Trabajo Final Deep Learning (DL - MDS18)

# Prediccion Tráfico de Bicicletas usando DL/LSTM

Alumno: Marcelo Rovai
Profesor: Pablo Figueroa

---

## Objective

Este trabajo utiliza el modelo LSTM de Deep Learning para predecir el número de bicicletas que pasan diariamente por la puente de Fremont Bridge en Seattle, WA EE.UU. El modelo es ajustado, variandose varios hiperparámetros como:

- Ventana de tiempo
- Número de neuronas
- Número de épocas
- Porcentaje de dropouts
- Optimizer

Además, el mejor modelo lSTM se enfrentará a otro modelo de previsión de series temporales, Prophet, desarrollado por Facebook.

# Dataset

Los sensores de bicicletas en el puente Fremont son dispositivos instalados bajo el pavimento que detectan y contan objetos metálicos a medida que los cruzan. Esos sensores proporcionan información valiosa sobre los patrones del ciclismo en Seattle. Los datos se generan 7 X 24 X 365 y se cargan una vez al día a las 5 de la mañana.

El conjunto de datos más actualizado tiene 62.760 instancias, que muestran el número de bicicletas que pasan por hora en las aceras del puente Fremont (Este y Oeste). La fecha va de octubre de 2012 hasta noviembre de 2019.

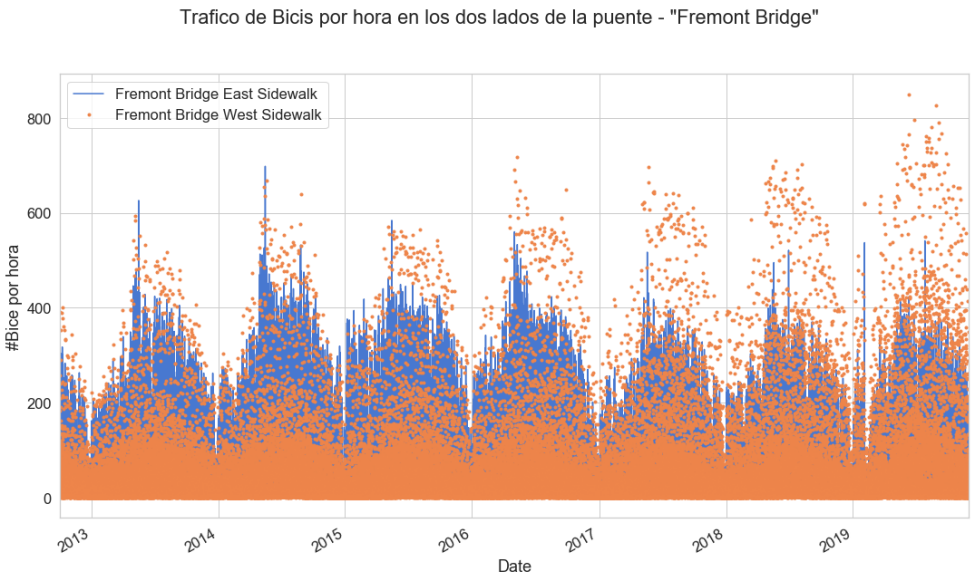
El conjunto de datos se puede descargar desde el sitio web oficial de Seattle: https://data.seattle.gov/api/views/65db-xm6k/rows.csv?accessType=DOWNLOAD

Acá se puede ver un ejemplo de la data:

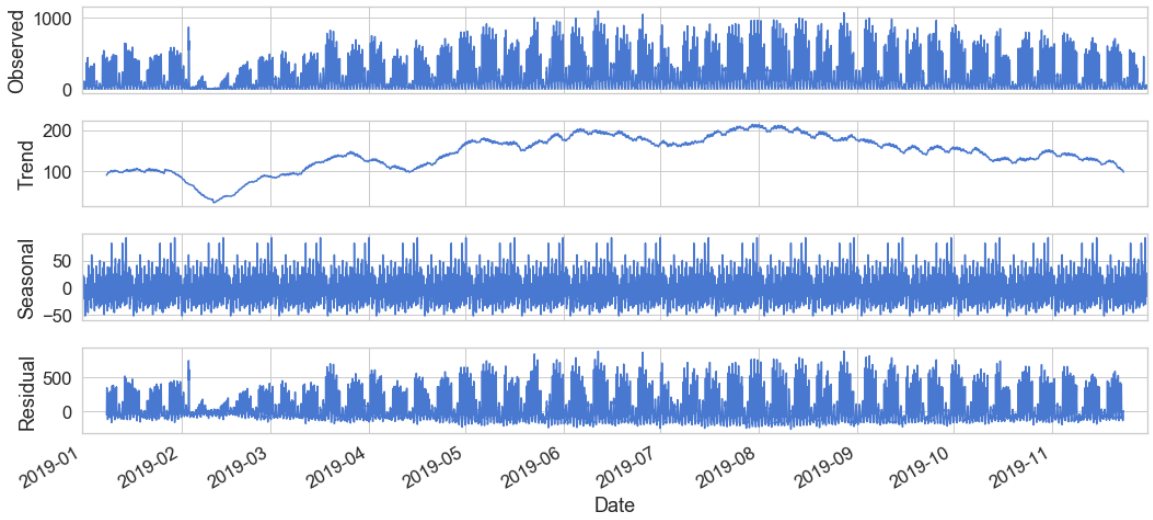
| Date | Fremont Bridge Total | Fremont Bridge East Sidewalk | Fremont Bridge West Sidewalk |
|---|---|---|---|
| 2012-10-03 00:00:00 | 13.0 | 4.0 | 9.0 |
| 2012-10-03 01:00:00 | 10.0 | 4.0 | 6.0 |
| 2012-10-03 02:00:00 | 2.0 | 1.0 | 1.0 |
| 2012-10-03 03:00:00 | 5.0 | 2.0 | 3.0 |
| 2012-10-03 04:00:00 | 7.0 | 6.0 | 1.0 |

# Limpieza y Exploración de la data (EDA)

Los datos tienen 10 registros nulos, que deben ser desechados, resultando en 62.750 datos. La figura abajo muestra el trafico de Bicis por hora en los dos lados de la Puente:



Mirando los datos con más detalle, es posible dividirlos en su tendencia (Trend), datos estacionales (Seasonal) y residuales (Residual):

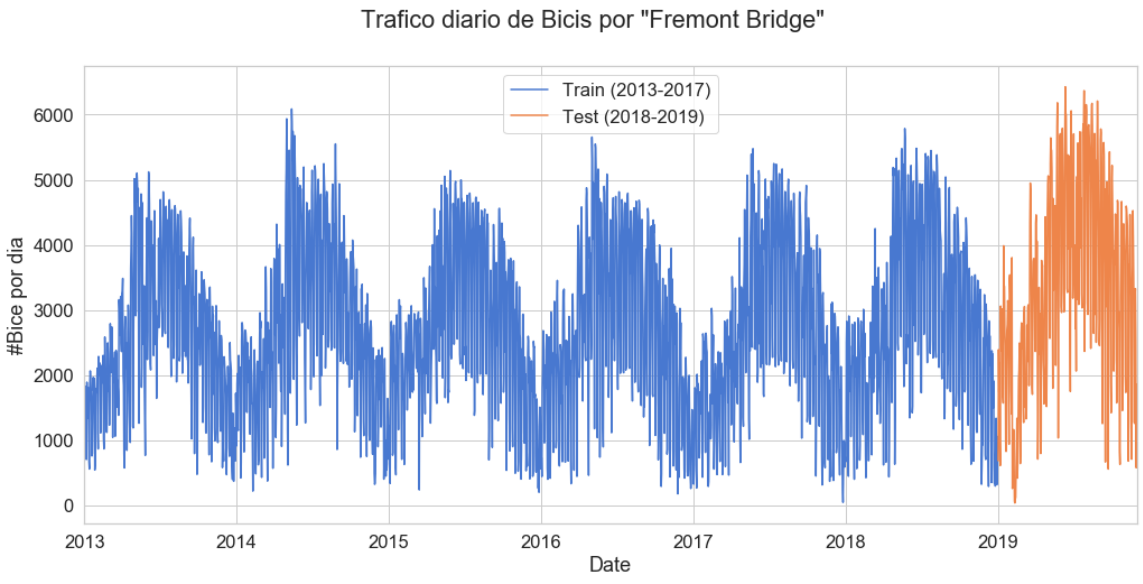



De la figura anterior, se puede observar que los datos no son completamente aleatorios, por lo que es posible hacer una predicción futura del número de bicis pasarán por el puente, solamente basándose en los datos históricos.

# Definir set entrenamiento/validación

Como los datos están por hora y el objectivo es hacer una previsión diaria, el dataset es entonces “resampled”, donde el numero de ciclistas por hora son somados para se obtener los datos de un dia.

El modelo es entrenado con los datos hasta deciembre de 2018, dejando los datos de 2019 para se testear como va el modelo.



# Normalización del set de entrenamiento

Los datos son normalizados desde 0 a 1, para facilitar el trabajo del modelo. Ademas una ventana ("time-step"), define el numero de entradas que deberá tener la red LSTM. Inicialmente se trabajará con una ventana de 60 dias. Posteriormente se reducirá a 30 dias para se estudiar sus efectos. El set es entonces "reshaped" para que se ajuste al modelo de Keras. Con esto se conforma el set de entrenamiento (X_train y Y_train).

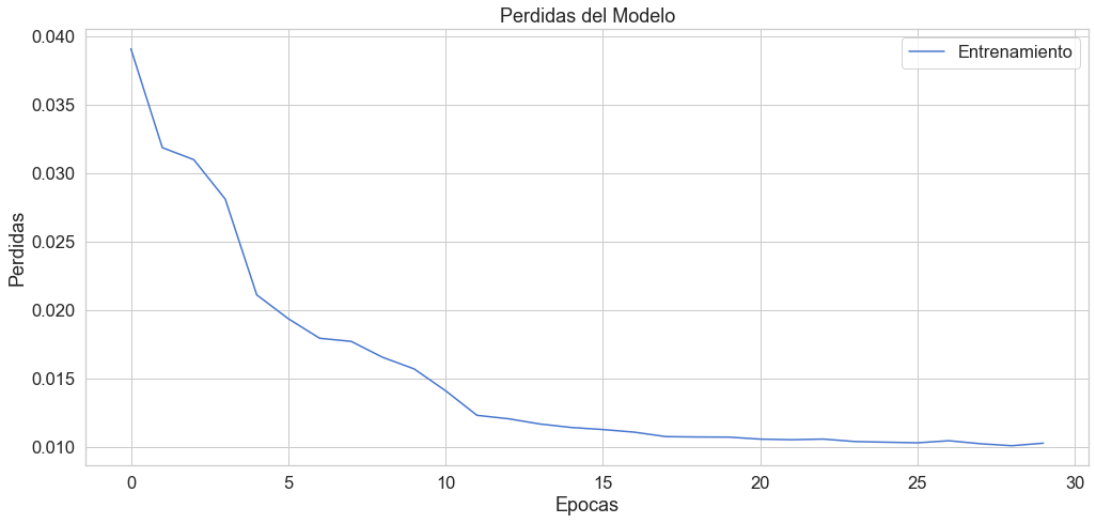
## La Red LSTM

El modelo terá:

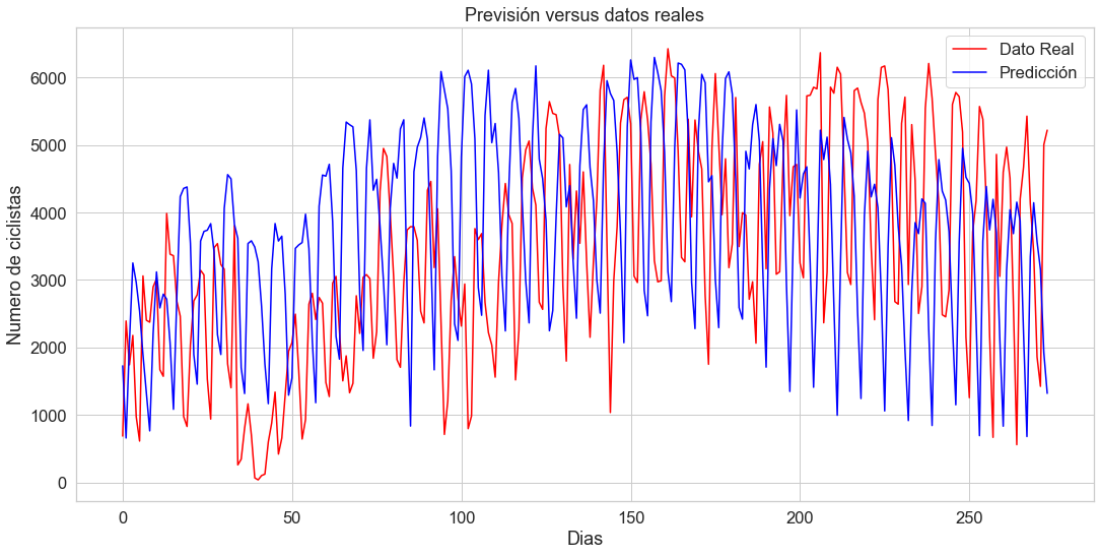
- Dimensión de entrada: (60, 1)
- Dimensión de salida:
- Número de unidades en la capa oculta (na) = 50
- Número de épocas iniciales: 30
- Tamaño del buffer: 32

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 50)                10400
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
```

Una vez corrido el modelo se puede observar el resultado de las perdidas a medida que se ejecutan las epocas:



En seguida es importante verificar graficamante como el modelo hice la previsión versus los datos reales (set de validación enero a noviembre 2019):




Claramente el modelo pudo seguir la tendencia general de los datos, pero se equivoca en los valores absolutos. Al calcular o error medio cuadratico (RMSE), se llega a 2.302 ciclistas. Un valor bien elevado. Pero, también es importante senalar que algo pasó en el Q2 de 2019, pues bajó mucho en numero de ciclistas en la puente y esto obviamante no fue capturado por el modelo. La parte central de la grafica parece que el modelo está mejor (en verano), donde el numero de ciclistas aumenta.

## Funciones para Benchmark

El proximo paso es definir una serie de funciones para auxiliar en el benchmark de los parámetros del modelo. Serán creadas las seguintes funciones:

1. create_train_data(data, time_step, INICIO, ENTREN, VALID)
2. create_model(X_train, na, epochs, batch_size, dropout, optimizer, loss)
3. fit_model (modelo, X_train, Y_train, epochs, batch_size, verbose)
4. predict (modelo, X_test, set_validacion, sc, tickers)

Y además la función test_model(), la cual receberá los parametros a ser testeados, llamará las cuatro funciones anteriores. Los resultados serán almacenados en la variable: *result* para posterior comparación. En el notebook se podrá ver todos las gráficas y resultados de los diversos benchmarks ejecutados. El mejor resultado fue un RMSE de 1,826.0 con los parametros abajo:

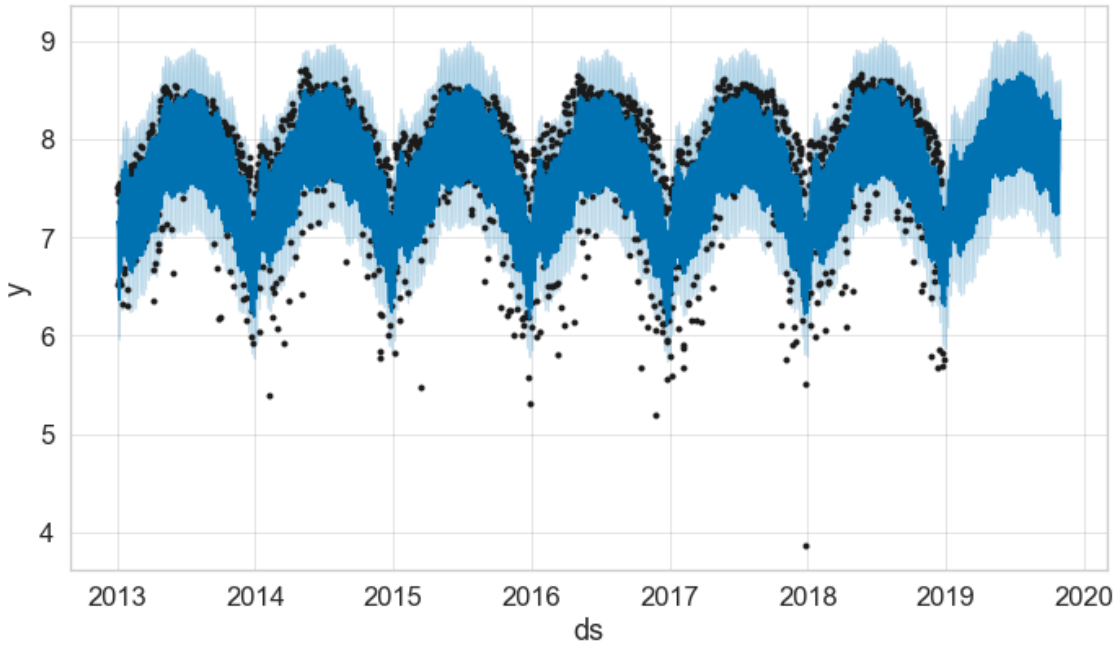
- time_step = 30
- optimizer = 'rmsprop'
- na = 100
- epochs = 100
- dropout = 0.0

La tabla abajo, resume los resultados del benchmark:

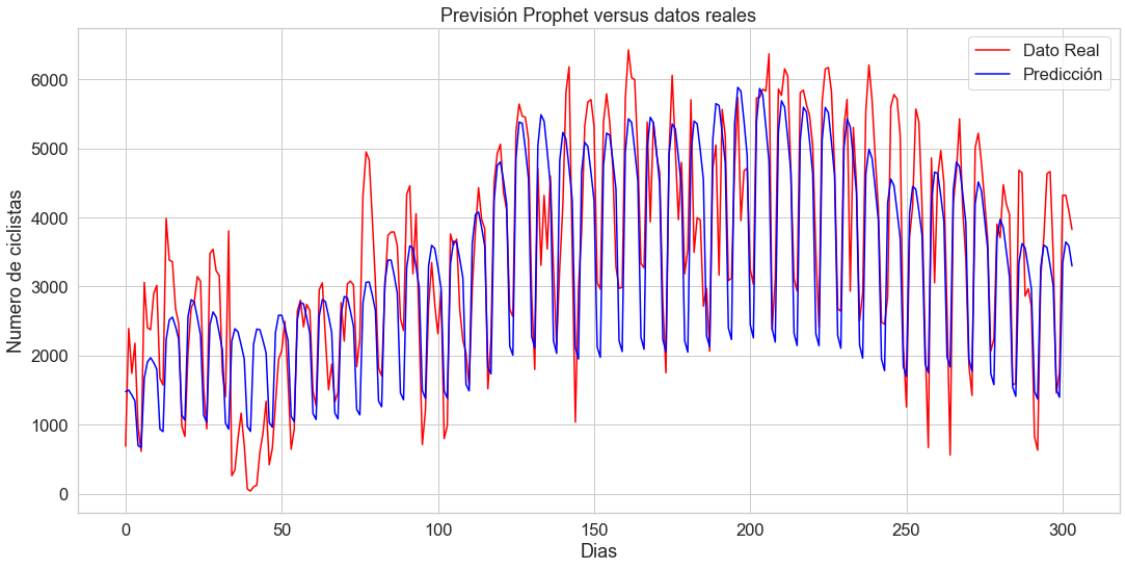
| | time_step | na | epochs | batch_size | dropout | optimizer | loss | rmse |
|---|---|---|---|---|---|---|---|---|
| **8** | 30 | 100 | 100 | 32 | 0.0 | rmsprop | mse | 1826.0 |
| **9** | 30 | 100 | 200 | 32 | 0.1 | rmsprop | mse | 1865.0 |
| **7** | 60 | 100 | 100 | 32 | 0.0 | rmsprop | mse | 2047.0 |
| **6** | 60 | 100 | 100 | 32 | 0.0 | adam | mse | 2075.0 |
| **1** | 60 | 50 | 100 | 32 | 0.0 | rmsprop | mse | 2084.0 |
| **0** | 60 | 50 | 30 | 32 | 0.0 | rmsprop | mse | 2191.0 |
| **2** | 60 | 50 | 100 | 32 | 0.2 | rmsprop | mse | 2219.0 |
| **3** | 60 | 50 | 100 | 32 | 0.4 | rmsprop | mse | 2306.0 |
| **4** | 60 | 100 | 100 | 32 | 0.4 | rmsprop | mse | 2380.0 |
| **5** | 60 | 100 | 100 | 32 | 0.4 | adam | mse | 2445.0 |

# Test con Prophet

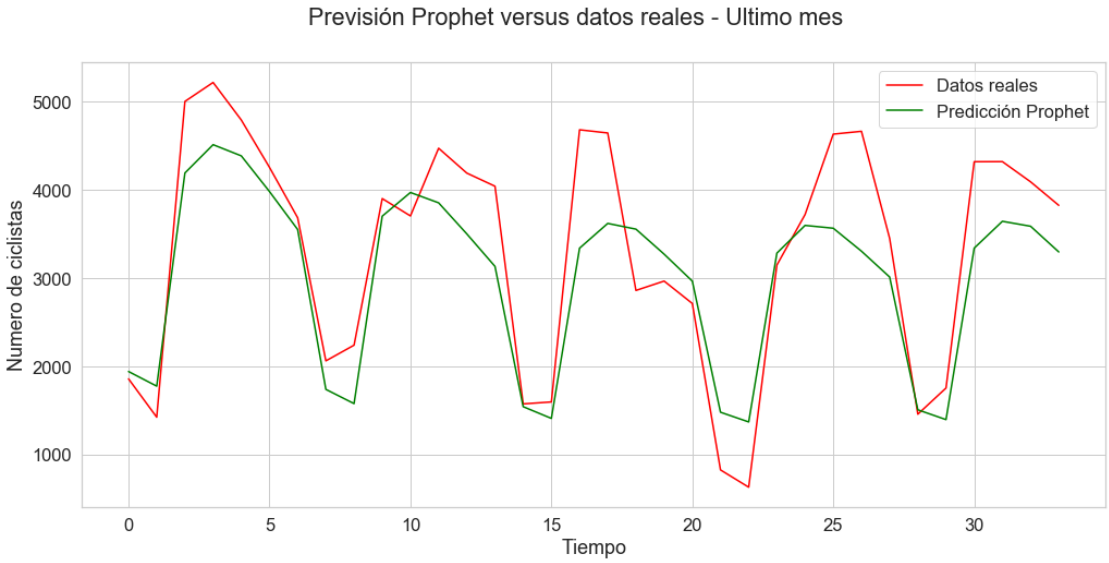
Por ultimo, se trabajó con Prophet de Facebook, un modelo para previsón de serie de tiempos. Abajo se puede ver el resultado da aplicación del modelo Prophet a los mismos datos usados como entrada a la red LSTM:



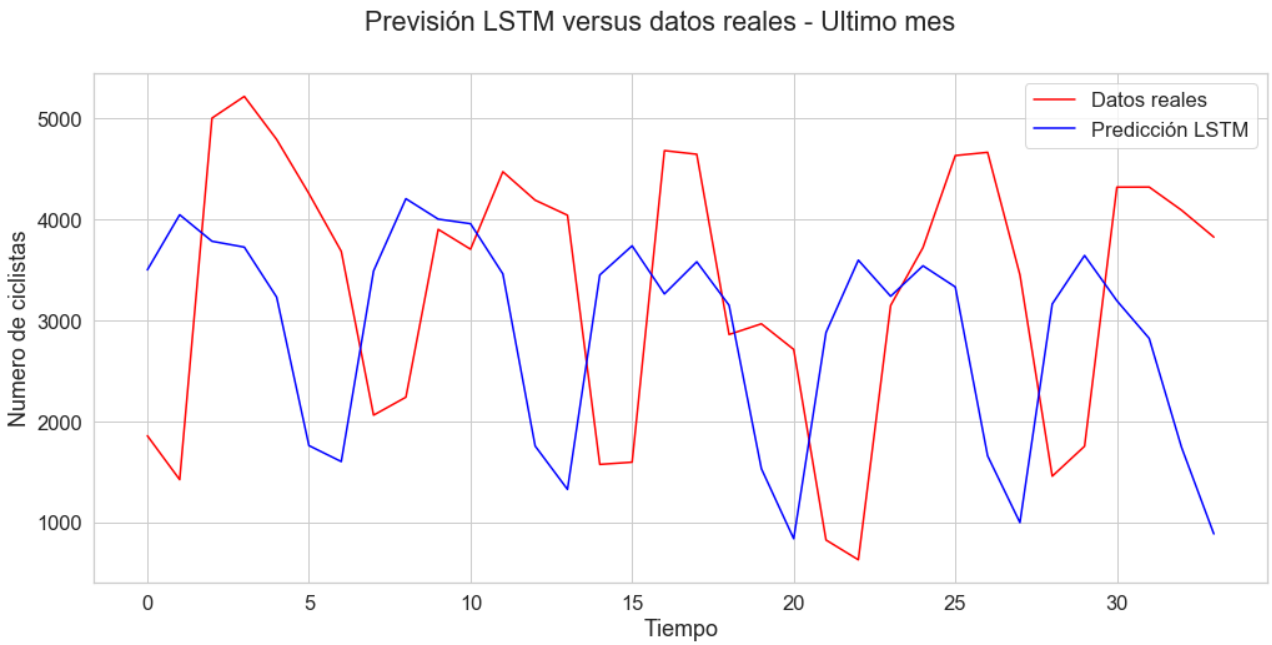
Al comparar la previsón de Prophet con la data de validación, se observa que el resultado es mucho mejor:




El RMSE fue de 618.2, mucho mejor de lo que se llegó con la red LSTM. Las graficas abajo hacen un “zoom” en lo último mes de previsión, donde se puede ver bien como los modelos trabajaron:




Prophet conseguió capturar tanto las tendencias cuanto los valores en general. Se observa que el modelo se pierde en los valores maximos y minimos, pero es bueno en lo general.




Con la red LSTM, el resultado es malo. A pesar de ser posible al modelo capturar las tendencias, pero se percebe un cierto “delay” en cambiar los rumbros.

# Notebook con los códigos

El codigo del traballo desarrollado está disponible en el entorno Colab y puede ser descargado a partir del link:

https://drive.google.com/file/d/15Ly-37mjqzA9WbfAmpclUCMMU7uYUJbL/view?usp=sharing

Santiago, 26 de diciembre 2019
Marcelo José Rovai