Laboratorio No. 3

Leidy D. Galindo Acuña

Diego N. Garcia Vargas

Marco D. Suarez Berdugo

Universidad ECCI

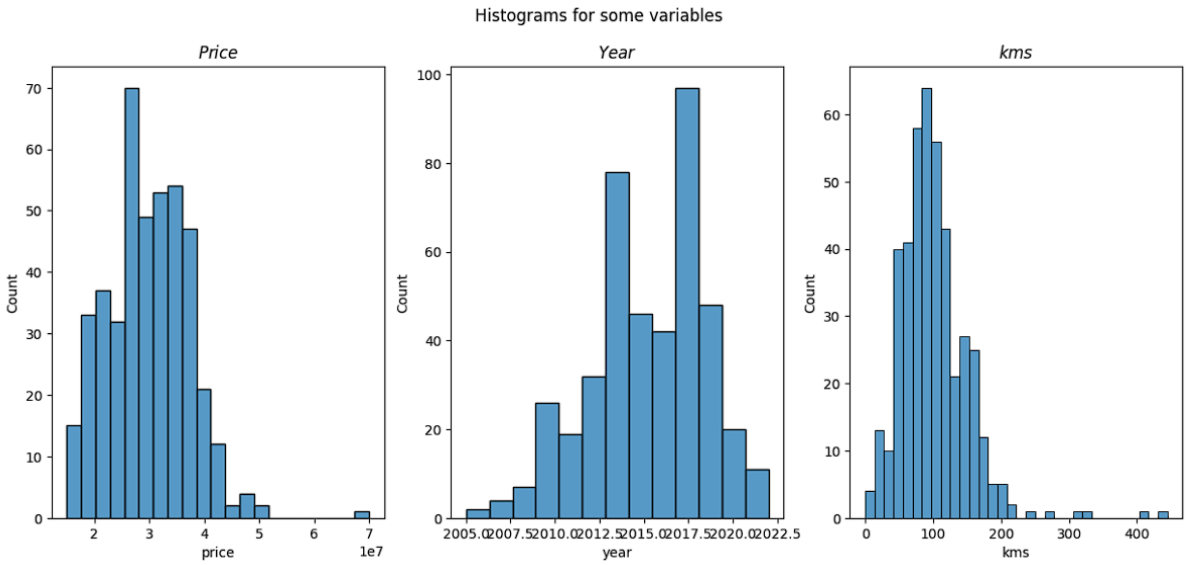
Seminario Big Data y Gerencia de datos

LABORATORIO #3

Desarrollo

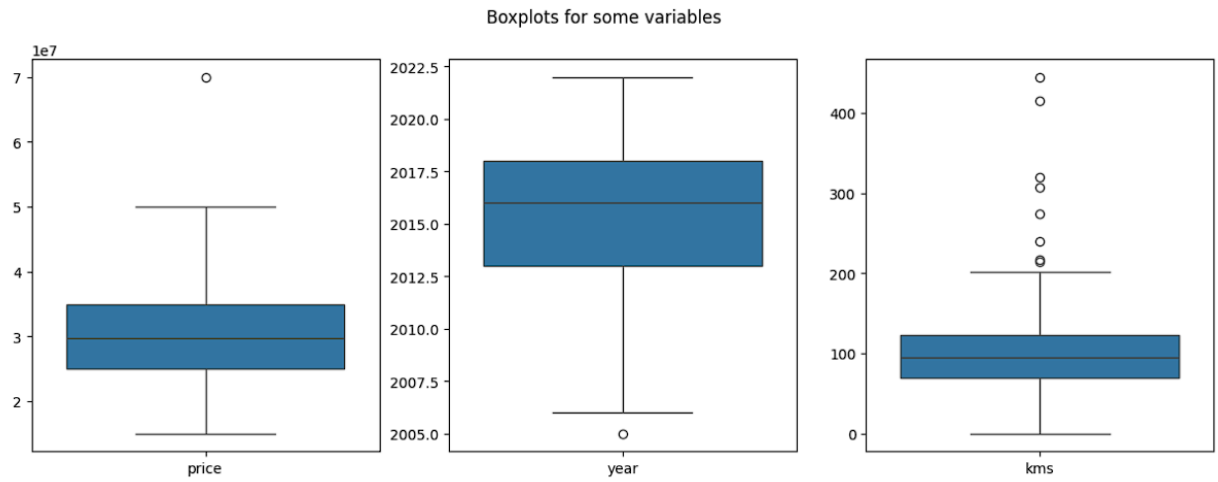
Se realizó el web scraping con la marca Chevrolet y el modelo Spark dando como resultado un archivo CSV de 432 registros.

En el Test1 se cargaron los registros sin hacer ningún filtro, dando como resultado los siguientes histogramas y Boxplots:



Estos histogramas muestran la distribución de los precios, años y kilometrajes de los carros Chevrolet Spark. El histograma de precios muestra que la mayoría de los carros Chevrolet Spark se vendieron por un precio entre $15,000,000 y $38,000,000. Por otro lado, el histograma de años muestra que la mayoría de los carros Chevrolet Spark se vendieron entre 2010 y 2017. Por último, en el histograma de kilometrajes muestra que la mayoría de los carros Chevrolet Spark se vendieron con menos de 100,000 kilómetros.

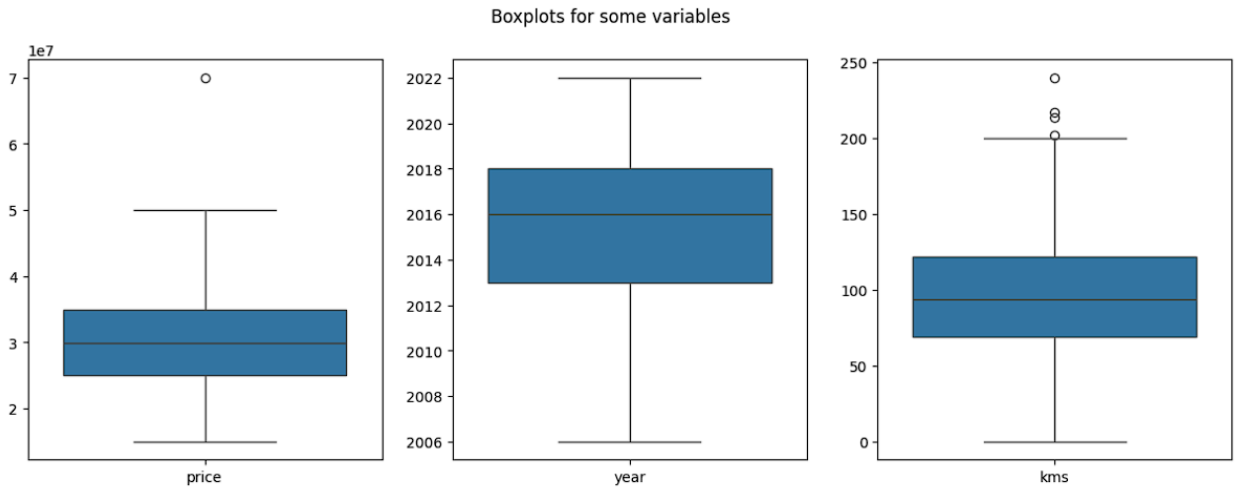
LABORATORIO #3



En el primer Boxplot de Precios se observa que hay un valor atípico significativo por encima de los 70 millones. Este valor es considerablemente mayor que el resto de los datos y podría indicar una venta excepcionalmente alta, posiblemente debido a un modelo especial o un conjunto de características únicas. En el Boxplot de los años del modelo hay un valor atípico notable por debajo del año 2005. Este podría ser un caso de un modelo muy antiguo que todavía está en venta. Para finalizar en el Boxplot de Kms hay varios valores atípicos por encima de 400,000 kilómetros.

Después del tratamiento de los outliers o valores atípicos se evidencia una disminución de estos valores en los Boxplots:

LABORATORIO #3



Para el Test2 se tomó en cuenta el histograma de precios muestra que la mayoría de los carros se venden por un precio entre 20 y 40 millones de pesos. Hay algunos carros que se venden por menos de 20 millones, y muy pocos los que se venden por más de 60 millones. Por lo tanto, se decidió tomar sólo los vehículos que se vendieron por menos de 60 millones.

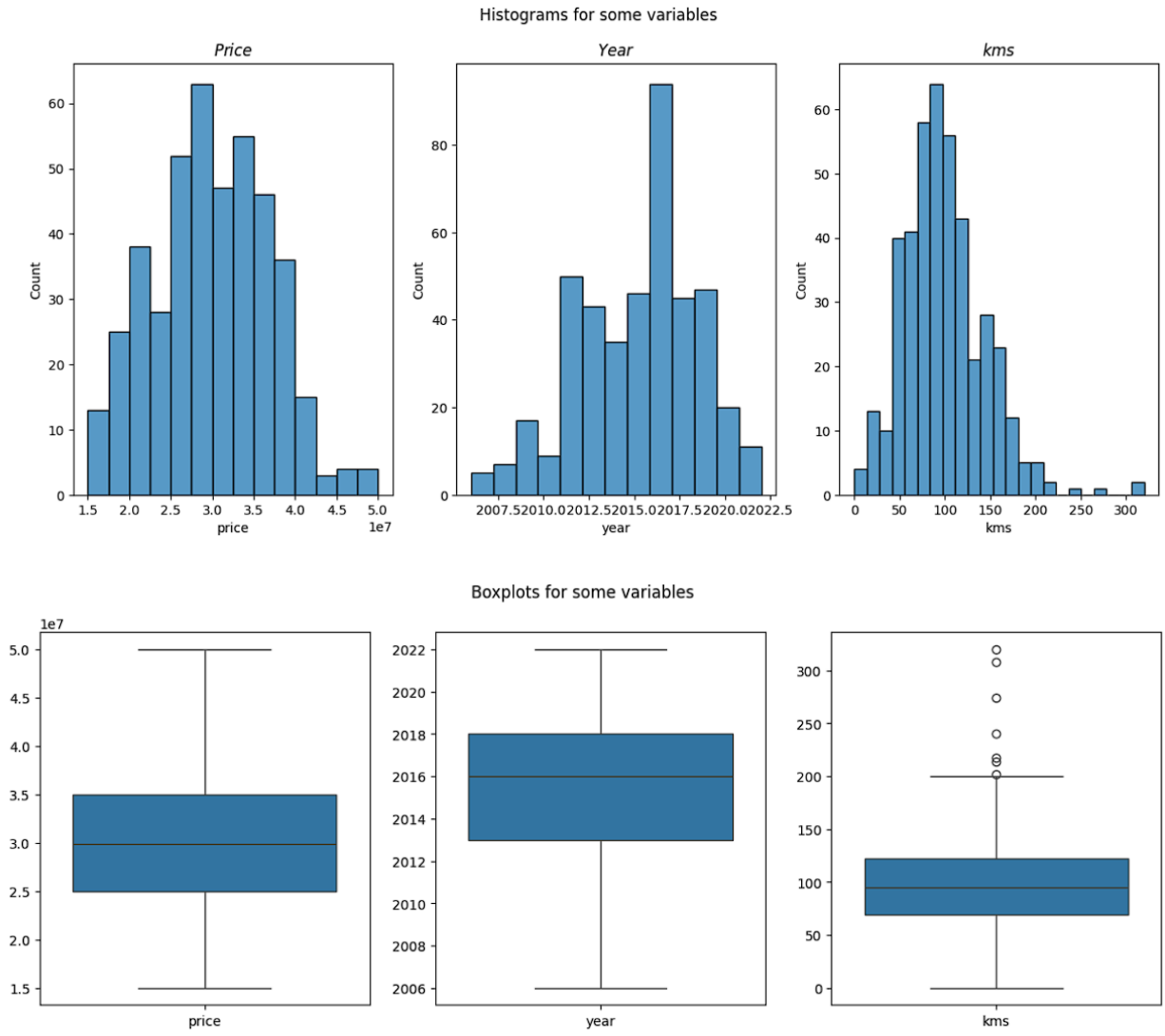
Además se tomó en cuenta el histograma y Boxplot de “kms”donde se evidencia una mayor cantidad de valores atípicos tomando solo los vehículos con kilometraje menor a los 400.000.

In [424…

```
#Filtrar los precios menores a 60.000.000 y kilometraje menor a 400000
datacc = datacc[datacc['price'] <= 60000000]
datacc = datacc[datacc['kms'] <= 400]

datacc.shape
# Verificar los primeros registros después del filtrado
print(data.shape)
data.head()
```

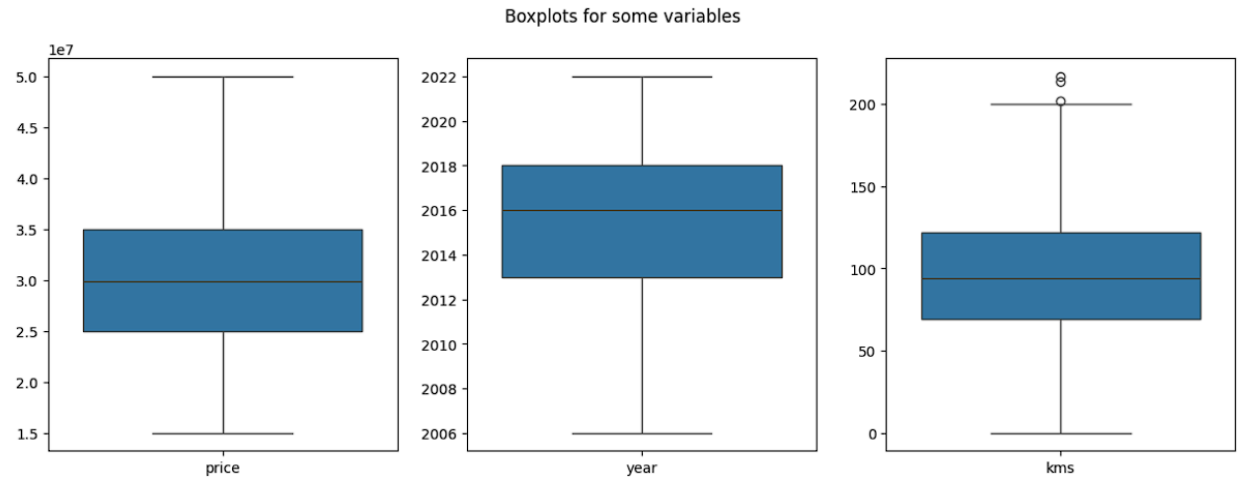
LABORATORIO #3



Como se puede observar en las anteriores gráficas los valores atípicos disminuyeron en comparación al Test1.

Después del tratamiento de los outliers o valores atípicos se evidencia una disminución de estos valores en el Boxplot de “kms”:

LABORATORIO #3



Comparación de los resultados de los modelos

Multivariate lineal regression: La regresión lineal múltiple permite generar un modelo lineal en el que el valor de la variable dependiente o respuesta (Y) se determina a partir de un conjunto de variables independientes llamadas predictores (X1, X2, X3…)

Test1	Test2

`# Define model and prediction`
`ols = LinearRegression()`
`model1 = ols.fit(X_train, y_train)`
`y_pred1 = model1.predict(X_test)`
 	
`# accuracy check`
`rmse = MSE(y_test, y_pred1, squared=False)`
`mae = MAE(y_test, y_pred1)`
`r2 = r2_score(y_test, y_pred1)`
`print("RMSE: %.2f" % rmse)`
`print("MAE: %.2f" % mae)`
`print("R2: %.2f" % r2)`
 	
`RMSE: 3675487.65`
`MAE: 3023620.71`
`R2: 0.74`
 	
La regresión lineal múltiple presentó un error cuadrático medio (RMSE) relativamente bajo y un coeficiente de determinación (R^2) alto, lo que sugiere un buen ajuste del modelo a los datos. Un R2 de 0.74 indica que el 74% de la variabilidad en los precios de venta de los vehículos puede ser explicada por este modelo.
	`# Define model and prediction`
	`ols = LinearRegression()`
	`model1 = ols.fit(X_train, y_train)`
	`y_pred1 = model1.predict(X_test)`
	
	`# accuracy check`
	`# RMSE más cercano a 0 diferencia del valor de la predicción`
	`# R2 mas cercano a 1 porcentaje de exactitud`
	`rmse = MSE(y_test, y_pred1, squared=False)`
	`mae = MAE(y_test, y_pred1)`
	`r2 = r2_score(y_test, y_pred1)`
	`print("RMSE: %.2f" % rmse)`
	`print("MAE: %.2f" % mae)`
	`print("R2: %.2f" % r2)`
	
	`RMSE: 3712421.37`
	`MAE: 2947323.68`
	`R2: 0.77`
	
	Similar al primer test, la regresión lineal múltiple mostró un RMSE bajo y un R2 alto, ligeramente mejor que en la primera prueba. Un R2 de 0.77 indica una buena capacidad predictiva.

Light GBM: es un algoritmo de aprendizaje automático de tipo ensemble basado en árboles de decisión. Es conocido por su velocidad y eficiencia, siendo uno de los algoritmos de boosting más rápidos disponibles. Se utiliza principalmente para tareas de regresión, pero también se puede usar para clasificación.

Test1	Test2

```
# Hyperparameters
params = {
    'task': 'train',
    'boosting': 'gbdt',
    'objective': 'regression',
    'num_leaves': 10,
    'learning_rate': 0.05,
    'metric': {'l2','l1'},
    'header' : 'true',
    'verbose': 0
}

# Laoding data
lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

# fitting the model
model2 = lgb.train(params,
                   train_set=lgb_train,
                   valid_sets=lgb_eval)
# Pred
y_pred2 = model2.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred2, squared=False)
mae = MAE(y_test, y_pred2)
r2 = r2_score(y_test, y_pred2)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 4148242.23
MAE: 3522189.49
R2: 0.67
```

Light GBM tuvo un RMSE más alto que la regresión lineal múltiple, indicando que el modelo tiene un mayor error de predicción. El R2 de 0.67 es inferior, lo que implica que el modelo explica el 67% de la variabilidad en los datos.

```
# Hyperparameters
params = {
    'task': 'train',
    'boosting': 'gbdt',
    'objective': 'regression',
    'num_leaves': 10,
    'learning_rate': 0.03,
    'metric': {'l2','l1'},
    'header' : 'true',
    'verbose': 0
}

# Laoding data
lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

# fitting the model
model2 = lgb.train(params,
                   train_set=lgb_train,
                   valid_sets=lgb_eval)
# Pred
y_pred2 = model2.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred2, squared=False)
mae = MAE(y_test, y_pred2)
r2 = r2_score(y_test, y_pred2)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 4038711.23
MAE: 3338791.78
R2: 0.72
```

Light GBM mejoró en comparación con la primera prueba. Se realizó un ajuste en el hiper parámetro learning_rate a 0.03, obteniendo como resultado un RMSE más bajo y un R2 mayor, lo que sugiere un mejor ajuste del modelo en esta segunda prueba.

Random Forest Regressor: es un algoritmo de aprendizaje automático de tipo ensemble basado en árboles de decisión. Se crea construyendo un conjunto de árboles de decisión aleatorios, donde cada árbol se entrena en un subconjunto aleatorio de datos. La predicción final se obtiene como el promedio de las predicciones de todos los árboles.

Test1	Test2

LABORATORIO #3

```
from sklearn.ensemble import RandomForestRegressor

model3 = RandomForestRegressor()
model3.fit(X_train, y_train)
y_pred3 = model3.predict(X_test)

# accuracy check
rmse = MSE(y_test, y_pred3, squared=False)
mae = MAE(y_test, y_pred3)
r2 = r2_score(y_test, y_pred3)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 4310134.13
MAE: 3476772.44
R2: 0.65
```

El Random Forest Regressor tuvo un RMSE aún mayor que Light GBM y un R2 de 0.65, lo que sugiere que este modelo es menos preciso en comparación con los dos modelos anteriores.

```
from sklearn.ensemble import RandomForestRegressor

model3 = RandomForestRegressor()
model3.fit(X_train, y_train)
y_pred3 = model3.predict(X_test)

# accuracy check
rmse = MSE(y_test, y_pred3, squared=False)
mae = MAE(y_test, y_pred3)
r2 = r2_score(y_test, y_pred3)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 3975092.77
MAE: 3208427.46
R2: 0.73
```

El Random Forest Regressor también mostró una mejora significativa en comparación con la primera prueba, con un RMSE más bajo y un R2 mayor, indicando un mejor rendimiento.

Xgboost regressor: es un algoritmo de aprendizaje automático de tipo ensemble basado en árboles de decisión. Es una implementación optimizada de Gradient Boosting Machine (GBM), que utiliza técnicas como el boosting secuencial y el aprendizaje por submuestreo para mejorar el rendimiento.

Test1	Test2

LABORATORIO #3

```
from sklearn.model_selection import cross_val_score, KFold
import xgboost as xgb
from sklearn.metrics import classification_report
```

```
#Define model
model4 = xgb.XGBRegressor(objective ='reg:squarederror',
                          booster='gbtree',
                          colsample_bytree = 1,
                          importance_type='gain',
                          learning_rate = 0.2,
                          max_depth = 5,
                          alpha = 5,
                          n_estimators = 200,
                          seed=123)
```

```
#Training
model4.fit(X_train, y_train)#,
#         eval_set=[(X_train, y_train), (X_test, y_test)], \
#         eval_metric='mlogloss', verbose=False)
```

```
#K-fold cross validation
scores = cross_val_score(model4, X_train, y_train, cv=10)
print("Mean cross-validation score: %.2f" % scores.mean())
```

```
Mean cross-validation score: 0.50
```

```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(model4, X_train, y_train, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores.mean())
```

```
K-fold CV average score: 0.50
```

```
# Pred
y_pred4 = model4.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred4, squared=False)
mae = MAE(y_test, y_pred4)
r2 = r2_score(y_test, y_pred4)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 4674025.50
MAE: 3786829.84
R2: 0.59
```

XGBoost Regressor presentó el mayor RMSE y el menor R2, lo que indica que este modelo tuvo el peor rendimiento en esta prueba.

```
from sklearn.model_selection import cross_val_score, KFold
import xgboost as xgb
from sklearn.metrics import classification_report
```

```
#Define model
model4 = xgb.XGBRegressor(objective ='reg:squarederror',
                          booster='gbtree',
                          colsample_bytree = 1,
                          importance_type='gain',
                          learning_rate = 0.2,
                          max_depth = 5,
                          alpha = 5,
                          n_estimators = 200,
                          seed=123)
```

```
#Training
model4.fit(X_train, y_train)#,
#         eval_set=[(X_train, y_train), (X_test, y_test)], \
#         eval_metric='mlogloss', verbose=False)
```

```
#K-fold cross validation
scores = cross_val_score(model4, X_train, y_train, cv=10)
print("Mean cross-validation score: %.2f" % scores.mean())
```

```
Mean cross-validation score: 0.50
```

```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(model4, X_train, y_train, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores.mean())
```

```
K-fold CV average score: 0.54
```

```
# Pred
y_pred4 = model4.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred4, squared=False)
mae = MAE(y_test, y_pred4)
r2 = r2_score(y_test, y_pred4)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 4532589.80
MAE: 3668010.35
R2: 0.65
```

XGBoost Regressor disminuyó el RMSE, y el R2 mejoró considerablemente a un 0.65 en comparación al 0.59 del test 1. Aún así, su rendimiento sigue siendo el más bajo entre los cuatro modelos.

Link código Test1: https://github.com/Diego-Garcia1/Repositorio-seminario-Big-Data/blob/bdddf623c36c0934803b668861711da984c74264/cars_price_prediction_Test1.ipynb

Link código Test2: https://github.com/Diego-Garcia1/Repositorio-seminario-Big-Data/blob/bdddf623c36c0934803b668861711da984c74264/cars_price_prediction_Test2.ipynb

LABORATORIO #3

Conclusiones

La regresión lineal múltiple resultó ser el método más preciso para predecir el precio de venta de los Chevrolet Spark, seguido de cerca por Light GBM y Random Forest Regressor. XGBoost Regressor tuvo el peor desempeño en ambas pruebas.

La regresión lineal múltiple mostró los valores más bajos de RMSE (error cuadrático medio) y los valores más altos de R2 (coeficiente de determinación), indicando una mejor precisión y capacidad explicativa. Por otro lado, Light GBM y Random Forest Regressor mostraron mejoras en la segunda prueba, acercándose al rendimiento de la regresión lineal múltiple, lo que sugiere su potencial como modelos alternativos. Por último, XGBoost Regressor, a pesar de ser un modelo potente en general, no presentó los mejores resultados.

LABORATORIO #3