FLIP ROBO

Used Cars Price Prediction Project

Submitted by

Raghavulu Patnala

ACKNOWLEDGMENT

Thanks for giving me the opportunity to work in FlipRobo Technologies as Intern and would like to express my gratitude to Data Trained Institute as well for trained me in Data Science Domain. This helps me to do my projects well and understand the concepts.
Resources Referred – Google, GitHub, Blogs for conceptual referring.

INTRODUCTION

- ## Business Problem Framing

  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.
  So, we must predict the used cars price predict for our client through machine learning models.

- ## Conceptual Background of the Domain Problem

  With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models.
  Some cars are in demand and making them costly and some are not in demand, and it will be cheaper.
  This will help our client to do better trade.

- ## Motivation for the Problem Undertaken

  Car is one of the most needed in everyone lives, and all people cannot afford to buy a new one and people who want to buy can exchange their old car in a good rate.
  Our Prediction will help the client to sell the car in a smart way.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem
  Here Our **target** Variable is **Price** and as the data is having continuous variables, hence this is **Regression Problem**.

- Data Sources and their formats
  The data is collected from One of the famous websites for used cars and price are in Euros and it has 8 columns and 9980 rows.

```
# Loading the dataset,

df = pd.read_excel("UsedCar.xlsx")
df
```

	Unnamed: 0	Brand	Price	Year	Fuel	Transmission	Running KM	Location
0	0	Mercedes-Benz A Class 2L AMG A35	26925	2018	Petrol	Automatic	8015	London
1	1	Toyota Yaris 1.6L GR Circuit T	26850	2015	Petrol	Automatic	11718	London
2	2	Toyota Yaris 1.6L GR Circuit T	26725	2020	Petrol	Manual	24308	London
3	3	BMW 1 Series 2L M135i	26400	2019	Petrol	Automatic	10849	London
4	4	BMW 1 Series 2L M135i	26375	2018	Petrol	Automatic	21716	London
...	...	...	...	...	...	...	...	...
9975	9975	Mercedes-Benz A Class 1.5L AMG Line A180d	25500	2020	Petrol	Automatic	11605	London
9976	9976	Mercedes-Benz A Class 1.3L AMG Line A180	25500	2020	Petrol	Automatic	8307	London
9977	9977	Mercedes-Benz A Class 1.3L AMG Line A200	25425	2019	Petrol	Automatic	2743	London
9978	9978	Mercedes-Benz A Class 1.5L AMG Line A180d	25425	2020	Electric	Automatic	11250	London
9979	9979	Mercedes-Benz A Class 1.3L AMG Line A200	25350	2019	Diesel	Manual	13398	London

9980 rows × 8 columns

Data is not having any null values and we are good to pre-process the data further.

- Data Pre-processing Done

```
#dropping unwanted columns as it will not have any impact,
df = df.drop(columns = ['Unnamed: 0'], axis = 1)
df.head()
```

	Brand	Price	Year	Fuel	Transmission	Running KM	Location
0	Mercedes-Benz A Class 2L AMG A35	26925	2018	Petrol	Automatic	8015	London
1	Toyota Yaris 1.6L GR Circuit T	26850	2015	Petrol	Automatic	11718	London
2	Toyota Yaris 1.6L GR Circuit T	26725	2020	Petrol	Manual	24308	London
3	BMW 1 Series 2L M135i	26400	2019	Petrol	Automatic	10849	London
4	BMW 1 Series 2L M135i	26375	2018	Petrol	Automatic	21716	London

```
# cheking infpormation of the each column in dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9980 entries, 0 to 9979
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Brand         9980 non-null   object
 1   Price         9980 non-null   int64 
 2   Year          9980 non-null   int64 
 3   Fuel          9980 non-null   object
 4   Transmission  9980 non-null   object
 5   Running KM    9980 non-null   int64 
 6   Location      9980 non-null   object
dtypes: int64(3), object(4)
memory usage: 545.9+ KB
```

information o f the datset tells there are no null values present in a dataset and total 9980 rows and 7 columns in a dataset and dtype of dataset is int type-1 and object type are 6

```
# Checking is there any null values present in dataset

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

```
Brand           0
Price           0
Year            0
Fuel            0
Transmission    0
Running KM      0
Location        0
dtype: int64
```

```
# Applying Label Encoder to encode categorical into numerical
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
cat = cat.apply(le.fit_transform)
```

```
cat
```

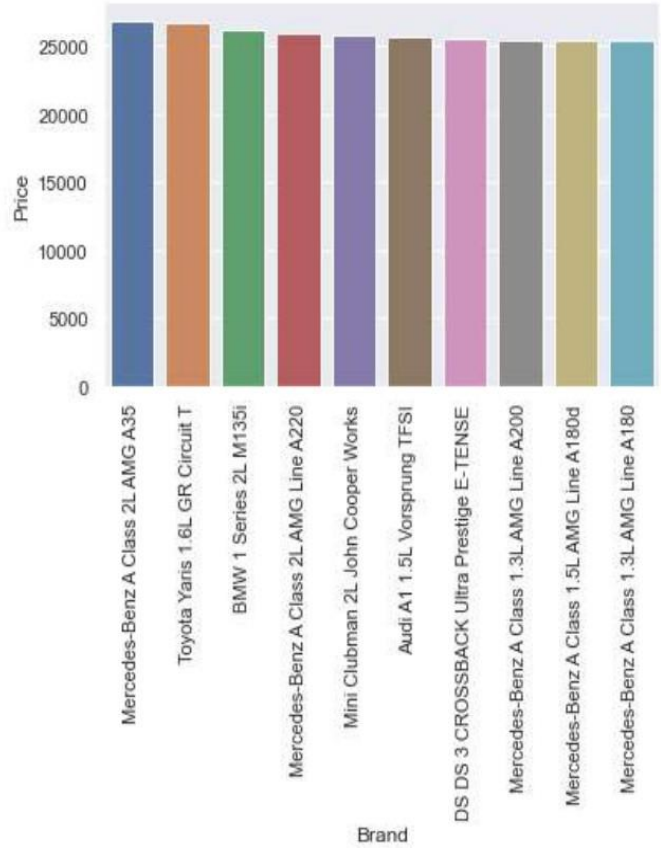
	Brand	Fuel	Transmission	Location
0	6	2	0	0
1	9	2	0	0
2	9	2	1	0
3	1	2	0	0
4	1	2	0	0
...	...	...	...	...
9975	5	2	0	0
9976	3	2	0	0
9977	4	2	0	0
9978	5	1	0	0
9979	4	0	1	0

9980 rows × 4 columns

- Data Inputs- Logic- Output Relationships

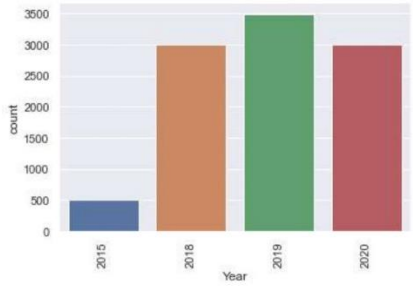
  Almost all car brands are in demand but among all popular brands, Mercedes, Toyota and BMW are having high price than other brand cars.

```
#let's visualize the barplot of brand,price using Seaborn
sns.set_theme()
sns.barplot(x = 'Brand', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



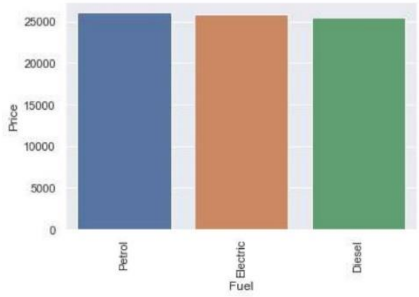
We can see from the below plot that most of the car registered year is from 2018,2019 and 2020.
Also, most of the cars are having fuel type is Electric , Diesel and petrol.

```
#Let's visualize the count of Year using seaborn
sns.set_theme()
sns.countplot(x = df['Year'])
plt.xticks(rotation = 90)
plt.show()
```



We can see from the below plot that most of the car registered year is from 2018,2019 and 2020.

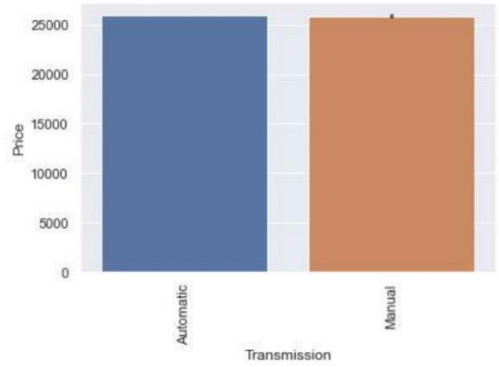
```
#Let's visualize the barplot of fuel,price using seaborn
sns.set_theme()
sns.barplot(x = 'Fuel', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



most of the cars are having fuel type is Electric,Diesel and petrol.

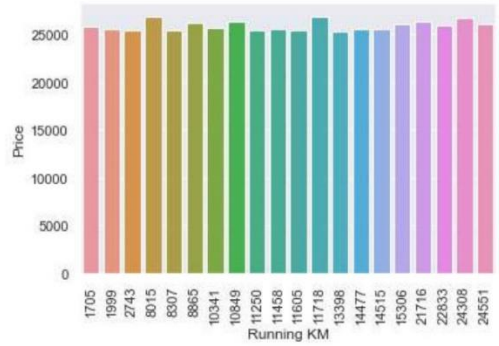
Most of the car transmission type is Automatic and the car which has minimum of > 1700 KM is having price of > 25000 Euros.

```
#let's visualize the barplot of Transmission,price using seaborn
sns.set_theme()
sns.barplot(x = 'Transmission', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



Most of the car transmission type is both Automatic and manual

```
#let's visualize the barplot of Running KM,price using seaborn
sns.set_theme()
sns.barplot(x = 'Running KM', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



Most of the car transmission type is Automatic and the car which has minimum of > 1700 KM is having price of

- Hardware and Software Requirements and Tools Used
  Model training was done on Jupiter Notebook. Kernel Version is Python3.

  Libraries – Scikit Learn, Pandas, NumPy

  Model Pre-process – **Standard Scaler** for normalize the ranges from 0-1.

  Label Encoder to encode the categorical values and convert into Numerical values.

  Metric – MSE, RMSE, R2 Score

  Model Selection – Train_Test_split for splitting the data into train and test dataset. **Cv Score** to check the model is over fit or under fit. **GridSearch Cv** for hyper parameter tuning the model.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  - Random Forest Regressor
  - K Neighbors Regressor
  - Gradient Boosting Regressor
  - Ada Boost Regressor

- Run and evaluate selected models

```
# Splitting X and Y values.

x = df_new.drop(columns = ['Price'], axis = 1)
y = df_new['Price']
```

```
#Scaling the data for normalize the range of values to 0-1.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_sc = scaler.fit_transform(x)
```

Model Building :

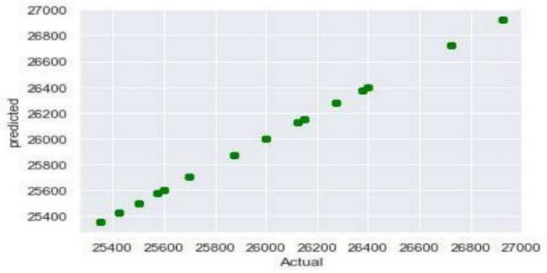
```
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
```

```
# Train test Split

x_train,x_test,y_train,y_test = train_test_split(x_sc,y, test_size = 0.20, random_state = 
```

```
#RandomForestRegressor Algorithm
rfr = RandomForestRegressor()
rfr.fit(x_train,y_train)
y_pred = rfr.predict(x_test)
scr_rfr = cross_val_score(rfr,x,y,cv=5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_rfr.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", rfr.score(x_train,y_train))
print("Test Score", rfr.score(x_test,y_test))
```

```
r2_Score 1.0
CV Score 1.0
MSE 0.0
RMSE 0.0
Train Score 1.0
Test Score 1.0
```

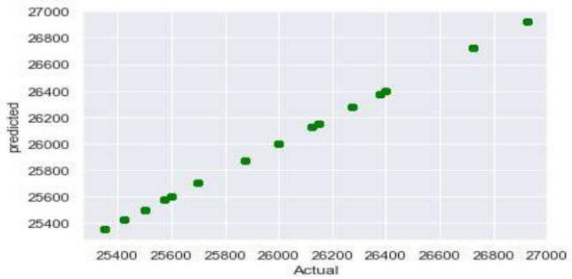
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix for Actual VS predicted fo
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```




```
#KNeighborsRegressor Algorithm

from sklearn.neighbors import KNeighborsRegressor

knr = KNeighborsRegressor(n_neighbors = 5)
knr.fit(x_train,y_train)
y_pred = knr.predict(x_test)
scr_knr = cross_val_score(knr,x,y,cv=5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_knr.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", knr.score(x_train,y_train))
print("Test Score", knr.score(x_test,y_test))
```

```
r2_Score 1.0
CV Score 1.0
MSE 0.0
RMSE 0.0
Train Score 1.0
Test Score 1.0
```

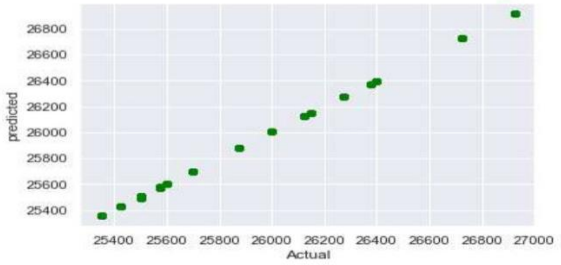
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix for Actual VS predicted fo
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



```
#GradientBoostRegressor Algorithm

from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(x_train, y_train)
y_pred = gbr.predict(x_test)
scr_gbr = cross_val_score(gbr,x,y,cv=5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_gbr.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", gbr.score(x_train,y_train))
print("Test Score", gbr.score(x_test,y_test))
```

```
r2_Score 0.9999286076335294
CV Score 0.9999472023945735
MSE 15.300643656442613
RMSE 3.9116037192490003
Train Score 0.9999295765220567
Test Score 0.9999286076335294
```

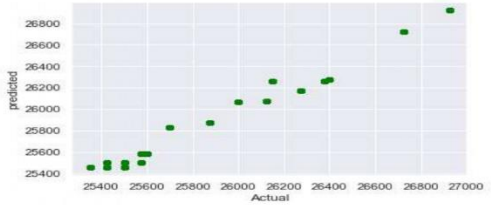
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix for Actual VS predicted fo
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



In [48]:
```
# AdaBoostRegressor Algorithm

from sklearn.ensemble import AdaBoostRegressor
abr = AdaBoostRegressor()
abr.fit(x_train, y_train)
y_pred = abr.predict(x_test)
scr_abr = cross_val_score(abr,x,y,cv=5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_abr.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", abr.score(x_train,y_train))
print("Test Score", abr.score(x_test,y_test))
```

```
r2_Score 0.9746734852134424
CV Score 0.9785465097759065
MSE 5427.9189353970005
RMSE 73.67441167323292
Train Score 0.9731772877362009
Test Score 0.9746734852134424
```

In [49]:
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix for Actual VS predicted fo
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



As we can see that random forest and KNeighbors are having high accuracy of 100%.

Let's apply hyper parameter tuning for RANDOM FOREST Model.

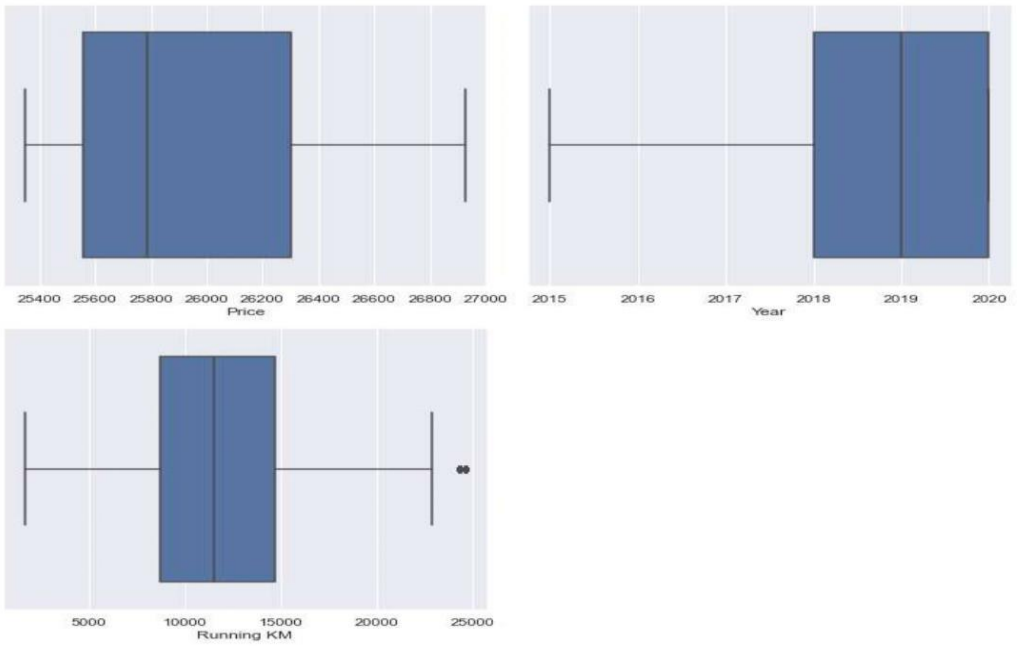
Hyperparameter Tuning :

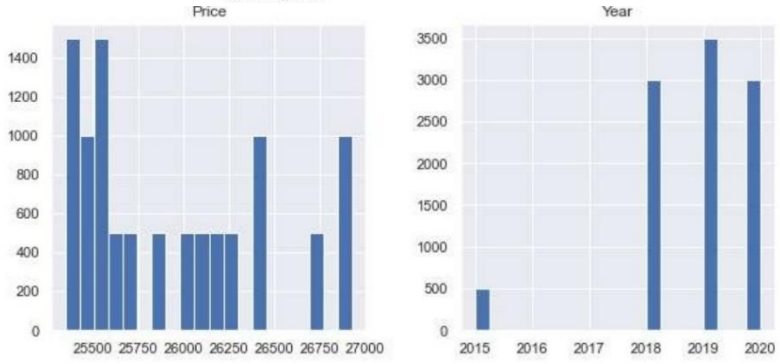
In [52]:
```
from sklearn.model_selection import GridSearchCV
```

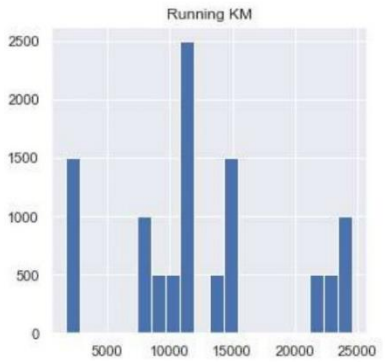
In [50]:
```
#Creating parameter list to pass in GridSearchCV
parameters={'criterion':['mse','mae'],
            'n_estimators':[40,60,80,100],
            'max_features':['auto','sqrt','log2']}
```

- Visualizations

```
plt.figure(figsize = (10,10))
pltnumber = 1
                                                    # boxplot for numerical columns for train data
for column in num:
    if pltnumber<=4:
        ax = plt.subplot(2,2,pltnumber)
        sns.boxplot(num[column])
        plt.xlabel(column,fontsize=12)
        pltnumber+=1
plt.tight_layout()
```







- Interpretation of the Results

Hyperparameter Tuning :

```
from sklearn.model_selection import GridSearchCV
```

```
#Creating parameter list to pass in GridSearchCV
parameters={'criterion':['mse','mae'],
            'n_estimators':[40,60,80,100],
            'max_features':['auto','sqrt','log2']}
```

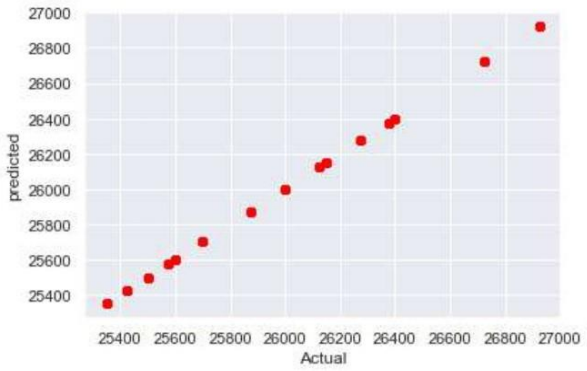
```
#Using GridSearchCV to run the parameters and checking final accuracy
rfr=RandomForestRegressor()
grid=GridSearchCV(rfr,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_) #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

```
{'criterion': 'mse', 'max_features': 'auto', 'n_estimators': 40}
1.0
```

```
#Using the best parameters obtained
RF=RandomForestRegressor(random_state=48, n_estimators=40, criterion='mse', max_features=
RF.fit(x_train,y_train)
pred=RF.predict(x_test)
print('r2_score: ',r2_score(y_test,pred))
```

```
r2_score:  1.0
```

```
plt.scatter(y_test,pred, color = 'red')      #Scatter Matrix for Actual VS predicted for t
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



Saving the model :

```
# Saving the model

import joblib
joblib.dump(RF,"UsedCar_Price.pkl")
```

```
['UsedCar_Price.pkl']
```

CONCLUSION

- Key Findings and Conclusions of the Study

  As this project is about predicting the prices of used cars, it is a regression problem as the target variables are continuous range.

  Used r2 score, MSE as a metrics to calculate the model accuracy.

  Data is Collected by me from theaa.com for used cars. The dataset doesn’t have any null or missing values.

- Learning Outcomes of the Study in respect of Data Science

  Random forest and K Neighbors Algorithm worked as a best model, and which have 100 % accuracy and I have used Grid Search CV for Hyper parameter tuning.