Proyección TRM diaria con Redes Neuronales

Maestría en ciencias de los datos y analítica
Universidad EAFIT

Liceth Mosquera Galvis - Código: 201910046228
Javier Rozo Alzate - Código: 201910053228
Juan Diego Estrada Pérez - Código: 201910042228
Programa: Inteligencia Computacional

11 de mayo de 2020

1. Descripción del problema

El problema que se desea resolver es tener un pronóstico de valores de la TRM para los primeros días hábiles de Enero del 2020.

Las variables financieras son de las más difíciles de pronosticar dada la volatilidad del corto plazo por la cantidad de fenómenos que generan grandes fluctuaciones. Dada la naturaleza de éste problema: donde se tiene entradas multivariadas en el modelo, ruido en los datos, modelado de las relaciones más complejas en los datos; una muy buena opción para abordar éste problema sería mediante redes neuronales.

Para lograrlo se plantean diferentes arquitecturas de redes neuronales aprendidas durante el curso: redes RNN (Recurrent Neural Network), Redes LSTM (Long Short Term Memory) y redes NARX (Nonlinear autoregressive models with exogenous inputs). La metodología utilizada para seleccionar el mejor modelo fue ejecutar un conjunto de experimentos con diferentes hiper-parámetros, para cada tipo de red neuronal. Luego comparamos resultados y se selecciona el modelo que tiene un menor error en la predicción.

2. Entendimiento de los datos

2.1. Escoger las variables:

Aunque la selección de variables no hace parte del curso, si es uno de los pasos importantes para obtener buenos resultados.

En la búsqueda de la literatura se pudo acceder a una tesis de maestría donde precisamente mostraban un modelo para pronosticar la TRM, se utilizó dicho trabajo para escoger las variables a tener en cuenta y los rezagos o modificaciones que debían tenerse en cuenta [**Quintero 2019**]. Las variables tomadas para que ayudan a predecir la TRM se muestran en la siguiente tabla: [1]

	TRM	TRM_logn	IBR	YTES1Y_COP	YTES5Y_COP	YTES10Y_COP	YTES1Y_UVR	YTES5Y_UVR	YTES10Y_UVR	FF_UB	WTI_USD	GOLD_USD	VAR_TRM	VAR_LNTRM	MM_5	MM_20
count	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000	1280.000000
mean	3048.219539	8.018003	0.052413	0.053287	0.063457	0.072009	0.015956	0.027207	0.033723	1.219727	52.163195	1292.769602	1.266305	0.000408	3045.650986	3036.331333
std	286.087760	0.092594	0.012612	0.009529	0.007822	0.007833	0.006387	0.004965	0.004495	0.798633	10.781998	134.213180	26.957999	0.008568	283.565014	274.227040
min	2360.580000	7.766663	0.035160	0.038254	0.050234	0.058742	0.004537	0.015435	0.021375	0.250000	-37.630000	1050.800000	-94.370000	-0.030407	2372.094000	2385.795000
25%	2902.192500	7.973222	0.042540	0.046093	0.058596	0.067309	0.011296	0.024805	0.032274	0.500000	46.437500	1208.050000	-13.947500	-0.004535	2901.454000	2909.559625
50%	3003.065000	8.007389	0.045170	0.048312	0.061477	0.070298	0.014619	0.027179	0.033740	1.250000	52.295000	1271.450000	0.395000	0.000132	3001.658000	3000.484250
75%	3194.892500	8.069309	0.062490	0.062428	0.067033	0.075008	0.020063	0.030237	0.035839	2.000000	59.022500	1334.400000	15.520000	0.005170	3190.723500	3200.109625
max	4153.910000	8.331805	0.077880	0.073609	0.088001	0.096859	0.039636	0.043588	0.045797	2.500000	76.410000	1768.900000	219.020000	0.059307	4110.698000	4031.675000

Figura 1: Resumen estadísticos variables consideradas modelo TRM

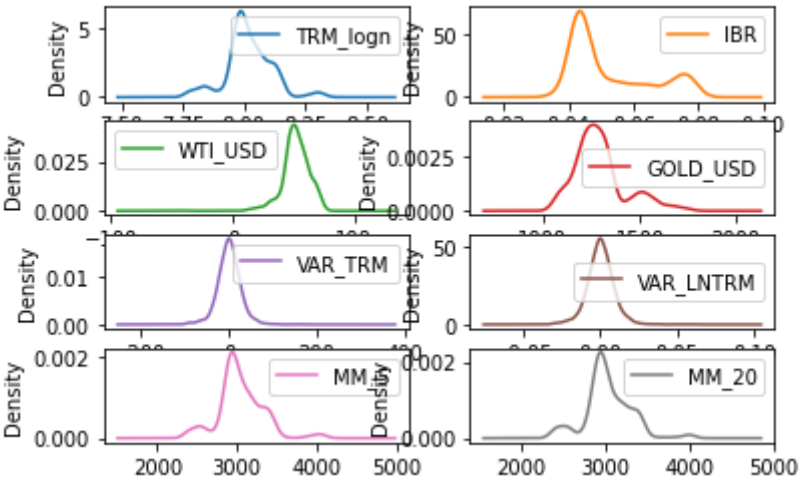

Figura 2: Densidad de algunas variables independientes

[1] https://repository.urosario.edu.co/handle/10336/20708

2.2. Dividir en train, test y value los datos:

De las variables escogidas se logró tener información diaria desde el 30 de enero de 2015 hasta el 30 de abril de 2020, sin embargo como la información no varia para los fines de semana se opta por eliminar esta información.

Como el objeto de este trabajo es pronosticara el primer día hábil de 2020, se optó por predecir 5 días hábiles para hacer una mejor evaluación de desempeño del modelo.

Se dividió la serie en train, test, validation, la serie de validation, se tomaron los datos solo de los días hábiles por lo que para el entrenamiento se contó con 959 datos, para la validación se tuvieron 239 y para el test 6. El test corresponde a los datos de los 9 días del año 2020 a predecir, es decir, el modelo no contiene información de los días a predecir.

3. Preparación de los datos

Se trabajó solo con la serie de tiempo de la TRM, se transformó con una estandarización, es decir, cada elemento se le restó la media y se dividió por la desviación estándar.

4. Red RNN (Recurrent Neural Network)

Esta arquitectura que es usada principalmente en procesamiento de lenguaje y en predicción en series de tiempo, tiene como característica en su arquitectura, que tiene en su forma mas simple 2 entradas al modelo, el vector x y el vector de predicciones del momento t-1, lo que implica que la red esta utilizando la información de su predicción reciente para su siguiente predicción.

Una característica de esta red es que tiene "poca memoria", ya que solo puede tener como entrada 1 predicción del pasado, puesto que si se incluye mas información, al momento de hacer el back propagation se presenta el problema de gradientes que explotan, en el que los valores pasados toman valores muy grandes, o se vuelven 0.

4.0.1. Implementación

Para la implementación de esta red usamos python y una librería especializada en redes neuronales pytorch.

para entrenar el modelo se realizó un ejercicio en el que se le suministro al modelo los 100 días anteriores y se le pide predecir el día siguiente, se realizó este ejercicio varias iteraciones, obteniendo las predicciones que se plantean en la gráfica.

4.0.2. Resultado del modelo

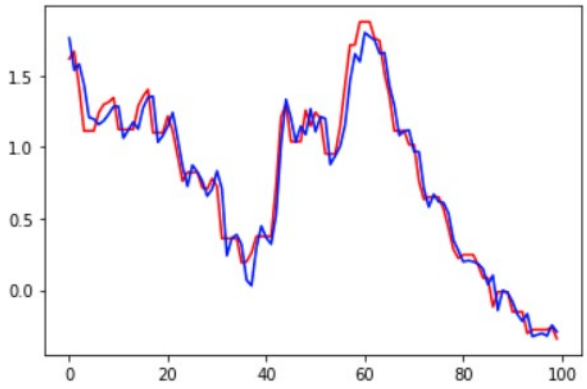

Figura 3: Resultados ultimos 100 dias

una vez se hizo el ajuste de los hiper-parámetros, se evaluó los resultados de los primeros 9 días calendario del 2020, obteniendo los siguientes resultados:

- el promedio de las diferencias de los 9 primeros dias es -3.7
- El MSE es 98.2
- El RMSE es 9.91

5. Red LSTM (long short term memory)

Esta red es un tipo de red recurrente, diseñada para superar el problema de la RNN respecto a los gradientes que explotan, permitiendo que la red aprenda dependencias de mayor plazo.

5.0.1. implementación

Para la implementación de esta red usamos python y una librería especializada en redes neuronales pytorch.

se tomaron 500 datos previo a la fecha de evaluación (1 semana del 2020) como datos de entrenamiento, después de el proceso de selección de hiperparametros, se seleccionó 1 sola capa oculta con 60 neuronas.

5.0.2. Resultado del modelo

.

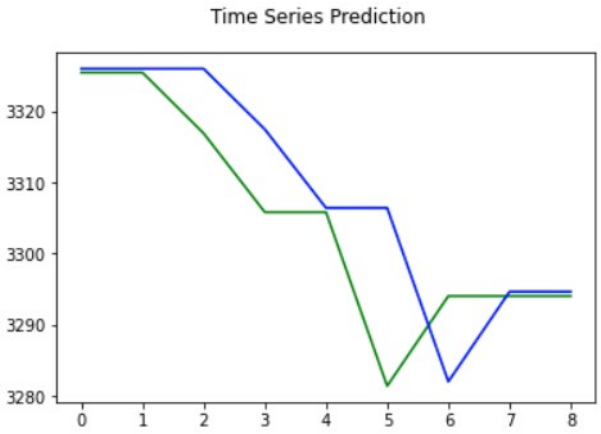

Figura 4: predicción 9 días 2020

En azul se observan los datos reales, en naranja la predicción, se obtienen los siguientes resultados:

- el promedio de las diferencias de los 9 primeros dias es -4.07
- El MSE es 110.21
- El RMSE es 10.49

6. NARX - Nonlinear autoregressive models with exogenous inputs

La red NARX es una arquitectura neural dinámica, comúnmente utilizada para modelar entradas y salidas de sistemas dinámicos no lineales. Su arquitectura es limitada en cuanto a feedback, en donde solo tiene una neurona de output para ello, en lugar neuronas de ocultas, para el caso de modelos de redes neuronales recurrentes.

Existen dos modos o formas de entrenar redes NARX. El primero, es el modo paralelo en donde el output es fed back para el input de la red neuronal feed forward, la cual es parte de la arquitectura NARX estándar.

El otro modo es llamado series en paralelo (SP) en donde el output de los datos reales son utilizados para alimentar la red neuronal, a diferencia de anterior modo, en donde se alimenta del output estimado.

Para llevar a cabo la construcción del modelo se realiza lo siguiente:

- La red neuronal en modo NARX de series en paralelo, es creada para entrenarse con los datos de entrada.
- Luego se convierte a modo paralelo para realizar las predicciones de los primeros

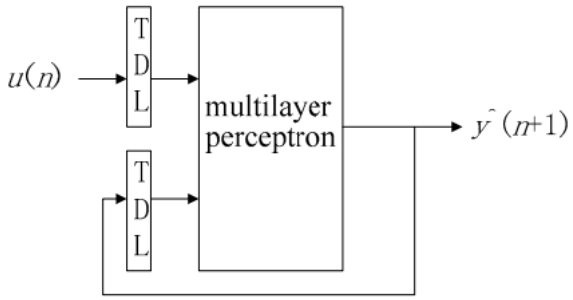

Figura 5: Modo P

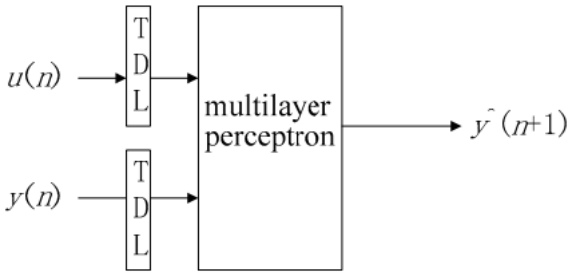

Figura 6: Modo SP

9 días hábiles de Enero.

6.0.1. Implementación

MATLAB permite implementar éste tipo de redes neuronales, con múltiples variables de entrada. Para ello se utiliza la función **narxnet(d1, d2, n)**, en donde *d1* son los rezagos para las entradas X o variables *exógenas*, *d2* son los rezagos para Y o la salida y n son el número de neuronas de la capa oculta.

El código utilizado para dichos experimentos se encuentra en el repositorio de git al cual se hace referencia en la sección de Anexos.

6.0.2. Resultados

A continuación la configuración de la red con diferentes hiper-parámetros y sus respectivos resultados:

	Modelo 1	Modelo 2	Modelo 3
Rezagos variables exógenas	[1:7]	[1:5]	[1:5]
Rezagos variable dependiente	[1:7]	[1:5]	[1:2]
Número de capas	3	3	1
Neuronas por capa	[10 20 30]	[10 10 10]	[30]
RMSE Predicción	29.75	66.36	19.46
epochs	10	50	50

Para los anteriores casos se escoge el modelo NARX donde el error RMSE es igual a 19.46. A continuación la configuración de la red tanto modo SP como P.

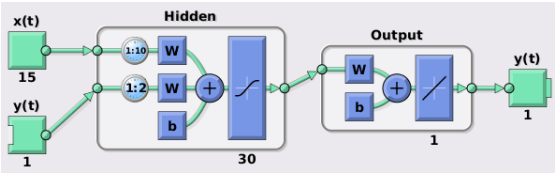

Figura 7: Arquitectura tres capas - Modo SP

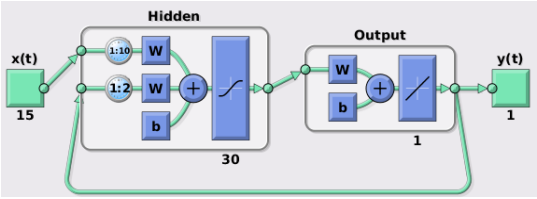

Figura 8: Arquitectura tres capas - Modo P

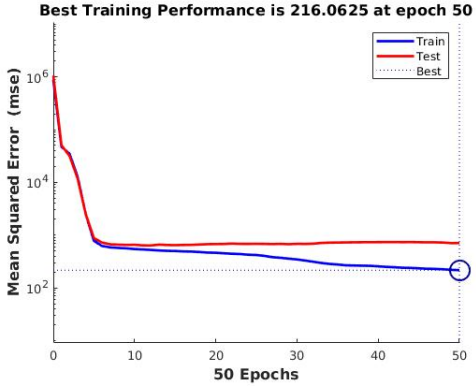

Figura 9: Mejor desempeño con epoch igual a 50

A continuación la estimación de los datos de prueba y finalmente la predicción de los primeros 8 días hábiles de Enero.

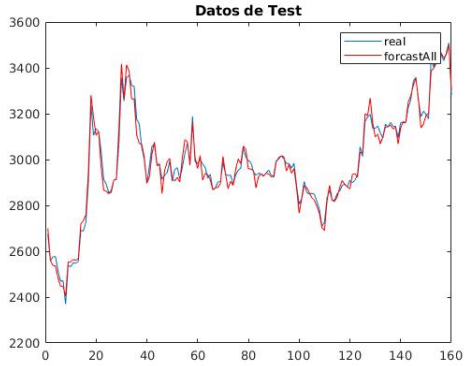

Figura 10: Predicción últimos 160 días año 2019

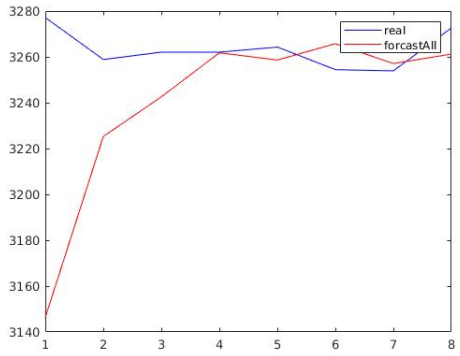

Figura 11: predicción 30 Diciembre a 14 de Enero 2020 (8 días hábiles)

7. RNN -LSTM cambiando hiper parámetros

En este caso se probaron otras variables y otros hiper parámetros, a continuación el paso a paso del preprocesamiento de los datos :

7.1. Implementación

A continuación se detallan los pasos seguidos para obtener la rd utilizada:

7.1.1. Escoger la red

Entre las opciones probadas la escogida fue una red neuronal recurrente de memoria de corto plazo (LSTM) dado que son capaces de modelar modelos con múltiples variables de entrada casi sin problema. Modelo programado en python utilizando la biblioteca de aprendizaje profundo de Keras.

7.1.2. Algoritmo de entrenamiento

El modelo escogido se define el LSTM con 3 capas ocultas en la primera capa oculta tiene 16 neuronas, una segunda LSTM con 8 neuronas, una tercera LSTM con 8 neuronas y una neurona en la capa de salida. la entrada sera de a un paso de tiempo con 16 características.

Model "sequential_1"

Layer (type)	output Shape	Param #
lstm_ 3 (LSTM)	(None, 1, 16)	2048
lstm_ 4 (LSTM)	(None, 1, 8)	800
lstm_ 5 (LSTM)	(None, 8)	544
dense_1 (Dense)	(None, 1)	9

Total params: 3,401
Trainable params: 3,401
Non-trainable params: 0

Se utiliza la función de pérdida de error absoluto medio (MAE) y la versión eficiente de Adam del descenso de gradiente estocástico.

7.1.3. Entrenar

El modelo se ajustará a 80 épocas de entrenamiento con un tamaño de lote de 5. El estado interno del LSTM en Keras se restablece al final de cada lote, por lo que un estado interno que es función de varios días puede ser útil.

Finalmente, se hace seguimiento de la pérdida de entrenamiento y prueba durante el entrenamiento al establecer el argumento validation_ data en la función fit ().

7.1.4. Evaluar la red

a continuación los resultados de salida que permiten evaluar la red escogida:

Métrica	valor
Mean absolute error (MAE)	209.866821
Mean squared error (MSE)	63024.886719
Root Mean squared error (RMSE)	251.047579
R square (R^2)	-3.292923

Quizá se requiera seguir ajustando los hiper parámetros para mejorar la red, se grafican tanto el entrenamiento como la pérdida de prueba.

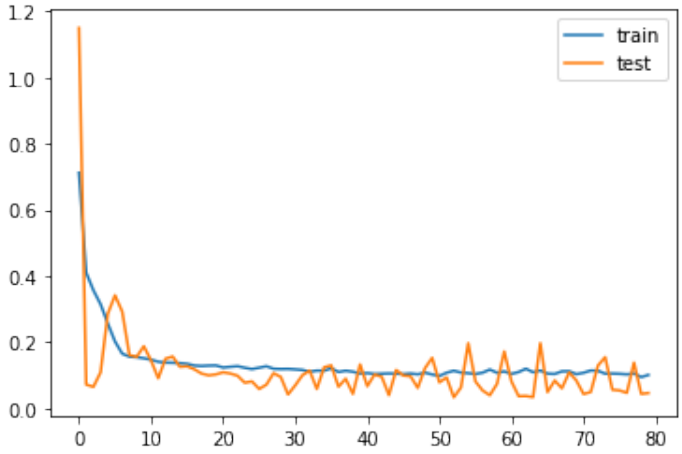

Figura 12: Perdida en la prueba

7.2. Resultado del modelo

Una vez el modelo es entrenado se hace las predicciones. se muestra la gráfica de el pronóstico Vs el real de los seis días en la escala original.

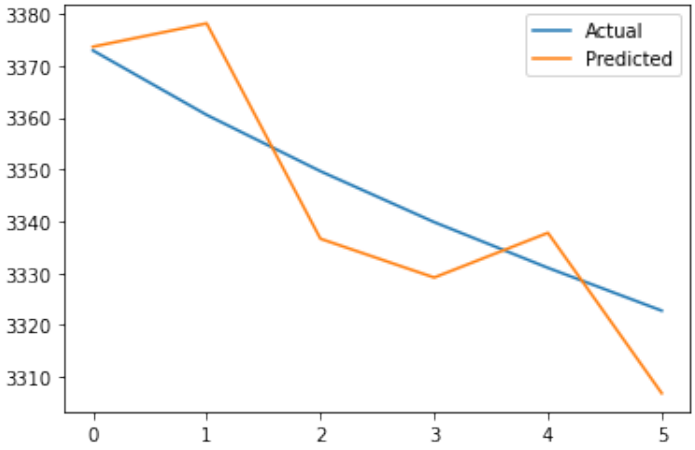

Figura 13: Perdida en la prueba

se muestra el resultado de esta predicción en métricas.

Métrica	valor
Mean absolute error (MAE)	10.790324
Mean squared error (MSE)	149.099091
Root Mean squared error (RMSE)	12.210614
R square (R^2)	0.489065

8. Conclusiones

En terminos generales todos los modelos logran un buen nivel de prediccion. los resultados son los siguientes:

- Red RNN con un RMSE de 9.91 y diferencia en pesos de -3.7.
- Red LSTM con un RMSE de 10.40 y diferencia en pesos de -4.7.
- Red NARX con un RMSE de 19.46.
- Red RNN y LSTM con variación en el planteamiento de problema e hiperparametros RMSE de 12.21

Al analizar los resultados anteriores, las redes RRN y LSTM presentan los mejores desempeños, todos tienen resultados muy similares a un RMSE de 10. de elegir un solo ejercicio seria el de la Red RNN, que presenta un mejor desempeño sobre los datos de validación.

9. Anexos

Todo el código utilizado para la construcción de estos modelos se encuentran en el siguiente repositorio de git:

`https://github.com/4JL/inteligencia_computacional`