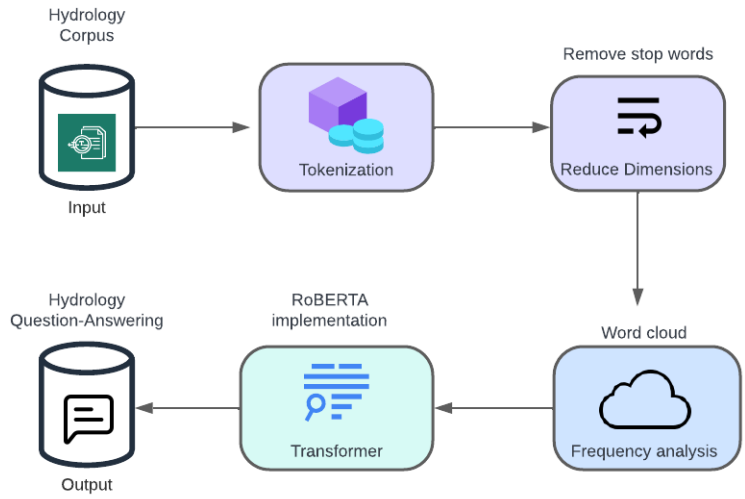


Figura 10: Chatbot implementation methodology

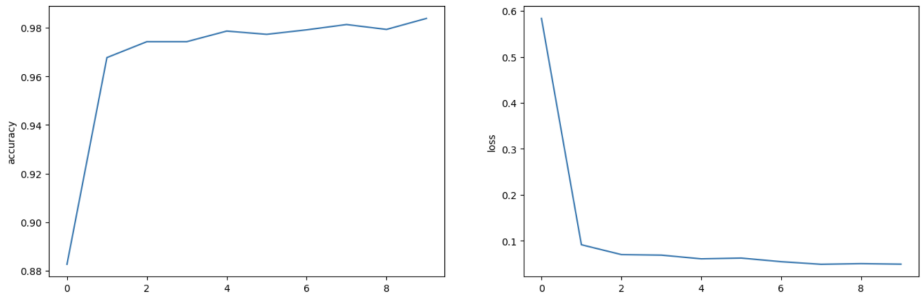


Figura 11: Accuracy and loss function curves for classification model

3.3. Deep Learning and Convolutional Neural Networks

In a first attempt to implement the convolutional neural network, even without the processed images, we developed the following models: simple VGG16, Resnet and a third model created by our own layers. The results of this initial approach on CNN are shown on Table 4. As can be seen, the metrics are very poor, being the best the simple CNN model.

Due to the poor performance of the models, the images had to be pre-processed. Afterwards, the results of both CNNs can be seen on Table 5 in terms of R^2 and MSE, also including the learning curves of the loss function and the accuracy of both models on Fig. 13.

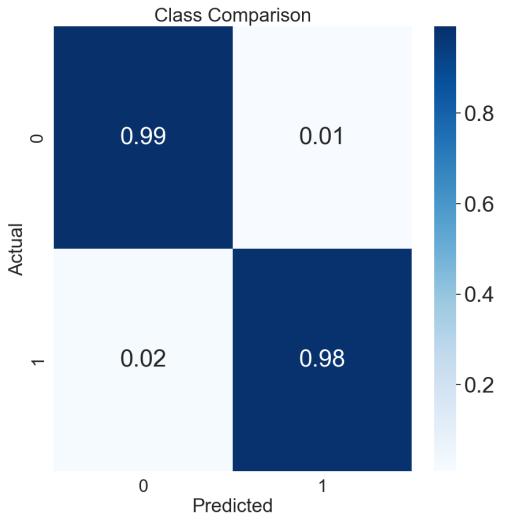


Figura 12: Confusion matrix for classification model

CNN R^2 and MSE before image processing			
	Simple VGG16	Simple CNN	Resnet
R^2	0.370	0.456	0.325
MSE	0.5006	0.3167	0.4481

Cuadro 4: CNN model results before image processing

CNN R^2 and MSE after image processing		
	Simple VGG16	Simple CNN
R^2	0.985	0.975
MSE	0.0079	0.0141

Cuadro 5: CNN model results after image processing

Taking into account the incorporation of both algorithms (classification and stage prediction) the best achieved combination that reaches more than 0.98 for both algorithms is, of course, with image pre-processing where classification reaches 0.98 of accuracy and stage prediction 0.985 R^2 using the best convolutional neural network for this study: VGG16.

The Fig. 14 shows the contrast between the stage prediction of the VGG16 model and the real value, highlighting the good performance of the study.

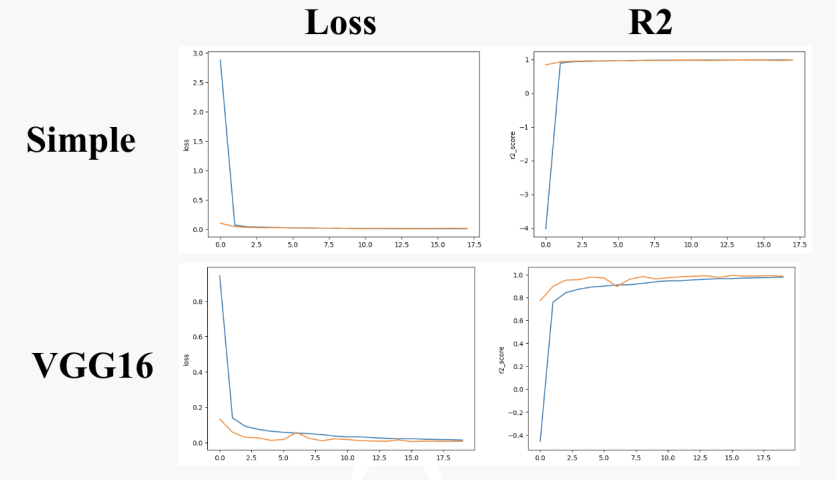


Figura 13: CNN loss function and accuracy curves

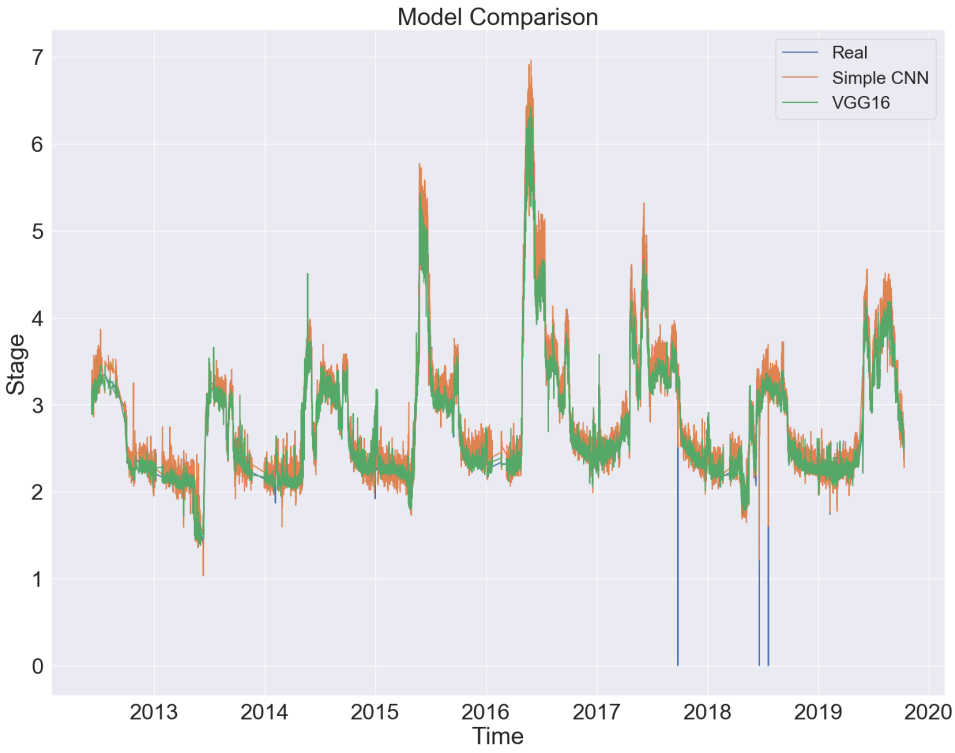


Figura 14: Real stage vs Simple CNN and VGG16 prediction

4. Conclusions

In terms of the stage-discharge relationship, one of the most notable difficulties is that the frequent measurement leads to many records of low and medium weir, that is normally, throughout winter and other average weather periods unlike the few data recovery on potential summer floods. This scarce of stream overflow records is generally because of the infrequency of the phenomena and the danger of measuring the weir discharge during the flood.

Although, the model we developed can be used only on the specific river that it was trained for (North Platte River) the pipeline of this study can be replicated in other streams or water bodies. However, the filtering classification model could be used on other rivers, to show the image utility on a later CNN application to predict the stage. This work had shown a significant enhancement against the previous study, reaching suitable results of the coefficient of determination (R^2) and implementing additional resources as the hydrology chatbot and the user interface to facilitate the application of the CNN models.

5. Further steps: a chatbot hydrology expert and the user interface

5.1. Chatbot

As mentioned above, our models might be useful only for this specific stream, but a further research could implement pre-trained new models based on this work.

On the other side, as a complement to the study, a chatbot expert in hydrology was implemented, which is to be incorporated into the user interface to answer questions about hydrology when there are no experts to interpret results.

Before the corpus analysis a word cloud was computed (Fig. 15), derived from its frequency of occurrence in the text, the chatbot is now an expert in hydrologic terms such as discharge and stage.

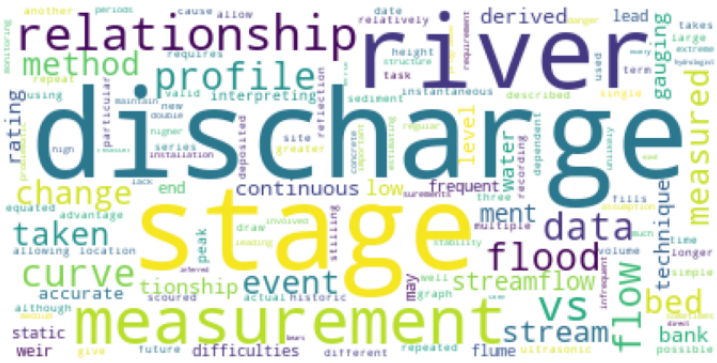


Figura 15: Word cloud derived from the training corpus

After feeding the transformer with the text corpus mentioned above, it was asked a few questions to evaluate its performance. (See chatbot example questions in Fig.16 where the blue square represents the chatbot and the gray one an average user)

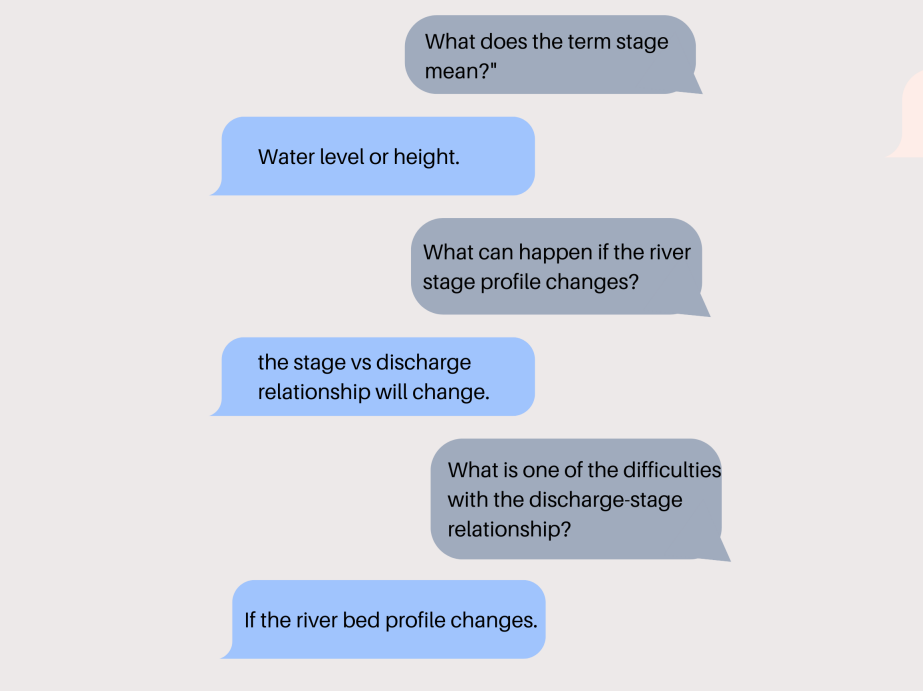


Figura 16: Chatbot conversation example

Although it is true that the transformer did not always get it right, his answers, even if there were redundant and brief, managed to make sense within the context. Through this simple implementation it is feasible to identify the different possibilities and applications that this model could mean for the familiarization of hydrological concepts within and outside the purposes of this study.

5.2. User interface

This section explains in detail, with screenshots, how the interface works.

After browsing a river image from the local files (see Fig. 17), if it is appropriate to calculate the stage, the following appears on display (Fig. 18) showing three different numbers: the real stage, and the predicted with our model and the VGG16.

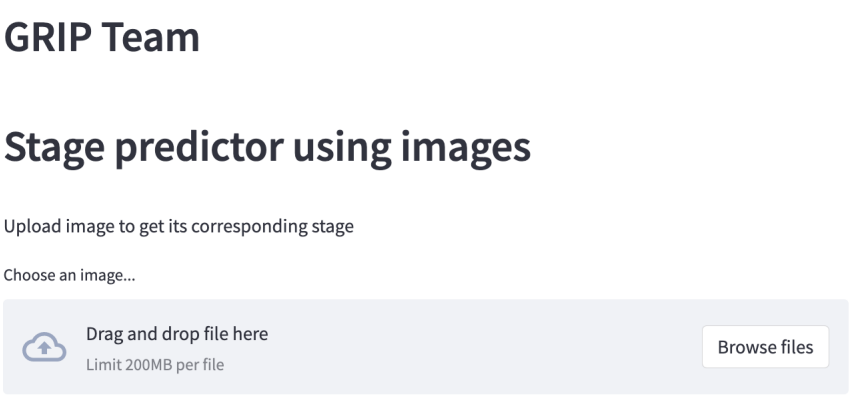


Figura 17: Initial browse from interface (input)

If the selected image is dark or blurry, then the following message is displayed with the possibility, if known, of selecting the desirable stage measurement date-time (shown on Fig. 19). After selecting this information, the exact, or the closest date-time stage measurement is returned as in Fig. 20.

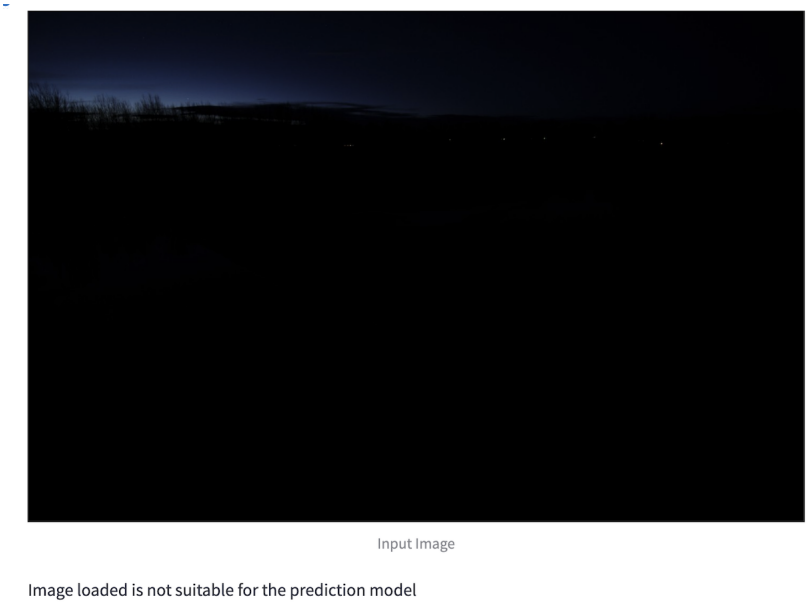


Figura 19: Dark image failing example

Sample Image

Predicted Stage:

Real: `3.0`

Our model: `3.02`

VGG: `3.08`

Figura 18: Optimal image selection and stage prediction

Please select the date of the Stage to load

2016/05/25

Select the time for that date

02:00

Select

The closest Stage to your date is: `6.35` which corresponds to: `2016-05-25 04:00:00`

Figura 20: Stage prediction based on the closest date-time introduced

6. Contributions

In this section, the team member contributions are briefly detailed:

- **Gerardo Villegas Contreras** : Calendar, interface and model development.
- **Renata Uribe Sánchez**: Chatbot, paper writing, data processing.
- **Ivan Emmanuel Gutiérrez Y López**: Model adjustment, development and data pre-processing.
- **Paola Naomi García Reyes** : Model development, pipeline, cloud adjustment.

7. Appendix

7.1. Resource code

Github repository, including all the model implementations is available at `https://github.com/PaolaGarci/IA-pt2.git`

7.2. Interface

The user interface demo can be found on `https://grip-stage-predictor.streamlit.app/`

7.3. Feature extracted variables

Cuadro 6: Feature extracted variables

Variable	**Description**
SensorTime	Sensor time and date
CaptureTime	Capture time and data
Filename	File name, corresponding to time and date
Agency	Capturing agency
SiteNumber	ID of the site where the capture was made
TimeZone	Three letters, time zone
Stage	Water level of the stream, measured in meters
Discharge	Volume of water that runs through a specific location at a specific time, normally expressed in cubic meters per second
CalcTimestamp	Capture date/time
width	nNumber of pixels across the width of the capture
height	Number of pixels across the height of the captures
exposure	Amount of light in the image
fNumber	Focal ratio of the lens
isoSpeed	Camera's sensibility to light
shutterSpeed	Time the lens was open
grayMean	Average of pixel intensities after conversion to grayscale
graySigma	Sum of pixel intensities after converting them to grayscale
entropyMean	Average Shannon entropy of pixels in grayscale
entropySigma	Shannon entropy sum of the grayscale pixels
hMean	Hue average taking each pixel in HSV format
hSigma	Hue summation taking each pixel in HSV format
sMean	Saturation average taking each pixel in HSV format
sSigma	Saturation summation taking each pixel in HSV format
vMean	Value averaging taking each pixel in HSV format
vSigma	Sum of value by taking each pixel in HSV format
grayMean 0	Average intensity of the pixels in grayscale, above the waterfall
graySigma 0	Sum of intensity of grayscale pixels above the waterfall
entropyMean 0	Average entropy of grayscale pixels after waterfall
entropySigma 0	Entropy sum of grayscale pixels after water fall off
hMean 0	Average hue taking each pixel in HSV format, after water fall-off
hSigma 0	Hue summation taking each pixel in HSV format, after the waterfall
sMean 0	Saturation average taking each pixel in HSV format, after water fall off
sSigma 0	Saturation summation taking each pixel in HSV format, after waterfall
vMean 0	Average value taking each pixel in HSV format, after waterfall
vSigma 0	Sum of value taking each pixel in HSV format, after the waterfall
grayMean 0	Average intensity of pixels in grayscale, before water drop
graySigma 0	Sum of intensity of pixels in gray scale, before water drop
entropyMean 0	Average entropy of grayscale pixels, before water fall off
entropySigma 0	Entropy sum of grayscale pixels, before water fall-off
hMean 0	Average hue taking each pixel in HSV format, before water fall-off
hSigma 0	Hue summation taking each pixel in HSV format, before water fall-off
sMean 0	Average saturation taking each pixel in HSV format, before waterfall
sSigma 0	Saturation summation taking each pixel in HSV format, before water drop

Cuadro 7: Feature extracted variables

Variable	**Description**
vMean 0	Average value taking each pixel in HSV format, before the waterfall
vSigma 0	Sum of value taking each pixel in HSV format, before the waterfall
WeirAngle	Angle where the waterfall is located
WeirPt1X	Far point for waterfall calculation, X axis
WeirPt1Y	Far point for water fall calculation, Y-axis
WeirPt2X	Near point for water fall calculation, X-axis
WeirPt2Y	Near point for water fall calculation, Y-axis
WwRawLineMin	Minimum distance between water fall and white water
WwRawLineMax	Maximum distance between water fall and white water
WwRawLineMean	Average distance between the water fall and the white water
WwRawLineSigma	Sum of distances between water fall and white water
WwCurveLineMin	Minimum distance between the water fall and the white water curve
WwCurveLineMax	Maximum distance between the water fall and the white water curve
WwCurveLineMean	Average distance between water fall and white water curve
WwCurveLineSigma	Sum of distances between water fall and white water curve
WwCurveLineMean	Sum of distances between water fall and white water curve

Bibliografía

[1] Chapman, K. W., Gilmore, T. E., Chapman, C. D., Mehrubeoglu, M., Mittelstet, A. R. (2020). Camera-based Water Stage and Discharge Prediction with Machine Learning. Hydrology and Earth System Sciences Discussions, 1-28.

[2] Davie, Tim. (2008). Fundamentals of Hydrology. Abingdon, Oxon: Routledge.

[3] Water Science School. (2018). Dictionary of Water Terms. USGS, science for a changing world

[4] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. (2000). CRISP-DM 1.0 Step-by-step data mining guide. The CRISP-DM consortium .

[5] edited Robin Mackay and Armen Avanessian. (2014). Accelerate : the accelerationist reader. Falmouth, United Kingdom : Berlin :Urbanomic Media Ltd. ; in association with Merve

[6] Xue, Minglong & Shivakumara, Palaiahnakote & Wu, Xuerong & Lu, Tong & Pal, Umapada & Blumenstein, Michael & Lopresti, Daniel. (2020). Deep invariant texture features for water image classification. SN Applied Sciences. 2. 10.1007/s42452-020-03882-w.

Uso de Imagenes para la Prediccion de Flujo y Descarga de Agua en Rios

Javier Lizarraga Beyles, Natalia Velasco Garcia, Jose Luis Rosa Cruz, Cesar Ivann Llamas Macias, and David Alejandro Velázquez Valdez

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

1. Introducción

Este trabajo se realizó en colaboración con la Universidad de Nebraska Lincoln, buscando una mejora en el uso de sensores convencionales los cuales miden el nivel y descarga de agua en ríos. Esto es debido a su gran costo, complicada instalación, operación y mantenimiento, lo cual ocasiona inconsistencia o desconfianza en los datos. Se busca la posibilidad de resolverlo con la implementación de modelos predictivos mediante el uso de cámaras para la interpretación de imágenes con inteligencia artificial y ciencia de datos. Se busca utilizar imágenes como una fuente por su fácil acceso económico, además de no necesitar mantenimiento extensivo ni experiencia especializada para su instalación como es el caso de sensores utilizados actualmente. El enfoque es resolver la falta e inconsistencia de datos en los sensores mediante errores. Esto se lograría con una coexistencia de ambas fuentes, necesarias para la medición constante del agua en ríos, lo cual funcionaria para estudios en campos de hidrología en la predicciones de desbordes de los cuerpos de agua estudiados, la construcción de puentes o infraestructura cercana, sistemas de agua y reservas dependientes del suplemento que brindan estos cuerpos de agua, los cuales dependen de un flujo constante y completo de datos para una precisión necesaria para tales trabajos. Todo el trabajo que se presenta se basa en el artículo (Chapman et al., 2022) brindado por los miembros de la universidad de Nebraska hacia nosotros, con nuestro objetivo siendo mejorar el trabajo ya hecho con la búsqueda de alternativas o mejores resultados, validando los recursos usados en esta investigación para encontrar un nuevo valor en estos datos como posible área de oportunidad para un mayor eficiencia en futuras iteraciones. La hipótesis principal es que al dividir los datos por temporadas se podrá entrenar los modelos para sus temporadas seleccionadas con mayor precisión.

2. Descripción de datos

Los datos utilizados son una base de datos obtenida mediante el procesamiento de miles de fotografías tomadas durante intervalos de 1 hora, posicionadas desde el mismo punto fijado en una presa con leves movimientos entre 4^o en rotación y 0.25^o en traslación. Esto causando un ligero ruido al intentar tomar

medidas de las imágenes debido a que fueron tomadas para un documental del área. A su vez se cuenta con medidas de sensores que nos ayudan a medir el nivel de agua y su flujo en intervalos de 15 minutos, contando con datos recopilados desde 2012 a 2019 de ambas fuentes. Estos datos fueron obtenidos del sitio North Platte River State Line Weir localizado entre Wyoming y Nebraska, siendo altamente afectado por el derretimiento de nieve en las cabezas de las montañas, seleccionado por su gran número de imágenes(57,544), la alta resolución que estas poseen(4288x2848 RGB) y su proximidad a la sección de control de rio de la USGS (estación USGS 06674500 NPRSLW, 2020) de donde se obtuvieron los datos de los sensores (State Line Gauge Weir | PBT, 2022). La comparación funciona con calcular la presa que se encuentra en las fotografías (Figure 1), identificando el área por encima de la presa, el área de la espuma blanca que causa el agua al caer y así obtener características que se pueden comparar con los valores de los sensores. A su vez midiendo el área del agua a comparación de la orilla para compararlo con el nivel del agua.

Lo primero que podemos observar es que las muestras clasificadas como agresivas poseen más muestras con valores extremos o irregulares, aunque no pertenecen únicamente a este grupo, también podemos observar que la variable del tiempo no podrá ser utilizada como una variable independiente o una variable que nos ayude a predecir por sí sola un resultado, pues las muestras fueron tomadas en segmentos ya clasificados por uno tras otro, por lo que no aportan mucho valor al modelo, dando la posibilidad de que nuestro modelo sea erróneo al momento de generar conclusiones (e.g. pudiera concluirse que en la mañana todos conducir normal, en medio día empiezan a conducir agresivamente para terminar el día conduciendo lento) en cambio lo podremos utilizar para comparar cambios bruscos de aceleración y dirección.

Estos datos ya fueron anteriormente limpiados por nuestros colaboradores en la universidad de Nebraska, removiendo datos faltantes y acoplando los datos de los sensores con sus respectivas fotografías, a su vez se eliminaron las imágenes que no presentaban una vista clara de la presa, impidiendo así la interpretación computacional del nivel y flujo del agua, esto sucedió por las siguientes razones: nieve, hielo o escombros en el sitio, humedad o hielo en el lente de la cámara, nivel de agua elevado, escombros en partes de la presa, imágenes muy oscuras para su correcto procesamiento.

3. Metodologías

Los modelos generados fueron para estimar el nivel y flujo del agua, esperando predecir estas variables en casos futuros sin necesidad de las variables específicas del ambiente, solo enfocándonos en la captura de imágenes para una predicción certera. Con este aclarado podemos decir que los datos con los que tratamos fueron obtenidos por nuestros colaboradores de la universidad de Nebraska, donde se encargaron de procesar las imágenes para obtener los datos con los que se trabajarán.

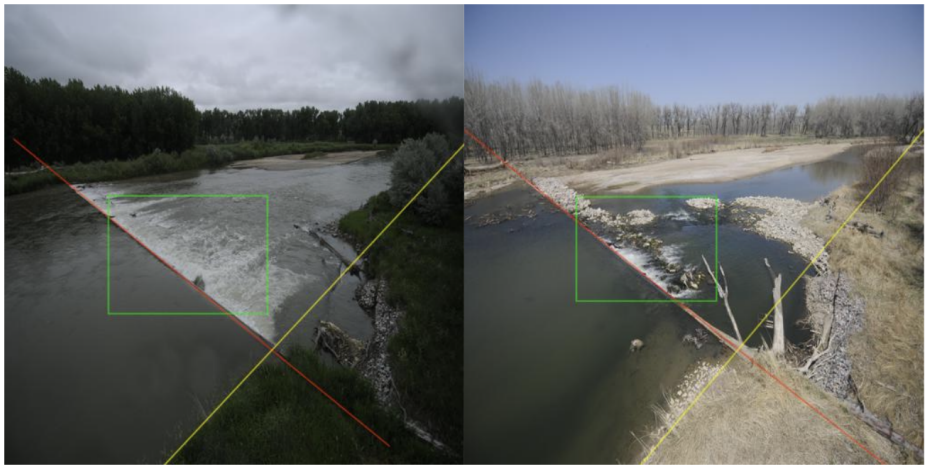


Figura 1: Comparación de dos imágenes del conjunto que se utilizaran en distintas etapas, la línea roja representando la presa, el recuadro verde marcando el área de descarga y la línea amarilla marcando la orilla.

El primer paso fue recrear los resultados obtenidos por el equipo de Nebraska. Estos se replicaron utilizando los datos que obtuvieron mediante el procesamiento de imágenes y aplicando los modelos seleccionados, la selección de estos modelos será explicada en la siguiente sección. El motivo de este paso fue para comenzar en el mismo punto en el que se concluyó el trabajo pasado, verificando que los datos sean correctos y la situación se pudo replicar exitosamente. Al replicar los resultados se optó por implementar modelos similares de regresión, debido a que el objetivo de estos modelos se enfoca en la predicción de variables, la selección de los modelos será especificada y justificada en la siguiente sección, en este caso el nivel del agua y su flujo. Nuestro enfoque fue encontrar modelos que nos brindaran resultados distintos a los ya obtenidos, buscando variación en los resultados utilizando la precisión como métrica, estos son vistos con mayor detalle en la sección de modelos.

Una alternativa que se buscó fue la especialización de los modelos a temporadas, dividiendo los datos en verano, otoño, invierno y primavera según las fechas que tienen los datos con los que contamos. Gracias a esto se creó un modelo para cada temporada por su cambio tan drástico en el río entre cada una de las estaciones, al mismo tiempo interpretaremos las imágenes fijando el área de donde se enfoca la información obtenida de la base de datos mediante el software de procesamiento de imágenes creado por nuestros compañeros de Nebraska. Este modelo se preparó no generando nuevos datos sino analizando la imagen para relacionarlo con los valores de los sensores, de esta forma utilizaremos otra fuente de datos nueva, no la información interpretada sino la imagen en sí. Por último se optó por probar una serie de tiempo para la predicción de datos faltantes, por la naturaleza de los datos la cuales cuentan con tiempos específicos.

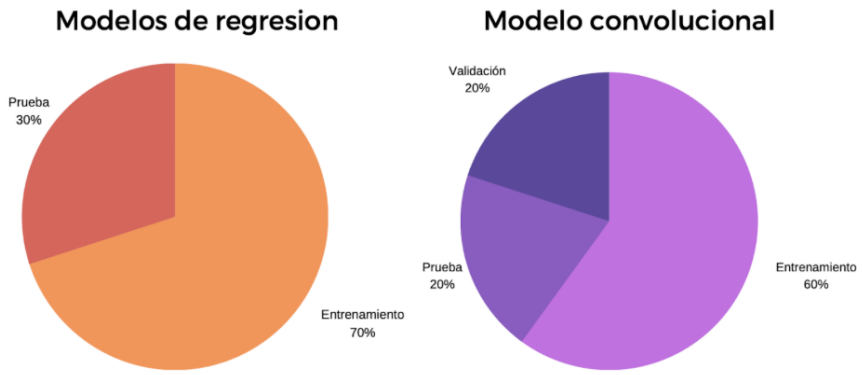


Figura 2: División de variables en los dos casos de modelos utilizados

Debido a las circunstancias distintas de nuestros puntos a seguir, los casos de prueba se organizaron de la siguiente manera (Figure 2), para los modelos de regresión en donde se reutilizaron los datos generados por el equipo de Nebraska, se dividió el 70

4. Modelos

Para recrear los resultados pasados debemos tomar en cuenta que se utilizó el modelo con mejores resultados, se especifica en su artículo (Chapman et al., 2022) que las características principales de sus modelos seleccionados son sostenibilidad para implementarlo con datos de nivel y flujo escalares, disponibilidad en herramientas conocidas de ciencia de datos como Weka (Frank et al., 2016), SciKit Learn (Buitinck et al., 2013), and R (R Core Team, 2016) y por ultimo modelos utilizados en otros casos similares, el modelo que seleccionamos fue:

- Random Forest Regression (RFR)

Ya contando con el modelo pasado se identificaron modelos similares a los anteriores para comparar el desempeño de ambos.

- Modelo de regresión linear
- Modelo de regresión Ridge
- Modelo de regresión Lasso
- Modelo de regresión K-Nearest Neighbor (KNN)
- Modelo de regresión con arbol de decision

Estos modelos se implementaron por su simplicidad, tanto como gran desempeño en trabajos de colinealidad y por la posibilidad de obtener mejores resultados para compararlos.

Cada modelos requería de dos versiones, una que estimara el nivel del agua (stage) y otro que estimara su flujo (discharge), por lo que al dividirlo por estaciones obtuvimos 48 versiones (12 por temporada), lo cual aplicamos a los modelos con mejor precisión.

Al utilizar las imágenes se realizaron dos pruebas, recortar imágenes para mostrar el área del flujo por la presa y el nivel del agua y aplicar un filtro de escalas grises como fuentes (Figure 3) se manejaron otros modelos especializados, ayudando a relacionar las imágenes con sus respectivos valores en los sensores. La esperanza es que este mismo modelo pueda identificar con solo la imagen los valores ya sea del nivel del agua tanto como el del flujo. El modelo específico para imágenes fue:

-Modelo de red convulsionar

Todos los modelos y sus resultados están accesibles en el repositorio del proyecto, anexado en este escrito.

Figura 3: Comparación entre imagen original e imagen utilizada con un recorte y filtro de escala de grises.

5. Resultados

Los primeros resultados obtenidos fueron la recreación de los resultados del equipo de Nebraska con el modelo de RFR como lo indicamos en la sección anterior. Los resultados obtuvieron la precisión del trabajo anterior, por lo que se pudo asegurar que los puntos de comienzo fueron exactos para continuar con una búsqueda de alternativas o mejoras. Los siguientes resultados fueron de los modelos de regresión lineal nuevos. En las series de tiempo no se lograron resultados aceptables, pues en cada caso solo se obtiene un promedio de los demás datos para remplazar los datos faltantes (Figure 4).

La variable utilizada para medir su rendimiento fue el MSE (Mean Square Error) y MAE (Mean Absolute Error). Estos representan la distancia del valor

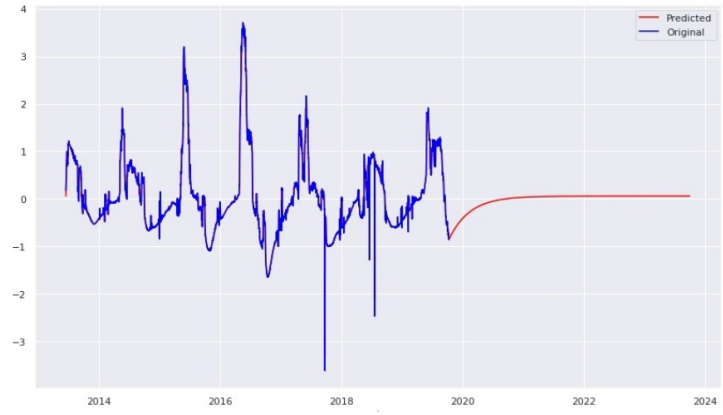


Figura 4: Resultados de predicción del nivel del agua con series de tiempo.

verdadero y el promedio del valor obtenido al cuadrado midiendo el error que puede obtener un modelo, mientras que el segundo es el promedio de este valor absoluto. Con esto se observa la precisión del modelo en los resultados tras evaluar sus pérdidas.

Por último se obtuvieron los resultados del modelo convolucional evaluado con las mismas variables, gracias a esto se logró una comparación aceptable (Table 1).

	Lineal	Ridge	Lasso	KNN	Decisión	RFR	CNN
MSE	0.784	0.245	0.443	0.195	0.398	0.398	0.143
% MAE	0.537	0.363	0.599	0.281	0.388	0.388	0.307

Cuadro 1: Comparación de resultados entre todos los modelos seleccionados.

Los mejores resultados se obtuvieron en todos los modelos al entrenar los modelos con los datos de una estación específica evaluandolos con esa misma temporada, la única excepción fue el modelo CNN en donde se entrenó con todos los datos y se evaluó por temporada.

Para elegir los mejores resultados se debe tomar en cuenta que aunque KNN muestra mejores resultados de forma general se encontró que estaba sobre entrenado. Esto se observa con que al seleccionar el valor de K el error es muy bajo por lo que no es preciso, mientras que al aumentarlo se dispara el error, dándonos a entender que el error aumentará hasta llegar a un número de K aceptable para el número de datos (Figure 5).

Ya con los resultados obtenidos se logró hacer una comparación en donde se encontró que el modelo de Ridge demostró mejores resultados (Table 2), aunque

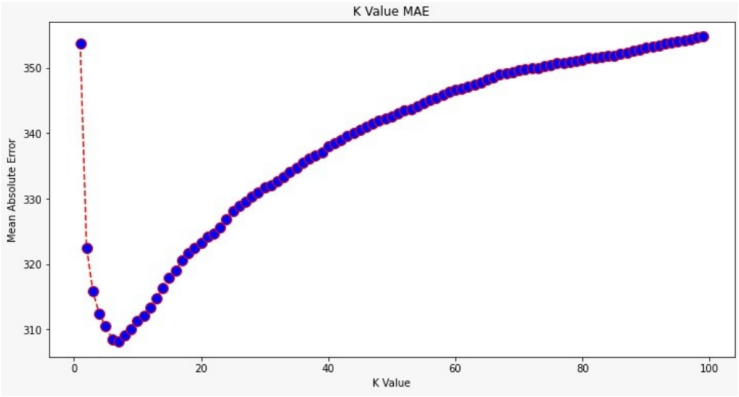


Figura 5: Demostración del valor de K a comparación del error.

el modelo de RFR mostró mayor precisión separado por estaciones (Table 3). El modelo CNN no mostraba resultados constantes variando la precisión, aunque en el mejor caso fue el mejor modelo (Table 4), considerando que la fuente de datos fue distinta.

Ridge	Primavera	Verano	Otoño	Invierno
MSE	0.423	0.332	0.475	0.797
% MAE	0.523	0.491	0.572	0.799

Cuadro 2: Resultados del modelo Ridge separado por temporada.

RFR	Primavera	Verano	Otoño	Invierno
MSE	0.176	0.176	0.176	0.175
% MAE	0.290	0.290	0.292	0.291

Cuadro 3: Resultados del modelo RFR separado por temporada.

6. Conclusion

Durante el trabajo se consideraron varias alternativas, dando la oportunidad de estudiar los datos a un fondo sorprendente, e impulsando a experimentar con soluciones creativas e innovadoras. La solución que mejores resultados fue tomar los datos y dividirlos por temporadas (primavera, verano, otoño e invierno),

CNN	Primavera	Verano	Otoño	Invierno
MSE	0.175	0.094	0.148	0.168
% MAE	0.342	0.256	0.272	0.356

Cuadro 4: Resultados del modelo CNN separado por temporada.

especializando un modelo para cada temporada, a su vez se implementó una solución que solo necesitaría imágenes con un leve procesamiento sin necesidad de un análisis complejo de las imágenes. Aunque el anterior muestra inconsistencia de resultados precisos se considera que fue la solución más creativa debido a que una vez entrenado podrá funcionar sin necesidad de alteraciones al sitio, implementando la solución de la hipótesis que creamos desde un inicio. Para concluir confirmamos la hipótesis inicial en la cual se creía que un modelo entrenado para temporadas específicas demostraría mejores resultados. Con esto aclarado los resultados con mayor precisión y consistencia de buenos resultados fue el modelo de RFR aplicado desde un comienzo por el equipo de Nebraska con la alteración de datos y segmentación de conjunto de entrenamiento y prueba.

Bibliografía

[1] Chapman, K. W., Gilmore, T. E., Chapman, C. D., Mehrubeoglu, M. Mittelstet, A. R. (2022). Camera-based Water Stage and Discharge Prediction with Machine Learning. University of Nebraska Lincoln.

[2] State Line Gauge Weir | PBT. (2022). https://plattebasintimelapse.com/explore/galleries/state-line-gauge-weir/

[3] Frank, E., Hall, M. A.,, Witten, I. H.: The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, fourth edn., https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016-appendix.pdf, 2016.

[4] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B.,, Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122, 2013.

[5] R Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/, 2016.

[6] GitHub - David Velazquez, AI WaterStageDischarge. (s. f.). GitHub. https://github.com/DavidVelazquez584/AI_WaterStageDischarge

Análisis de tendencias y predicción de fallas en sistemas de bases de datos relacionales

Diana Guadalupe García Aguirre

Tecnológico de Monterrey, Campus Guadalajara
Guadalajara, México

Resumen Una plataforma de sistemas de bases de datos relacionales contiene, en sus capas física y lógica, un gran número de componentes que permiten que almacene información de manera confiable y robusta, siendo todos ellos importantes para la persistencia de los datos y el rendimiento, escalabilidad, mantenimiento y disponibilidad de sus contenedores. En tiempos modernos, las compañías dedicadas a ofrecer soluciones empresariales de almacenamiento de datos físico y en la nube -un servicio conocido como *Cloud Storage*-, han dedicado esfuerzos titánicos para implementar herramientas que permitan simplificar el mantenimiento de sus sistemas y asegurar la detección de errores a sus clientes. El uso de algoritmos de aprendizaje profundo supone, por ende, una mejora que puede coadyuvar de forma potencial en el cometido antes descrito y reducir la cantidad de recursos necesarios y previsibles errores humanos. En el presente documento, se abordará el análisis e implementación de un algoritmo de esta índole, que se basa en los datos arrojados por una herramienta de chequeo de salud para este tipo de sistemas de almacenamiento, propiedad de Oracle®, y que ayudará a predecir el momento en el que dichos sistemas se encuentren en riesgo de incurrir en un error fatal.

Keywords: Plataforma de bases de datos relacionales · ORAchk® · Exadata · *Relational Database Management System (RDBMS)* · *Cloud Storage* · *Oracle Engineered Systems* · *Oracle Grid Infrastructure* · detección de errores automatizada · tendencias · predicción de errores críticos

1. Antecedentes e introducción

El buen funcionamiento de un sistema de software depende, de manera usual, en la forma en la que se ha configurado . Así, cualquier error de configuración, por mínimo que parezca, puede conseguir que los sistemas presenten comportamientos indeseados, con errores importantes y difíciles de diagnosticar y reparar. [9] [4] Esta misma premisa podría extrapolarse a los sistemas de hardware, que fungen un rol vital en la supervivencia del software y la existencia del almacenamiento de datos. En relación con esto último, en particular, si se habla de

sistemas que almacenan información muy importante y la distribuyen a numerosos clientes, en niveles de implementación corporativos, se tiene la necesidad imperiosa de contar con marcos de diagnóstico de errores fiables, que permitan mantener un nivel de desempeño óptimo de manera sostenida.

En los inicios de la década del 2010, Oracle contaba ya con una herramienta para auditar las configuraciones establecidas en sus sistemas, en categorías tales como: parámetros del *kernel* del sistema operativo, paquetes de software importantes para *Oracle Real Application Clusters* -en adelante RAC-, parámetros de funcionamiento de *Oracle Grid Infrastructure* y *Oracle Automatic Storage Management* -en adelante ASM-, así como configuraciones de bases de datos; llamada RACcheck® [21]. Brindaba un escaneo no intrusivo de todas estas áreas, que se ejecutaba por medio de un *daemon* integrado, lo que permitía ejecuciones silenciosas, programadas y sin intervención humana necesaria para ello. Creaba una serie de logs que le informaban al cliente sobre el estado actual de los servidores de bases de datos de Oracle, -por lo que se recomendaba correrlo especialmente al tener una instancia de base de datos cargada y ejecutándose-, y un reporte `.HTML` detallado que contenía beneficios, impactos, riesgos, acciones por tomar e información de reparación de varios aspectos del sistema analizado, según se requiriera.[14]

Asimismo, durante la década pasada, la ola del *Cloud Computing* comenzó a alzarse más y más, obligando a los proveedores de servicios e infraestructura computacional a tomar un segundo aire para replantearse sus modelos de negocio. Oracle, líder de la industria de las bases de datos desde hace más de treinta años, no se quedó atrás y, además de redefinir su oferta de servicios para ponerla a disposición de la nube, evolucionó su producto estrella de detección de errores de bases de datos automatizado, dando paso a lo que hoy por hoy se conoce como ORAchk®/EXAchk® y dejando atrás al antepasado RACcheck®. Este nuevo *framework* integraba las utilidades de RACcheck® con nuevas funcionalidades, que van perfeccionándose gracias a los clientes y a los desarrolladores, que van implementando mejoras de manera continua y conjunta.

Se describe en el manual *Oracle Engineered Systems for High Availability* [6] el funcionamiento de ORAchk® y EXAchk®, que luego de este párrafo se referenciarán únicamente como ORAchk®. Ambas herramientas son útiles para obtener chequeos de salud y pruebas de buenas prácticas en entornos Oracle de una forma comprensible para el usuario. Se pueden englobar a la postre sólo como ORAchk®, como ya se ha mencionado, pues su funcionamiento es equivalente en todo excepto en lo que concierne a la plataforma de Exadata [16], -que engloba un amplio servicio de bases de datos de Oracle en la nube-, como los chequeos a las celdas de almacenamiento, por ejemplo. En este documento, debido a la naturaleza del proyecto, resulta más útil mencionar a ORAchk®, por ser la versión que se encarga exclusivamente de productos de Oracle, pues su contraparte es capaz de analizar sistemas que no pertenecen al *Oracle Database Appliance* y el objetivo es poder ampliar el banco de conocimiento de esta compañía sobre su propia línea de negocio principal.

Conviene mencionar que ORAchk® ya viene preinstalado como parte del kit de herramientas de *Oracle Grid Infrastructure*. Sin embargo, puede descargarse también desde servidores de Oracle de manera directa, algo recomendado para asegurar siempre que se cuenta con la versión más reciente del programa [19]. Incluso, convendrá siempre conectar cualquier plataforma que ejecute la herramienta a la red para que pueda mantenerse actualizada. ORAchk® es un programa no intrusivo, igual que su antepasado. Corre de manera silenciosa y ágil, en segundo plano, gracias a su *daemon*, configurado por el usuario, y es capaz de almacenar los resultados obtenidos por el tiempo que el usuario determine y de enviar notificaciones y alertas de funcionamiento del sistema vía correo electrónico [6] [18] [17] [15].

Al ser una herramienta en constante evolución, se ha planteado la posibilidad de mejorarla por medio de modelos de aprendizaje de máquina. En este caso, se ha sugerido, mediante un levantamiento de requerimientos exhaustivo, que se describe en diversos lugares de este documento, según sea pertinente, que una de las mejoras más prometedoras consistiría en poder anticiparse a un estado de falla de los sistemas que ORAchk vigila. Es decir, en contar con un modelo, implementado dentro de la misma suite de software, capaz de predecir un estado de error basándose en el contexto proveído por los datos obtenidos a partir de los escaneos realizados. El camino proyectado para ello no es corto ni falto de dificultades, pero resulta, cuando menos, prometedor. En el presente documento, se relatará el proceso y los hallazgos obtenidos en la primera iteración en la búsqueda de dicho modelo, donde se proponen dos modelos estadísticos y uno de aprendizaje profundo que buscan predecir un estado de falla a partir de un conjunto de variables.

A manera de descargo de responsabilidad, vale la pena mencionar que este reporte incluye únicamente los hallazgos, metodologías y aprendizajes de once semanas de trabajo. El proyecto, cuyos alcances en cuestión de tiempo podrían verse incrementados cuantiosamente en el futuro en vista del alcance y recursos disponibles hasta el momento de redacción de este escrito, consideró también los entrenamientos previos en materia de inteligencia artificial y conocimientos técnicos internos de la empresa involucrada. [7].

2. Metodología

La metodología CRISP-DM, cuyas fases se ilustran en la figura 1, fue acuñada en 1996 por expertos provenientes de tres organizaciones que, en aquel momento, eran punta de lanza del mercado en ciernes de la minería de datos, como una manera de estandarizar y regular las prácticas que gobernarían esta industria, aún inmadura y con un potencial tan enorme que resultaba casi atemorizante. Se trataba de NCR, SPSS y Daimler Chrysler. [1]

De acuerdo con lo expuesto por Haya (2021), la estandarización no buscaba únicamente regular este tipo de prácticas, sino también dar control a quienes se dedican a este ramo sobre sus resultados. Se trata de una metodología capaz de iterar sobre el propio proceso que se sigue, de tal forma que se pueda resolver el problema planteado de manera eficaz. En palabras del Espinoza-Zúñiga (2020), IBM es la empresa que más promueve el uso de esta metodología, si bien se trata en nuestros tiempos de un estándar de facto en el desarrollo de soluciones de minería de datos, por lo que no solamente IBM lo usa, sino también cientos de mineros de datos en otras industrias y corporaciones y, cabe resaltar, es el más usado en la academia.

CRISP-DM son las siglas en inglés de *CRoss Industry Standard Process for Data Mining* y describen el proceso que idealmente debería seguirse para la Ciencia de Datos. Consiste en seis etapas:

- Entendimiento del negocio, o *business understanding*, etapa crucial en la que se entiende lo más posible la problemática y se conocen los requisitos, restricciones y beneficios de ejecutar el proyecto. De igual forma resulta útil para marcar un punto de partida, un marcador de inicio, que permita contrastar el estado del negocio previo y posterior a la minería de datos al llegar a la quinta etapa.
- Entendimiento de los datos, o *data understanding*, etapa en la que se identifican claramente los formatos, contenidos y roles de los datos con los que se cuenta, de forma que sea posible manipularlos posteriormente con soltura y eficacia.
- Preparación de los datos, o *data preparation*, etapa que, por regla general, suele consumir más tiempo, dado que permite seleccionar aquellos datos que sean útiles para el posterior modelado y eliminar aquellos que no aporten realmente -o cuya aportación sea virtualmente nula- en ese sentido.
- Modelado, o *modeling*, etapa en la que se procesa lo aprendido en la etapa previa para generar modelos estadísticos y algorítmicos mediante técnicas apropiadas de acuerdo con el problema a resolver. También implica el cambio de parámetros iterativo hasta encontrar un modelo viable y óptimo.
- Evaluación, o *evaluation*: En la penúltima etapa, por medio de métricas concretas, se evalúa la calidad del modelo previamente planteado, ya sean medidas estadísticas o un análisis con los clientes al aplicar la solución propuesta. De ser necesario, se iterará a partir del paso previo conveniente, pudiendo ser el caso de tener que comenzar desde cero.
- Despliegue, o *deployment*, etapa final en la que se documenta el proceso para recabar todo el conocimiento adquirido en un corpus concreto y se monitorea la implantación de la solución, en aras de detectar áreas de oportunidad.

Durante once semanas se ha trabajado siguiendo los pasos marcados por esta metodología. Cabe hacer mención del hecho de que el proyecto principal del que se desprende este reporte se encuentra actualmente en las fases de entendimiento y preparación de datos, por lo que sólo se alcanzará el estatus de evaluación y, en el presente documento, se plasmará el proceso en una suerte de despliegue previo.

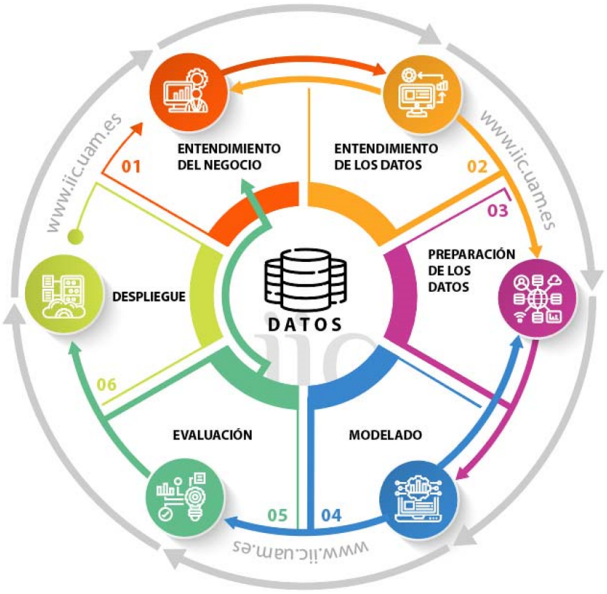


Figura 1: Esquema del ciclo CRISP-DM estándar. Tomado del sitio web del Instituto de Ingeniería del Conocimiento de la Universidad Autónoma de Madrid, en https://www.iic.uam.es/innovacion/metodologia-crisp-dm-ciencia-de-datos/

3. Análisis exploratorio de los datos

Con el objetivo de realizar cualquier tipo de análisis de datos, se requiere, por supuesto, describirlos, conocerlos y visualizarlos en primera instancia. En este punto del documento, cabe destacar algunos puntos importantes respecto a su manejo y disponibilidad:

- El conjunto de datos que se utilizará en este documento para evidenciar la estructura con la que se cuenta, así como el usado para entrenar el modelo aquí descrito, han sido previamente sanitizados y anonimizados para proteger los intereses de todas las partes involucradas, así como de los entes propietarios de dichos datos, de acuerdo con las legislaciones aplicables en nuestro país y el estado de Jalisco y lo que consta en las políticas de la compañía que los provee, misma que previamente recolectó los datos gracias a un acuerdo de confidencialidad y anonimato absolutos con sus antiguos propietarios. Todos los datos recabados se usan e ilustran en este documento con fines meramente académicos y se han censurado de manera adicional para poder mostrarlos aquí.
- El acceso a los datos se logra a través de un canal especializado y confidencial, del que sólo los interesados tienen conocimiento, y que cumple con las normativas expresadas en el acuerdo entre la compañía propietaria de

los conjuntos de datos y demás entes involucrados en el desarrollo de este proyecto.
- Los datos recabados hasta este momento, debido a los costes de sanitización, son limitados. Se usarán para realizar un análisis exploratorio que se pueda mostrar en este documento y, posteriormente, los modelos propuestos y construidos se probarán de manera interna en la compañía que los provee. Aquí se incluirán únicamente los resultados obtenidos en dichas pruebas, pero no se compartirá, bajo ningún concepto, el conjunto de datos de prueba, al no encontrarse debidamente sanitizado ni anonimizado.
- El manejo de los datos sanitizados no requiere del uso de herramientas de *big data* por la cantidad tan baja de registros y el hecho de que son archivos de texto. No obstante, para trabajar con los datos no sanitizados y restringidos, sí se han manejado herramientas de esa índole, si bien no pueden compartirse en este reporte por las razones antes citadas.
- Dado lo anterior, no se compartirán datos de ninguna índole mediante repositorios de GitHub ni con personas ajenas a la realización del proyecto, siendo la única demostración del trabajo realizado el presente documento. Se anexa al final un documento que prueba la participación de la autora en el proyecto.

Dicho lo anterior, resulta preciso señalar la manera en la que ORAchk trabaja, de manera general. La herramienta en cuestión realiza chequeos de diversas índoles en el sistema de almacenamiento, a niveles lógico y físico. Posteriormente, despliega los resultados obtenidos por medio de múltiples archivos en formato `.JSON`, `.HTML` y `.OUT`. A pesar de que existe una amplia variedad de archivos por analizar, por cuestiones de tiempo y alcance, para el presente análisis, se usará sólo uno de los archivos de salida de tipo `.JSON`, que contiene un condensado de resultados de las pruebas realizadas e informa del estado de cada uno de los chequeos. Para el entrenamiento, contamos con treinta reportes de salida, de los que se extraerá el archivo `.JSON` que se ha mencionado previamente y se concatenarán en un único DataFrame. A partir de este archivo, se buscará encontrar tendencias concretas que permitan realizar predicciones de estados futuros. El resto de los archivos de salida no se usará, además, por cuestiones relacionadas con la sanitización de datos, pues no todos los archivos pueden limpiarse y anonimizarse correctamente y quedarían fuera de un posible análisis externo a Oracle.

Si se carga el archivo en una variable de tipo DataFrame, usando la librería Pandas de Python, se obtendrá una tabla con 25 columnas. El conjunto de datos sanitizados cuenta con 34,484 registros de chequeos.

A continuación, se lista una descripción de las columnas del DataFrame obtenido:

1. `modelVersion`: Describe la versión del software que realiza las pruebas que se está ejecutando en cada chequeo.

2. `modelVersion`: Describe la configuración física del rack donde reside el sistema sobre el que se ejecuta la prueba.
3. `engineeredSystems`: Describe el tipo de contenedor físico donde reside el sistema sobre el que se ejecuta la prueba.
4. `RackIdentifier`: Contiene el identificador del rack donde reside el sistema sobre el que se ejecuta la prueba.
5. `OSVersion`: Describe la versión del sistema operativo del sistema.
6. `DBVersion`: Describe la versión de la base de datos sobre la que se ejecuta la prueba -sólo se presenta en chequeos de la base de datos, lo que excluye otros tipos de chequeos-.
7. `ExadataVersion`: Describe la versión del contenedor físico donde reside el sistema sobre el que se ejecuta la prueba.
8. `exachkExecTimestamp`: Contiene el *timestamp* de inicio de ejecución de cada chequeo, incluyendo fecha, hora en formato hh:mm:ss y zona horaria en formato de tres letras.
9. `exachkID`: Contiene el identificador único del chequeo realizado
10. `exachkType`: Describe el componente sobre el que se ejecuta el chequeo -por ejemplo, sistema operativo, SQL, etc.
11. `exachkName`: Contiene el nombre del chequeo realizado.
12. `exachkMessage`: Contiene una descripción corta del chequeo realizado.
13. `exachkAlertType`: Contiene el estado de falla del chequeo en cuestión, pudiendo ser `FAIL`, `WARNING`, `INFO` o `CRITICAL`, en función del chequeo *per se*.
14. `exachkTargetType`: Describe el componente sobre el que se ejecuta el chequeo, en términos de los componentes del sistema -por ejemplo, host, storage cell, switch, `ASM`, etc.
15. `exachkActualValue`: Expresa el valor real del chequeo.
16. `exachkStatus`: Expresa el estado final del chequeo, pudiendo ser `PASS` o un fallo del chequeo. Si esto último ocurre, el chequeo podrá encontrarse en diversos estados, en función de cada uno, como se mencionó en el punto quince de esta lista.
17. `exachkStatusCode`: Expresa el estado final del chequeo de manera numérica.
18. `exachkReturnCode`: Expresa el valor numérico de retorno del chequeo.
19. `exachkMsgDetail`: Contiene mensajes de información adicional de cada chequeo.
20. `exachkOutfilePath`: Contiene la ruta en la que se almacenó el archivo `.OUT` del chequeo, que contiene los logs obtenidos por el chequeo. Nótese que puede estar en blanco.
21. `NodeName`: Contiene el nombre del nodo sobre el que se ejecutó el chequeo.
22. `exachkTypeInHtml`: Contiene el componente sobre el que se corrió el chequeo.
23. `PreviousStatus`: Expresa el valor obtenido en la corrida previa de ese mismo chequeo (nota: este valor únicamente se obtendrá a partir de corridas programadas mediante el *daemon* del sistema operativo sobre el que se ejecuta el chequeo).
24. `DBName`: Contiene el nombre de la base de datos sobre la que se ejecutó el chequeo (nótese que este campo sólo aplica para los chequeos propios de base de datos).

25. `InstanceName`: Contiene el nombre de la instancia sobre la que se ejecutó el chequeo.

Adicionalmente, conviene echar un vistazo al reporte que entrega la herramienta en formato `.HTML`, que también describe de forma clara, especialmente dirigida a los clientes, algunos puntos importantes:

A partir de la descripción anterior, de conversaciones sostenidas con los desarrolladores de la herramienta y del análisis de lo que cada columna representa, se obtienen los siguientes hallazgos y tareas:

- La columna de `exachkID` resulta de vital importancia para filtrar algunos registros que no aportarán nada al análisis. Al observar que la columna `exachkAlertType` debería contener únicamente los potenciales estatus de error de cada chequeo y que en nuestro conjunto de datos existen algunos registros con valores `PASS` para esa categoría, se averiguó que existen chequeos que no deberán ser tomados en cuenta por ser genéricos y estar ligados a la instalación de la herramienta y no propiamente a la salud y funcionamiento de la plataforma de base de datos. Estos chequeos tienen un ID que comienza con el prefijo `GCHECKID`, por lo que para el análisis deberán desecharse del conjunto de datos. Se realizará el filtrado por medio de expresiones regulares, quedando con 33,092 registros útiles.
- Se realizaron gráficos de comparación entre pares de variables para observar la incidencia de errores en relación con las variables que, de acuerdo con los desarrolladores de la herramienta, cuentan con un mayor impacto de desempeño en la variable de salida identificada, en este caso, el `exachkStatus`, que indica si el chequeo fue exitoso o falló: Es importante mencionar que cada aspecto auditado funciona de manera similar a como lo hace un interruptor: cada *check* solamente puede tener dos estados: el de `PASS` y el de `FAIL`. No obstante, la falla puede ser de cinco tipos diferentes: `CRITICAL`, `FAIL`, `WARNING`, `INFO` y `SKIP`, que dependerán de cada *check* en particular y cuya ponderación se observa en el reporte generado por la herramienta en `.HTML`: A partir de los gráficos previos, se puede inferir lo siguiente: El análisis de `PASS` y `FAIL` de cada *check* deberá tomarse con dos perspectivas posibles. En la primera, se puede tratar a la variable de salida como una variable de Bernoulli, en la que sólo se cuenta con dos estados posibles. De hacer esto, se pueden considerar todas las fallas como un cero y todos los `PASS` como un uno y realizar modelos en consecuencia. En la segunda perspectiva, se toma en consideración que cada falla pondera de manera distinta en el resultado, con lo que claramente valdría la pena centrar la atención en aquellas fallas que descuentan más puntos, como los `CRITICAL`, que son, por supuesto, los menos deseados en los resultados de ejecución. En esta segunda perspectiva las figuras antes mostradas resultan de utilidad, pues es evidente que hay aspectos específicos en los que se observan *check* con resultado `CRITICAL`. Por ejemplo, en la figura 2 se puede ver que los *check* con esta salida son únicamente aquellos que validan el estado del switch. Esto, de acuerdo con

Oracle Exadata Assessment Report

System Health Score is 93 out of 100 (detail)

System Health Score is derived using following formula.

- Every check has 10 bhbpeq
- Critical will deduct 10 bhbpeq
- Failure will deduct 7 bhbpeq
- Warning will deduct 5 bhbpeq
- Info will deduct 3 bhbpeq
- Skip will deduct 3 bhbpeq

Critical checks	0
Failed checks	70
Warning checks	29
Info checks	69
Passed checks	1054
Skipped checks	43
Total checks	1265

..Hide

Cluster Summary

Heading	Description
Cluster Name	Cluster-c1
OS/Kernel Version	LINUX X86-64 OELRHEL 7 4.14.35-2047.505.4.4.el7uek.x86_64
CRS Home - Version	/u01/app/21.0.0.0/grid - 21.3.0.0.0
DB Home - Version - Names	/u01/app/oracle/product/21.0.0.0/dbhome_1 - 21.3.0.0.0 - hczh213 database /u01/app/oracle/product/19.0.0.0/dbhome_1 - 19.12.0.0.0 - vmau61m database /u01/app/oracle/product/18.0.0.0/dbhome_1 - 18.14.0.0.0 - cmkz12d database /u01/app/oracle/product/12.2.0.1/dbhome_1 - 12.2.0.1.210720 - hczh122 database /u01/app/oracle/product/12.1.0.2/dbhome_1 - 12.1.0.2.210720 - 3 databases
Exadata Version	21.2.4.0.0.210909
Number of nodes	8
Database Rlaodxd	2
Storage Rlaodxd	3
IB Switches	3
EXAchk Version	22.2.0_20220707
Collection	exachk_odfq95malqtj01_hnl48b_080322_113355_CRONTAB_COLLECTION
Duration	40 mins, 3 seconds
Executed by	root
Arguments	-hardwaretype X4-2 -clusternodes odfq95malqtj01.tg.qbfulm.xhj,odfq95malqtj02.tg.qbfulm.xhj -cells 192.157.0.47,192.157.0.45,192.157.0.56 -ibswitches oxke28we-yzz0.tg.qbfulm.xhj,oxke28we-hke0.tg.qbfulm.xhj,oxke28we-fka0.tg.qbfulm.xhj -withperfdata -silentforce -showpass -show_critical -tag CRONTAB_COLLECTION -nocleanup
Collection Date	03-Aug-2022 11:37:27

Figura 2: Reporte de resultados de la herramienta ORAchk en formato HTML. Obtenida de los reportes compartidos de manera confidencial.

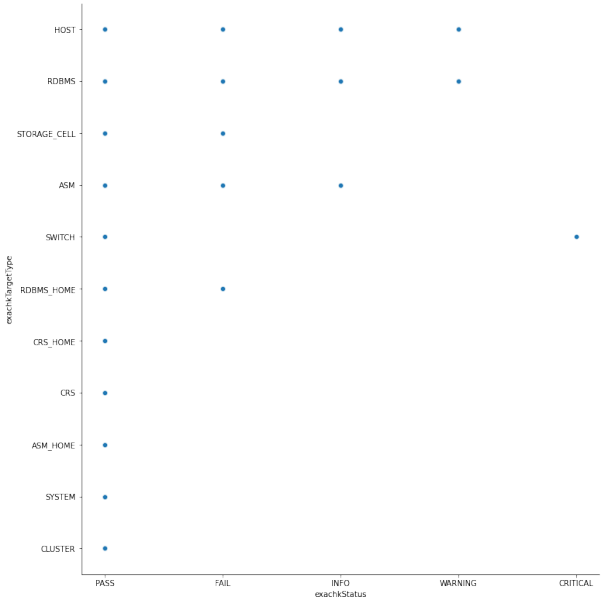


Figura 3: Relación entre exachkTargetType y exachkStatus. El TargetType describe el componente de la plataforma Exadata sobre el que se ejecuta la comprobación de estado.

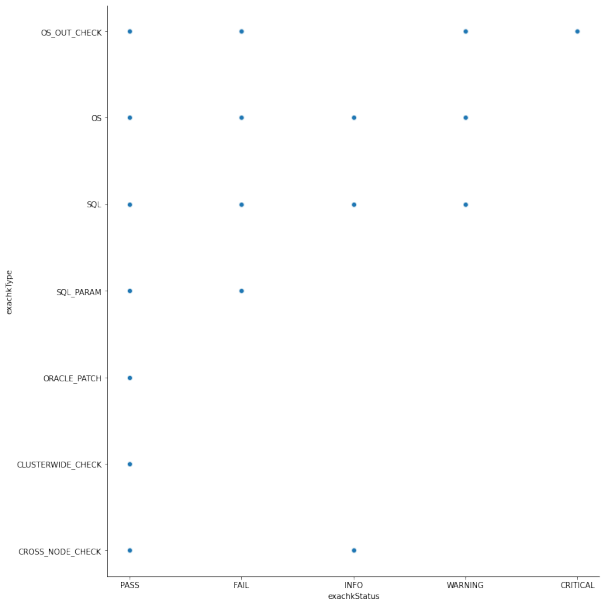


Figura 4: Relación entre exachkType y exachkStatus. El EXAchk Type describe el tipo de comprobación de estado ejecutada a nivel de software.

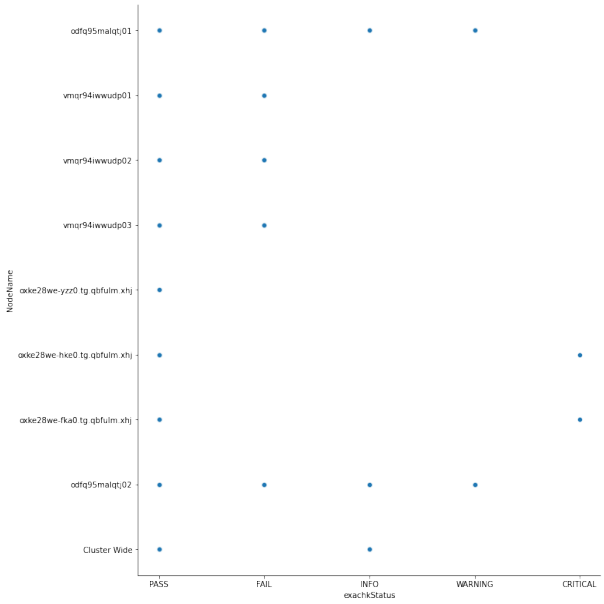


Figura 5: Relación entre NodeName y exachkStatus. El Node Name describe el nodo sobre el que se ejecutó el chequeo.

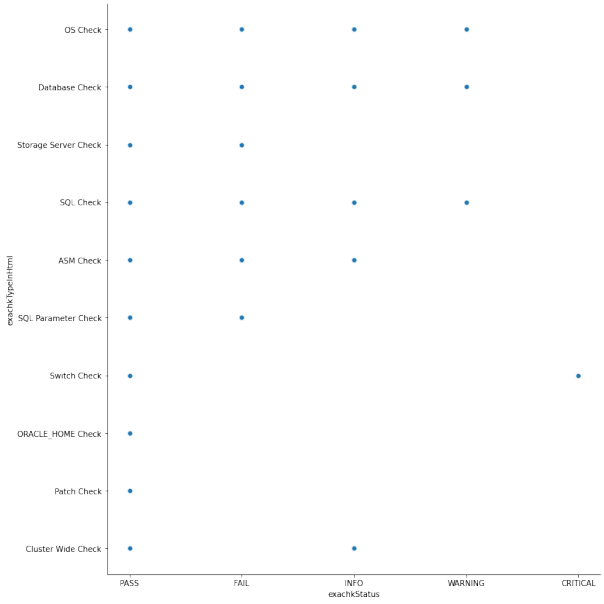


Figura 6: Relación entre exachkTypeInHTML y exachkStatus. El EXAchk Type in HTML describe el tipo de comprobación de estado ejecutada en general.

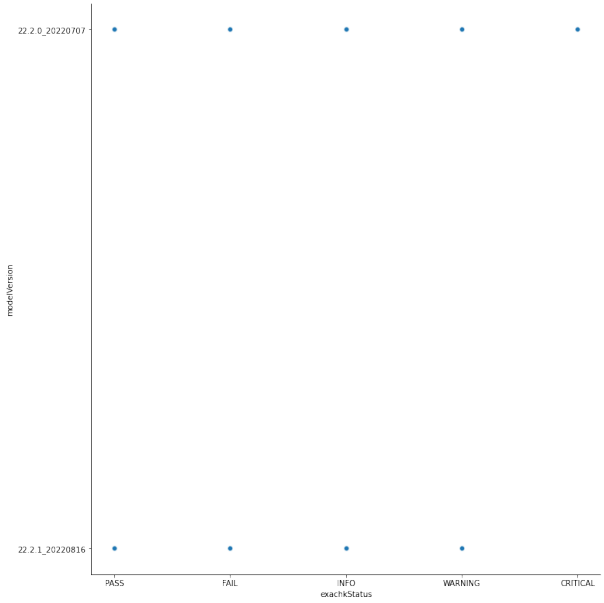


Figura 7: Relación entre modelVersion y exachkStatus. El Model Version describe la versión de ORAchk ejecutada.

Oracle Exadata Assessment Report

System Health Score is 93 out of 100 (detail)

System Health Score is derived using following formula.

- Every check has 10 bhbpeq
- Critical will deduct 10 bhbpeq
- Failure will deduct 7 bhbpeq
- Warning will deduct 5 bhbpeq
- Info will deduct 3 bhbpeq
- Skip will deduct 3 bhbpeq

Critical checks	0
Failed checks	70
Warning checks	29
Info checks	69
Passed checks	1054
Skipped checks	43
Total checks	1265

..Hide

Figura 8: Deducciones en la puntuación de salud del sistema con base en el tipo de falla generada.

los desarrolladores de la herramienta, es más bien producto de observar la arquitectura de un sistema de Exadata.

En el diagrama, se puede observar que el switch es responsable de admi-

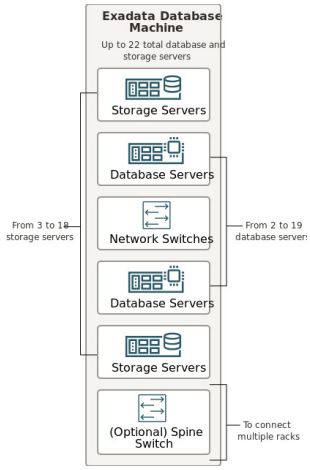


Figura 9: Arquitectura de un Engineered Systems Exadata. Tomada de Oracle Docs, en https://docs.oracle.com/en/engineered-systems/exadata-database-machine/edbid/ [20]

nistrar la comunicación entre el resto de los componentes del Exadata e, inclusive, de la comunicación entre varios racks. Por ende, puede suponerse que una falla ahí será mucho más importante que una falla en los servidores de almacenamiento o de base de datos, si bien las fallas en estos son también negativas. Un `FAIL` implica una deducción de cinco puntos, por lo que será el siguiente en importancia. La mitad de los componentes: `HOST`, `RDBMS`, `ASM`, `STORAGE_CELL` y `RDBMS_HOME`, tienen fallas de tipo `FAIL`. De ahí existen errores `WARNING` e `INFO`, que deducen cinco y tres puntos, respectivamente. Los conjuntos de datos de entrenamiento y pruebas están libres de incidencias de `SKIP`. No obstante, resulta interesante el hecho de que en `CRS`, `CRS_HOME`, `ASM_HOME`, `SYSTEM` y `CLUSTER` no hay incidencia de errores, los *check* pasan la prueba en su totalidad. Resultados similares se observan con componentes y nombres y versiones de nodos: los fallos críticos son exclusivos de ciertos tipos, hay otros que siempre pasan las pruebas y hay algunos otros que tienen una mescolanza de tipos de fallos y aciertos.

Cualquiera de las perspectivas, tanto la dicotómica como la que contempla una salida múltiple, son dignas de estudio y, con un alcance de proyecto mayor, *a posteriori*, podrán implementarse para observar contrastes de desempeño en los modelos obtenidos. Además, la interpretación de los datos ofrece un panorama de conocimiento mayor de la estructura arquitectónica del sistema analizado y permite hacer hipótesis pequeñas de correlación entre

variables, que habrán de comprobarse. Para los propósitos de este documento, se optará primordialmente por tratar a todos los fallos como uno solo, de modo que cada muestra aleatoria -o *check*-, sea en cierta manera similar a un experimento de Bernoulli, si bien se realizarán modelos de contraste para observar si hay mejoras de desempeño.

Es importante remarcar una premisa mencionada en la introducción de este documento de la cual sería útil extraer información, pero para la cual no se cuenta con los datos suficientes: ORAchk® implementa un *daemon* que permite ejecuciones silenciosas y no intrusivas, que el usuario configura manualmente por medio de un *cronjob*. En un escenario realista, lo ideal sería configurar este programa para que se ejecute con cierta periodicidad sobre el producto de Oracle en cuestión. El usuario entonces deja al sistema a cargo de su propia detección de errores, pues el programa implementa también funcionalidades como: envío automatizado de notificaciones y alertas vía correo electrónico y tiempo límite de ejecución, que asegura que el programa no se quedará colgado si encuentra una excepción, sino que se saltará la ejecución del *check*, -dando como resultado un `SKIP`-, lo que, como ya se ha mencionado, restará puntos al estado de salud en general y permitirá la solución de los errores que se encuentren o provoquen un salto de *check*. No obstante, ORAchk® también puede ejecutarse sin el *daemon*, lo que genera resultados que no cuentan con registros de estados previos y tampoco poseen una periodicidad exacta de ejecución. Los registros con los que se cuenta para entrenar y hacer exploración de datos, que, como ya se ha mencionado, están anonimizados y sanitizados y, cabe añadir, son fruto de ejecuciones realizadas sin el *daemon* y en intervalos irregulares, por lo que un análisis sobre tiempo de ejecución y periodicidad es imposible. Asimismo, luego de analizar los datos disponibles, si bien los *timestamps* se encuentran presentes en las ejecuciones, no permiten conocer el tiempo que le toma correr a los *check* con exactitud. Debido a lo anterior, si bien resultaría interesante analizar el fenómeno basándonos en el tiempo para modelarlo de manera estocástica, se carece de los datos necesarios para hacerlo de esa forma y, por ello, se propondrán en su lugar modelos deterministas, como Redes Neuronales Convolucionales.

Tal como se mencionó previamente, se ha de considerar como un factor relevante lo mencionado por los desarrolladores respecto a las variables con mayor influencia en el desempeño general de un sistema de bases de datos. Conocer ese dato y sopesar el contenido del DataFrame en su totalidad permite el discernimiento de aquellas variables que pueden resultar útiles y aquellas que no. En ese sentido, se considera pertinente filtrar el DataFrame para enfocarnos, al menos en esta primera etapa, en analizar la correlación existente entre las variables regresoras más importantes para la compañía y la variable de salida (`exachkStatus`)

Por otro lado, resulta necesario compartir también en este reporte otro de los enfoques con el que se ha intentado analizar tendencias y patrones en los datos. Se trata de un modelo de análisis estadístico automatizado que

se ejecuta por medio de una interfaz de línea de comandos, -en adelante `CLI`-. Desde la línea de comandos, se pueden descargar, computar y generar gráficas de contraste con un banco de reportes disponible únicamente de manera interna y reservada -motivo por el cual no se compartirá ni el código implementado ni los datos analizados-, pero cuyos resultados son dignos de mención en el análisis exploratorio de datos de este documento. Bien, ¿qué pasaría si se intentara visualizar el porcentaje de éxito en la ejecución de una comprobación basándonos en el desempeño de varios sistemas individuales? Aquí se muestran algunos ejemplos de las visualizaciones obtenidas por medio de la herramienta antes descrita:

La figura 10 muestra cuatro de las gráficas que se han generado por me-

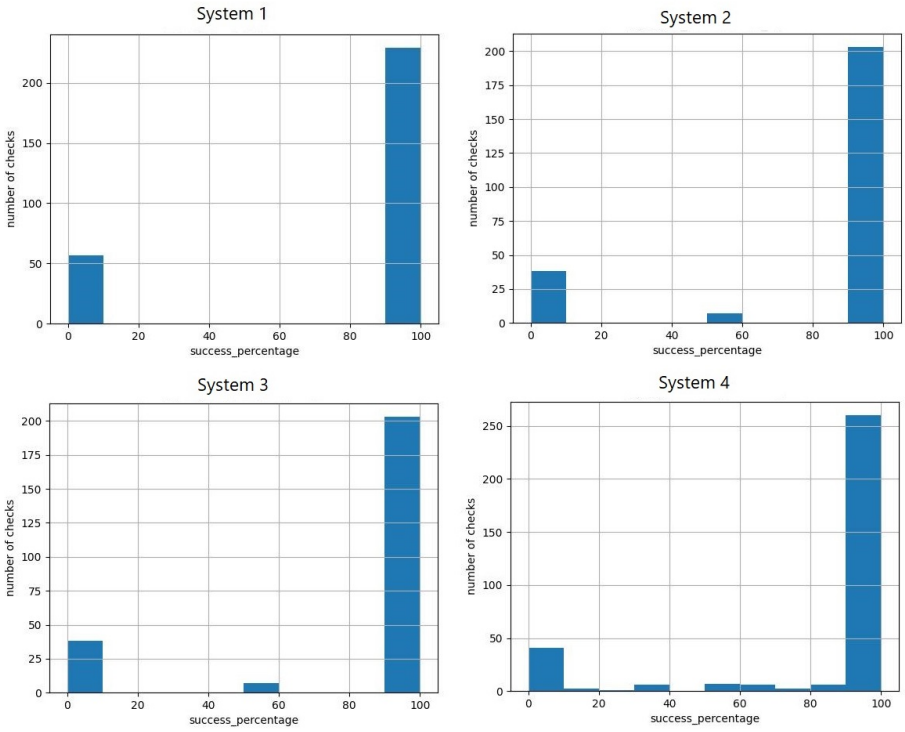


Figura 10: Visualización del porcentaje de éxito en las ejecuciones de cuatro sistemas de bases de datos. Los nombres de los sistemas están anonimizados. Gráficos de elaboración interna.

dio de la CLI. En el eje de las x, se encuentra el porcentaje de éxito de los *checks* expresado como porcentaje, pero basado en el estatus: por ejemplo, un `PASS` tendría un porcentaje de éxito equivalente a 100, mientras que uno de `CRITICAL` tendría uno de 0. En el eje de las y se encuentra la cantidad

de *checks* encontrados en cada sistema por estatus. A partir de su análisis se hallaron cosas interesantes. Por cuestiones de espacio, no se mostrarán en este documento todas las gráficas generadas, pero todas ellas siguen estos mismos cuatro patrones: existen ejecuciones en los que los *checks* se distribuyen en dos columnas únicamente, con porcentajes de éxito de cien y cero, sin puntos medios, otras comparten el mismo patrón, pero tienen una o más barras intermedias que denotan a cierta cantidad de *checks* cuyo desempeño ronda el 20 y el 80 por ciento (es decir, se trata de *checks* con estatus de `WARNING`, `INFO` o `FAIL`). La tendencia se inclina hacia este último ejemplo.

4. Implementación de modelos *benchmark* de análisis estadístico: Regresión lineal multivariada

Toda vez que se han analizado los datos disponibles, se mostrará en este apartado del documento la manera en la que se ha seleccionado, implementado y evaluado un modelo *benchmark* de análisis estadístico del problema.

Los problemas de clasificación basados en características son muy comunes en el mundo real. Entender los conceptos de causalidad y correlación para quienes se dedican a la minería de datos es esencial, dado que, para construir modelos fiables, se requiere analizar los datos con los que se cuenta, en aras de descubrir si las variables involucradas en un resultado en específico se encuentran tan intrínsecamente relacionadas que, podría decirse, causan un comportamiento, o bien, sólo influyen levemente en que ocurra o la manera en la que lo hace. El proyecto que aquí se está describiendo no ha carecido en absoluto de este tipo de análisis, ya que la pregunta que el equipo de investigadores trata de resolver es, precisamente, ¿cuáles de los datos que recaba la herramienta pueden ser de utilidad para intentar predecir errores? De ser así, ¿cuál es el modelo que lo sustenta?

Si bien cabe hacer notar que el proyecto principal se encuentra todavía en una fase exploratoria de los datos, pues comprenderlos no ha sido una tarea sencilla, para efectos de este documento nos centraremos en uno de los archivos `.JSON` de salida, que conjunta, cuando menos, un condensado de los datos que, hasta el momento, se han considerado más relevantes. A partir de ahí, se ha considerado prudente implementar un modelo de regresión multivariada, que brinde una primera aproximación al problema, basada en lo aprendido durante el curso. Se ha elegido este modelo porque los modelos de regresión son capaces de predecir con base en características y son relativamente fáciles de evaluar y de ilustrar en este documento, si bien se prevén resultados no prometedores en vista de que, como ya se verá más adelante, no existen relaciones lineales entre las variables de entrada y la de salida. En el apartado de aprendizaje profundo, se implementará otro modelo, conocido como *K-Nearest Neighbors* para contrastar los resultados obtenidos con Redes Neuronales Convolucionales, que, si bien no se evalúa con el mismo detalle que la regresión multivariada en este documento, sí presenta resultados prometedores.

Como ya se ha explicado en párrafos anteriores, los desarrolladores, con base en su experiencia, han indicado cuáles de las variables les parecen más importantes y, por ende, que merecen tener un lugar asegurado en el análisis realizado mediante modelos estadísticos.

Por este motivo, la primera fase de la implementación del modelo consistió en limpiar los datos [2] [12] basándonos en tres premisas:

- Incluir los apartados indicados por los desarrolladores: `modelVersion`, `exachkTargetType`, `exachkType` y `NodeName`.
- Eliminar el resto de las columnas que, dada su naturaleza, no aportan mucho a un modelo de regresión, tales como mensajes de salida, variables que describen cosas similares, logs y claves únicas de identificación, dejando también la columna `exachkTypeInHtml`".
- Eliminar los registros de chequeos que no forman parte de la ejecución, sino de la instalación, cuyo ID comienza con `GCHECKID`.

Asimismo, teniendo en consideración que los datos están desbalanceados, se tratará a la variable de salida como una salida dicotómica, pudiendo solamente indicar un `PASS` (con 1) o un `FAIL` (con 0), con lo que podremos alimentar al algoritmo con más información sobre el contexto de los fallos. En adelante, en caso de tener más datos disponibles, como un trabajo futuro, puede considerarse la necesidad de volver a establecer cinco variables de salida en lugar de dos.

Luego de la limpieza de los datos, se prepararán [8] [10] [11] [13] por medio de la función `LabelEncoder` de `sklearn`, con lo que las etiquetas categóricas de todos los datos se convertirán en números, según corresponda. En el caso de la variable de salida, se establece que todo aquel estatus de salida que no sea `PASS`, será considerado `FAIL`. Se verifica que no existan datos faltantes antes de proceder con algunas de las métricas iniciales de evaluación, siendo el caso que no existen datos por imputar. Como puede observarse en las gráficas [3] de la figura 10, los datos, al provenir enteramente de variables categóricas, no presentan, a primera vista, una distribución normal. Esto se evaluará a través de métricas que se citarán posteriormente. Además de analizar la distribución, conviene observar si existe correlación entre las variables, usando tres tipos de métrica: Pearson, Spearman y Kendall. En la figura 12 se observa que, en términos de la de Pearson, la correlación entre las variables regresoras y la de salida es bastante baja, algo esperable en vista de que este tipo de correlación mide directamente relaciones lineales, que no se observan en los datos. Preocupantemente, se puede observar que, de manera contraria a lo que pasa con la variable de salida, entre las variables regresoras, sí existen correlaciones mucho más fuertes, como es el caso de `NodeName` y `exachkType` con `exachkTargetType`. Para intentar paliar esto, podrá elegirse la eliminación de las variables que se encuentren más correlacionadas entre sí y se construirá otro modelo con distinto número de variables regresoras, con el objetivo de verificar cómo se modifican las métricas de desempeño y ver si es posible mejorarlo.

No obstante, como ya se mencionó, la correlación de Pearson no es la única que existe. Es por esta razón que se presentan aquí las tablas de valores de correlación de Spearman y Kendall:

Cuadro 1: Correlación de Spearman

Variable regresora	r	$p-val$
modelVersion	-0.001	0.853
exachkTargetType	-0.041	4.587
exachkType	0.07	1.024
NodeName	0.078	2.135
exachkTypeInHtml	0.125	7.785

Cuadro 2: Correlación de Kendall

Variable regresora	r	$p-val$
modelVersion	-0.001	0.853
exachkTargetType	-0.038	4.696
exachkType	0.064	1.254
NodeName	0.073	2.917
exachkTypeInHtml	0.110	6.275

Puede observarse que, aún intentando medir la correlación con otros paradigmas, no puede observarse un aumento significativo en los números obtenidos para el valor de r. Conviene recordar que la correlación de Spearman no es lineal, al menos no necesariamente, por lo que este caso es muy interesante. Se creará un modelo de regresión multivariada y se observará su desempeño. La salida de consola al crear el modelo es la siguiente:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           exachkStatus   R-squared:                       0.031
Model:                          OLS   Adj. R-squared:                  0.031
Method:               Least Squares   F-statistic:                     210.2
Date:              Sat, 19 Nov 2022   Prob (F-statistic):          1.95e-221
Time:                      17:33:35   Log-Likelihood:                -9574.6
No. Observations:             33092   AIC:                         1.916e+04
Df Residuals:                 33086   BIC:                         1.921e+04
Df Model:                         5
Covariance Type:          nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const              1.0020      0.010    101.830      0.000       0.983       1.021
modelVersion      -0.0007      0.005     -0.159      0.874      -0.010       0.008
exachkTargetType  -0.0309      0.001    -23.479      0.000      -0.034      -0.028
exachkType        -0.0254      0.003     -9.603      0.000      -0.031      -0.020
exachkTypeInHtml   0.0326      0.002     20.763      0.000       0.030       0.036
NodeName           0.0025      0.001      1.849      0.064      -0.000       0.005
==============================================================================
Omnibus:                    13144.761   Durbin-Watson:                   0.816
```

Prob(Omnibus):	0.000	Jarque-Bera (JB):	39151.803
Skew:	-2.187	Prob(JB):	0.00
Kurtosis:	6.044	Cond. No.	52.5

Los resultados mostrados en la parte superior dan cuenta de la prueba de bondad de ajuste realizada. Se observa que el valor de F probabilístico es prácticamente cero, con lo que se rechaza la hipótesis nula. Las probabilidades T de las variables regresoras, asimismo, son cero, por lo que es posible aventurar que son independientes entre sí -excepto una- y que aportan de manera potencialmente significativa al modelo. El valor de r^2 obtenido en el resumen del modelo no es nada significativo, lo que casa con lo antes mencionado sobre la falta de linealidad de los datos, que se puede observar de forma más detallada en la figura 11.

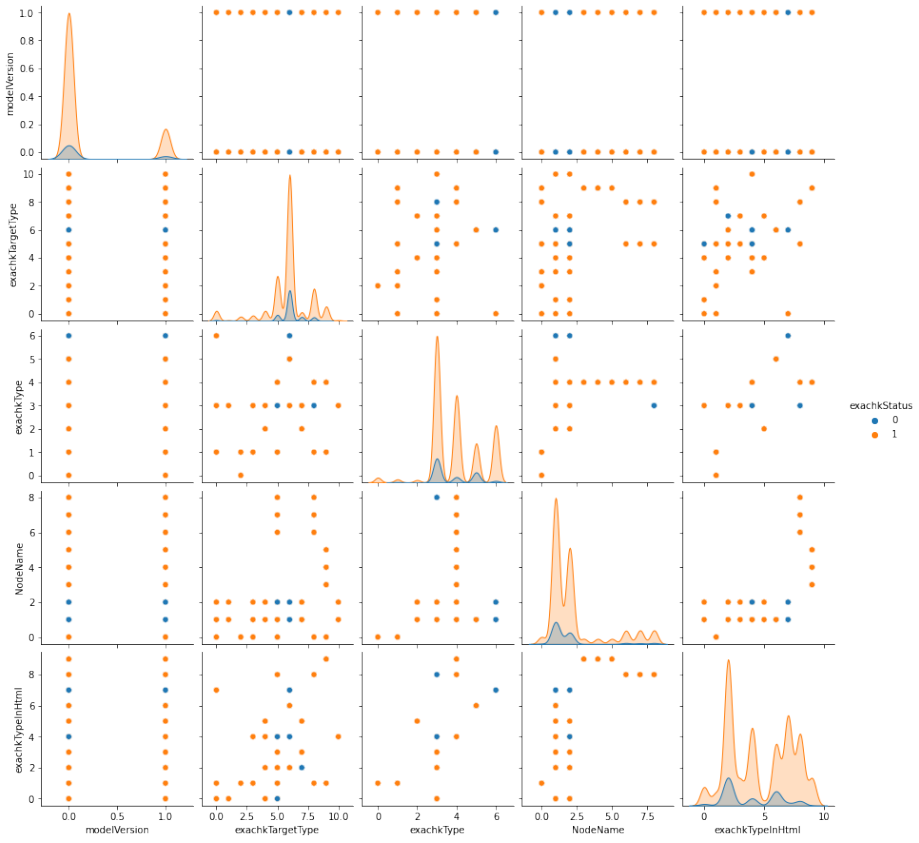


Figura 11: Gráficas par a par realizadas sobre el conjunto sanitizado de datos. Elaboración propia.

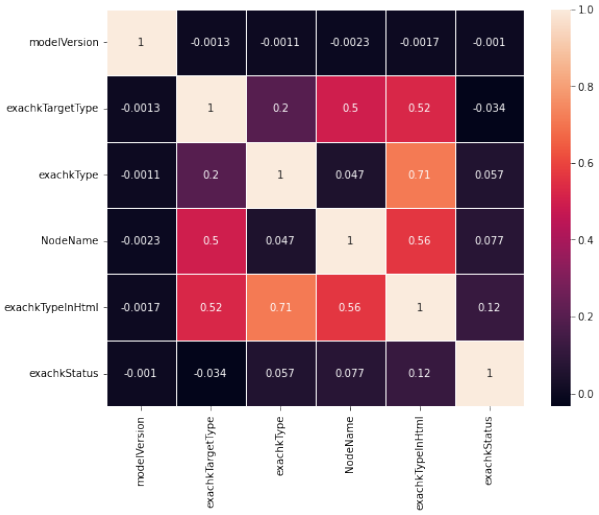


Figura 12: Mapa de calor que ilustra la correlación de Pearson entre las variables. Elaboración propia.

Como se mencionó anteriormente, hay métricas que ayudan a comprobar que los datos no siguen una distribución normal, -en virtud de que no siempre una primera impresión de los histogramas es prueba de ello-, y ayudan a visualizar mejor su comportamiento. *A priori*, conviene recordar las gráficas presentadas en la fase de exploración de datos de este documento, en el que se muestra que los datos se concentran principalmente en los extremos de las gráficas presentadas. Si se grafica su distribución, se observa lo siguiente:

A partir de estas gráficas, puede suponerse, entonces, que los datos sí están distribuidos en dos puntos principales, ubicados en los extremos y no son, de ningún modo, normales. Sin embargo, se realizaron también otras pruebas estadísticas para comprobar esto, obteniendo los siguientes valores:

Cuadro 3: Pruebas de normalidad

Prueba	r
Interpretación	
Normalidad de residuos	0.000
Al ser menor a 0.05, indica que los datos no provienen de una distribución normal.	
Heterocedasticidad de los residuos / varianza constante de los residuos	1.282e-84
Al ser menor a 0.05, indica que los residuos no tienen una varianza constante.	

Puede concluirse entonces que el modelo propuesto no tendrá, claramente, un buen desempeño y que lo que se observaba en principio al explorar los datos sí los describe en la realidad. La falta de normalización y de linealidad hace que

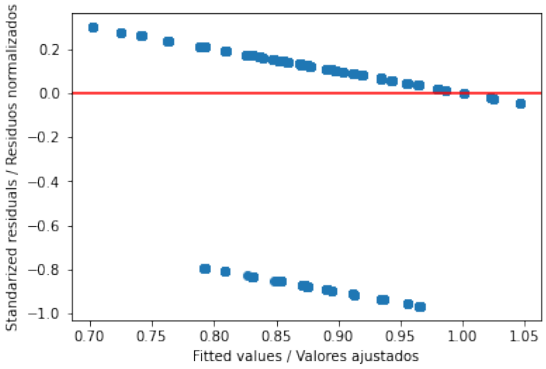


Figura 13: Gráfico de residuos normalizados vs. valores ajustados. Elaboración propia.

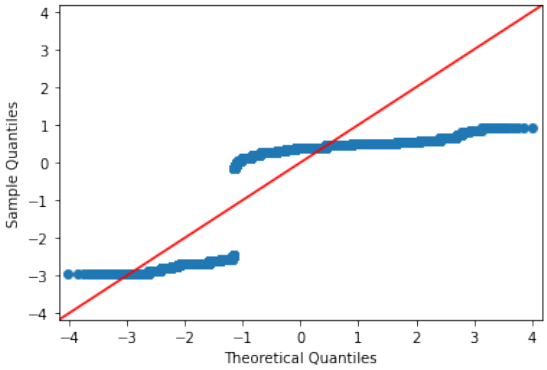


Figura 14: Gráfica cuantil - cuantil. Elaboración propia.

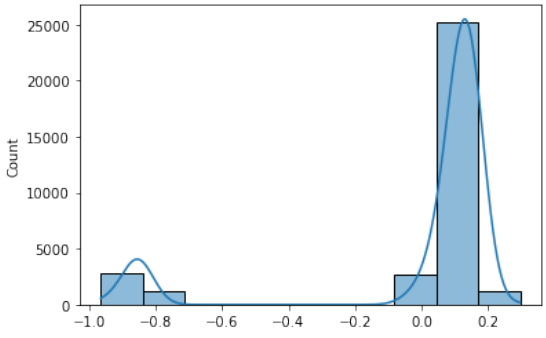


Figura 15: Histograma de residuos. Elaboración propia.

éste no funcione adecuadamente, algo que también se ve reflejado en el valor de r ajustado obtenido en el resumen del modelo. No obstante, se destaca el hecho de que este análisis también da un precedente sobre que, aunque las pruebas de correlación no muestren que existe una relación de esta índole entre las variables regresoras y la de salida, lo cierto es que numéricamente, los valores de f estadístico parecen mostrar que las variables sí pueden estar relacionadas. Si se construye el modelo sin incluir las variables correlacionadas entre sí -que convergen en `exachkTargetType` y `exachkTargetTypeInHtml`-, se obtiene lo siguiente:

OLS Regression Results

Dep. Variable:	exachkStatus	R-squared:	0.009
Model:	OLS	Adj. R-squared:	0.009
Method:	Least Squares	F-statistic:	98.08
Date:	Sun, 27 Nov 2022	Prob (F-statistic):	3.35e-63
Time:	00:47:37	Log-Likelihood:	-9945.5
No. Observations:	33092	AIC:	1.990e+04
Df Residuals:	33088	BIC:	1.993e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7940	0.006	125.581	0.000	0.782	0.806
modelVersion	-0.0007	0.005	-0.143	0.886	-0.010	0.009
exachkType	0.0145	0.001	9.813	0.000	0.012	0.017
NodeName	0.0122	0.001	13.588	0.000	0.010	0.014

Omnibus:	13679.148	Durbin-Watson:	0.825
Prob(Omnibus):	0.000	Jarque-Bera (JB):	42446.933
Skew:	-2.264	Prob(JB):	0.00
Kurtosis:	6.206	Cond. No.	17.5

Demostrando que aún pueden realizarse más pruebas en trabajos futuros, quizá incluso con más datos disponibles, para comprobar la correlación real que existe entre las variables, pues el modelo empeora al quitar algunas de las variables regresoras, incluso si éstas están correlacionadas entre sí. Adicionalmente, plantea retos adicionales de balanceo de datos que se discutirán en profundidad en el siguiente capítulo y algunos hallazgos interesantes relacionados con los puntos de influencia, que conducen a la eliminación de datos atípicos y, que, en este ejercicio, también resultan en que el modelo tenga, incluso, un peor desempeño que el original.

Cabe recordar en este punto que se ha realizado también un modelo simple de índole *K-Nearest Neighbors*, que basa su funcionamiento en premisas geométricas, obteniendo un puntaje de *accuracy* o precisión de 0.882, medido en función de su desempeño con un conjunto de datos de prueba no sanitizado del mismo

tamaño que el que se describió en capítulos previos. Dicho desempeño es visiblemente mejor que el del modelo de regresión, algo probablemente relacionado con la forma en la que se distribuyen los datos al graficarse.

5. Construcción e implementación de modelos de aprendizaje profundo

En el apartado anterior se pudo dar cuenta de la implementación de un modelo de clasificación usando regresión con múltiples variables. Fue evidente que su formulación, si bien sirve como un buen punto de partida para entender el problema y los datos con los que se cuenta, no siempre puede resultar tan eficaz como se desearía.

Debido a esto, surge la necesidad de sugerir métodos aún más sofisticados de aprendizaje de máquina, que permitan la formulación de un modelo matemático que resuelva el problema y permita, en este caso concreto, hacer predicciones sobre los estados de error del sistema. Para ilustrar este apartado, se implementó una red neuronal convolucional de entre tres y cuatro capas, incluyendo una o dos ocultas, que permite ajustar una función matemática mediante retropropagación, con resultados mucho mejores a los obtenidos mediante modelos de regresión. De manera distinta a como se abstrajo el modelo estadístico, en esta implementación se realizaron cuatro sub-implementaciones, que contemplan más o menos capas y variables de salida dicotómicas y múltiples, con el objetivo de contrastar el desempeño de todas ellas y decidir cuáles son los parámetros más adecuados para el problema presentado. Dado que este tipo de redes reciben datos en forma de matrices o tensores, se realizó una pequeña implementación de `reshape` mediante la librería `NumPy`, que permitió ingresar los datos dentro de la red.

En la construcción de estos modelos, se usó el ya popular y reputado framework `TensorFlow-Keras` [5], que permite la creación de este tipo de redes neuronales mediante Python. Se dividieron los datos en conjuntos de entrenamiento, prueba y validación, en los siguientes porcentajes: 85 para los datos de validación y prueba en x y y y 15 para los datos de entrenamiento y validación en x y y. Los cuatro modelos que se construyeron en la primera iteración cuentan con los siguientes parámetros:

- Variable de salida dicotómica (`PASS` y `FAIL`), una sola capa Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación sigmoidal. Se compila con una función de pérdida `binary_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal -entendiéndose ésta como un porcentaje de las observaciones, tanto positivas como negativas, que fueron clasificadas correctamente-. 171 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 32:
- Variable de salida dicotómica (`PASS` y `FAIL`), dos capas Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación sigmoidal. Se compila con una función de pérdida

`binary_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal. 171 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 32:
- Variable de salida múltiple (`PASS`, `INFO`, `WARNING`, `CRITICAL` y `FAIL`), una sola capa Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación `softmax`. Se compila con una función de pérdida `categorical_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal. 61 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 64:
- Variable de salida múltiple (`PASS`, `INFO`, `WARNING`, `CRITICAL` y `FAIL`), dos capas Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación `softmax`. Se compila con una función de pérdida `categorical_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal. 61 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 64:

Luego de tal iteración, se observaron los resultados y se realizaron los siguientes cambios para la siguiente:

- Implementación de *early stopping* para evitar sobre entrenamiento del modelo.
- Eliminación definitiva de capas adicionales, que sólo entorpecen el desempeño.
- Implementación de técnicas de regularización *ridge* y *lasso* para asegurar un buen desempeño a través de las épocas de los conjuntos de validación y prueba.

En la figura 17, se muestra el contraste de pérdida y exactitud de los modelos realizados. A continuación, se enlistan algunos detalles observados que permiten entender mejor qué sucedió, cuál modelo podría funcionar mejor y por qué y, luego, se mencionará cuál podría ser el trabajo futuro para mejorar el modelo:

- En cuestión de modelos convolucionales, lo más complejo no siempre es lo más recomendable. Los gráficos y las matrices de confusión obtenidas muestran que aquellos modelos con más de una capa oculta funcionan mucho peor que aquellos que sólo incluyen una. Evidentemente, las redes neuronales deberán modelarse en función del problema y los datos con los que se cuenta. Si bien existen modelos sumamente profundos, modelos pre entrenados y otras herramientas más de esta índole, en el caso de nuestros datos parece ser que lo más simple es lo más funcional. En realidad, el desempeño de la red neuronal, tomando en cuenta su exactitud como punto referencial, es bastante aceptable con una sola capa oculta y muy pocas épocas. La técnica de *early stopping* funcionó muy bien para evitar el sobre entrenamiento del modelo y el cómputo innecesario de épocas que, además, no mejoraban nada el desempeño.
- Hablando de los datos que se poseen, es sumamente complejo establecer si el análisis realizado podría ser útil. Si bien la cantidad de registros no es

tan pequeña, el desbalanceo de clases puede resultar un problema muy serio. No obstante, aplicar técnicas de *data augmenting* puede ser un arma de doble filo, pues, en realidad, el modelado del fenómeno es bastante acertado. En la realidad, los sistemas de Exadata suelen tener un número muy bajo de errores y un catálogo extensísimo de chequeos con estatus usual de `PASS`, como se ha observado gracias a la herramienta basada en `CLI` de uso interno. Tratar de nivelar los datos es, entonces, un poco incoherente con el fenómeno *per se*, por lo que, en adelante, el trabajo futuro consistirá en recabar más datos y en buscar una técnica que permita mejorar el desempeño del modelo, basándose en el tratamiento de los datos *a priori*.

- Siguiendo el hilo del punto anterior, los modelos dicotómicos se comportan de manera más adecuada y resuelven mejor el problema que aquellos que cuentan con varias salidas, una diferente para cada tipo de error. Tomemos el ejemplo de un `exachkStatus` de `CRITICAL`. Como ya se ha mencionado previamente, sólo los chequeos del switch pueden encontrarse en este estado, un porcentaje muy bajo del total de *check* con los que se cuenta. Al modelar con varias salidas, en casi todas las ocasiones, excepto en un caso, la variable se pierde. Es decir, no es sólo que en la matriz de confusión termine en ceros -se ejemplifican en la parte inferior dos de las matrices de confusión obtenidas, tanto con salidas dicotómicas como múltiples-, sino que la red termina eliminándola del análisis al recomputar sus parámetros mediante retropropagación. Por ende, se considerará mejor el modelo de dos salidas por, además, contar con mejor desempeño.

Matriz de confusión de una salida múltiple.
Se observa que la matriz termina en ceros para precision y recall en varias salidas. Nótese que los dígitos corresponden a la etiqueta de datos para cada clase codificada con el encoder, con dos salidas desaparecidas en la matriz, 0 y 4:

	precision	recall	f1-score	support
1	0.06	1.00	0.11	296
2	0.00	0.00	0.00	181
3	0.00	0.00	0.00	4374
accuracy			0.06	4964
macro avg	0.01	0.20	0.02	4964
weighted avg	0.00	0.06	0.01	4964

Matriz de confusión de una salida dicotómica.
Se observa que la matriz termina en ceros en precision y recall para FAIL.

	precision	recall	f1-score	support

FAIL	0.00	0.00	0.00	631
PASS	0.87	1.00	0.93	4333
accuracy			0.89	4964
macro avg	0.44	0.50	0.47	4964
weighted avg	0.76	0.87	0.81	4964

6. Resultados

Después de la implementación de tres modelos: una regresión de múltiples variables, un *K-Nearest Neighbors* y una red neuronal convolucional, se obtuvieron los siguientes valores para R cuadrada / exactitud en cada uno, en su implementación con mejoras añadidas, cambios de parámetros y demás elementos descritos en párrafos previos y luego de probarlo en un conjunto de datos que toma parte de los datos sanitizados y no sanitizados de uso interno. Se usará este valor como referencia para poder comparar su desempeño, en vista de que son modelos de implementaciones y fundamentos diferentes:

Cuadro 4: Contraste de desempeño de modelos

Variable regresora	**Exactitud de predicción**
Regresión multivariada	0.031
K-Nearest Neighbors con siete vecinos	0.882
Red Neuronal Convolucional CNN (dicotómica, una sola capa oculta)	0.888

Gracias a esta comparación, así como a los resultados del modelo de Red Neuronal Convolucional en términos de precisión y *recall* de precisión, -cuyos valores finales son 0.87 y 1, respectivamente- se deduce que la implementación de CNN es la más apropiada para resolver el problema, pues es capaz de predecir con una exactitud aceptable el resultado. Una de sus fallas mayores está en la medida de sensitividad (0.13), dado que el desbalanceo de datos provoca una pérdida muy importante en este rubro. No obstante, una de las ventajas que ofrece es la capacidad de aprendizaje y retropropagación, que hará que se pueda continuar optimizando si se añaden datos nuevos al set de entrenamiento.

7. Conclusiones

A manera de conclusión, se resalta la enorme influencia y poder del análisis de tendencias de datos en el rumbo de las grandes compañías. Los intereses del mundo entero están puestas en aquello que los científicos de datos pasan el día

completo analizando. Se trata de una labor que requiere paciencia, perspicacia e, incluso, un poco de audacia: atreverse a formular hipótesis, incluso en ocasiones luchando con lo que la propia literatura dice, para poder llegar al resultado esperado. El estado del arte de la minería de datos se transforma, evoluciona constantemente. Del curso académico se aprendió el enorme valor del tiempo: de que no existe un único método para llegar a una solución y de que sólo se requiere tener cierta experiencia para lograr construir modelos cada vez más certeros por medio de cambios en los parámetros que, muchas veces, incluso carecen de fundamento, pero, increíblemente, funcionan. La implementaciones aquí descritas constituyen un fundamento sólido en materia de descripción y entendimiento de los datos y medidas de desempeño y, si bien no resuelven el problema aún, sí pueden aumentar el banco de conocimiento con el que se trabaja para seguir proponiendo modelos.

8. Trabajo futuro

Entre las cuestiones más interesantes de este caso de estudio, se encuentra el hecho de que, en realidad, lo aquí expuesto es sólo parte de una pequeña primera iteración independiente e individual realizada por la autora del documento. El proyecto *per se* tiene aún futuro y posibilidad de ampliar horizontes. La herramienta analizada forma parte de una suite de software líder y, por ende, inmensa, que seguramente esconde aún muchos secretos por descubrir. ¿Qué es la minería de datos sino el mero interés de encontrar joyas en el medio de cantidades ingentes de información? Definitivamente, todo lo que queda aún por descubrir y documentar resultará interesante y, probablemente, ayude a mejorar en demasía lo aquí mostrado. Adicionalmente, los datos siguen produciéndose en algún lugar, el software evoluciona y el proyecto avanza. Muchas de las implementaciones sugeridas sólo requieren de un poco más de datos o de mentes abiertas dispuestas a aportar ideas nuevas y es lo que se espera de esto en los próximos meses y años.

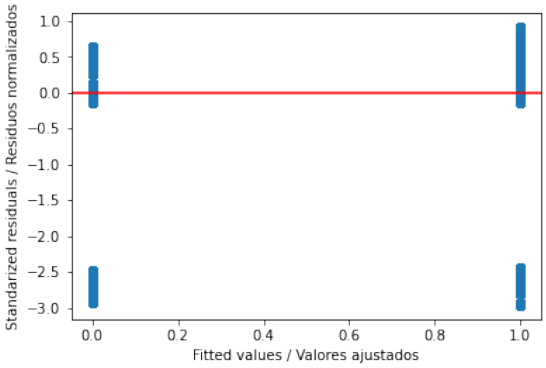


Figura 16: Gráfica de residuos contra valores ajustados (validación cruzada). Elaboración propia.

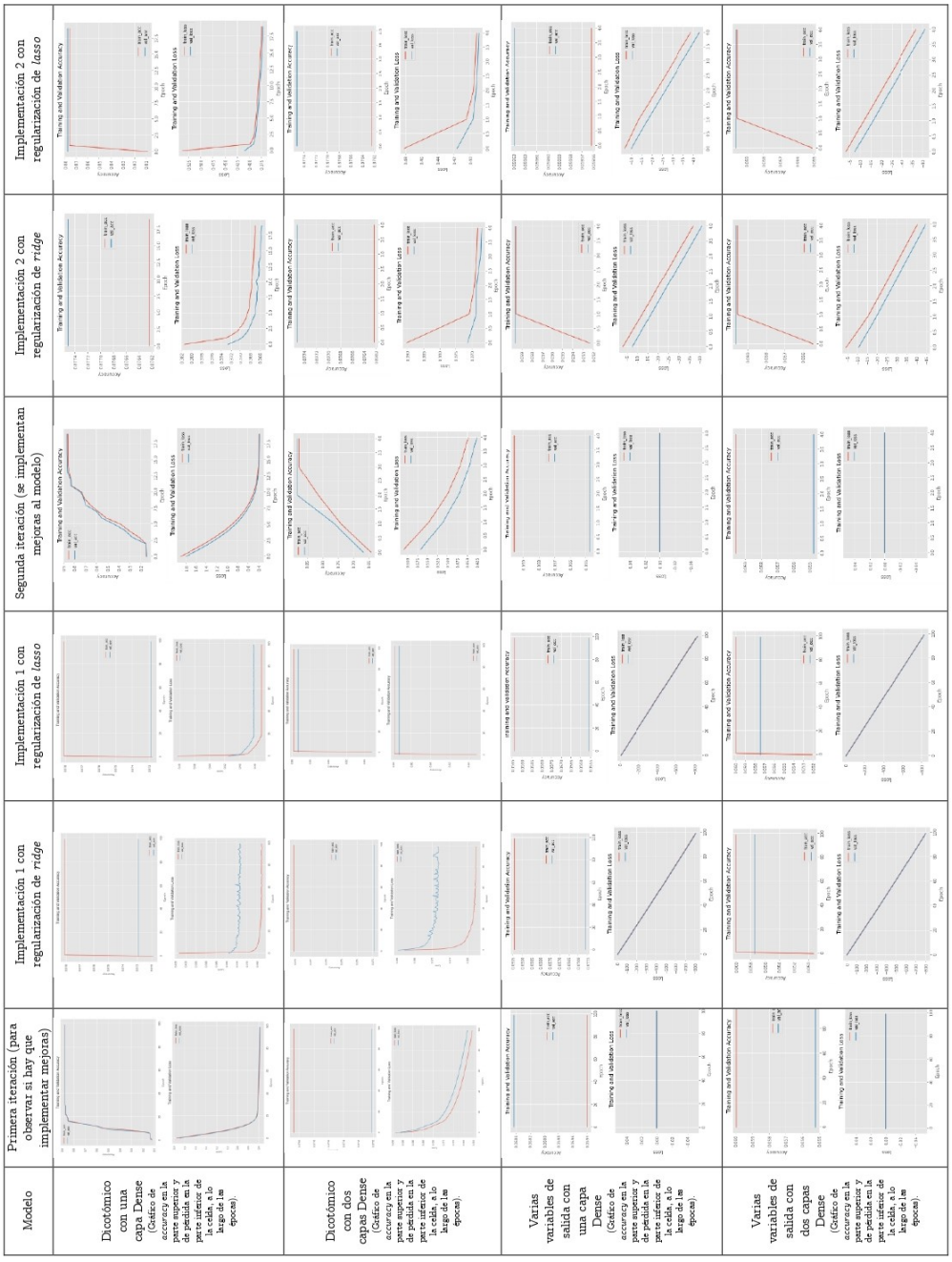


Figura 17: Tabla de contraste de desempeño de modelos de CNN. Elaboración propia.

Bibliografía

[1] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. & Wirth, R. CRISP-DM 1.0: Step-by-step data mining guide. (2000)

[2] Harris, C., Millman, K., Walt, S., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M., Brett, M., Haldane, A., Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. Array programming with NumPy. *Nature*. **585**, 357-362 (2020,9), `https://doi.org/10.1038/s41586-020-2649-2`

[3] Hunter, J. Matplotlib: A 2D Graphics Environment. *Computing In Science & Engineering*. **9**, 90-95 (2007)

[4] Attariyan, M. & Flinn, J. Using Causality to Diagnose Configuration Bugs.. (2008,1)

[5] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. & Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. *Proceedings Of The 12th USENIX Conference On Operating Systems Design And Implementation*. pp. 265-283 (2016)

[6] Kumar, Y., Basha, N., M, K., Sharma, B. & Kerekovski, K. Oracle High Availability, Disaster Recovery, and Cloud Services: Explore RAC, Data Guard, and Cloud Technology. (Apress,2019), `https://books.google.com.mx/books?id=GOWYDwAAQBAJ`

[7] Pane, S. Being proactive and keeping your databases healthy. *Database Trends & Applications*. **32**, 40 (2018), `https://link.gale.com/apps/doc/A558368471/AONE?u=googlescholar&sid=sitemap&xid=0775bd13`

[8] Virtanen, P., Gommers, R., Oliphant, T., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Van der Walt, S., Brett, M., Wilson, J., Millman, K., Mayorov, N., Nelson, A., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E., Harris, C., Archibald, A., Ribeiro, A., Pedregosa, F., Van Mulbregt, P. & SciPy 1.0 Contributors SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. **17** pp. 261-272 (2020)

[9] Zhang, S. & Ernst, M. Automated diagnosis of software configuration errors. *ICSE 2013, Proceedings Of The 35th International Conference On Software Engineering*. pp. 312-321 (2013,5)

[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. & Others Scikit-learn: Machine learning in Python. *Journal Of Machine Learning Research*. **12**, 2825-2830 (2011)

[11] Seabold, S. & Perktold, J. statsmodels: Econometric and statistical modeling with python. *9th Python In Science Conference*. (2010)

[12] Team, T. pandas-dev/pandas: Pandas. (Zenodo,2020,2), https://doi.org/10.5281/zenodo.3509134

[13] Waskom, M. seaborn: statistical data visualization. *Journal Of Open Source Software*. **6**, 3021 (2021), https://doi.org/10.21105/joss.03021

[14] Oracle Oracle Engineered Systems. (2016), https://www.oracle.com/mx/engineered-systems/

[15] Oracle Oracle Enterprise Manager ORAchk Healthchecks Plug-in User's Guide, 13.2.1.0. (2016), https://docs.oracle.com/cd/E73210_01/PICHK/PICHK.pdf

[16] Oracle Oracle Exadata. (2016), https://www.oracle.com/mx/engineered-systems/exadata/

[17] Oracle Database Concepts: Introduction to Oracle Database. (2016), https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm#CNCPT001

[18] Oracle Ejecución de comprobaciones de estado de orachk (CLI). (2016), https://docs.oracle.com/cd/E78252_01/html/E78266/gqteq.html

[19] Oracle Installation Guide for HP-UX Itanium: Downloading and Installing the ORAchk Health Check Too. (2016), https://docs.oracle.com/en/database/oracle/oracle-database/12.2/hpdbi/downloading-and-installing-the-orachk-health-check-tool.html#GUID-19914928-49B2-444A-8F1B-D4398C264AAD

[20] Oracle Oracle Exadata Database Machine Technical Architecture. (2016), https://docs.oracle.com/en/engineered-systems/exadata-database-machine/edbid/

[21] Oracle Real Application Clusters Administration and Deployment Guide: Oracle RAC Configuration Audit Tool. (2016), https://docs.oracle.com/cd/E11882_01/rac.112/e41960/racchk.htm#RACAD8365