Used Cars Price Prediction Project

Submitted by:

Bramee Venkatesan

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 7 columns and 9980 rows.

```
# Loading the dataset,

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

	Unnamed: 0	Brand	Price	Year	Fuel	Transmission	Running KM
0	0	BMW 1 Series 2L M135i	34075	2020	Petrol	Automatic	3234
1	1	BMW 1 Series 2L M135i	33900	2020	Petrol	Automatic	8362
2	2	BMW 1 Series 2L M135i	33025	2020	Petrol	Automatic	6970
3	3	Mercedes-Benz A Class 1.3L AMG Line A250e	32225	2020	Plug_in_hybrid	Automatic	3548
4	4	Mini Clubman 2L John Cooper Works	32175	2020	Petrol	Automatic	4962
...	...	...	...	...	...	...	...
9975	9975	Mercedes-Benz A Class 1.3L AMG Line A200	27750	2019	Petrol	Automatic	14530
9976	9976	Mercedes-Benz A Class 1.3L AMG Line A200	27200	2019	Petrol	Automatic	8623
9977	9977	Mercedes-Benz A Class 2L AMG Line A200d	27050	2020	Diesel	Automatic	13038
9978	9978	Mercedes-Benz A Class 1.3L AMG Line A180	26900	2019	Petrol	Automatic	4359
9979	9979	BMW 1 Series 1.5L M Sport 118i	26850	2020	Petrol	Automatic	3551

9980 rows × 7 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
0	BMW 1 Series 2L M135i	34075	2020	Petrol	Automatic	3234
1	BMW 1 Series 2L M135i	33900	2020	Petrol	Automatic	8362
2	BMW 1 Series 2L M135i	33025	2020	Petrol	Automatic	6970
3	Mercedes-Benz A Class 1.3L AMG Line A250e	32225	2020	Plug_in_hybrid	Automatic	3548
4	Mini Clubman 2L John Cooper Works	32175	2020	Petrol	Automatic	4962

```
# Info of the each column in dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9980 entries, 0 to 9979
Data columns (total 6 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 
dtypes: int64(3), object(3)
memory usage: 467.9+ KB
```

```
# 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
dtype: int64
```

```
num = df.select_dtypes(exclude = object)
cat = df.select_dtypes(include = object)
```

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

```
cat
```

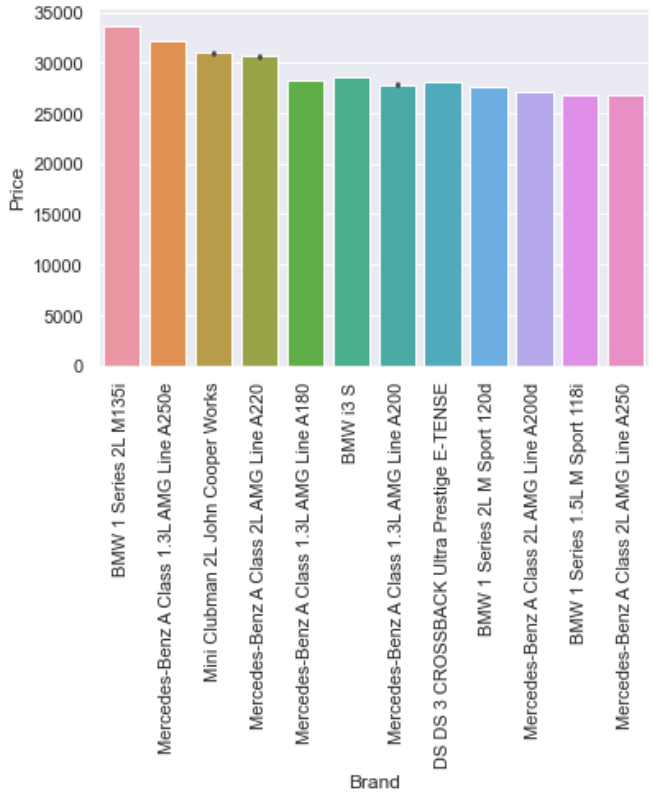
	Brand	Fuel	Transmission
0	2	2	0
1	2	2	0
2	2	2	0
3	7	3	0
4	11	2	0
...	...	...	...
9975	6	2	0
9976	8	0	0
9977	5	2	0
9978	0	2	0
9979	10	0	0

9980 rows × 3 columns

- Data Inputs- Logic- Output Relationships

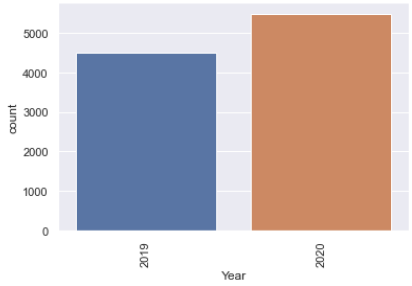
  Almost all car brands are in demand but among all popular brands, BMW, Mercedes, Mini Clubman are having high price than other brand cars.

```
sns.set_theme()
sns.barplot(x = 'Brand', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```

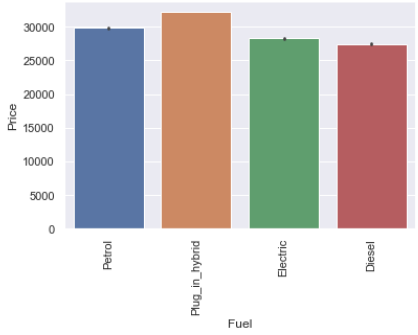


  We can see from the below plot that most of the car registered year is from 2019 and 2020.
  Also, most of the cars are having fuel type is Plug in Hybrid and petrol.

```
sns.countplot(x = df['Year'])
plt.xticks(rotation = 90)
plt.show()
```

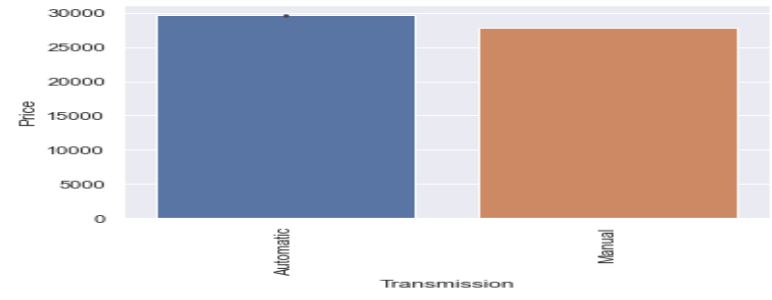



```
sns.barplot(x = 'Fuel', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```

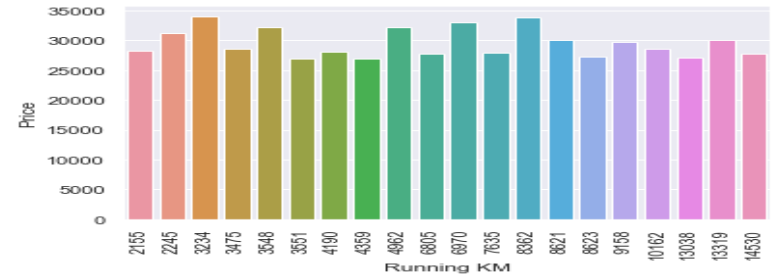


Most of the car transmission type is Automatic and the car which has minimum of > 2500 KM is having price of > 25000 Euros.

```
sns.barplot(x = 'Transmission', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```




```
sns.barplot(x = 'Running KM', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



- 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. **Randomized Search 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 = df1.drop(columns = ['Price'], axis = 1)
y = df1['Price']
```

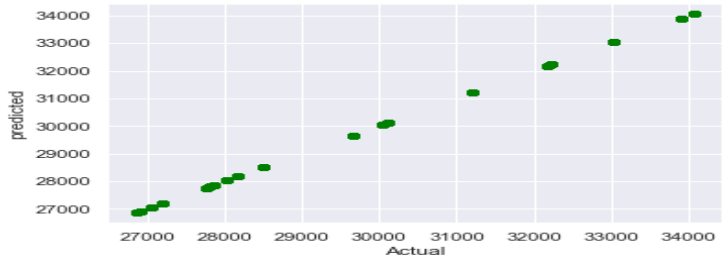
```
#Scaling the data for normalize the range of values to 0-1.

scaler = StandardScaler()
x_sc = scaler.fit_transform(x)
```

```
# Train test Split

x_train,x_test,y_train,y_test = train_test_split(x_sc,y, test_size = 0.20, random_state = 555)
```

```
# Random forest algorithm

from sklearn.ensemble import RandomForestRegressor

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
```

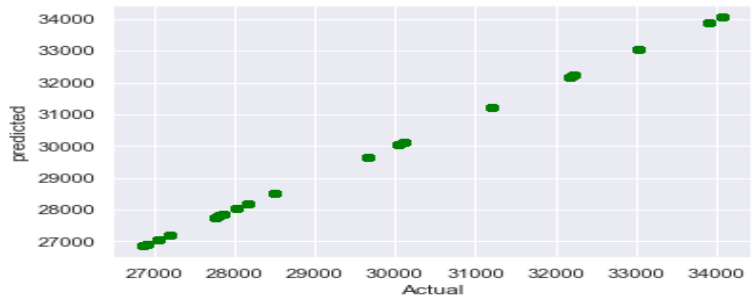
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```




```
#K Neighbors

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
```

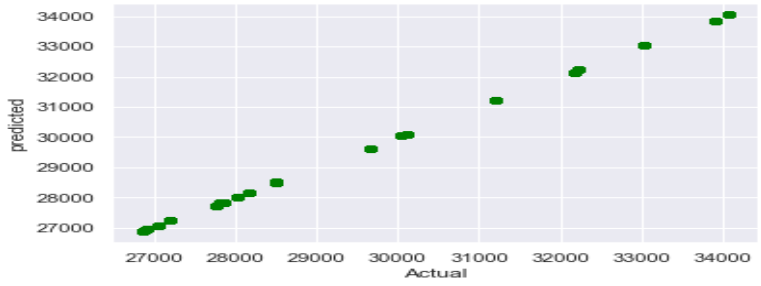
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



```
#Gradient Boost Regressor

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 0.9998731520563713
CV Score 0.9999217097684492
MSE 699.6136273652209
RMSE 26.450210346332234
Train Score 0.9998770071137332
Test Score 0.9998731520563713
```

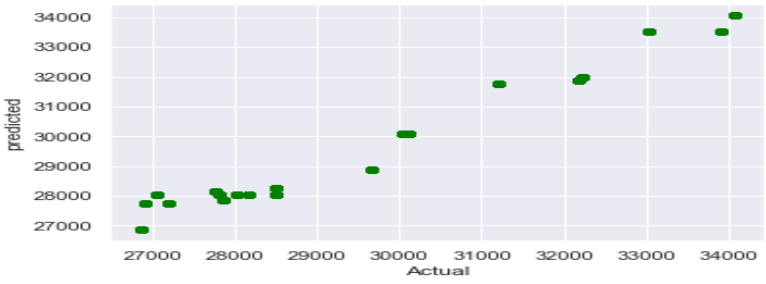
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



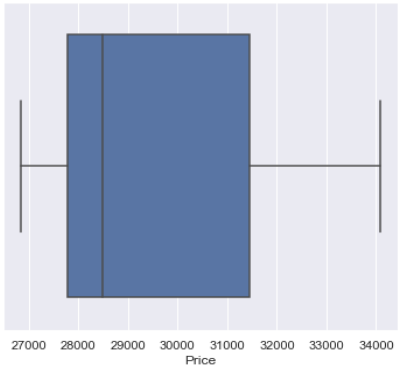

```
# Ada boost

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))
```

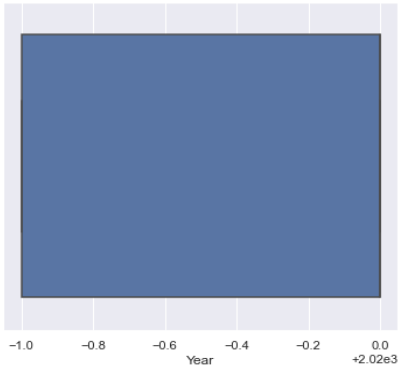
```
r2_Score 0.964570867450033
CV Score 0.9476175710478933
MSE 195404.854257941
RMSE 442.04621280805134
Train Score 0.9643246954630958
Test Score 0.964570867450033
```

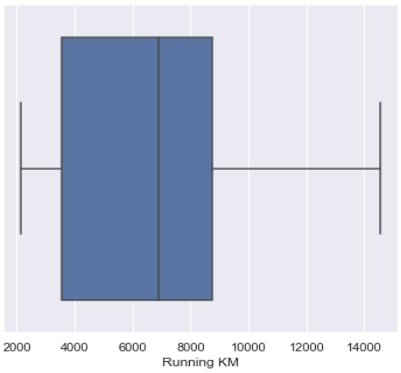
```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



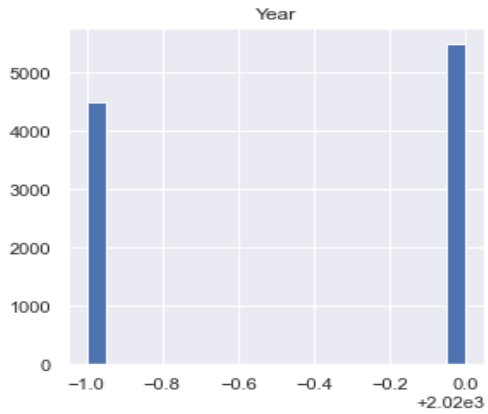
- Visualizations

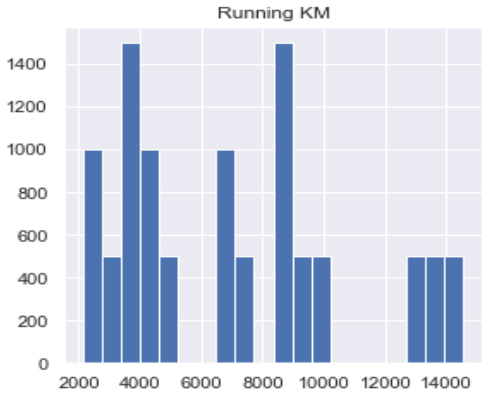






dtype=object)





- Interpretation of the Results

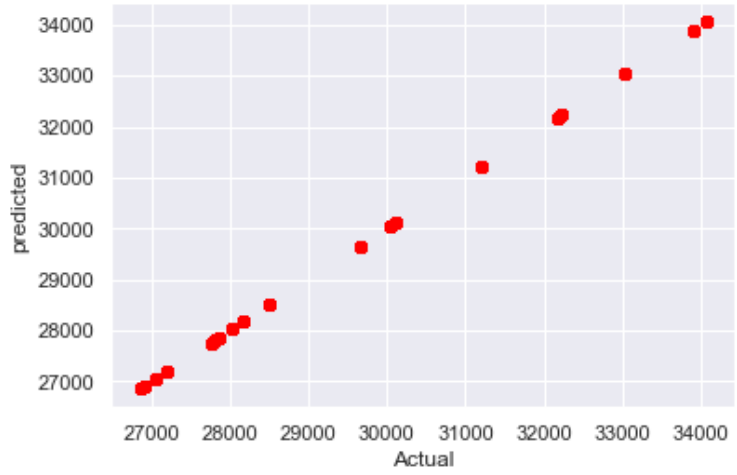
```
gs = RandomizedSearchCV(rfr, param_distributions = param, cv= 5)     # Hyper paramete
gs.fit(x_train,y_train)
gs.best_params_
```

```
{'n_estimators': 100,
 'min_samples_split': 3,
 'min_samples_leaf': 6,
 'max_depth': 12,
 'criterion': 'mse'}
```

```
final = RandomForestRegressor(n_estimators=80 , criterion = 'mse', max_depth = 20,
                              min_samples_leaf =8,min_samples_split =6)

final.fit(x_train,y_train)
pred = final.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
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



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 Randomized Search CV for Hyper parameter tuning as it is faster than Grid.

  This is kind of different as the data is not present and we need to collect it to build a model but helps me to learn more and most important is that I am getting hands-on experience more on Data Science Concepts.

  Thanks, FlipRobo and Data Trained for this wonderful Opportunity!! 😊