Realizado por:
Luis Daniel Beltrán Rodríguez
Diego Alejandro Caballero Rincón
Johan Camilo Lasso Figueroa

Predicción de comportamiento del Dólar

Introducción

Este Proyecto se ha realizado con el fin de predecir el movimiento del dólar respecto a los datos conocidos de históricos pasados, para esto se utilizan tres históricos, entre estos el histórico del cambio de peso dólar, el de Nasdaq y de la relación del petróleo, a través de dos métodos distintos de regresión se comparan para conocer cuál se aproxima más al dato real.

Objetivos

A través de los siguientes objetivos se busca llegar al desarrollo total del proyecto.

Objetivo general

Desarrollar diferentes algoritmos de los vistos en clase para lograr hacer una predicción a tiempo futuro sobre el dólar.

Objetivos específicos

- Manejo de datasets para procesamiento de estos.
- Realizar la división del dataframe en conjunto de entrenamiento y de prueba.
- Realizar regresión con los datos obtenidos.
- Realizar el pronóstico de un valor a futuro del dólar mediante series de tiempo.
- Comparar los métodos utilizados y el dato real para corroborar cuál es el método más acertado.

Marco Teórico

Una serie temporal es un conjunto de muestras tomadas a intervalos de tiempo regulares. Se puede analizar su comportamiento a mediano y largo plazo, esto intentando detectar patrones y poder así realizar pronósticos de cómo será su comportamiento futuro. [1]

El pronóstico de las series de tiempo significa que extendemos los valores históricos al futuro, donde aún no hay mediciones disponibles. El pronóstico se realiza generalmente para optimizar áreas como los niveles de inventario, la capacidad de producción o los niveles de personal.

Existen dos variables estructurales principales que definen un pronóstico de serie de tiempo:

El período, que representa el nivel de agregación. Los períodos más comunes son meses, semanas y días en la cadena de suministro (para la optimización del inventario). Los centros de atención telefónica utilizan períodos de cuartos de hora (para la optimización del personal).

El horizonte, que representa la cantidad de períodos por adelantado que deben ser pronosticados. En la cadena de suministro, el horizonte es generalmente igual o mayor que el tiempo de entrega.[2]

Desarrollo

Para hacer nuestra implementación utilizamos la herramienta de Google colab, la cual puede acceder mediante este enlace: https://colab.research.google.com/

Implementación Series de tiempo:

Para este primer método descargamos el dataset del precio histórico del dólar en el siguiente enlace:

https://es.investing.com/currencies/usd-cop-historical-data

Los datos corresponden a un periodo de un año empezando el día 17 de noviembre de 2019 y terminando el 17 de noviembre de 2020.

Ilustración 1. Fechas seleccionadas.

El dataset original contiene puntos para separar las unidades de mil y comas para los decimales, por lo cual se tienen que adecuar para ser trabajados en colab.

Plazo: Diario | Descargar datos | 17/11/2019 - 17/11/2020

Fecha	Último	Apertura	Máximo	Mínimo	% var.
17.11.2020	3.642,50	3.644,40	3.652,90	3.617,65	0,19%
16.11.2020	3.635,65	3.643,40	3.648,17	3.630,28	-0,14%
13.11.2020	3.640,90	3.644,05	3.651,00	3.630,68	-0,02%
12.11.2020	3.641,55	3.639,00	3.667,45	3.632,00	0,21%
11.11.2020	3.634,00	3.631,50	3.646,47	3.623,81	0,17%
10.11.2020	3.627,75	3.649,21	3.673,50	3.625,90	-0,53%

Ilustración 2. Demostración de los datos

```
[ ]  #Librerias necesarias
     import pandas as pd
     import numpy as np
     %matplotlib inline
     import matplotlib.pyplot as plt
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.metrics import r2_score
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.callbacks import EarlyStopping
     from keras.optimizers import Adam
     from keras.layers import LSTM
     import sys
     import cv2 as cv
     from google.colab.patches import cv2_imshow
```

Se carga el Data frame ya modificado:

```
[ ]  # Se importa los datos
     df = pd.read_csv('dolar.csv',sep=';', header=0)
```

```
     #el tratamiento previo se realizo directamente en el csv
     print(df.head(10))
```

```
   Fecha   Ultimo  Apertura   Maximo   Minimo   var  Siguiente
0  44152  3644.50   3644.40  3652.90  3617.80  0.24    3643.40
1  44151  3635.65   3643.40  3648.17  3630.28  0.14    3644.50
2  44148  3640.90   3644.05  3651.00  3630.68  0.02    3635.65
3  44147  3641.55   3639.00  3667.45  3632.00   2.1    3640.90
4  44146  3634.00   3631.50  3646.47  3623.81  1..7    3641.55
5  44145  3627.75   3649.21  3673.50  3625.90  0.53    3634.00
6  44144  3647.08   3723.00  3723.93  3622.50  1.98    3627.75
7  44141  3720.75   3754.00  3775.00  3719.15  0.83    3647.08
8  44140  3751.75   3815.82  3819.03  3752.00  1.62    3720.75
9  44139  3813.57   3819.50  3825.50  3787.00   0.1    3751.75
```

Graficando los datos de *train* y *test*:

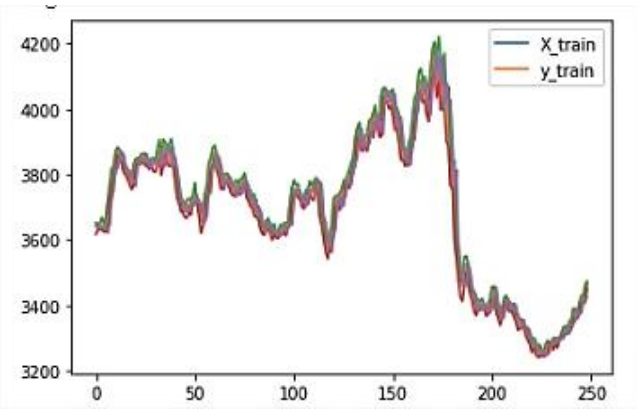


Ilustración 3. Gráfica de train.

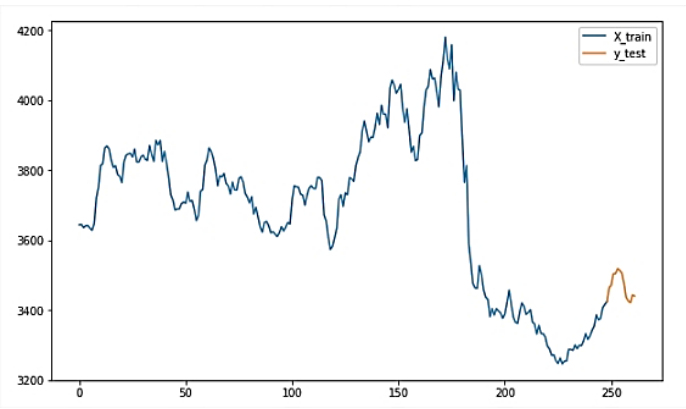
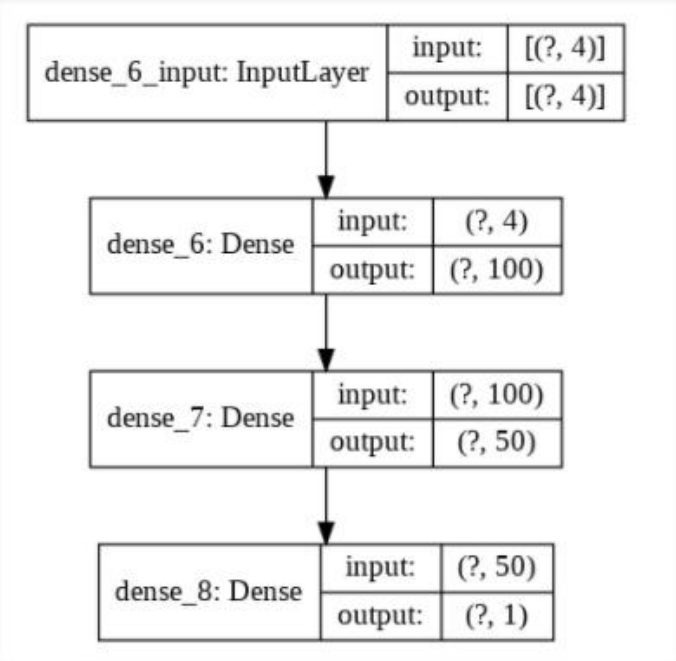


Ilustración 4. Gráfica de test.

Después de normalizar los datos se procede a diseñar el modelo de predicción con una capa de entrada, una de salida y dos capas internas.



Al hacer R cuadrado al modelo se visualiza que el valor predicho estaría un poco alejado de la realidad.

```
El R2 en el conjunto de train es:       -9.519
El R2 en el conjunto de test es:        -657.001
```

Para comprobar lo anterior se gráfica la siguiente comparación:

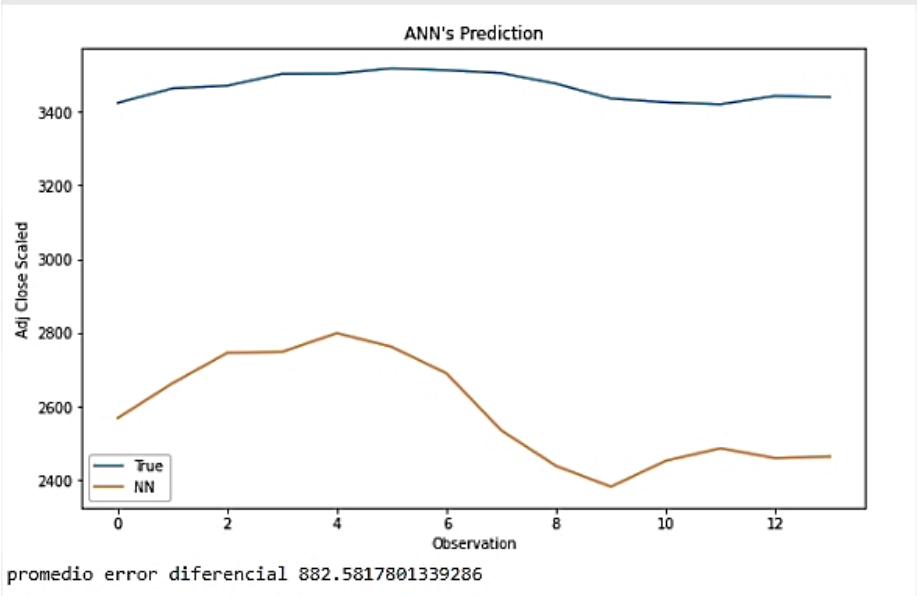


Ilustración 5. Predicción ANN, comparación true vs NN

De la siguiente forma se calcula la predicción del modelo de series de tiempo simple:

```
[ ]  #Hoy: Fecha      Último   Apertura  Máximo  Mínimo
     #Hoy: 17.11.2020 3644.50  3643.40   3648.17 3644.50
     XHoy=np.array([[3644.50,3643.40,3648.17,3644.50]])
     predicion_nueva = nn_model.predict(scaler.transform(XHoy))
     print("Valor de mañana: ",predicion_nueva[0][0])
```

ANN 8 entradas:

Para la segunda parte de series de tiempo, añadimos un intervalo de tiempo más por filas.

La preparación del data set se hizo por medio de la herramienta Excel.

```
#el tratamiento previo se realizo directamente en el csv
print(df.head(10))
```

```
   Fecha   Ultimo  Apertura   Maximo  ...  Maximo3  Minimo4   var5  Siguiente
0  44152  3644.50   3644.40  3652.90  ...  3648.17  3630.28   0.14    3643.40
1  44151  3635.65   3643.40  3648.17  ...  3652.90  3617.80   0.24    3644.50
2  44148  3640.90   3644.05  3651.00  ...  3648.17  3630.28   0.14    3635.65
3  44147  3641.55   3639.00  3667.45  ...  3651.00  3630.68   0.02    3640.90
4  44146  3634.00   3631.50  3646.47  ...  3667.45  3632.00    2.1    3641.55
5  44145  3627.75   3649.21  3673.50  ...  3646.47  3623.81   1..7    3634.00
6  44144  3647.08   3723.00  3723.93  ...  3673.50  3625.90   0.53    3627.75
7  44141  3720.75   3754.00  3775.00  ...  3723.93  3622.50   1.98    3647.08
8  44140  3751.75   3815.82  3819.03  ...  3775.00  3719.15   0.83    3720.75
9  44139  3813.57   3819.50  3825.50  ...  3819.03  3752.00   1.62    3751.75

[10 rows x 12 columns]
```

Graficando lo datos de *train* y *test* para el nuevo data set:

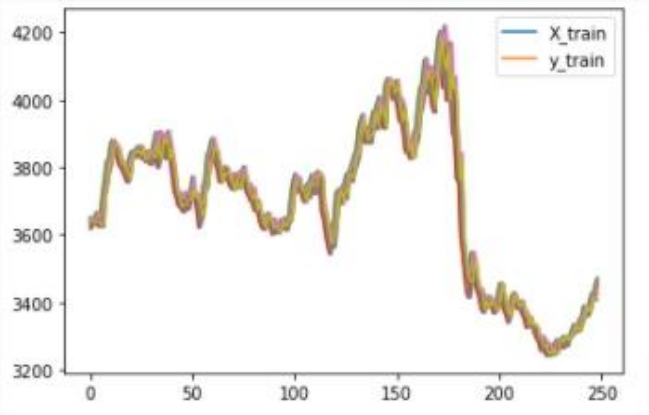


Ilustración 6. Datos train.

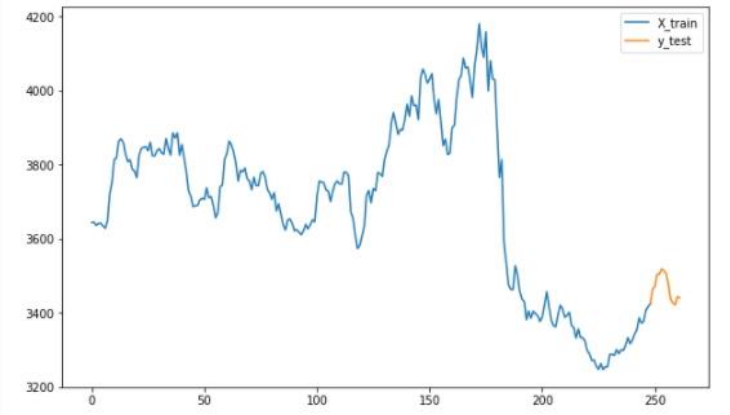
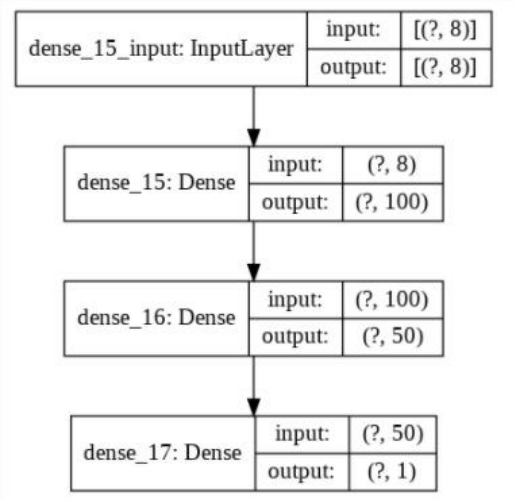


Ilustración 7. Datos test

Cómo se puede apreciar este modelo duplica el número de entradas en la primera capa:



El coeficiente de terminación o R cuadrado para este modelo mejora considerablemente:

```
El R2 en el conjunto de train es:       0.986
El R2 en el conjunto de test es:        0.791
```

Realizando la gráfica comparativa entre los reales y los predichos es la siguiente:

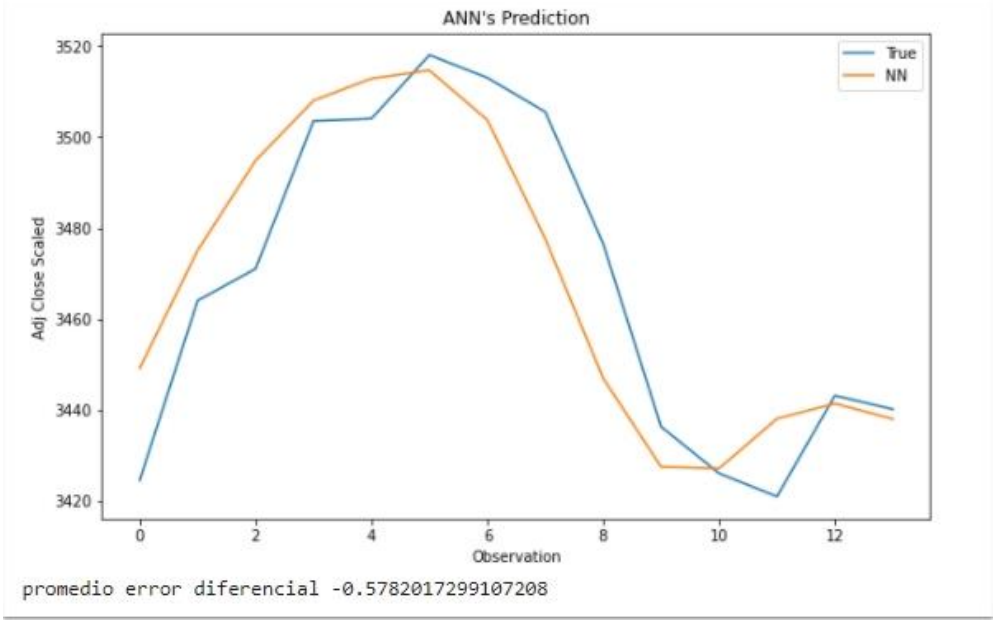


Ilustración 8. Nueva comparación True vs NN predicción ANN

El Código para realizar la predicción fue el mismo, las columnas que tienen un número a su final corresponde a los valores que toman esas variables en el periodo de tiempo siguiente

```
# hoy: 17.11.2020
#Fecha Ultimo','Apertura','Maximo','Minimo','Ultimo1','Apertura2','Maximo3','Minimo4'
XHoy=np.array([[3644.50,3643.40,3652.90, 3617.8,3635.65,3643.4,3648.17,3630.28]])
nueva_prediccion = nn_model.predict(scaler.transform(XHoy))
print("Valor de mañana: ",nueva_prediccion[0][0])
```

Implementación regresión[5]:

Para el segundo método descargamos el dataset del precio histórico del dólar en el siguiente enlace:

https://es.investing.com/currencies/usd-cop-historical-data

Los datos corresponden a un periodo de un año empezando el día 17 de noviembre de 2019 y terminando el 17 de noviembre de 2020.

```
[ ]  #librerias necesarias
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sb
     from scipy.spatial import distance
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import r2_score

     from sklearn import linear_model
```

```
[ ]  # Se importa los datos
     df = pd.read_csv('dolar.csv',sep=';', header=0)
     df1 = pd.read_csv('nasdaq.2.csv',sep=';', header=0)
     df2 = pd.read_csv('petroleo.csv',sep=';', header=0)

     #el tratamiento previo se realizo directamente en el csv
     print('dolar' "\n",df.head(10))
     print('nasdaq' "\n",df1.head(10))
     print('petroleo' "\n",df2.head(10))
```

Se realiza el debido preprocesamiento directamente a los CSV utilizados para la implementación, utilizando las diferentes librerías que brinda Sklearn para realizar las regresiones lineales.

Para el modelo de regresión de 11 entradas el valor que se predijo del dólar en pesos colombianos fue:

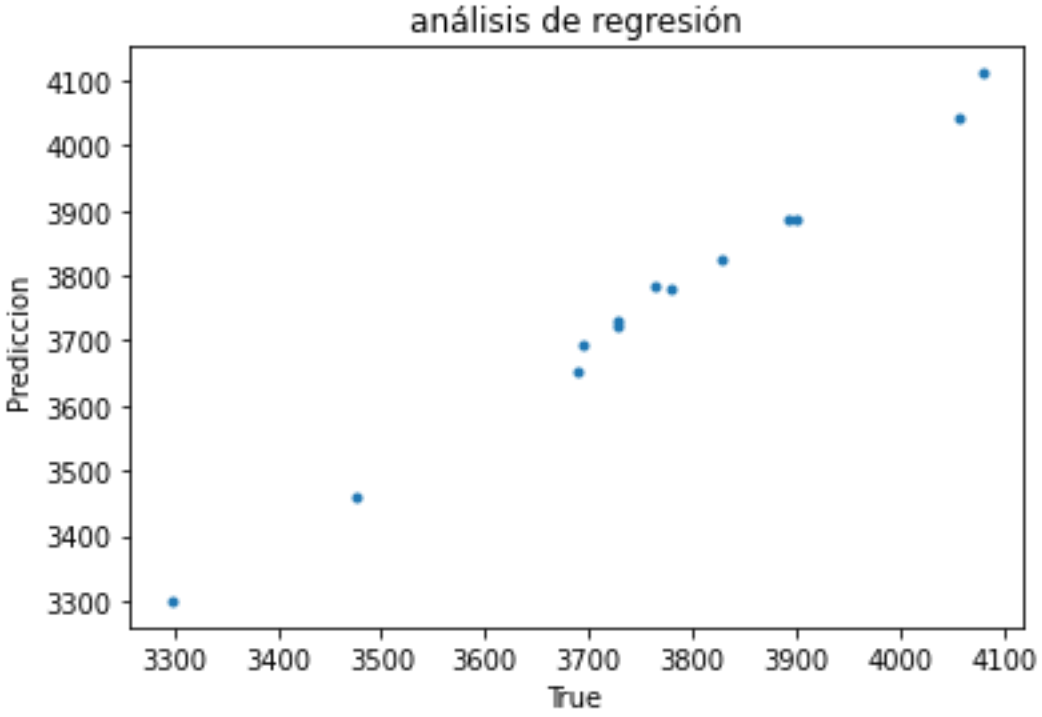


Ilustración 9. Lasso

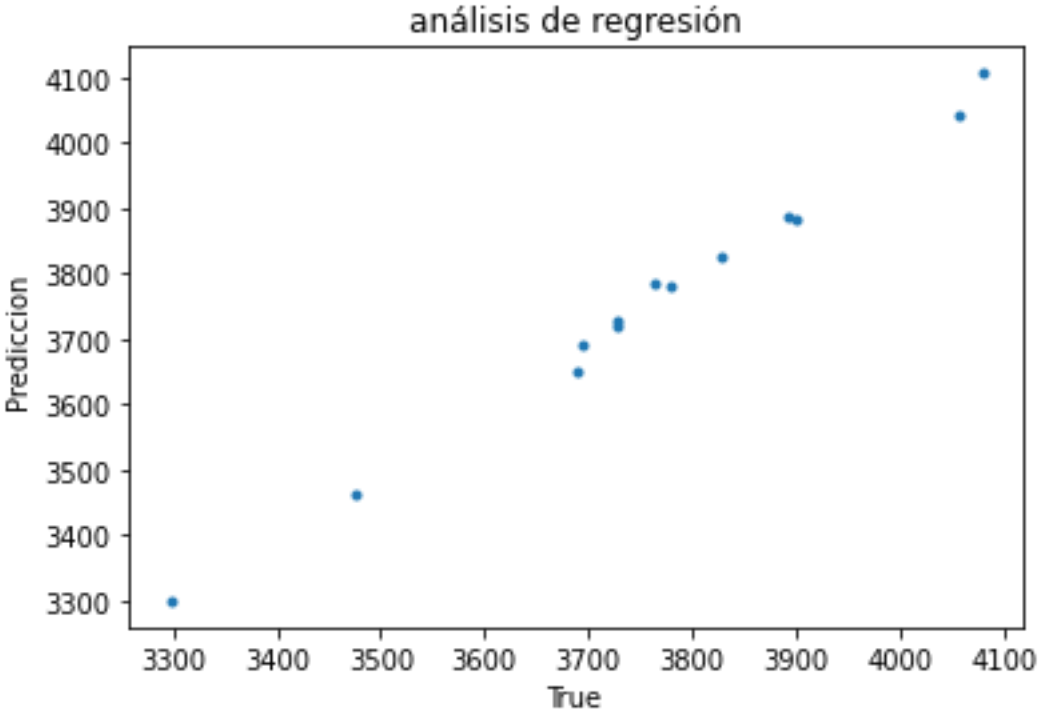


Ilustración 10. Ridge.

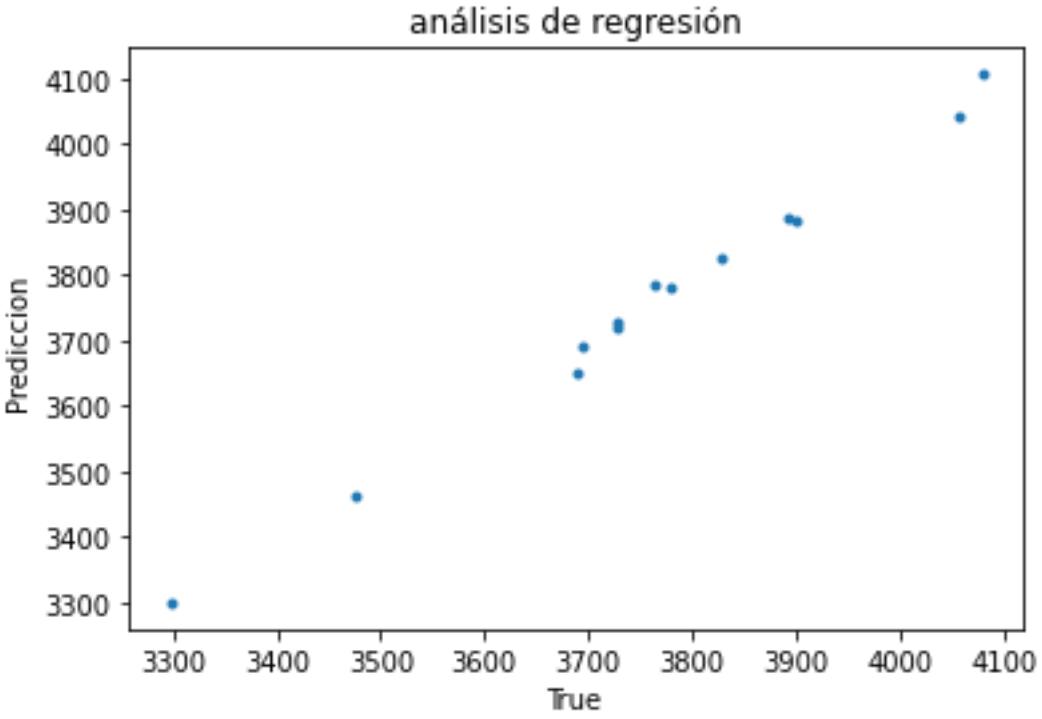


Ilustración 11. Lineal

Implementación donde solo se tiene entradas de apertura de los diferentes dataframe

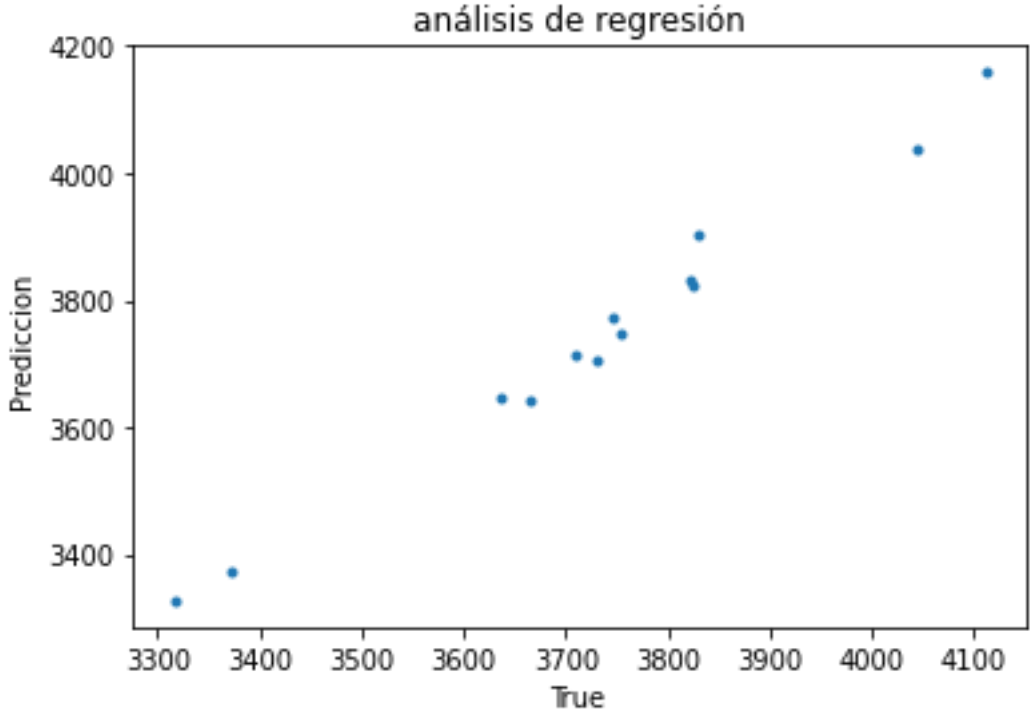


Ilustración 12. Lasso

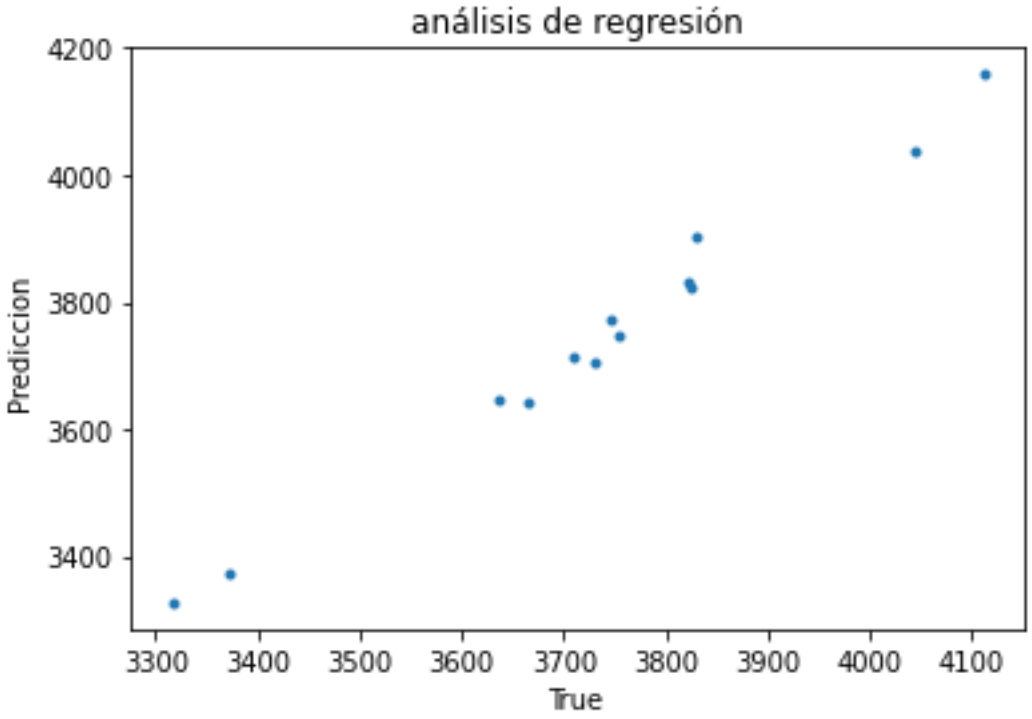


Ilustración 13. Ridge

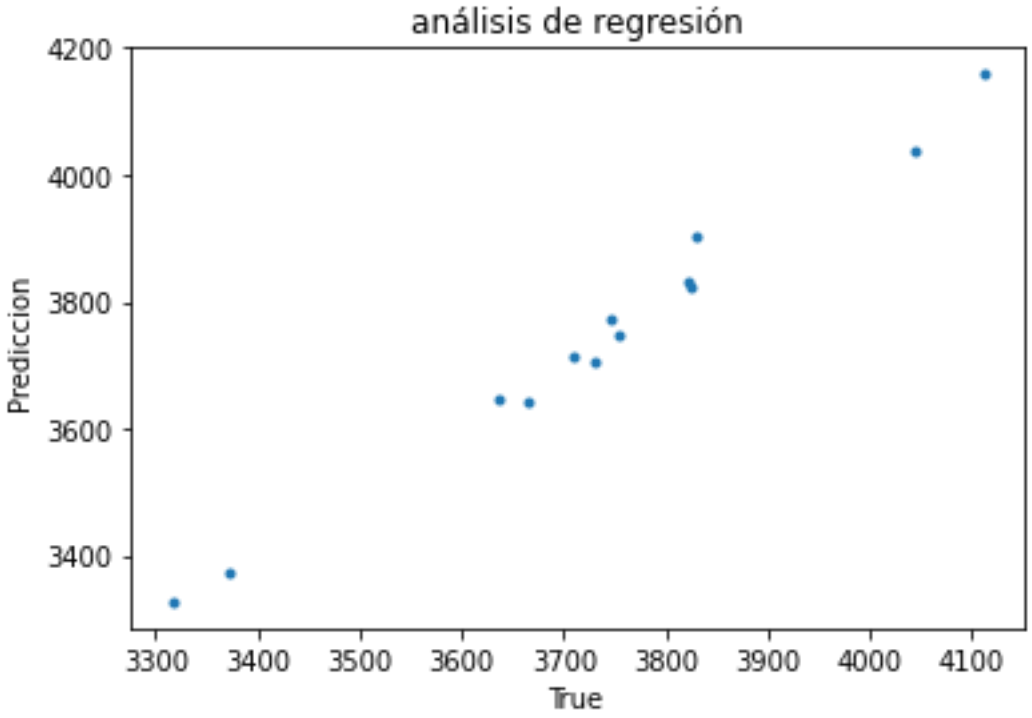


Ilustración 14. Lineal

Resultados

El valor esperado real que obtuvo el dólar el día 18 de noviembre según Dólar Colombia fue:

Por parte de nuestros dos modelos implementados en series de tiempo obtuvimos las siguientes predicciones:

Modelo simple de 4 entradas:

```
Valor de mañana:  3312.6519
```

Modelo ANN de 8 entradas:

```
Valor de mañana:  3639.3052
```

Error de exactitud:

Modelo simple de 4 entradas:

$$E\% = \left|\frac{3312.65 - 3635.19}{3635.19}\right| * 100 = 8.9\%$$

Modelo ANN de 8 entradas:

$$E\% = \left|\frac{3639.30 - 3635.19}{3635.19}\right| * 100 = 0{,}11\%$$

Modelo de Regresion

Modelo con 11 entradas:

```
Predicion_nueva con Lasso: [3631.5697128]
```

$$E\% = \left|\frac{3631.57 - 3635.19}{3635.19}\right| * 100 = 0{,}1\%$$

Predicion_nueva con Ridge: [3632.09433804]

$$E\% = \left|\frac{3632.09 - 3635.19}{3635.19}\right| * 100 = 0{,}085\%$$

Predicion_nueva con R lineal: [3632.1018824]

$$E\% = \left|\frac{3632.09 - 3635.19}{3635.19}\right| * 100 = 0{,}085\%$$

Modelo con entradas de apertura:

Predicion_nueva con Lasso: [3647.32665321]

Predicion_nueva con Ridge: [3647.3572383]

Predicion_nueva con R lineal: [3647.35776593]

$$E\% = \left|\frac{3647.35 - 3635.19}{3635.19}\right| * 100 = 0{,}33\%$$

Conclusiones

Los métodos lineales, Lasso y ridge utilizados para predecir el dólar realmente son bastante confiables según métricas como R2 y el accuracy entregado por el propio método, y a la hora de predecir la diferencia es mínima, entre predicción y dato real, estos métodos se pueden considerar buenos sobre todo para predecir el valor del dólar en un futuro no muy lejano, incluso cuando solo se tiene la apertura de los diferentes data set utilizados.

El manejo de cual método se utiliza para el análisis de cada caso es diferente y es importante que se entrenen estos conocimientos al manejar todas.

Referencias

[1]"Pronóstico de Series Temporales con Redes Neuronales en Python", *Aprende Machine Learning*, 2020. [Online]. Available: https://www.aprendemachinelearning.com/pronostico-de-series-temporales-con-redes-neuronales-en-python/. [Accessed: 04- Dec- 2020].

[2]"Qué es el pronóstico de series de tiempo: Una introducción no técnica", *Lokad.com*, 2020. [Online]. Available: https://www.lokad.com/es/que-es-el-pronostico-de-series-de-tiempo#:~:text=Una%20serie%20de%20tiempo%20es,un%20valor%20(un%20n%C3%BAmero).&text=Visualmente%2C%20es%20una%20curva%20que,como%20una%20serie%20de%20tiempo. [Accessed: 04- Dec- 2020].

[3]"1.1. Linear Models — scikit-learn 0.23.2 documentation", *Scikit-learn.org*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#lasso. [Accessed: 04- Dec- 2020].

[4]"1.1. Linear Models — scikit-learn 0.23.2 documentation", *Scikit-learn.org*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression-and-classification. [Accessed: 04- Dec- 2020].

[5] Implementación regresión:
https://colab.research.google.com/drive/1uLX1ArQkZr5clPiJdbwVHE8NkGDYCVC7?usp=sharing

[6] Implementación Series de tiempo:
https://colab.research.google.com/drive/1uLX1ArQkZr5clPiJdbwVHE8NkGDYCVC7?usp=sharing

[7] Implementación Series de tiempo:

https://youtu.be/NJLVEKaKaF4