Entrega final proyecto

Por:

Luis David Morales Aguilar

Materia:
Introducción a la inteligencia artificial

Profesor:

Raul Ramos Pollan

UNIVERSIDAD DE ANTIOQUIA
FACULTAD DE INGENIERÍA

MEDELLÍN

1. Introducción.

Para entrar en un mercado y ser competitivo, las empresas deben realizar estudios sobre el producto que quieren comercializar en determinada región. Estos estudios se basan en las necesidades que deben suplir al cliente con los artículos que ofrecen, y al mercado existente en la zona por parte de otras empresas. Para que una empresa de automóviles extranjera pueda entrar al comercio de autos local, deben entender los factores que afectan el precio de los autos locales ya que se pueden presentar diferencias significativas con los factores propios. Para esto deben identificar las variables más significativas y así tener la posibilidad de predecir el precio de un vehículo. Una vez se identifica esto, se ajusta un valor adecuado al producto con el fin de garantizar una ganancia al fabricante y a su vez generar competitividad en el mercado.

1.1 Dataset

El dataset a utilizar se encuentra en la plataforma Kaggle(https://www.kaggle.com/datasets/hellbuoy/car-price-prediction?select=CarPrice_Assignment.csv) con el título de "Car price prediction multiple linear regressión".Este dataset consiste en información de las variables más relevantes que sedeben tener en cuenta para definir el precio de un vehículo, contando con 26columnas y 205 filas. Las columnas se identifican como sigue:'car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration','doornumber', 'carbody','drivewheel','enginelocation','wheelbase', 'carlength', 'carwidth', 'carheight','curbweight','enginetype', 'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio','stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg','price'.Dentro de los archivos que se pueden obtener de Kaggle, se encuentra unollamado "CarPrice_Assignment.csv", en el cual se encuentra toda la informaciónnecesaria para abrir y utilizar el dataset en pandas.

1.2 Métricas

Una de las métricas a emplear es el "Root Mean Squared Logarithmic Error" (RMSLE), seleccionada para indicar el error porcentual de acierto entre los datos reales y los datos obtenidos por predicciones, la cual indica que entre menor sea el valor obtenido, menor es la separación entre datos reales y obtenidos, y mayor precisión. La ecuación que la describe es la que sigue:

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(log(p_i+1) - log(a_i+1)\right)^2}$$

Dónde n es el número total de observaciones en el dataset, pi es la predicción de la variable objetivo y ai es el valor real de la variable objetivo.

La otra métrica, consiste en el R^2, la cual otorga una indicación de qué ten precisa es la predicción haciendo uso de la porción de datos utilizada para test.

Para el precio de los vehículos, una empresa requiere que el error de predicción sea igual o inferior al 30%, por lo que el precio puede ser 30% superior o inferior al original. Como métrica de negocio se espera que el error en la predicción sea el más bajo posible, y superior al valor original, para garantizar ganancia y competitividad en el mercado de vehículos local.

1.3 Importación de datos y preparación.

Para llevar a cabo el correcto funcionamiento de los datasets, se realiza inicialmente la importación de los datos directamente desde Kaggle utilizando el archivo kaggle.json, y su preparación para el trabajo posterior.

1.4 Descripción y preparación del dataframe.

Se procede a la eliminación de datos de las columnas 'aspiration', 'drivewheel', 'fuelsystem' y 'doornumber'. Estas columnas se seleccionan debido a que proporcionan datos muy específicos que no afectan en gran medida el valor de los vehículos. Uno de los requisitos para trabajar con el dataset es que tenga un faltante del 5% en datos, ya que este no cumple con lo pedido se procede a eliminar este porcentaje de datos, por lo que se hace el cálculo en función de los datos disponibles, el cual es 205 (cantidad de filas) x 30 (cantidad de columnas), dando un valor de 308 datos que deben ser eliminados para cumplir con el requisito. De las columnas anteriormente mencionadas, se eliminan de forma aleatoria algunos valores hasta llegar a la cantidad de datos faltantes pedida.
El dataset original sólo contenía 25 columnas, por lo que se adicionan columnas sintéticas basadas en las existentes, realizando operaciones matemáticas sobre los datos las cuales se indican en el notebook. A continuación, se muestra una captura de las líneas de código utilizadas.

```
ds1= dc.assign(Log_peakrpm=log_pr)
ds2= ds1.assign(Log_wheelbase=log_wb)
ds3= ds2.assign(Square_root_wp=sqrt_hp)
ds4= ds3.assign(Exp2_stroke=exp_st)
dsf= ds4.assign(Compressionratio_norm=comp_nor)
```

En esta sección también se describe el df original, con la cantidad de datos y el tipo de los mismos.

1.5 Limpieza de datos.

Antes de comenzar a manipular el dataframe, se realiza una limpieza de datos con el fin de llenar los datos faltantes con otros que puedan llegar a ser útiles. La siguiente captura describe las líneas de comando utilizadas para dicho fin:

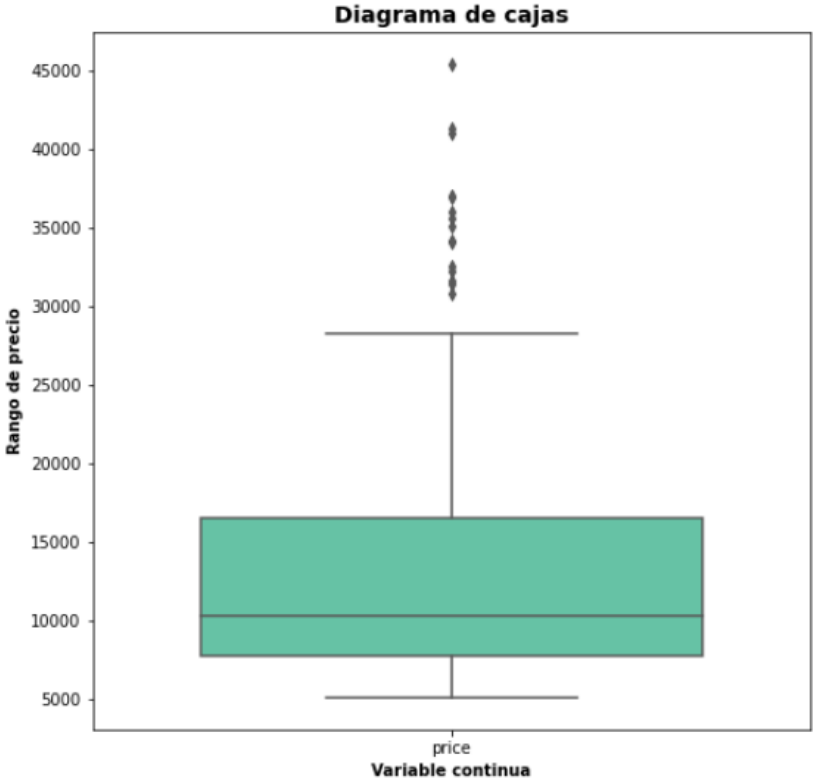
```
ds["aspiration"]=ds.aspiration.fillna("Unk")
ds["drivewheel"]=ds.aspiration.fillna("Unk")
ds["fuelsystem"]=ds.aspiration.fillna("Unk")
ds["doornumber"]=ds.doornumber.fillna(int(np.mean(ds["doornumber"]))+1)
ds
```

Los datos de las columnas "aspiration", "drivewheel" y "fuelsystem" contienen información muy específica de los autos que no necesariamente afectan o tienen una correlación con su precio, por lo cual los datos faltantes se llenaron con el dato "UnK" que siganifica Unknown o desconocido en español. Por esta razón se utilizan otras variables para predecir el precio de los vehículos. En la columna "doornumber" los datos faltantes se completaron con su promedio, el cual dio de 3.14, pero debido a que el dato debe ser entero (ya que un carro comúnmente tiene un numero entero de puertas) se convirtió en entero, danto un valor de 3, a lo que se realiza una suma de 1 para así obtener un valor de 4. Esta decisión se debe a que los vehículos de 4 puertas son más comunes, por ello el promedio estaba más cerca de 4 que a 2. Luego se convierten los datos de la columna doornumber de letras a números, los four y two por 4 y 2.

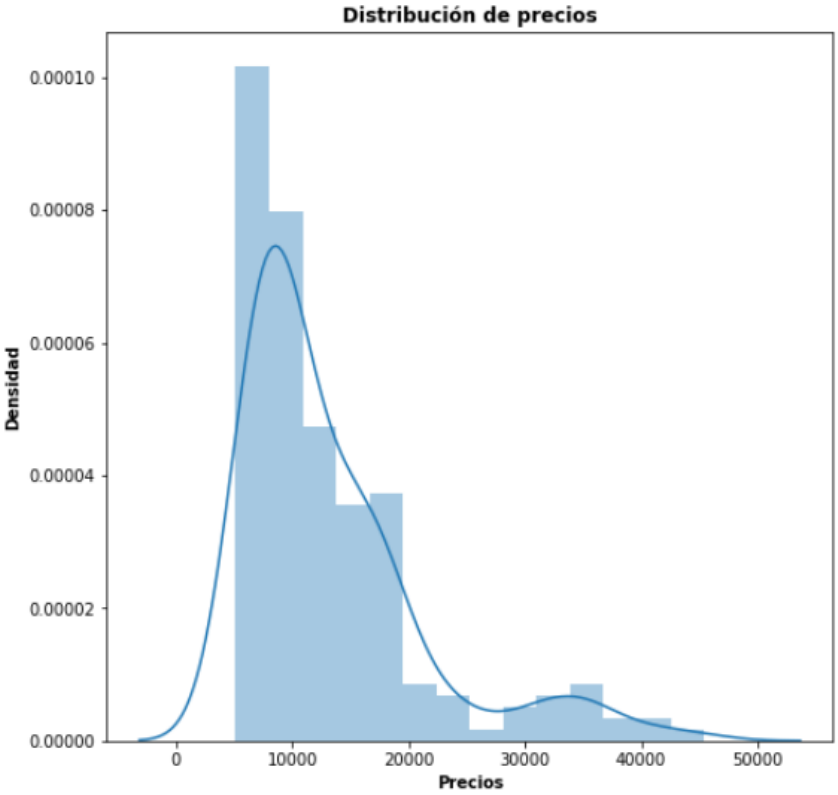
2. Exploración descriptiva del dataset.

En esta sección se da una descripción gráfica de los datos, haciendo uso de las herramientas estadísticas como las cajas de bigotes, diagrama de distribución y diagramas de correlación.

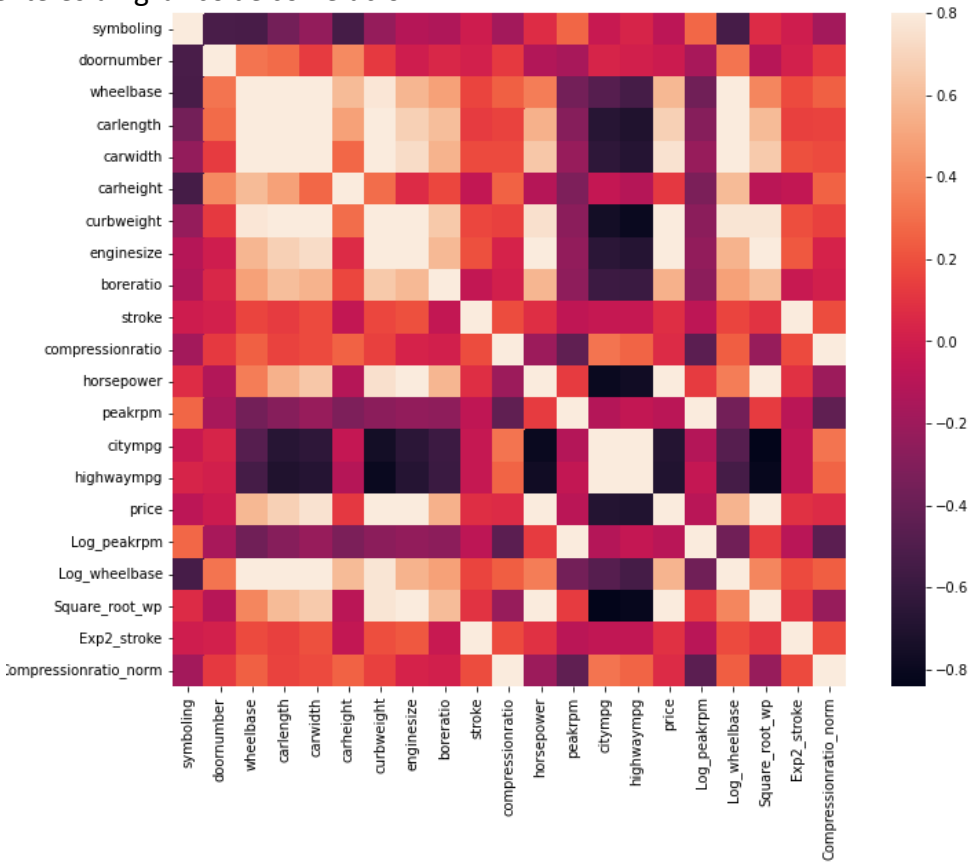
Haciendo uso de un diagrama de cajas y bigotes, se observan algunos datos atípicos de la variable de precios, tal y como se muestra a continuación.



Realizando un gráfico de distribución de precios, se puede determinar que es una distribución no tan abierta, en la cual se evidencia que la mayoría de los vehículos poseen un precio inferior a 18000.



El siguiente es un gráfico de correlación:



El gráfico anterior muestra la correlación entre las variables numéricas del dataframe, en el cual se evidencian varias cosas:
- La distancia entre ejes posee una alta relación con la longitud del vehículo, el ancho y la altura. En el siguiente gráfico se puede observar que entre mayor la separación de los ejes, mayores las demás dimensiones del vehículo, algo que tiene sentido ya que se requiere proporcionalidad en el vehículo, tanto por temas de funcionalidad como estéticos. El logaritmo de este valor posee las mismas correlaciones.
- El tamaño del motor también posee una correlación con las dimensiones del vehículo, pero su mayor relación esta con el peso y los caballos de fuerza. En el siguiente gráfico se evidencia que entre mayor el tamaño del motor, mayor peso y también una tendencia a incrementar la cantidad de caballos de fuerza.
- El precio, que es la variable de interés, está fuertemente relacionada con los caballos de fuerza, el tamaño del motor y las dimensiones del vehículo. En el siguiente gráfico, se nota una tendencia de aumento de la variable precio en función a las anteriormente descritas. Mayores dimensiones del vehículo requieren un motor más grande debido a un mayor peso, y por ende mayor cantidad de caballos de fuerza y a su vez un incremento en el precio.

2.1 Conversión y visualización de variables categóricas.

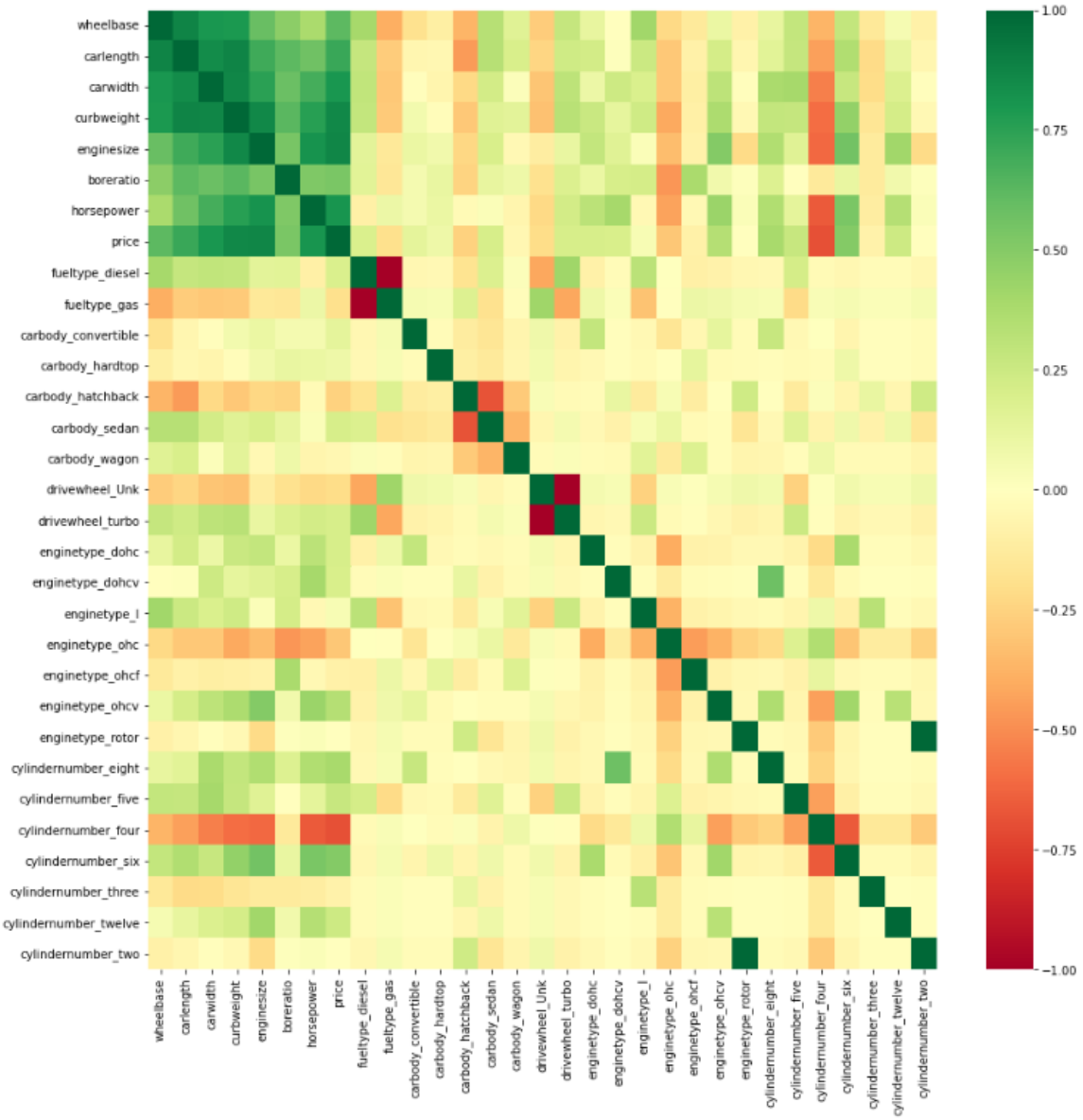
En esta sección se eliminan las columnas:
'fueltype','carbody','drivewheel','enginetype','cylindernumber','Log_wheelbase','Log_peakrpm','Square_root_wp','Exp2_stroke','Compressionratio_norm','symboling','aspiration','drivewheel','fuelsystem','doornumber','carheight','stroke','compressionratio','peakrpm','citympg','highwaympg','CarName','enginelocation'
Ya que algunas columnas contienen información que es muy específica y no aporta mucho a la predicción y otras no tienen información relacionada directamente con la variable 'price'.
Posteriormente, las columnas:
'fueltype','carbody','drivewheel','enginetype','cylindernumber' se convierten de columnas con datos categóricos a columnas con información de cada uno de los descriptores, tal como se puede apreciar en la siguiente imagen:

	fueltype_diesel	fueltype_gas	carbody_convertible	carbody_hardtop	carbody_hatchback	carbody_sedan	carbody_wagon	drivewheel_Unk	drivewheel_turbo
0	0	1	1	0	0	0	0	1	0
1	0	1	1	0	0	0	0	1	0
2	0	1	0	0	1	0	0	1	0
3	0	1	0	0	0	1	0	1	0
4	0	1	0	0	0	1	0	1	0

2.2 Separación y normalización de datos.

En esta sección se normalizan los datos entre 0 y 1, con el fin de eliminar los problemas de escala entre los datos y obtener así una mejor comparabilidad entre estos. Luego se hace la separación de los datos de train y test de forma aleatoria. Con los datos de train se realiza

otro mapa de correlación, fijando la variable Price como la de interés. De este gráfico se determina que las columnas más importantes para predecir el valor de los vehículos son las siguientes:
'wheelbase','carlength','carwidth','curbweight','enginesize','boreratio','horsepower','carbody_hatchback' ,'enginetype_ohcv','price'



Por esto se reacondiciona el df a estar solo con estas 10 columnas.

3. Iteraciones de desarrollo.

3.1 Modelos supervisados.

3.1.1 Partición de datos.

Para poder utilizar los modelos y realizar predicciones y entrenamiento se deben separar aleatoriamente los datos en dos grupos, uno de test y otro de entrenamiento. Para esto se

hace uso del siguiente código, teniendo en cuenta que el 70% de los datos harán parte del entrenamiento y el 30 de test.

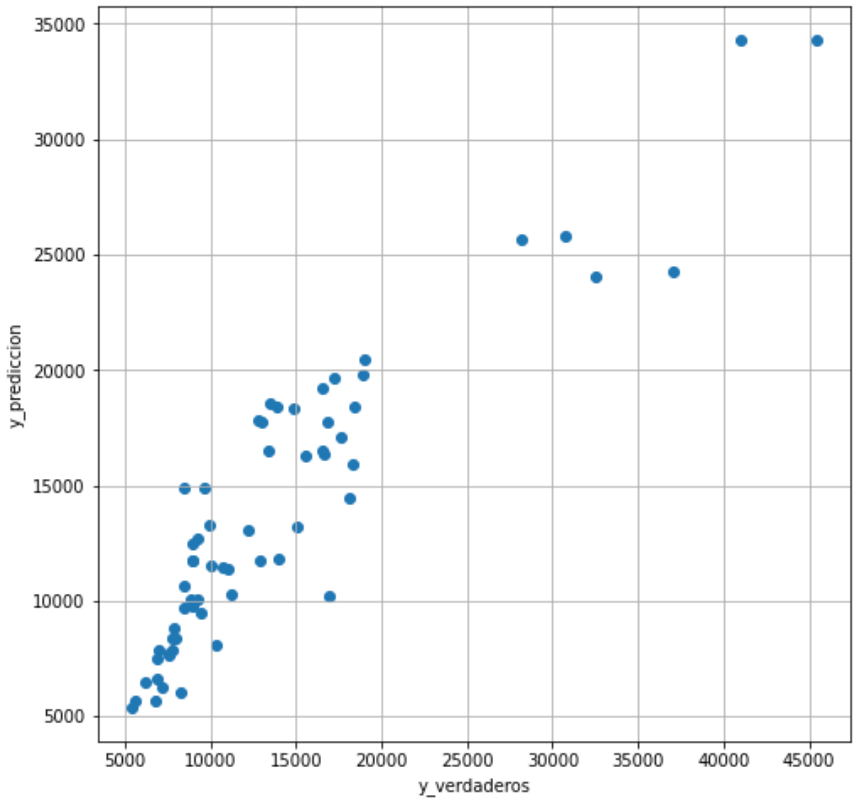
```
[ ]  y1 = X1.pop('price')
```

```
#--------Partición de los datos normalizados------------

from sklearn.model_selection import train_test_split

test_size = 0.3
val_size  = test_size/(1-test_size) # Elementos de validación


print (X1.shape, y1.shape)
print ("test size %.2f"%test_size)
print ("val size is %.2f (relative to %.2f) "%(val_size, 1-test_size))

Xtr, Xts, ytr, yts = train_test_split(X1, y1, test_size=test_size)
print (Xtr.shape, Xts.shape)
```

```
(205, 9) (205,)
test size 0.30
val size is 0.43 (relative to 0.70)
(143, 9) (62, 9)
```

3.1.2 Modelo lineal

El primer modelo utilizado fue el lineal. Con este primer modelo se obtuvo un coeficiente de error cuadrático de 0.81 para los datos de test y un 0.84 para datos de train. Además, el RMSLE es de 0.487 para train y 0.2169 para test. Esta fue una primera aproximación a la predicción de los precios en los vehículos. Posteriormente se implementan otros métodos supervisados que pretenden mejorar la precisión en cuanto a la predicción. A continuación se muestra la gráfica de los datos predichos vs los datos reales para el modelo lineal.




```
lr = LinearRegression()
lr.fit(Xtr, ytr)
lr.score(Xtr, ytr), lr.score(Xts, yts)
```

```
(0.8462975550082843, 0.8182875710609149)
```

```
[145] RMSLE(lr, Xtr, ytr), RMSLE(lr, Xts, yts)
```

```
(0.4287203175670387, 0.21692664959280258)
```

3.1.3 Modelo Decision Tree Regresor

Se decide trabajar con este modelo ya que presenta mejor desempeño aparente en cuanto a la predicción de precios. Para poder utilizarlo se define el mejor hiperparámetro de profundidad utilizando el siguiente código.

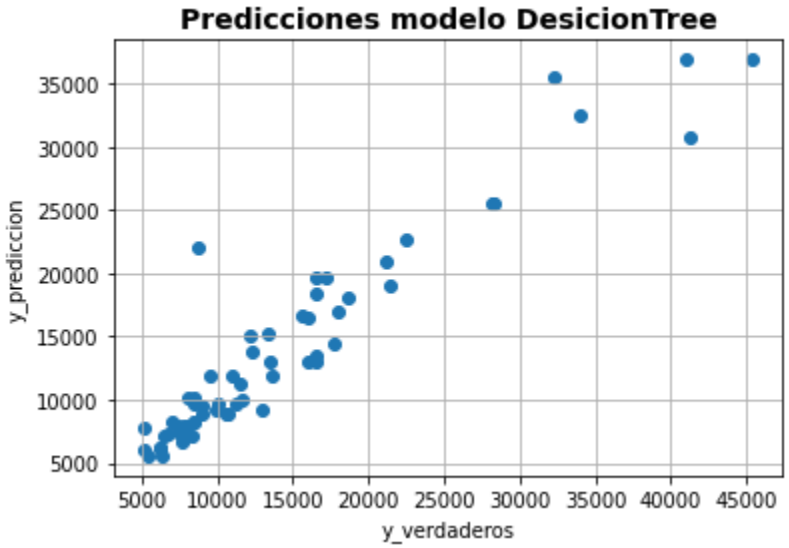
```
decission_tree = GridSearchCV(estimator = estimator2,
                          param_grid = parametros,
                          cv = ShuffleSplit(n_splits= 10, test_size=val_size),
                          scoring = 'neg_mean_squared_error',
                          verbose = 1,
                          return_train_score = True,
                          n_jobs = -1)

decission_tree.fit(Xtr, ytr)
```

```
Fitting 10 folds for each of 4 candidates, totalling 40 fits
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=None, test_size=0.4285714285714286,
       train_size=None),
             estimator=DecisionTreeRegressor(max_depth=5), n_jobs=-1,
             param_grid={'max_depth': [2, 5, 11, 17]}, return_train_score=True,
             scoring='neg_mean_squared_error', verbose=1)
```

```
33] print("Mejor estimador Decision Tree: ",decission_tree.best_estimator_)
    print("Mejores parámetros para el estimador Decision Tree: ", decission_tree.best_params_)
```

```
Mejor estimador Decision Tree:  DecisionTreeRegressor(max_depth=17)
Mejores parámetros para el estimador Decision Tree:  {'max_depth': 17}
```

Una vez se determinó el mejor hiperparámetro, se implementó en el modelo final obteniendo un RMSLE de 0.01 en train y 0.11 en test. Ahora el R^2 fué de 0.99 en train y 0.89 en test, evidenciando una mejora significativa en el cuanto a la precisión de predicción. A continuación se observa la gráfica de valores reales vs predichos.




```
[85] Des_tree.score(Xtr, ytr), Des_tree.score(Xts, yts)
```

```
(0.9999843600878122, 0.8935624527927633)
```

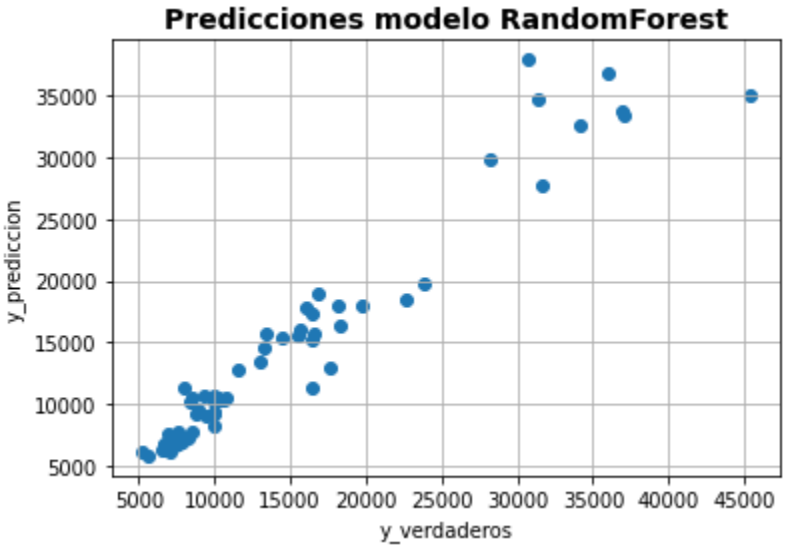
3.1.4 Modelo Random Forest

Para este modelo también se buscaron los mejores hiperparámetros entorno a la cantidad de estimadores y profundidad. Los resultados obtenidos varían debido a que en cada ciclo se eligen datos diferentes. Para este caso el resultado fue de 5 estimadores y una profundidad máxima de 10.

```
Rand_forest= RandomForestRegressor(n_estimators = 5,max_depth = 10)
Rand_forest.fit(Xtr, ytr)
```

```
RandomForestRegressor(max_depth=10, n_estimators=5)
```

Haciendo uso de estos parámetros, se realizaron las predicciones obteniendo un RMSLE de 0.07 en train y 0.12 en test. El R^2 para train es de 0.97 y para test es de 0.93, obteniéndose una mejora enorme en comparación con el modelo lineal. La gráfica de datos reales vs predichos se presenta a continuación.




```
print(Rand_forest.score(Xtr, ytr),Rand_forest.score(Xts, yts))
```

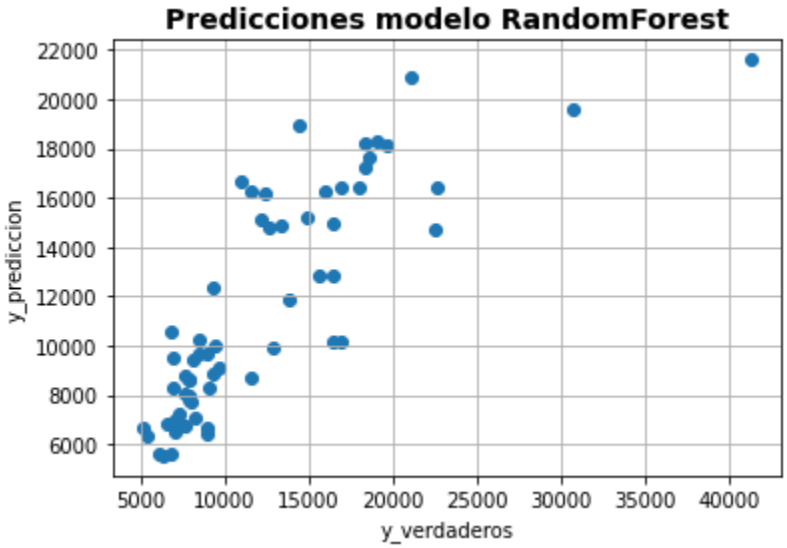
```
0.97112962743173 0.9347431679228212
```

3.2 Modelos no supervisados

3.2.1 PCA

Inicialmente se determinó la mejor cantidad de componentes con un RandomForest fijo (el mejor de los modelos supervisados). Este valor fue de 7. Posteriormente con este dato, se buscan los mejores hiperparámetros para el modelo RandomForest a partir de las muestras transformadas por el PCA utilizado con los componentes anteriormente definidos.

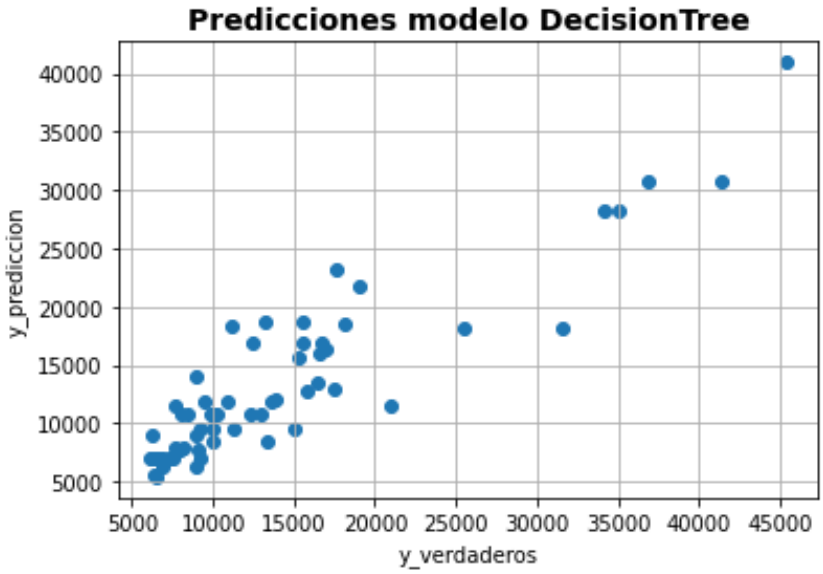
Teniendo definidos todos los parámetros, se hizo uso del modelo RandomForest obteniendo un RMLSE de 0.07 para train y 0.22 para test. El R^2 para train es de 0.98 y para test es de 0.64. Se observa que no se obtiene un buen resultado con este método, ya que incluso es inferior a los valores obtenidos por el método lineal. La gráfica de valores reales vs predichos se muestra a continuación.




```
RMSLE del Random Forest en entrenamiento:  0.07197
RMSLE del Random Forest seleccionado:  0.22523
Score train = 0.98274 Score test = 0.64543
```

3.2.2 NMF

Con la finalidad de hacer uso de este modelo, se debe definió la mejor cantidad de compones a utilizar, por lo que se realizaron varias iteraciones, al igual que en los modelos anteriores, y así se determinó el valor correspondiente. Para este modelo, la cantidad de componentes que presenta el mejor RMSLE es la asociada al valor de 3. Cabe aclarar que se hace uso también del modelo DesiciónTree con una profundidad máxima constante, definida a partir del mejor resultado de hiperparámetros de los modelos supervisados. Con los componentes definidos, se recalculan los hiperparámetros para el modelo DesiciónTree haciendo uso de los datos transformados. Una vez realizado esto se hace uso del modelo para generar las predicciones. El RMSLE en test fue de 0.23 y el R^2 de 0.81.



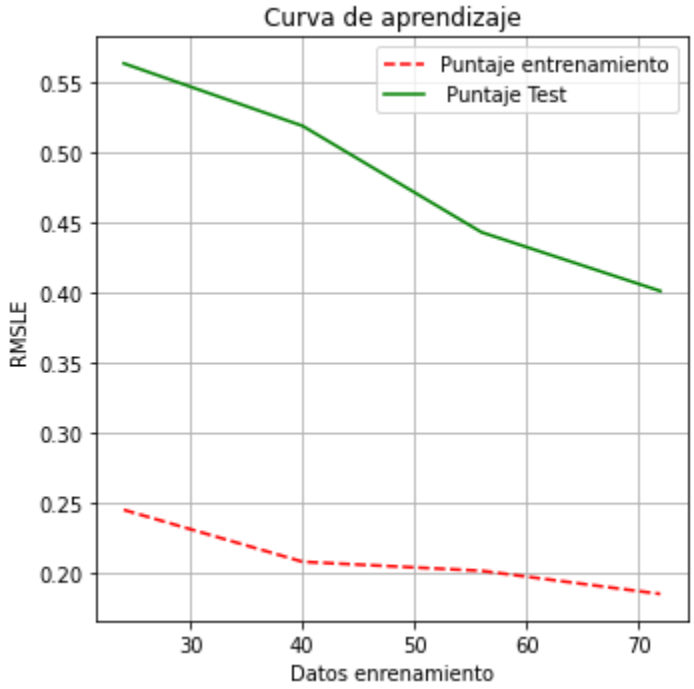

```
RMSLE del Decision Tree en entrenamiento:  0.03783
RMSLE del Decision Tree seleccionado:  0.23453
Score train = 0.99409 Score test = 0.81013
```

4. Curvas de aprendizaje

Una vez se trabajó con los modelos, se realizan curvas de aprendizaje para determinar el comportamiento de los modelos en función de la cantidad de datos utilizados y la métrica de RMSLE. A continuación, se mostrarán las curvas para algunos modelos.

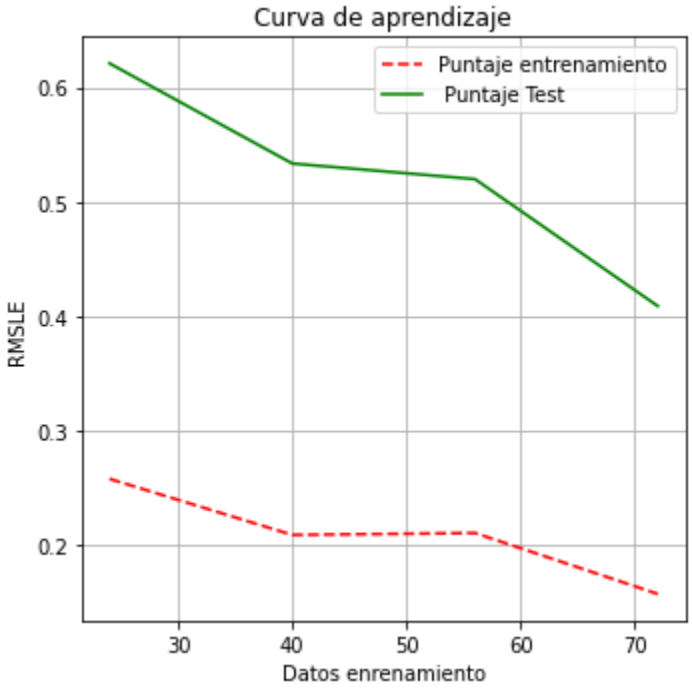
4.1 Métodos no supervisados

4.1.1 DecisionTree



Esta curva corresponde al modelo Decision Tree, y se observa que a medida que se aumenta la cantidad de datos en entrenamiento, disminuye el RMSLE en Test, mientras que la tendencia del entrenamiento se mantiene mucho más constante.

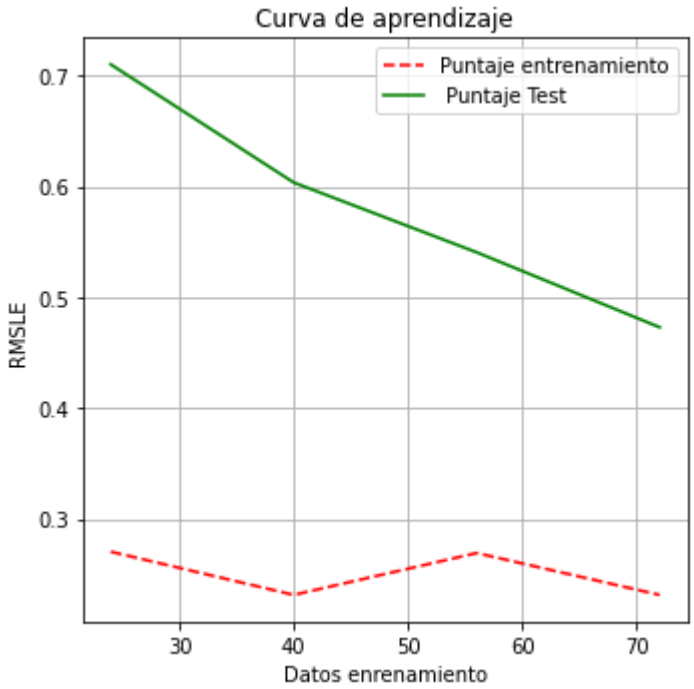
4.1.2 *RandomForest*



Esta curva corresponde al modelo RandomForest (con los mejores estimadores de los modelos supervisados). Se observa una tendencia de reducción en los valores del RMSLE para los datos de test a medida que aumentan los datos en entrenamiento.

4.2 *Métodos no supervisados*

4.2.1 PCA + RandomForest



Esta curva corresponde al modelo PCA + RandomForest, se evidencia que a medida que aumenta la cantidad de datos en entrenamiento, disminuye el valor de la métrica de RMSLE para los datos de test.

4 Retos y consideraciones de despliegue.

Con el fin de probar el desempeño del modelo, se deben conseguir más datos que se encuentren acorde a los utilizados para la producción de los modelos. Las variables para tener en cuenta pueden centrarse sólo en las de mayor peso identificadas en el trabajo, así se evitaría la obtención de datos innecesarios para este objetivo, lo cual significa un ahorro económico. El modelo entonces debe cumplir con una buena estimación del precio de un vehículo en función de las características más importantes previamente definidas. La estimación debe tener la menor cantidad de error, para que así, la empresa china interesada en incurrir en el mercado de vehículos estadounidenses pueda establecer precios competitivos y ubicarse en la competencia de los vendedores de vehículos locales.

Para que haya un despliegue del modelo, es necesario entonces que sea evaluado con una mayor cantidad de datos, y así nuevamente definir su desempeño en función de las métricas puestas. Una vez se tenga esta información, se replantea la cuestión de la viabilidad del modelo. Cabe aclarar que los modelos supervisados presentaron valores de RMSLE más bajos que los modelos no supervisados, y además, también, valores de R^2 más cercanos a 1. A parte, se observa que la tendencia en los resultados es más constante en los modelos supervisados que en los no supervisados, debido a que en cada iteración del modelo, se toman diferentes datos, por lo que varían los resultados. Teniendo en cuenta esto, se elige el mejor modelo para plantear en el despliegue de evaluación de datos, que genere buenas predicciones de acuerdo con las métricas definidas para esta situación.

5 Conclusiones.

No toda la información que se tiene respecto a un vehículo sirve para determinar su precio. Por tal motivo, dentro de los datos dados, se deben elegir las columnas con las variables que presenten mayor influencia en cuanto al valor del precio en los vehículos.

Los modelos se deben ajustar a variables óptimas, esto quiere decir que se pretende evitar un overfitting o un bias, por lo que se evalúan dentro de un loop, haciendo uso de diferentes parámetros y de la métrica RMSLE para determinarlos.

Los modelos supervisados presentan buena precisión a la hora de predecir precios a vehículos comerciales, y su varianza en cada iteración del Notebook es menor comparada con los modelos no supervisados.

6 Referencias.

[1] CarPrice Prediction MLR+RFE+VIF. (2022). Tomado de: https://www.kaggle.com/code/hellbuoy/carprice-prediction-mlr-rfe-vif