PREDICCIÓN DE INGRESOS Y EGRESOS DE VEHÍCULOS EN LA CIUDAD DE BUENOS AIRES

UTN FRBA - 2021
Sofía Ploschuk y Javier Laudadío

RESUMEN
En el siguiente informe se estudia la cantidad de vehículos que ingresan y egresan a la Ciudad Autónoma de Buenos Aires, en diferentes días y horarios, tomando como dato los registros de las cámaras de seguridad en diferentes accesos de la ciudad.

1. Introducción

Dentro del marco académico de la materia Ciencia de Datos de la Universidad Tecnológica Nacional (FRBA), se propone realizar un estudio sobre el flujo de circulación de vehículos en la Ciudad de Buenos Aires. El dataset fue descargado de la librería pública del Gobierno de la Ciudad, quienes ofrecen de manera abierta y gratuita los datos recolectados en diferentes ámbitos y medios.

El código se desarrolló en Python, con Pandas, Numpy, Matplotlib, Seaborn, Scikit Learn y TensorFlow como principales librerías y Jupyter Notebook como herramienta de trabajo. El mismo se estructura partiendo de un Análisis Exploratorio de Datos, continuando con una comparación entre un modelo KNN Regression, uno de Redes Neuronales, y un tercero de Bosque Aleatorio, para finalmente concluir en una función optimizada con el modelo más apropiado.

2. Descripción del Dataset

Para el análisis tomamos un dataset llamado Flujo Vehícular, extraído de la página web BA Data (para descargar haga click acá). El portal de datos abiertos BA Data fue creado en el año 2012, y recopila más de 300 conjuntos de datos de 13 áreas de gobierno, los cuales se encuentran publicadas en formato abierto para que cualquier persona los pueda descargar y reutilizar para crear nuevos análisis, aplicaciones o ideas.

Originalmente el dataset cuenta con seis columnas que reflejan el código de locación (string), la latitud y longitud de la ubicación del sensor (ambos string), el sentido de circulación (string), fecha del registro (date) y cantidad de autos detectados en la franja horaria (int).

Por otro lado, se utilizó también un dataset calendario que detalla los días hábiles y no hábiles según la fecha, con el objetivo de poder diferenciar la dinámica del flujo de circulación de vehículos en estos días. En las columnas de este dataset se incluye la

fecha (date); día de la semana, mes y año (los tres en int); día de la semana identificado con un número (int); el número de semana, mes, trimestre y año (int)

3. Análisis Exploratorio de Datos

El dataset cuenta originalmente con 150.980 registros. Decidimos eliminar los valores NaN de la columna del código de locación ya que los consideramos registros vacíos, así como los valores que corresponden a un sentido de circulación interna, dejando sólo los ingresos y egresos. De esta manera, quedaron 70.739 registros. Por otro lado, la fecha que estaba en formato date la separamos en dos columnas: días de la semana y hora, y la concatenamos con el segundo dataset para poder determinar los días hábiles y no hábiles. Asimismo, a partir de los puntos de localización, se pudo asociar los códigos de locación con un barrio de la ciudad.

De esta manera, podemos observar el promedio diario de vehículos que circulan por día de la semana, en función de los ingresos y egresos:

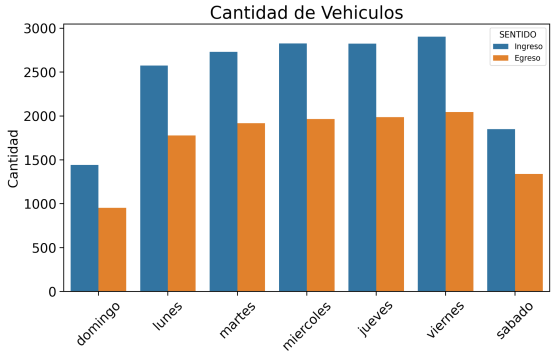

Figura 1 - Cantidad media de vehículos que circulan según el día de semana, entre todos los accesos. En azul el sentido "ingreso" y en naranja el "egreso"

Se puede ver que en estos accesos específicos, son más los vehículos que transitan en el sentido de circulación de ingreso. Inferimos por lo tanto que hay accesos en donde sucede lo contrario, pero estos no fueron incluídos en el dataset otorgado por BA Data. Por otro lado, se puede graficar la distribución del flujo de circulación según el horario:

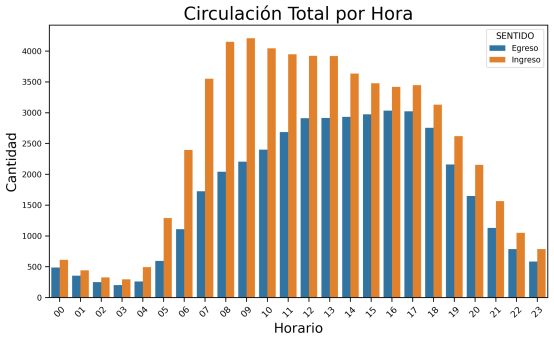

Figura 2 - Cantidad media de vehículos que circulan según el horario en un día de semana, entre todos los accesos. En azul el sentido "ingreso" y en naranja el "egreso"

Este gráfico refuerza que los ingresos son mayores que los egresos, y demuestra un incremento en el flujo de circulación entrante creciente desde las 03hs, con un pico a las 09hs y que luego decrece de manera abrupta a partir de las 18hs. Por el contrario, el flujo de ingresos crece desde las 03hs hasta llegar a su pico a las 17hs. Por lo tanto, en la franja de 05hs a 13hs es donde se percibe una mayor diferencia de circulación entre ambos sentidos.

Por otro lado, se puede observar que la cuarentena obligatoria alteró la circulación vial, por lo que los datos anteriores a agosto de 2020 también fueron eliminados.

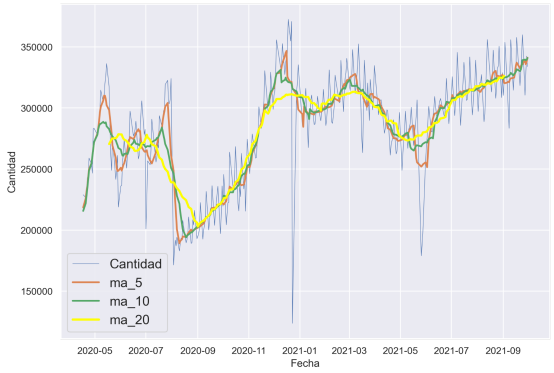

Figura 3 - Tendencia sobre la cantidad total de vehículos circulando diariamente desde mayo de 2020 hasta septiembre de 2021. Promedios móviles: en naranja ventana de 5, en verde de 10 y en amarillo de 20.

De los datos obtenidos luego de la limpieza de los mismos, se utilizará un 80% para entrenamiento y 20% para testeo.

4. Procesamiento de Datos

4.1 Modelo KNN

Como primera instancia, se desarrolló un modelo K-Nearest Neighbor (KNN) para aproximar la función. La característica de este método es que se aproxima una función a través de las muestras vecinas, definiendo como hiper-parámetro a los "K" vecinos. Como desventaja, debido a que este modelo es comparativo, es necesario contar siempre con el dataset de entrenamiento cada vez que se quiere aplicar el modelo.

Para determinar el mejor hiper parámetro, se realizó un Gridsearch, que arrojó que 22 era la cantidad óptima de vecinos a considerar. Luego, estimamos la Raíz del Error Cuadrático Medio (RMSE) y graficamos la función versus los datos reales, resultando de la siguiente manera:

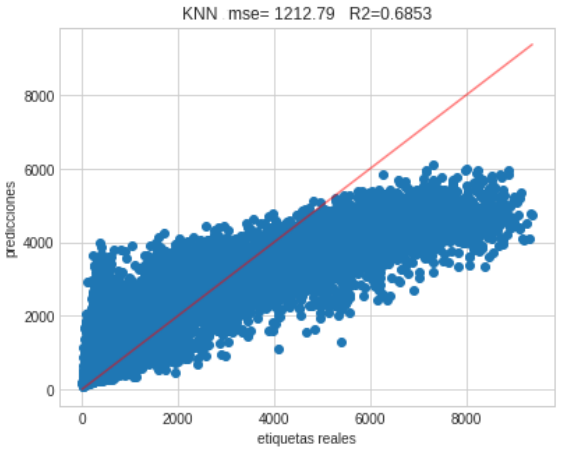

Figura 4 - Modelo KNN: en azul los datos de testeo, en rojo la curva de predicción.

Como se puede observar en el gráfico anterior, el RMSE es de 1212,79 y un R2 de 0.6853. Por otro lado, la función generada parecería no responder correctamente a los datos de entrenamiento, por lo que suponemos que debería existir un modelo que sea más apropiado para generar predicciones.

4.2 Redes Neuronales (NN)

Como segunda opción, se aplicó un modelo de redes neuronales. La idea es simplificar la función eliminando dimensiones entre diferentes capas, siendo cada capa una variable aprendida por la combinación de variables en la capa anterior.

Utilizando TensorFlow como principal librería, se utilizaron dos capas con un regularizador de 0.01, una tasa de aprendizaje del 1%, 25 épocas (epochs) y un lote (batch) de 5. En la siguiente figura de pérdida (Loss) en función de epoch, se puede observar cómo a partir de los 20 epochs, la pérdida se regulariza con una tendencia a cero:

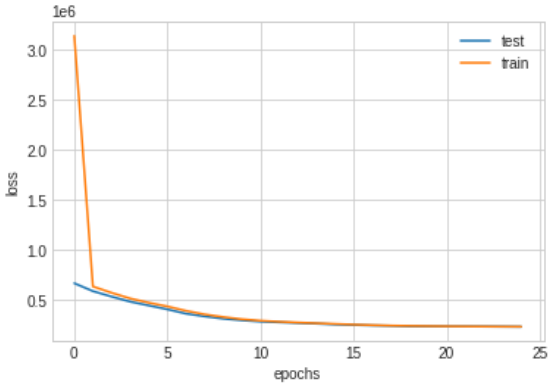

Figura 5 - Modelo de Redes Neuronales: evolución de la pérdida en función de las épocas. En naranja los datos de entrenamiento y en azul los de testeo.

De esta manera, se redujo sustancialmente el RMSE hasta 485.59 y el R2 se elevó a 0.9501:

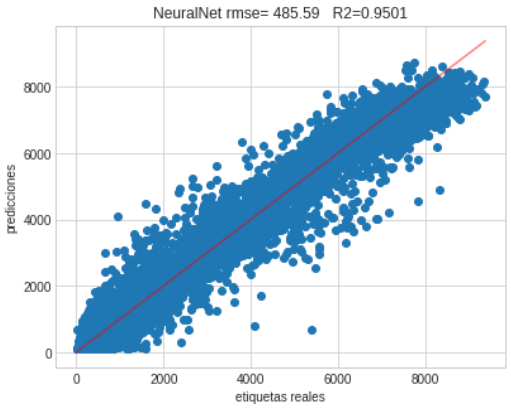

Figura 6 - Modelo de Redes Neuronales: en azul los datos de testeo, en rojo la curva de predicción

4.3 Random Forest

Por último, utilizamos un modelo de regresión Random Forest (Bosque Aleatorio) para generar una función. Este modelo desarrolla varios árboles de decisión en su objetivo de predecir la variable de interés, y suele ser frecuentemente utilizado para series de tiempo. Al realizar un Gridsearch, se determinó que los mejores hiperparámetros consistían en lotes de 200 muestras con una profundidad máxima de siete ramas. Estos valores generaron un RSME de 1423.73 con un R2 de 0.59, junto con la siguiente figura:

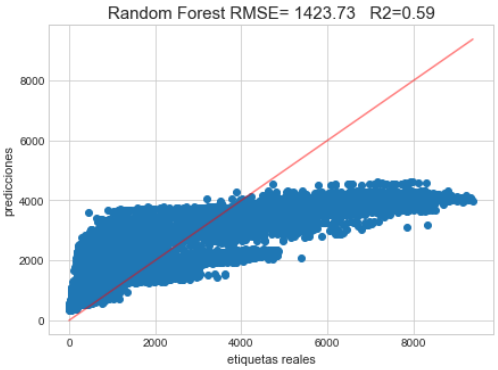

Figura 7 - Modelo Random Forest: en azul los datos de testeo, en rojo la curva de predicción

5. Conclusiones

El trabajo concluye con una comparativa entre los tres modelos, para determinar cuál fue el que logró un mejor aprendizaje. Para esto, se consideran tres recursos clave: la raíz cuadrada del error cuadrático medio (RMSE), el score R2 y los gráficos donde se exponen las funciones generadas junto con los datos de testeo. El RMSE se calcula como:

$$\boldsymbol{RMSE} = \sqrt{\frac{\sum (\hat{y}_t - y_t)^2}{n}}$$

Donde $\hat{y}_t$ son los valores predecidos por la función generada, y_t son los datos de testeo y n es el número de muestras. Como es de suponer, se trata de que el RMSE sea el menor posible, ya que cuanto mejor sea la función generada por el modelo, menor va a ser la diferencia entre el valor real y el predecido. Se tiende a utilizar este índice por sobre otros similares como el MSE (error cuadrático medio) ya que al aplicar la raíz cuadrada,

se reducirá el efecto que pueden producir los outliers.

Por otro lado, el R2 se define con la siguiente fórmula:

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}}$$

Siendo TSS la sumatoria de la diferencia entre los valores Y de testeo y la media, elevado al cuadrado, y RSS la sumatoria de los errores al cuadrado. Por lo tanto, cuanto menor sea el error de predicción, más va a tender R2 a valer uno.

En conclusión, el mejor modelo se va a seleccionar considerando aquel que tenga un menor RSME y un R2 con mayor tendencia a uno. Los resultados fueron:

Modelo	RMSE	R2
KNN	1212.79	0.69
NN	485.59	0.95
Rand. For.	1423.73	0.59

A su vez, en la figura 6 se puede observar que el modelo de redes neuronales es el que mejor ajustó su curva de predicción con los datos de testeo, en comparación con los otros planteados (figuras 4 y 7). Por lo tanto, determinamos que el mejor modelo a aplicar es el de redes neuronales.

Referencias

(1) Gareth James, Daniela Witten, Trevor Hastie & Robert Tibshirani, "An Introduction to Statistical Learning with Applications in R", Segunda edición, 2021.

(2) Pádraig Cunningham & Sarah Jane Delany, "k-Nearest Neighbour Classifiers 2nd Edition (with Python examples)", Irlanda, 2020.

(3) Dipanjan Sarkar, Raghav Bali & Tamoghna Ghosh, "Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras", India, 2018.

(4) Muhammad Waseem Ahmad, Monjur Mourshed & Tacine Rezgui, "Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption", Reino Unido, 2017.