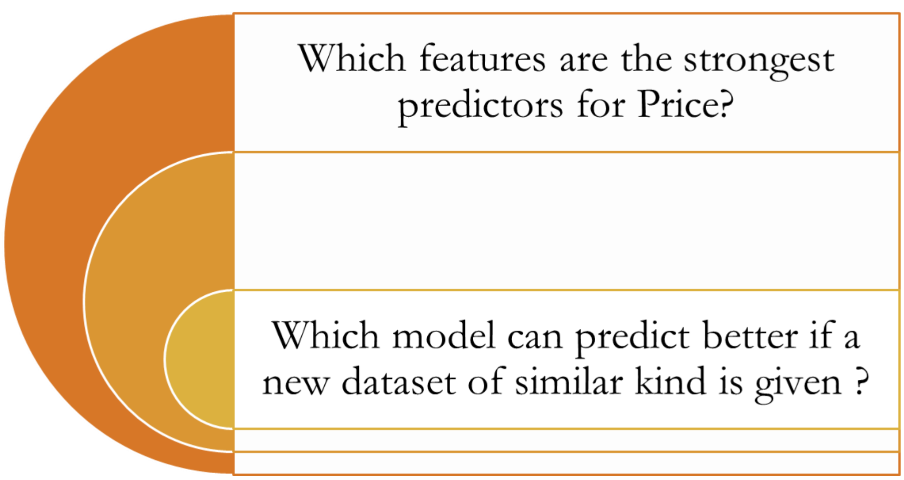
Used Car Price Prediction using Machine Learning Model.

OBJECTIVE



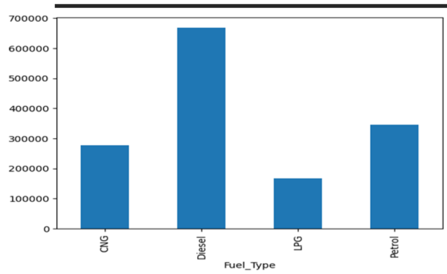
Target Variable:

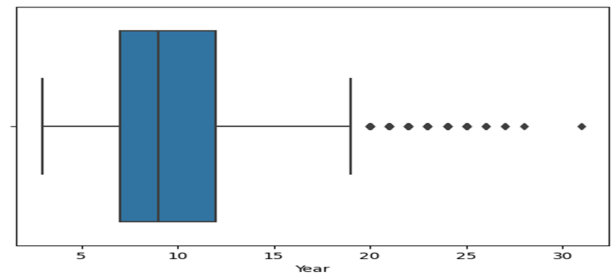
- Selling Price

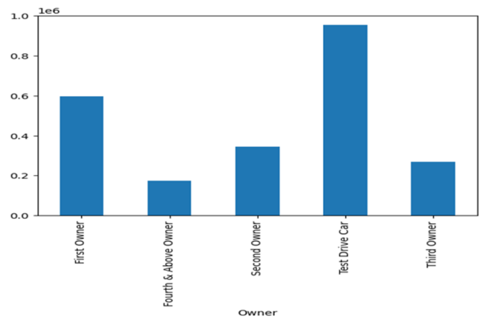
Predictor Variables:

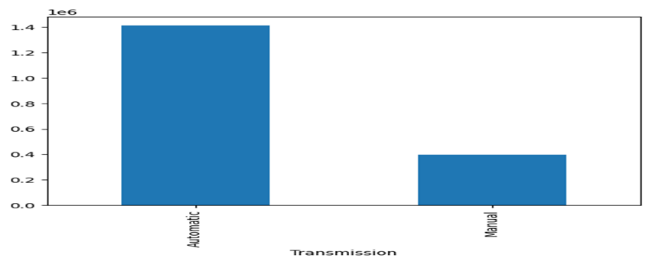
- Car Name
- Year
- Kms_Driven
- Fuel_Type
- Seller_Type
- Transmission
- Owner
- Day Mins
- Day Calls
- Monthly Charge
- Overage Fee
- Roam Mins

EXPLORATORY DATA ANALYSIS

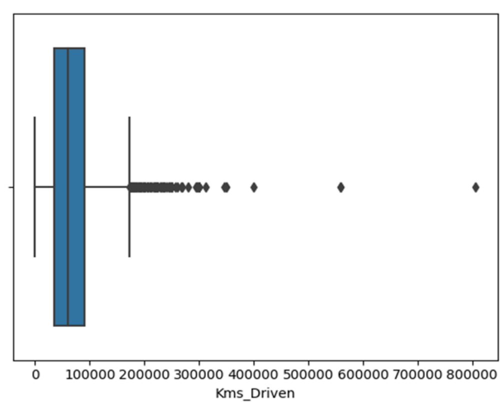


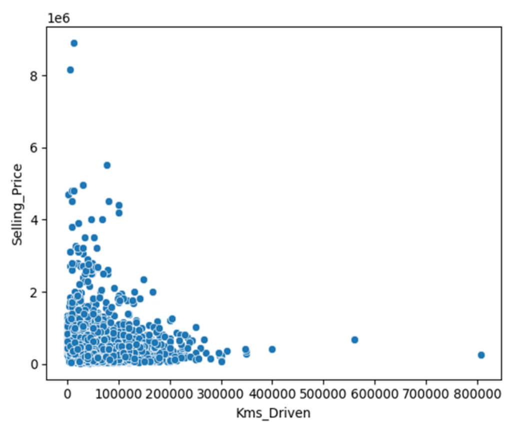






EXPLORATORY DATA ANALYSIS





FEATURE ENGINEERING

- Indexing Categorical Variables
- Encoding Categorical Variables.
- Vectorising
- Scaling

FEATURE ENGINEERING

```
from pyspark.ml.feature import VectorAssembler

type_indexer = StringIndexer(inputCol="Fuel_Type", outputCol="fuel_indexer").fit(new_data)
new_data = type_indexer.transform(new_data)

type_encoder = OneHotEncoder(inputCol="fuel_indexer", outputCol="fuel_Vector").fit(new_data)
new_data = type_encoder.transform(new_data)

type_indexer = StringIndexer(inputCol="Seller_Type", outputCol="seller_type_indexer").fit(new_data)
new_data = type_indexer.transform(new_data)

type_encoder = OneHotEncoder(inputCol="seller_type_indexer", outputCol="seller_type_vector").fit(new_data)
new_data = type_encoder.transform(new_data)

type_indexer = StringIndexer(inputCol="Owner", outputCol="owner_indexer").fit(new_data)
new_data = type_indexer.transform(new_data)

type_encoder = OneHotEncoder(inputCol="owner_indexer", outputCol="owner_vector").fit(new_data)
new_data = type_encoder.transform(new_data)

type_indexer = StringIndexer(inputCol="Transmission", outputCol="transmission_indexer").fit(new_data)
new_data = type_indexer.transform(new_data)

type_encoder = OneHotEncoder(inputCol="transmission_indexer", outputCol="transmission_vector").fit(new_data)
```

Scaling of features has been done here after encoding and assembling.

MODELLING

- Liner RegressionPipeline
- Decision Tree Regression Pipeline

LINEAR REGRESSION

Training the Model

```
[114] # Import `LinearRegression`
      from pyspark.ml.regression import LinearRegression
```

```
[115] # Initialize `lr`
      lr = LinearRegression(labelCol="Selling_Price", maxIter=10, regParam=0.3, elasticNetParam=0.8)
```

```
[116] # Fit the data to the model
      linearModel = lr.fit(train_data)
```

```
# Generate predictions
predicted = linearModel.transform(test_data)
predicted.show()
```

```
+--------------------+-------------+-------------+----------+---------+------------+---------------+-----------------+------------------+-------------+---------------+------------------
|                name|Selling_Price|Present_Price|Kms_Driven|Fuel_Type|fuel_indexer|    fuel_Vector|seller_type_indexer|seller_type_vector|owner_indexer|   owner_vector|transmission_indexe
+--------------------+-------------+-------------+----------+---------+------------+---------------+-----------------+------------------+-------------+---------------+------------------
|                 800|         0.35|         2.28|    127000|   Petrol|         0.0|(2,[0],[1.0])|              1.0|        (1,[],[])|          0.0|(2,[0],[1.0])|                0.
|   Bajaj Avenger 150|         0.75|          0.8|      7000|   Petrol|         0.0|(2,[0],[1.0])|              1.0|        (1,[],[])|          0.0|(2,[0],[1.0])|                0.
|   Bajaj Avenger 220|         0.75|         0.95|      3500|   Petrol|         0.0|(2,[0],[1.0])|              1.0|        (1,[],[])|          0.0|(2,[0],[1.0])|                0.
|       KTM 390 Duke |         1.15|          2.4|      7000|   Petrol|         0.0|(2,[0],[1.0])|              1.0|        (1,[],[])|          0.0|(2,[0],[1.0])|                0.
|           KTM RC390|         1.35|         2.37|     21700|   Petrol|         0.0|(2,[0],[1.0])|              1.0|        (1,[],[])|          0.0|(2,[0],[1.0])|                0.
| Mahindra Mojo XT300|         1.15|          1.4|     35000|   Petrol|         0.0|(2,[0],[1.0])|              1.0|        (1,[],[])|          0.0|(2,[0],[1.0])|                0.
```

Getting the outcome

```
118] # Coefficients for the model
     linearModel.coefficients

     DenseVector([-0.0, 0.0, 3.6723, 3.9266, 0.0, 0.0, -4.797])
```

```
119] # Intercept for the model
     linearModel.intercept

     5.854445158715909
```

```
# Get the RMSE
linearModel.summary.rootMeanSquaredError

3.1442264124177766
```

```
121] # Get the R2
     linearModel.summary.r2

     0.5732903949548369
```

Decision Tree Regresion

```
[122] # Decision Tree Regression
```

```python
from pyspark.sql import SparkSession
from pyspark.ml.regression import DecisionTreeRegressor
from pyspark.ml.evaluation import RegressionEvaluator



# Create an instance of the DecisionTreeRegressor
dt = DecisionTreeRegressor(featuresCol='features', labelCol='Selling_Price')

# Train the model
model = dt.fit(train_data)

# Make predictions on the test data
predictions = model.transform(test_data)

# Evaluate the model using a regression evaluator
evaluator = RegressionEvaluator(labelCol='Selling_Price', metricName="rmse")
rmse = evaluator.evaluate(predictions)

# Print the Root Mean Squared Error (RMSE)
print("RMSE:", rmse)
```

```
RMSE: 3.939194759463442
```

MODEL COMPARISON

```python
# Create an instance of RegressionEvaluator
evaluator = RegressionEvaluator(labelCol="Selling_Price", predictionCol="prediction", metricName="r2")

# Calculate the R-squared
r2 = evaluator.evaluate(predictions)
r2
```

```
0.5058637298018713
```

```
[118] # Coefficients for the model
      linearModel.coefficients

      DenseVector([-0.0, 0.0, 3.6723, 3.9266, 0.0, 0.0, -4.797])

[119] # Intercept for the model
      linearModel.intercept

      5.854445158715909

[120] # Get the RMSE
      linearModel.summary.rootMeanSquaredError

      3.1442264124177766

      # Get the R2
      linearModel.summary.r2

      0.5732903949548369

[122] # Decision Tree Regression
```

```
RMSE: 3.939194759463442
```

CONCLUSION

Decision Tree has given us better result which is just marginally better than Linear Regression.
However the r2 score is better in case of Decision Tree.
Hence, Decision Tree can be used to predict selling price of used car.

Submitted By.:
Abhishek Lal