

Reporte Etapa 4: Modeling

| | |
|---|-----------|
| Objetivo: | 3 |
| Selección de modelo | 3 |
| Técnicas de modelo a emplear | 3 |
| Generar el diseño de las pruebas | 4 |
| Diseño de las pruebas | 4 |
| Construcción del modelo | 5 |
| Modelos | 5 |
| • Regresión lineal | 5 |
| • Random Forest | 5 |
| • Gradient Boosted | 5 |
| • Extra trees Regressor | 5 |
| • Densely-Connected Neural Network (Sequential) | 5 |
| Ajuste de parámetros | 5 |
| Lista de los parámetros y sus valores elegidos para las herramientas de modelado, junto con las razones por las cuales se eligieron esos valores. | 5 |
| Modelos | 6 |
| • Regresión Lineal | 7 |
| • Random Forest Model | 7 |
| • Gradient Boosted | 8 |
| • Extra trees Regressor | 8 |
| • Densely-Connected Neural Network (Sequential) | 9 |
| Descripción del modelo | 10 |
| • Regresión Lineal | 10 |
| • Random Forest | 10 |
| • Gradient Boosted | 11 |
| • Extra Trees Regressor | 11 |
| • Densely-Connected Neural Network (Sequential) | 11 |
| Evaluar el modelo Y parámetros revisados y ajustados | 11 |
| • Regresión Lineal | 12 |
| • Random Forest | 12 |
| • Gradient Boosted | 13 |
| • Extra Trees Regressor | 13 |
| • Densely-Connected Neural Network (Sequential) | 14 |
| Comparación de modelos | 17 |

Objetivo:

Predecir la cantidad de viajes que realizan las comunas a las universidades en determinados horarios.

Selección de modelo

Técnicas de modelo a emplear

Ya que nuestra problemática está enfocada en una predicción, seleccionamos los siguientes modelos:

- ★ **Modelo de Regresión Lineal:** Este tipo de modelo describe la relación entre una variable dependiente y una o más variables independientes, para nuestro caso este modelo resulta de utilidad dado que buscamos hacer una predicción de una cantidad. Como ventajas tenemos que es un modelo fácil de implementar, interpretar y entrenar, tiene un desempeño bueno hablando de datos linealmente separados.
- ★ **Random Forest:** Este es un modelo que consiste en varios árboles de decisión, el cual puede ser implementado en problemas de regresión y clasificación. Es un posible modelo ya que puede llegar a mejorar la precisión de las predicciones, reduce problemas de overfitting y de varianza.
- ★ **Gradient Boosted:** Este modelo se basa en crear varias instancias independientes de tipo Random Forest en paralelo, algunos árboles mejores que otros, esto con el propósito de que el siguiente árbol sea capaz de corregir los errores del árbol pasado. Estos modelos son ideales para problemas con decisiones complejas, la desventaja es que estos modelos pueden llevar a overfitting muy rápido.
- ★ **ExtraTree Regressor:** Este modelo se caracteriza por su utilización en la construcción de modelos de clasificación o regresión, en su proceso entrena numerosos árboles de decisión y agrega los resultados del grupo de árboles de decisión para generar una predicción. Cabe mencionar que tiene un sesgo más alto y una varianza más baja que Random Forest, además es más rápido, además de que se reduce en costo computacional.
- ★ **Densely-Connected Neural Network:** Este es un tipo de red neuronal que puede minimizar errores de predicción al ajustar distintas variables del modelo. Este tiene distintas neuronas conectadas entre sí para modificar los pesos aplicados en cada una. Este tipo de red funciona por medio de capas de neuronas que reciben y generan información para las siguientes, esto resulta útil ya que busca encontrar una cantidad por medio de datos previos, el cual es uno de sus propósitos principales.

Generar el diseño de las pruebas

Diseño de las pruebas

Después del procesamiento de datos, se quedaron las siguientes columnas:

- **timestamp:** dentro de esta columna, se describe la hora en formato de 0 a 23, donde 0 es la hora 12 am y 23 es la hora 11.
- **universidad:** La universidad destino del viaje.
- **comuna_origen:** comuna de origen del viaje.
- **tipo_de_viaje:** Si el viaje se lleva a cabo en la misma comuna de la universidad o si es de otra comuna externa.
- **tipo_de_universidad:** Si la universidad es privada o pública.
- **tiene_residencia:** si la universidad tiene residencias para los estudiantes.
- **tamaño_uni:** Área de la universidad en metros cuadrados
- **n_carreras:** Cantidad de carreras por sede.

Para más información sobre cómo se obtuvieron estas columnas y la lista de supuestos que se está tomando en cuenta, se puede consultar el reporte de la Etapa 3, que se encuentra dentro de la carpeta Etapa 3 del github del proyecto.

Los modelos serán contruidos y entrenados con un conjunto de entrenamiento, su calidad estimada con los conjuntos de validación y finalmente se probará el modelo con un conjunto de datos para pruebas.

La división del dataset se determina el 75% de datos para entrenamiento y 25% de datos para pruebas. Del 75% de los datos de entrenamiento se toma un 20% para la validación del modelo.

Las métricas empleadas con las que evaluaremos los modelos son:

- Mean Absolute Percentage Error (MAPE)
- R Square (r²)
- Root mean square error (RMSE) (Para modelos enfocados a árboles)

Dentro de los objetivos de minería de datos que se establecieron en la primera etapa, se habla de dos puntos:

- I. Identificar la comuna de origen que registre los mayores viajes a cada una de las universidades, para el entrenamiento del modelo, se va a tomar en cuenta la cantidad de viajes que se hace de cada comuna hacia las universidades.
- II. Determinar los rangos de horarios donde se hacen más viajes hacia las universidades, para esto, se va a tomar en cuenta la cantidad de viajes que se hace a cada hora en el transcurso de un día.

Construcción del modelo

Modelos

Para todos los modelos se tuvo que realizar one hot encoding para convertir las variables de string a variables categóricas numéricas. De esta manera los modelos aceptarán y entenderán la entrenar.

- **Regresión lineal**

En la implementación del modelo de regresión lineal nos auxiliaremos del modelo de scikit-learn `LinearRegression()`.

- **Random Forest**

Para la implementación de este modelo se utilizó el modelo `tf.keras.RandomForestModel()` de la librería tensorflow decision forests

- **Gradient Boosted**

Para este modelo, se utilizó el `tf.keras.GradientBoostedTreesModel()` de la librería tensorflow decision forests.

- **Extra trees Regressor**

En este modelo, se usó el `from sklearn.ensemble import ExtraTreesRegressor` de la librería sklearn ensemble.

- **Densely-Connected Neural Network (Sequential)**

En la implementación de este modelo se utilizó TensorFlow Keras. Creamos un modelo secuencial para ir añadiendo capas densas y tener neuronas que vayan moviendo sus valores eficientemente a través de la función Adam, funciones de activación y funciones para desactivar aleatoriamente y de forma completa ciertas neuronas para evitar overfitting.

Ajuste de parámetros

Lista de los parámetros y sus valores elegidos para las herramientas de modelado, junto con las razones por las cuales se eligieron esos valores.

| Modelo | Parámetros Iniciales (el primer modelo realizado) | Parámetros finales |
|-------------------|---|---|
| Regression Lineal | positive=False(permite valores negativos) | positive=True(no permite valores negativos) |
| Random Forest | Número de árboles = 73, Profundidad max = 10 | número de árboles = 200, profundidad max = 50 |
| Gradient Boosted | Número de árboles = 73, Profundidad max = 10 | número de árboles = 63, profundidad max = 13 |

| Extra tree Regressor | Max_features = 1.0 | Max_features = 50 |
|----------------------------------|--|---|
| Densely-Connected Neural Network | <p>Número de capas = 1 Número de entradas = 7 Capa 1:</p> <ul style="list-style-type: none"> Capa densa con 7 neuronas, activación relu. <p>Número de salidas = 1</p> | <p>Número de capas = 4 Número de entradas = 7 Capa 1:</p> <ul style="list-style-type: none"> Capa densa con 100 neuronas, activación softmax <p>Dropout de 60%</p> <p>Capa 2</p> <ul style="list-style-type: none"> Capa densa de 80 neuronas, activación relu. <p>Dropout de 20%</p> <p>Capa 3:</p> <ul style="list-style-type: none"> Capa densa con 60 neuronas , activación relu <p>Capa 4:</p> <ul style="list-style-type: none"> Capa densa de 8 neuronas, activación sigmoid. <p>Número de salidas = 1</p> |

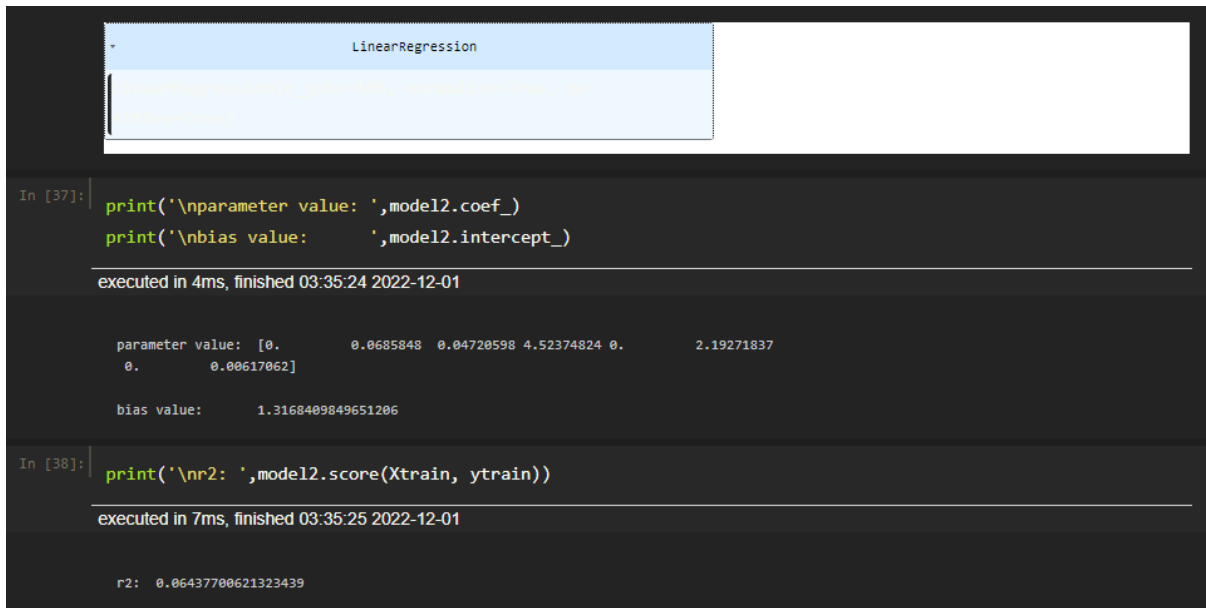
Modelos

Los siguientes modelos se pueden encontrar en este enlace de GitHub:

https://github.com/AnHrt/RetolA/tree/master/Etapa4_Modeling/Modelos

• Regresión Lineal

```
model.fit(Xtrain, ytrain)
```



The screenshot shows a Jupyter Notebook interface. At the top, a variable 'model' is assigned a 'LinearRegression' object. Below this, two print statements are executed to show the model's coefficients and bias. The output shows a vector of coefficients and a scalar bias value. Finally, the model's R-squared score is printed on the training data.

```
LinearRegression
```

```
LinearRegression(n_jobs=100, normalize=True, po
               ...)
```

```
In [37]: print('\nparameter value: ',model2.coef_)
         print('\nbias value:      ',model2.intercept_)

executed in 4ms, finished 03:35:24 2022-12-01

parameter value: [0.          0.0685848  0.04720598  4.52374824  0.          2.19271837
                 0.          0.00617062]

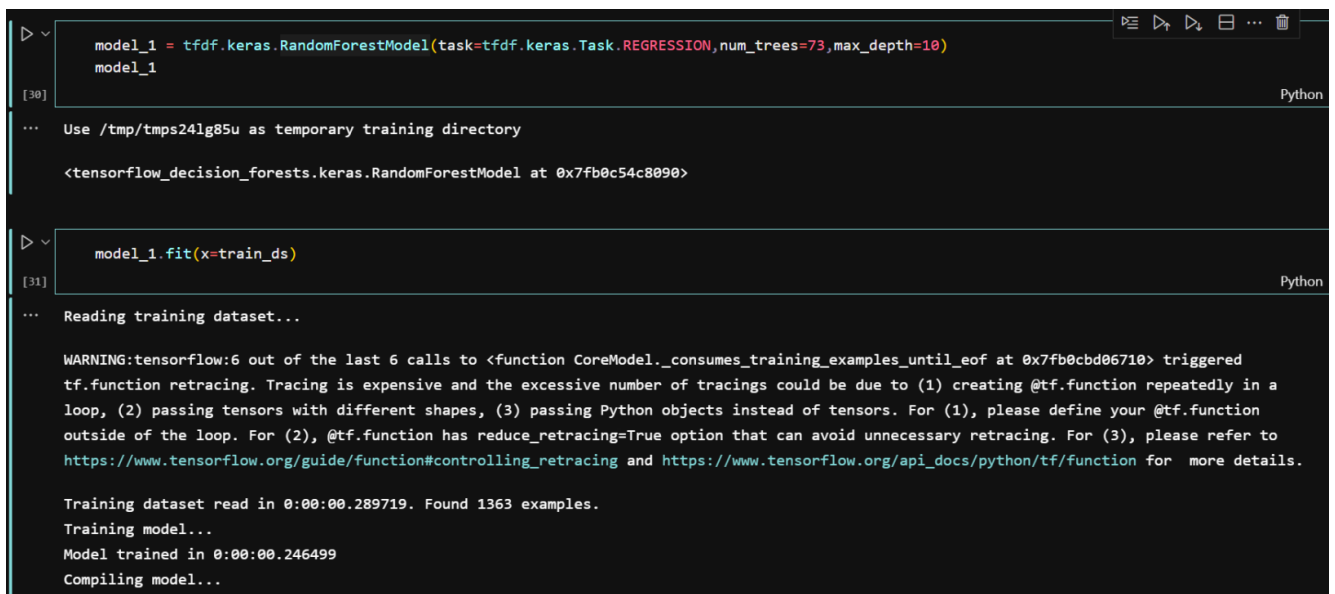
bias value:      1.3168409849651206

In [38]: print('\nr2: ',model2.score(Xtrain, ytrain))

executed in 7ms, finished 03:35:25 2022-12-01

r2:  0.06437700621323439
```

• Random Forest Model



The screenshot shows a Jupyter Notebook interface for training a Random Forest model using TensorFlow Keras. The first cell creates a 'model_1' object with specific parameters. The second cell calls 'fit' on the model with training data. The output shows the model being trained and the R-squared score on the training data.

```
model_1 = tfidf.keras.RandomForestModel(task=tfidf.keras.Task.REGRESSION,num_trees=73,max_depth=10)
model_1
```

```
[30] Python
```

```
... Use /tmp/tmps24lg85u as temporary training directory

<tensorflow_decision_forests.keras.RandomForestModel at 0x7fb0c54c8090>
```

```
model_1.fit(x=train_ds)
```

```
[31] Python
```

```
... Reading training dataset...

WARNING:tensorflow:6 out of the last 6 calls to <function CoreModel._consumes_training_examples_until_eof at 0x7fb0cbd06710> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a
loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

Training dataset read in 0:00:00.289719. Found 1363 examples.
Training model...
Model trained in 0:00:00.246499
Compiling model...
```

- Gradient Boosted

```

model = tfdf.keras.GradientBoostedTreesModel(task = tfdf.keras.Task.REGRESSION,num_trees=63, max_depth=13)

[]

... Use /tmp/tmpvk24yr2s as temporary training directory

model.fit(train_ds)

[]

... Reading training dataset...
Training dataset read in 0:00:00.384075. Found 1360 examples.
Training model...
Model trained in 0:00:03.650146
Compiling model...
Model compiled.

<keras.callbacks.History at 0x7f28f1bd4a50>

print(model.summary())

[]

... Output exceeds the size limit. Open the full output data in a text editor
Model: "gradient_boosted_trees_model_40"

Layer (type)                 Output Shape                 Param #
=====
Total params: 1
Trainable params: 0
Non-trainable params: 1

Type: "GRADIENT_BOOSTED_TREES"
Task: REGRESSION
Label: "__LABEL__"

```

- Extra trees Regressor


```

model = ExtraTreesRegressor(max_features=50)
✓ 0.3s

grid_result = model.fit(X_train, y_train)
✓ 0.4s

print('Model name', model)
from sklearn import metrics
y_pred = model.predict(X_test)
print('Mean Absolute Error (MAE): ', round(metrics.mean_absolute_error(y_test, y_pred), 3))
print('Mean Squared Error (MSE): ', round(metrics.mean_squared_error(y_test, y_pred), 3))
print('Root Mean Squared Error (RMSE): ', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)), 3))
print('R2 Score: ', round(metrics.r2_score(y_test, y_pred), 5))

def MAPE(y_test, y_pred):
    y_test, y_pred = np.array(y_test), np.array(y_pred)
    return np.mean(np.abs((y_test - y_pred) / y_test)) * 100

# Evaluation of MAPE
result = MAPE(y_test, y_pred)
print('Mean Absolute Percentage Error (MAPE): ', round(result, 2), '%')
✓ 0.1s

Model name ExtraTreesRegressor(max_features=40)
Mean Absolute Error (MAE): 1.796
Mean Squared Error (MSE): 15.141
Root Mean Squared Error (RMSE): 3.891
R2 Score: 0.76329
Mean Absolute Percentage Error (MAPE): 52.97 %

```

- **Densely-Connected Neural Network (Sequential)**

```
[81] Python
... Model: "sequential_12"

Layer (type)                 Output Shape              Param #
=====
dense_57 (Dense)              (None, 100)               800
dropout_23 (Dropout)          (None, 100)               0
dense_58 (Dense)              (None, 80)                8080
dropout_24 (Dropout)          (None, 80)                0
dense_59 (Dense)              (None, 60)               4860
dense_60 (Dense)              (None, 8)                 488
dense_61 (Dense)              (None, 1)                 9
=====
Total params: 14,237
Trainable params: 14,237
Non-trainable params: 0
=====

# Lo entrenamos
hist = model.fit(x_train, y_train, epochs=100, validation_data=(x_test, y_test))

[82] Python
... Epoch 1/100
43/43 [=====] - 1s 7ms/step - loss: 85.2166 - MAPE: 100.4604 - val_loss: 82.6014 -
val_MAPE: 100.0050
Epoch 2/100
```

Descripción del modelo

● Regresión Lineal

El modelo consiste en una regresión lineal de mínimos cuadrados ordinarios, es un algoritmo de aprendizaje automático basado en aprendizaje supervisado. La regresión genera predicciones en relación a las variables independientes, se ajusta a un modelo lineal con coeficientes para minimizar la suma residual de cuadrados entre las observaciones en el conjunto de datos y la aproximación lineal. Este modelo se conforma de 5 parámetros con valores por defecto:

- *fit_intercept = True* - Calcula la intercepción del modelo.
- *normalize = False* - Normaliza los valores de la observación en X antes de la regresión.
- *copy_X = True* - Genera una copia de la observación.
- *n_jobs = None* - Proporciona aceleración en la realización de los cálculos.
- *positive = False* - Permite obtener predicciones negativas o limitarlas a valores positivos

● Random Forest

Este modelos se caracteriza por componerse de diferentes árboles de decisiones para entrenar y al finalizar su entrenamiento se promedia su respuesta. Este modelo tiene diferentes parámetros para modificar y poder ajustar el modelo.

- **Gradient Boosted**

Este modelo, es ideal para problemas con decisiones complejas debido a su arquitectura, especializada para reducir el error, por otro lado, con este tipo de modelos el overfitting es un problema muy común. Este modelo toma distintos árboles de decisiones con diferentes parámetros y los conecta paralelamente, esto con el propósito de hacer que el árbol actual sea capaz de aprender de los errores de los árboles anteriores

- **Extra Trees Regressor**

Este modelo usa todo el conjunto de datos para entrenar los árboles de decisión y para garantizar suficientes diferencias entre los árboles individuales y selecciona aleatoriamente los valores en los que dividir una característica y crear nodos secundarios, cuenta con diferentes parámetros para su posible modificación, entre estos se encuentran principalmente el “max_feature” el cual como su nombre lo indica el número de características a considerar al momento de buscar la mejor división y “min_samples” el cual es el número de muestras requeridas en el nodo de una de las hojas del árbol.

- **Densely-Connected Neural Network (Sequential)**

Este modelo secuencial está compuesto por varias capas densas. Lo que hacen estas capas es conectar a todas las neuronas entre sí, para realizar cálculos en base a su valor y peso asignados. Cada capa contiene un determinado número de neuronas y una función de activación que determina si se cumple con un cierto rango de valores para que esta de resultados óptimos.

Dentro de este modelo se aplica una función dropout que hace que no se activen ciertas neuronas de forma aleatoria, de esta manera evitamos que el modelo aprenda sólo de sí mismo y que ajuste otras neuronas para encontrar la mejor solución sin hacer overfitting. Mover las funciones de activación también resulta útil por los tipos de valores que esperas y cómo deseas que vayan cambiando de acuerdo a la forma en la que tus datos están.

Evaluar el modelo Y parámetros revisados y ajustados

Durante la creación de cada uno de los modelos, estuvimos intentando y revisando la manera de obtener buenas predicciones para cumplir con los objetivos del negocio.

Las características de cada modelo en precisión y calidad en cada una de las diferentes pruebas realizadas, considerando también los ajustes de los parámetros fueron los siguientes:

- **Regresión Lineal**

Inicialmente se corrió el modelo con parámetros predeterminados en la librería, recibiendo únicamente el conjunto de datos destinado para el entrenamiento.

Después de entrenar el modelo, evaluamos el desempeño basándonos en las métricas escogidas, identificamos que nuestro MAPE es bajo al arrojar un porcentaje de 1.98%, por lo que definimos que el error de predicción se encuentra alejado un 1.98%. Para concluir con nuestro primer acercamiento, observamos el valor del coeficiente de determinación arrojado que fue de 8%, el cual nos indica el nivel de confianza que puede tener el modelo para explicar el comportamiento de la predicción.

Los resultados de la predicción generan valores negativos, por lo que es necesario implementar que los valores sean únicamente positivos, ya que no se pueden obtener menos viajes.

Modelo 2:

- positive=True
- normalize=True
- n_jobs=100

Se realizaron ajustes de parámetros para encontrar un mejor desempeño. Haciendo una comparación con el primer modelo tenemos un coeficiente de determinación del 5.6%, disminuyendo el nivel de confianza al limitar los valores y obteniendo un MAPE de 1.86% del error de predicción.

- **Random Forest**

Para esta solución se modificaron los parámetros que indican la cantidad de árboles y la profundidad de estos.

Inicialmente, se tenían los siguientes parámetros:

Modelo 1:

- Número de árboles = 73
- Profundidad max = 10

Este modelo no se quedaba corto y su resultados eran deficientes teniendo un MAPE de 92.43% el cual nos indica que en promedio, la predicción de modelos está alejada de la realidad por 92.43%. Así mismo, nos indicaba que el problema probablemente iba a necesitar un modelo un poco más complejo, para esto primero se intentó modificar los parámetros de este Random Forest antes de pasar a otro tipo de modelo.

Modelo 2:

- número de árboles = 200
- profundidad max = 50

Esta versión del modelo, obtuvo mejores resultados al tener un 75.14% de MAPE, el cual nos indica que en promedio, la predicción está alejada de la

realidad por un 75.14%. Este porcentaje no es lo suficientemente bueno, por lo que se intentará con un modelo un poco más complejo.

- **Gradient Boosted**

Igual que en el caso anterior, en este modelo se modificaron los mismos parámetros de cantidad de árboles y su profundidad.

Modelo 1:

- Número de árboles = 73
- Profundidad max = 10

Con esta iteración, se obtuvo un 88.73% MAPE, indicando que la predicción está 88.73% alejada de la realidad. Debido a que este no es un modelo eficiente, se volvió a modificar los parámetros:

Modelo 2:

- número de árboles = 63
- profundidad max = 13

Donde se obtuvo que las predicciones están a 73.31% alejadas de la realidad, aunque el modelo sí mejora un poco en intentar predecir, se tiene que tomar la decisión de tomar un modelo más complejo, debido a que este no está dando el ancho.

- **Extra Trees Regressor**

Tomando esto en cuenta y considerando los posibles cambios se probaron los siguientes parámetros:

Modelo 1:

- Max_Features: 1.0
- Min_samples: 1

Para este modelo se dejaron los parámetros por default, en el cual se vieron resultados considerablemente buenos, ya que se obtuvo un total de 53.71% de MAPE, un valor de R^2 de 0.768 y un RMSE de 3.891. Al probar el modelo se realizaron diferentes predicciones, las cuales fueron un poco más cercanas a lo que se busca predecir.

Modelo 2:

- Max_Features: 40
- Min_samples: 1

Para el segundo modelo se configuró el hiper parámetro de Max_Features, aumentando su valor a 40, dicho cambio mejoró un poco el MAPE, obteniendo un 52.97%. Sin embargo los valores de R^2 y RMSE. Al mejorar el error del modelo, se obtuvieron mejores resultados al momento de realizar

las predicciones.

- **Densely-Connected Neural Network (Sequential)**

Este modelo en un principio comenzó con una sola capa de 7 neuronas y una función de activación relu.

Modelo 1:

- MAPE train: 80.40
- MAPE test: 79.90
- MAPE val: 86.01
- Lr: 0.001
- Epochs: 30

Como podemos observar, los resultados eran deficientes a la hora de entrenar y realizar predicciones.

Con el primer intento de hacer que mejorara el modelo añadimos otra capa con más neuronas con el mismo método de activación y aumentamos el número de épocas a 40 del entrenamiento con la esperanza de que algo cambiara. Esto es lo que obtuvimos:

Modelo 2:

- MAPE train: 78.92
- MAPE test: 76.12
- MAPE val: 78.89
- Lr: 0.001
- Epochs: 40

Observamos que el modelo mejoró, así que partimos de ahí modificando el número de neuronas de la primera capa de 7 a 14, cambiando el método de activación de la segunda capa a sigmoid y aumentando el número de épocas a 50. El resultado para entrenamiento fue mejor, pero ahora para las pruebas y validaciones era peor. Desde ese momento estábamos teniendo overfitting en el modelo.

Modelo 3:

- MAPE train: 71.33
- MAPE test: 79.12
- MAPE val: 79.55
- Lr: 0.001
- Epochs: 50

Ya que el modelo aún no estaba haciendo buenas predicciones y observábamos que todavía faltaba bastante para obtener lo deseado, nos

enfocamos primero en deshacerse del overfitting por lo que implementamos un dropout para desactivar neuronas de forma aleatoria.

Al mismo tiempo, añadimos una tercera capa, modificamos las neuronas de la primera capa a 50, cambiamos el método de activación relu por softmax ya que esta regulariza un los resultados que entran para intentar generar números que salgan positivos.

Las neuronas de la segunda capa fueron modificadas a 40, quitamos la función sigmoid y nos basamos más en relu para obtener números más grandes. Para la tercera capa pusimos 12 neuronas y también el método de activación relu. Entre la primera y la segunda capa fue donde implementamos un dropout ya que de la primera a la segunda capa notamos que se estaba comportando mejor para evitar overfitting. Estos son los resultados que obtuvimos:

Modelo 4:

- MAPE train: 68.45
- MAPE test: 70.56
- MAPE val: 74.34
- Lr: 0.0001
- Epochs: 60

Para este punto, comenzamos a ver que el modelo no mejoraba tanto, las predicciones comenzaban a salir casi siempre en 1, las métricas . Descubrimos que la razón era que tamaño del dataset que no era suficiente para entrenar al modelo, también porque la mayoría de los datos eran de 1 viaje por comuna en 1 hora, por lo que el entrenamiento estaba intentando hacer un fit sobre ese valor al ser el más abundante. Intentamos mejorar la calidad y cantidad de estos datos con correcciones a los algoritmos pero no logramos mucho. Aún así intentamos con otro modelo.

En la última iteración del modelo se introdujo una nueva capa de 8 neuronas y con activación sigmoideal para considerar un aumento de bajos y altos valores, la primera capa cambió a 100 neuronas, la segunda a 80, y la tercera a 60. Se añadió un dropout adicional de 20% entre la segunda y tercera capa. El learning rate se redujo y el número de épocas aumentó a 100. Los resultados mejoraron un poco y de hecho ahora el overfitting era menor. Esto fue lo que obtuvimos:

Modelo 5:

- MAPE train: 51.40
- MAPE test: 45.76
- MAPE val: 45.76
- Lr: 0.00001

- Epochs: 100

Finalmente concluimos que este modelo no podía proveernos la solución, no se generaba algún patrón en concreto o al menos uno que esperábamos y las predicciones que esperábamos nunca se acercaron a lo que teníamos para validar. Necesitábamos tener más datos, no variables, tener un modelo más avanzado o en el peor de los casos regresarnos a cambiar los objetivos de negocio para generar otros datos y predecir alguna otra cosa.

La precisión para generar datos parecidos o acercados a los de validación o incluso con los que había entrenado al modelo estaban muy alejados como para considerar que las predicciones fueran de buena calidad, pues, casi todos los datos generados eran de 1, y al intentar predecir con datos que dieran como resultado un número como 50, apenas quería subir a 2.

Con esto no era posible de ninguna manera crear datos que cumplieran los objetivos de negocio en base a predicciones de manera confiable y cien por ciento verídica.

Debido a que los modelos no tuvieron un desempeño eficaz, los criterios de éxito, tanto de negocio como de minería de datos, no se pueden cumplir al 100% ya que las predicciones tendrán un porcentaje de error relativamente alto, por otro lado se realizará en la siguiente etapa un análisis estadístico que será de utilidad para conocer más acerca de la movilidad que generan las universidades en Chile.

Una vez que se analizó el desempeño de cada uno de los modelos, se determinó que a pesar de que las predicciones no tienen una alta confiabilidad, los modelos que demostraron mejores resultados fueron el modelo de Extra Trees Regression y la Densely-Connected Neuronal Network.

Al realizar este proceso se descubrió que una gran cantidad de personas entran a las zonas universitarias entre las 12 de la mañana a 3:59 de la mañana, a la 1 de la tarde, a las 2:59 de la tarde y a las 9 de la noche hasta las 9:59 de la noche. Además se observaron horarios usuales de movimientos dentro de las zonas entre 12 de la mañana a 3:59 de la mañana, 11 de la mañana a 2:59 de la tarde y de 8 de la noche a 9:59 de la noche. Dentro de estos rangos de horario es donde pudimos encontrar más actividad a partir de la media de todos los viajes por hora.

Pudimos observar que una gran parte de la población viaja a las comunas de zonas universitarias que les quedan cerca y que muy pocas personas son las que viajan a una comuna cuando les queda muy lejos, son casos que apenas tienen registros. Adicionalmente descubrimos que normalmente se registran menos viajes internos que externos al ingresar o cambiar de antena dentro de la zona universitaria.

Comparación de modelos

Durante la construcción de cada uno de los modelos, estuvimos cambiando parámetros constantemente con la finalidad de mejorar la precisión, calidad y confiabilidad de las predicciones. Realizamos varias iteraciones para encontrar el mejor modelo e intentar alcanzar los objetivos de negocio.

| Modelo | HyperParametros | MAPE | R ² | RMSE |
|----------------------------------|--|-------|----------------|-------|
| Regresión lineal | <i>fit_intercept=True</i> <i>normalize=False</i> <i>copy_X=True</i> <i>n_jobs=None</i> <i>positive = False</i> | 1.98 | 0.08 | 6.59 |
| | <i>positive=True</i> <i>normalize=True</i> <i>n_jobs=100</i> | 1.86 | 0.056 | 6.65 |
| Extra Trees Regressor | Max_features = 1.0 | 53.71 | 0.7687 | 3.846 |
| Extra Trees Regressor | Max_features = 40 | 52.97 | 0.7692 | 3.891 |
| Random Forest Model | Número de árboles = 73, Profundidad max = 10 | 92.53 | - | 13.86 |
| | Número de árboles = 200, profundidad max = 50 | 75.14 | - | 13.86 |
| Gradient Boosted | Número de árboles = 73, Profundidad max = 10 | 88.73 | - | 4.5 |
| | Número de árboles = 63, profundidad max = 13 | 73.31 | - | 4.5 |
| Densely-Connected Neural Network | Número de capas = 4 Número de entradas = 7 Capa 1: | 45.76 | - | 9.83 |

| | | | | |
|--|--|--|--|--|
| | <ul style="list-style-type: none"> • Capa densa con 100 neuronas, activación softmax <p>Dropout de 60%</p> <p>Capa 2</p> <ul style="list-style-type: none"> • Capa densa de 80 neuronas, activación relu. <p>Dropout de 20%</p> <p>Capa 3:</p> <ul style="list-style-type: none"> • Capa densa con 60 neuronas, activación relu <p>Capa 4:</p> <ul style="list-style-type: none"> • Capa densa de 8 neuronas, activación sigmoid. <p>Número de salidas = 1</p> | | | |
|--|--|--|--|--|