CAR PRICE PREDICTION PROJECT

Submitted by:

KARISHMA YADAV.

ACKNOWLEDGMENT

I have taken efforts in this project. To solve this problem, the log transformation on the target variable is applied when it has skewed distribution and we need to apply an inverse function on the predicted values to get the actual predicted target value.

Due to this, for evaluating the model, the *RMSLE* is calculated to check the error and the *R2 Score* is also calculated to evaluate the accuracy of the model.

INTRODUCTION

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model.

1) Regression analysis: Regression analysis techniques aim mainly to investigate and estimate the relationships among a set of features. Regression includes many models for analysing the relation between one target/response variable and a set of independent variables. Logistic Regression (LR) is the appropriate regression analysis model to use when the dependent variable is binary. LR is a predictive analysis used to explain the relationship between a dependent binary variable and a set of independent variables.

2) Decision Tree: Decision Tree (DT) is a model that generates a tree-like structure that represents set of decisions. DT returns the probability scores of class membership. DT is composed of: a) internal Nodes: each node refers to a single variable/feature and represents a test point at feature level; b) branches, which

represent the outcome of the test and are represented by lines that finally lead to c) leaf Nodes which represent the class labels. That is how decision rules are established and used to classify new instances. DT is a flexible model that supports both categorical and continuous data. Due to their flexibility they gained popularity and became one of the most commonly used models for prediction.

3) Random Forest Random forests (RF) are an ensemble learning technique that can support classification and regression. It extends the basic idea of single classification tree by growing many classification trees in the training phase. To classify an instance, each tree in the forest generates its response (vote for a class), the model choses the class that has receive the most votes over all the trees in the forest. One major advantage of RF over traditional decision trees is the protection against over fitting which makes the model able to deliver a high performance.

4) KNN

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as

for Classification but mostly it is used for the Classification problems. K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

- R^2 Score: It is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. 0% indicates that the model explains none of the variability of the response data around its mean.

The first step before applying the selected analytical models on the dataset, explanatory data analysis for more insights into dataset was performed. Based on the observations data was pre-processed to be more suitable for analysis.

Analytical Problem Framing

Data cleaning :

As a next step, check missing values for each feature.

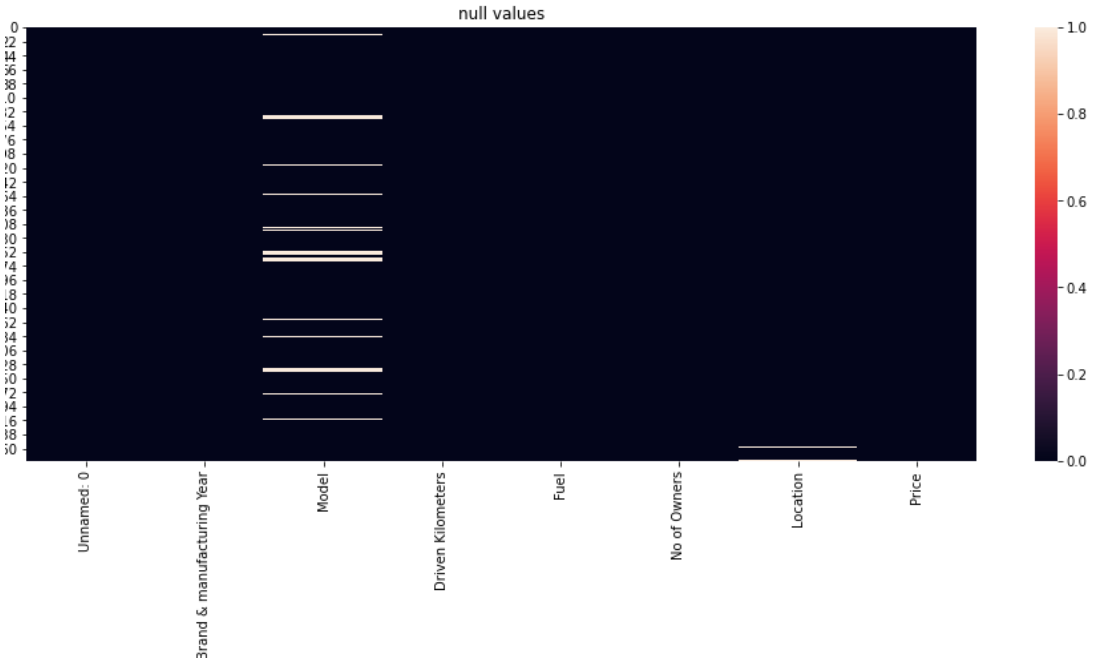
Missing values before data cleaning process :

```
df.isnull().sum()
```

```
Unnamed: 0                    0
Brand & manufacturing Year    0
Model                        43
Driven Kilometers             0
Fuel                          0
No of Owners                  0
Location                      3
Price                         0
dtype: int64
```

There are 43 missing values in Model and 3 missing values in Location.

White colour shows the distribution of null values



Next, now missing values were filled with appropriate values by an appropriate method.

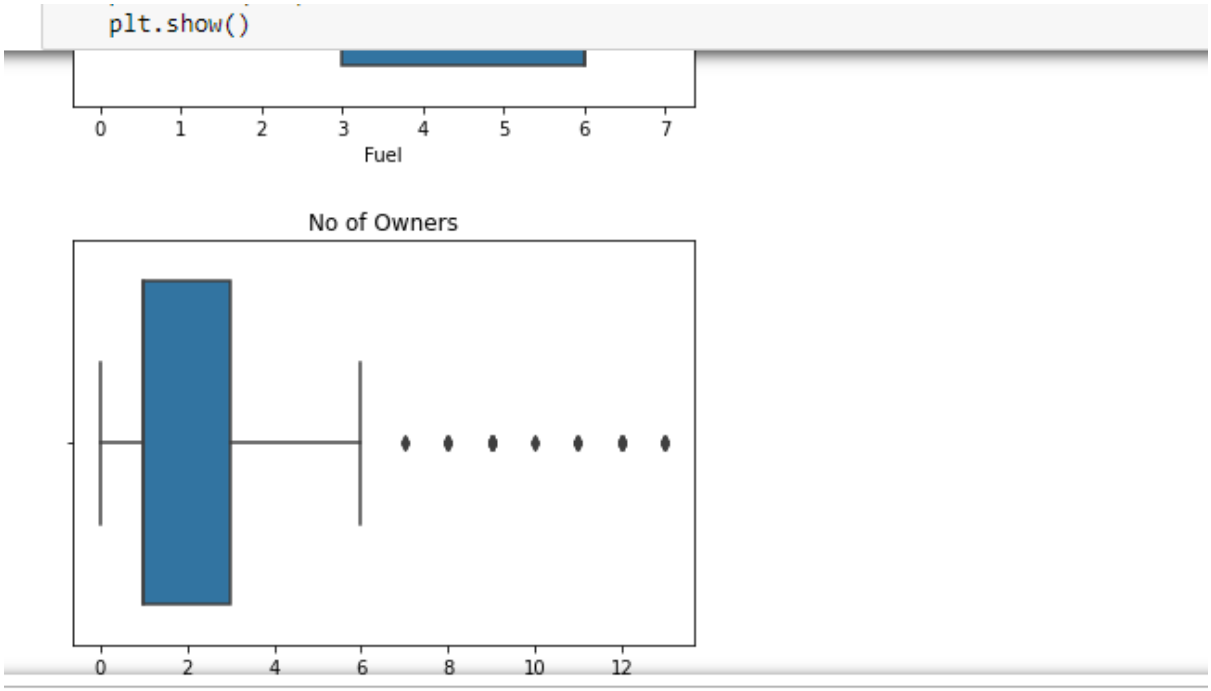
Missing values after data cleaning process :

```
df.isnull().sum()
```

```
Unnamed: 0                  0
Brand & manufacturing Year  0
Model                       0
Driven Kilometers           0
Fuel                        0
No of Owners                0
Location                    0
Price                       0
dtype: int64
```

At last, after dealing with missing values there zero null values.

Outliers: zscore method is used to remove the outliers from the data.



Data preprocessing:

Label Encoder: In our dataset, 7 features are categorical variables and To apply the ML models, we need to transform these categorical variables into numerical variables. And sklearn library *LabelEncoder* is used to solve this problem.

Normalization: The dataset is not normally distributed. All the features have different ranges. Without normalization, the ML model will try to disregard coefficients of features that have low values because their impact will be so small compared to the big value. Hence to normalized, *sklearn library i.e. standard Scaler* is used.

Train the data. In this process, 90% of the data was split for the train data and 10% of the data was taken as test data.

Hardware and Software Requirements and Tools Used

RAM: 8GB

ROM: I3 processor

SOFTWARE: Python 3.9.6

LIBRARIES: Numpy, Pandas, Seaborn, Sklearn, Scipy .

Tools: Jupiter notebook, Matplotlib, Scikit-learn, Excel.

- Testing of Identified Approaches (Algorithms)

1) K-Neighbors Regressor
2) Decision Tree Regressor
3) Random Forest Regressor
4) Ridge
5) Lasso
6) Logistic Regressor

- Run and Evaluate selected models

- MODELS

RandomForestRegressor

```
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
y_train_pred = rdr.predict(x_train)
y_test_pred = rdr.predict(x_test)
```

```
print("Train Results for Random Forest Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Random Forest Regressor Model:
--------------------------------------------------
Mean_absolute_error 16.17355042016807
Mean_squared_error 470.31394222689073
R-squared: 21.68672271752675
```

```
print("Test Results for Random Forest Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Random Forest Regressor Model:
--------------------------------------------------
Mean_absolute_error 71.65499071158787
Mean_squared_error 6514.930452168333
R-squared: 80.7151191052106
```

Logistic Regressor

```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_train_pred = logreg.predict(x_train)
y_test_pred = logreg.predict(x_test)
```

```
print("Train Results for LogisticRegression Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for LogisticRegression Model:
--------------------------------------------------
Mean_absolute_error 55.44117647058823
Mean_squared_error 7423.752100840336
R-squared: 86.16119834844648
```

```
print("Test Results for LogisticRegression Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Test Results for LogisticRegression Model:
--------------------------------------------------
Mean_absolute_error 67.80849531973156
Mean_squared_error 6125.673334109533
R-squared: 78.26668086810334
```

Ridge

```
rd = Ridge()
rd.fit(x_train,y_train)
y_train_pred = rd.predict(x_train)
y_test_pred = rd.predict(x_test)
```

```
print("Train Results for Ridge Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Ridge Model:
--------------------------------------------------
Mean_absolute_error 67.80849531973156
Mean_squared_error 6125.673334109533
R-squared: 78.26668086810334
```

```
print("Test Results for Ridge Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Test Results for Ridge Model:
--------------------------------------------------
Mean_absolute_error 67.80849531973156
Mean_squared_error 6125.673334109533
R-squared: 78.26668086810334
```

Lasso

```
lss = Lasso()
lss.fit(x_train,y_train)
y_train_pred = lss.predict(x_train)
y_test_pred = lss.predict(x_test)
```

```
print("Train Results for Lasso Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Lasso Model:
--------------------------------------------------
Mean_absolute_error 67.81831408316285
Mean_squared_error 6126.114681465278
R-squared: 78.26950032717264
```

```
print("Test Results for Lasso Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Lasso Model:
--------------------------------------------------
Mean_absolute_error 71.64179139387477
Mean_squared_error 6511.1412802071545
R-squared: 80.69164318693203
```

Decision Tree Regressor

```
dtc = DecisionTreeRegressor()
dtc.fit(x_train,y_train)
y_train_pred = dtc.predict(x_train)
y_test_pred = dtc.predict(x_test)
```

```
print("Train Results for Decision Tree Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Decision Tree Regressor Model:
--------------------------------------------------
Mean_absolute_error 0.0
Mean_squared_error 0.0
R-squared: 0.0
```

```
print("Test Results for Decision Tree Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Decision Tree Regressor Model:
--------------------------------------------------
Mean_absolute_error 54.09803921568628
Mean_squared_error 6884.313725490196
R-squared: 82.97176462803594
```

K Neighbors Regressor

```
knn = KNeighborsRegressor()
knn.fit(x_train,y_train)
y_train_pred = knn.predict(x_train)
y_test_pred = knn.predict(x_test)
```

```
print("Train Results for K Neighbors Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for K Neighbors Regressor Model:
--------------------------------------------------
Mean_absolute_error 48.20840336134454
Mean_squared_error 3691.962521008403
R-squared: 60.7615217140618
```

```
print("Test Results for K Neighbors Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for K Neighbors Regressor Model:
--------------------------------------------------
Mean_absolute_error 62.844117647058816
Mean_squared_error 6056.954705882352
R-squared: 77.82643963257186
```

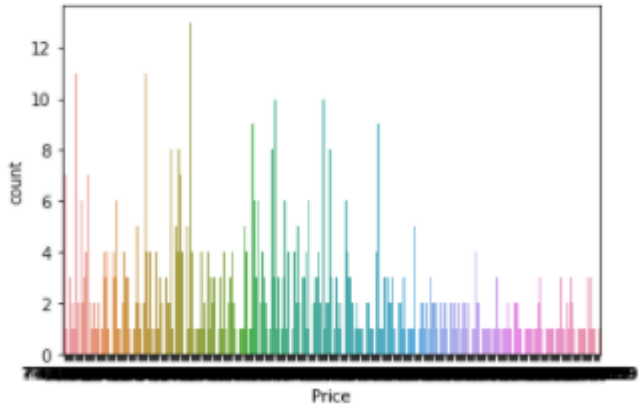
- Visualizations

    1. Count plot
    2. Boxplot
    3. Distplot
    4. Scatterplot
    5. Heatmap

1) Countplot

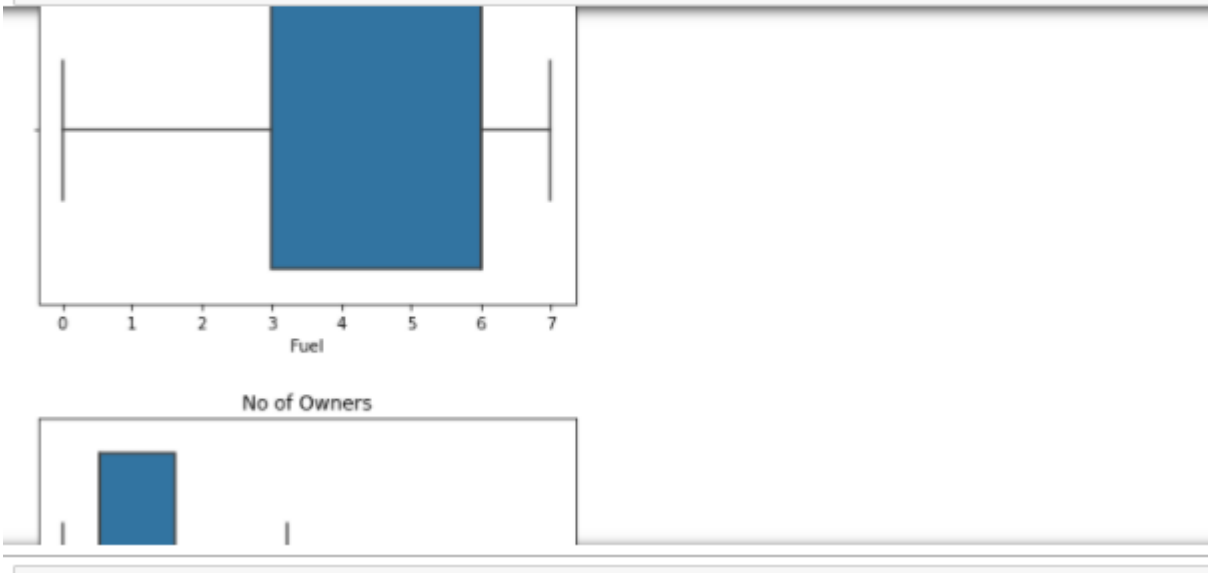
```
sns.countplot(df["Price"])
```

```
<AxesSubplot:xlabel='Price', ylabel='count'>
```



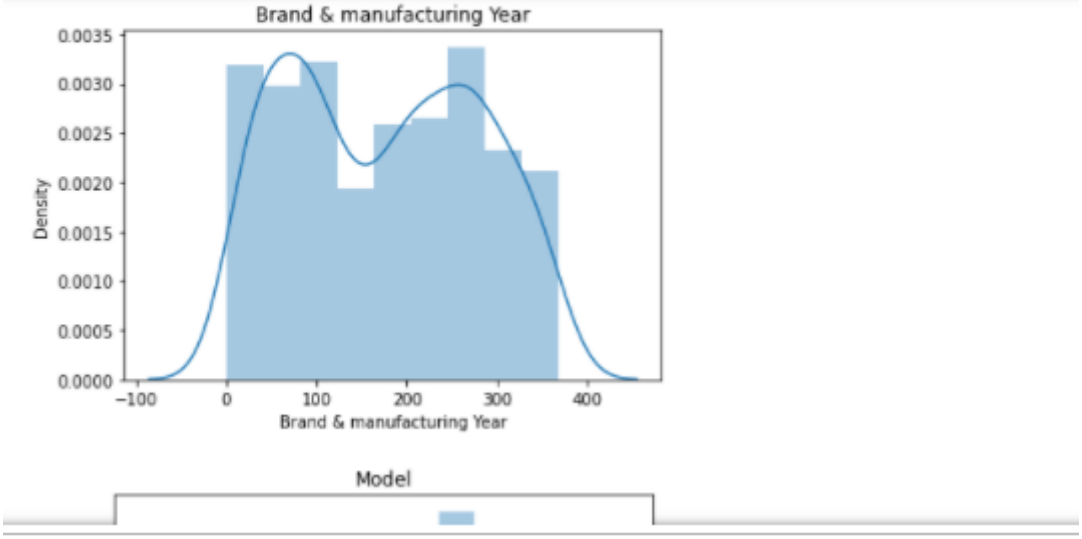
2) Boxplot

```
for col in ['Brand & manufacturing Year', 'Model',
       'Driven Kilometers', 'Fuel', 'No of Owners', 'Location', 'Price']:
    sns.boxplot(df[col].dropna(),orient='v')
    plt.title(col)
    plt.show()
```



3) Distplot

```
for col in ['Brand & manufacturing Year', 'Model',
       'Driven Kilometers', 'Fuel', 'No of Owners', 'Location', 'Price']:
    sns.distplot(df[col])
    plt.title(col)
    plt.show()
```



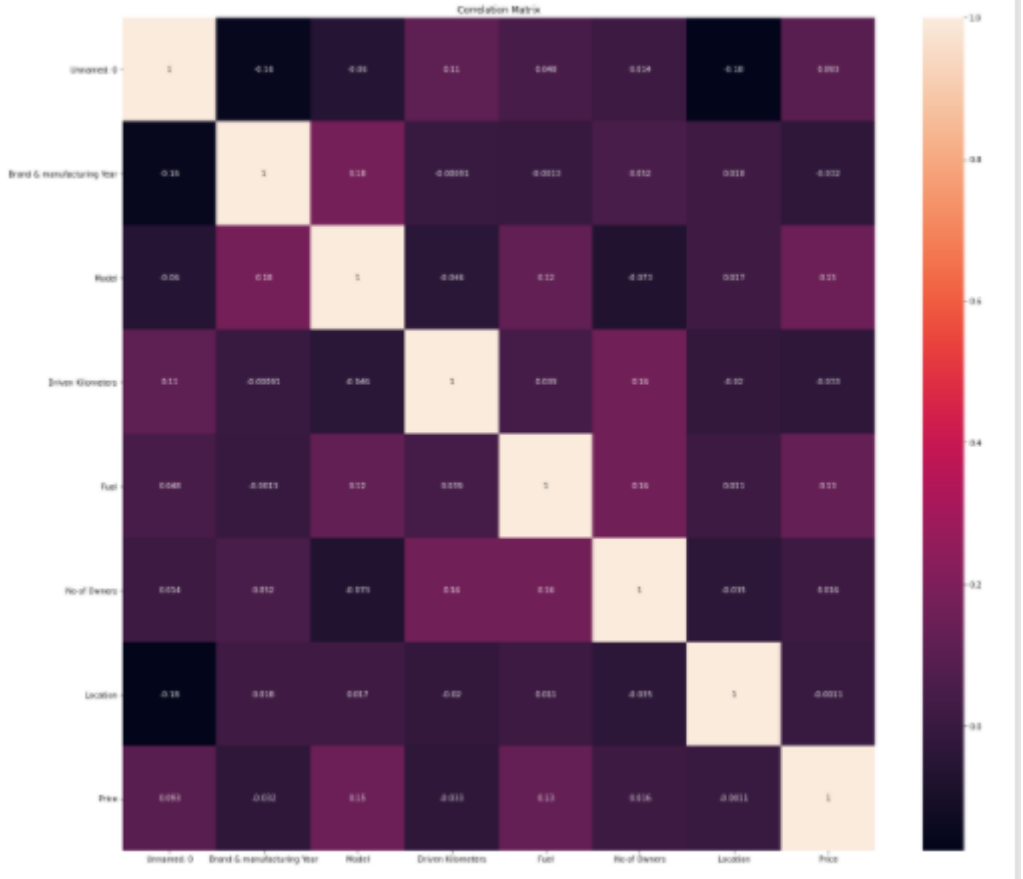
4) Scatterplot

```
# example of Bivariate analysis
plt.scatter(df["Brand & manufacturing Year"],df["Model"],alpha=0.3,c=(0,0,0),edgecolors='r')
plt.title("Brand & manufacturing Year vs Model")
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping wil
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array wit
u intend to specify the same RGB or RGBA value for all points.
```

5) Heatmap

```
plt.figure(figsize=[22,20])
sns.heatmap(cor,annot=True)
plt.title("Correlation Matrix")
plt.show()
```



CONCLUSION

In these project, Decision Tree Regressor, Random Forest Regressor, K-Neighbors Regressor, Ridge, Lasso these models are used.

Decision Tree Regressor is r2 score ,Mean absolute error, Mean_squared_error and R-squared is good than other models, Hence Decision Tree Regressor is performing is good.

Decision Tree Regressor is best model for this dataset.

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	194	195	196	197	198	199	200	201	202
predicted	220.56	99.91	60.21	197.47	120.54	140.3	165.86	102.74	267.94	56.2	...	138.27	264.9	157.25	125.33	113.65	131.24	156.11	155.48	106.83
original	274.80	112.40	111.80	145.20	134.00	140.4	194.80	142.00	220.40	130.2	...	142.60	268.2	145.40	66.80	171.60	107.80	163.00	130.60	184.20

2 rows × 204 columns