NAME OF THE PROJECT

Car Price Prediction Project.

Submitted by:

Rahul Singh

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 training me in Data Science Domain. This helps me to do my projects well and understand the concepts.

Dataset – FlipRobo Tech

Resources used – 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. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation 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 and help in his/her business to grow.

- **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 Modeling 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.

```
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_excel("UsedCar.xlsx")
df
```

Out[2]:

	Unnamed: 0	Brand	Price	Year	Fuel	Transmission	Running KM	Location
0	0	Mercedes-Benz A Class\n2L AMG S Plus A45	58525	2021	Diesel	Automatic	11101	London
1	1	Mercedes-Benz A Class\n2L AMG A35	37300	2019	Hybrid	Automatic	30108	London
2	2	Toyota Yaris\n1.6L GR Circuit T	36500	2021	Petrol	Automatic	48445	London
3	3	Toyota Yaris\n1.6L GR Circuit T	36300	2021	Petrol	Manual	10680	London
4	4	Toyota Yaris\n1.6L GR Circuit T	36150	2021	Petrol	Automatic	10810	London
...	...	...	...	...	...	...	...	...
9975	9975	Mercedes-Benz A Class\n2L AMG Line A200d	27525	2019	Petrol	Automatic	4912	London
9976	9976	Mercedes-Benz A Class\n1.5L AMG Line A180d	27350	2019	Petrol	Manual	12801	London
9977	9977	Mercedes-Benz A Class\n1.3L AMG Line A200	27250	2019	Petrol	Automatic	30440	London
9978	9978	Mercedes-Benz A Class\n1.3L AMG Line A200	26775	2019	Petrol	Automatic	26198	London
9979	9979	Mercedes-Benz A Class\n1.3L AMG Line A200	26375	2019	Electric	Automatic	22554	London

9980 rows × 8 columns

In [3]:
```
df = df.drop(columns = ['Unnamed: 0'], axis = 1)
df.head()
```

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

- **Data Preprocessing Done**

In [3]:
```
df = df.drop(columns = ['Unnamed: 0'], axis = 1)
df.head()
```

Out[3]:

	Brand	Price	Year	Fuel	Transmission	Running KM	Location
0	Mercedes-Benz A Class\n2L AMG S Plus A45	58525	2021	Diesel	Automatic	11101	London
1	Mercedes-Benz A Class\n2L AMG A35	37300	2019	Hybrid	Automatic	30108	London
2	Toyota Yaris\n1.6L GR Circuit T	36500	2021	Petrol	Automatic	48445	London
3	Toyota Yaris\n1.6L GR Circuit T	36300	2021	Petrol	Manual	10680	London
4	Toyota Yaris\n1.6L GR Circuit T	36150	2021	Petrol	Automatic	10810	London

unnamed column has no use in the dataset so we dropped that column from a dataset

In [4]:
```
# 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
```

In [16]:
```
# Applying label Encoder to encode categorical into numerical
from sklearn.preprocessing import LabelEncoder

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

In [17]:
```
cat
```

Out[17]:

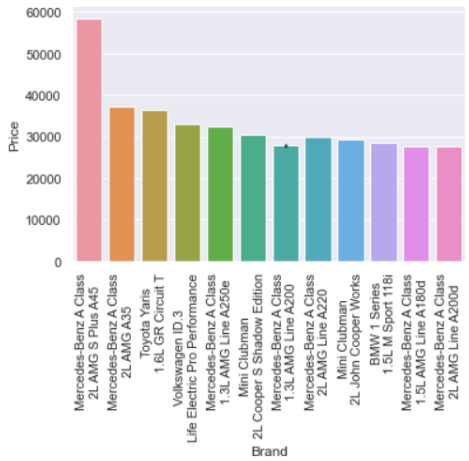
	Brand	Fuel	Transmission	Location
0	7	0	0	0
1	4	2	0	0
2	10	3	0	0
3	10	3	1	0
4	10	3	0	0
...	...	...	...	...
9975	5	3	0	0
9976	3	3	1	0
9977	1	3	0	0
9978	1	3	0	0
9979	1	1	0	0

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.

EDA

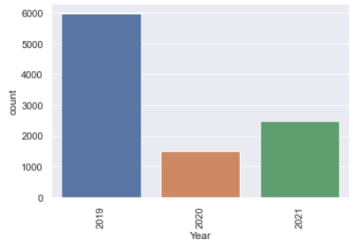
In [7]:
```
#let's visualize the barplot of brand,price using Seaborn
sns.set_theme()
sns.barplot(x = 'Brand', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



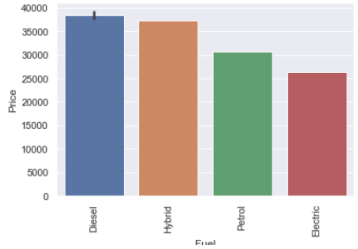
We can see from the below plot that most of the car registered year is from 2019, 2020 and 2021.

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

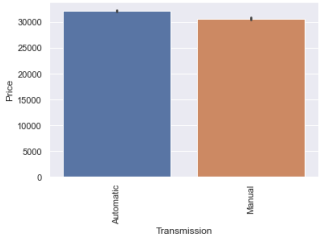
```
sns.set_theme()
sns.countplot(x = df['Year'])
plt.xticks(rotation = 90)
plt.show()
```



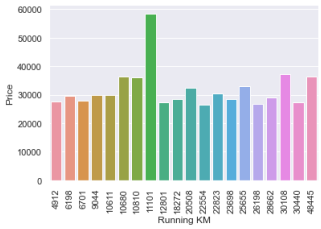
In [9]:
```
#Let's visualize the barplot of fuel,price using seaborn
sns.set_theme()
sns.barplot(x = 'Fuel', y = 'Price', data = df)
plt.xticks(rotation = 90)
plt.show()
```



In [10]:
```
#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()
```



In [11]:
```
#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()
```



Hardware and Software Requirements and Tools Used

Model training was done on Jupiter notebook. Kernel Version is python 3.

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.
Metrics- MSE, RMSE, R2 Score
Model Selection – Train_Test_split for spilitting 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**

In [33]:
```
# Splitting X and Y values.

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

In [34]:
```
#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 :

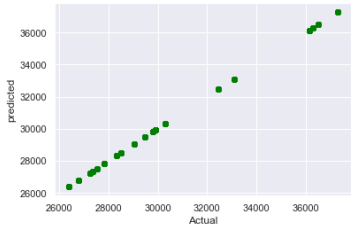
In [35]:
```
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
```

In [36]:
```
# Train test Split

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

In [37]:
```
#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
```

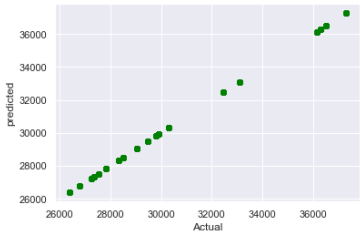
In [38]:
```
plt.scatter(y_test,y_pred, color = 'green')      #Scatter Matrix for Actual VS predicted for the model
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



In [39]:
```
#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
```

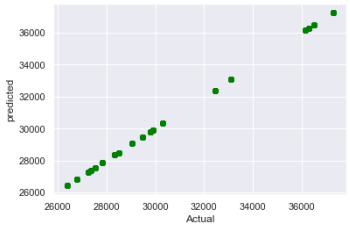
In [40]:
```
plt.scatter(y_test,y_pred, color = 'green')      #Scatter Matrix for Actual VS predicted for the model
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



In [41]:
```
#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 0.9999516533918015
CV Score 0.9998593071201656
MSE 617.1784820327791
RMSE 24.8430771450072
Train Score 0.999950913741859
Test Score 0.9999516533918015
```

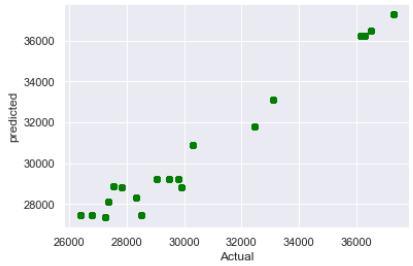
In [42]:
```
plt.scatter(y_test,y_pred, color = 'green')      #Scatter Matrix for Actual VS predicted for the model
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



In [43]:
```
# 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 0.9643451927232546
CV Score 0.9548019466862296
MSE 455158.71024192905
RMSE 674.65451176282
Train Score 0.9639903529082678
Test Score 0.9643451927232546
```

In [45]:
```
plt.scatter(y_test,y_pred, color = 'green')      #Scatter Matrix for Actual VS predicted for the model
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 :

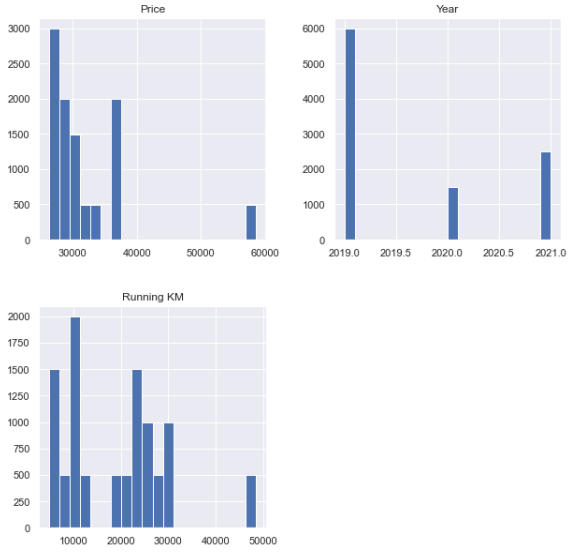
In [46]:
```
from sklearn.model_selection import GridSearchCV
```

In [47]:
```
parameters={'criterion':['mse','mae'],
            'n_estimators':[40,60,80,100],
            'max_features':['auto','sqrt','log2']}
```

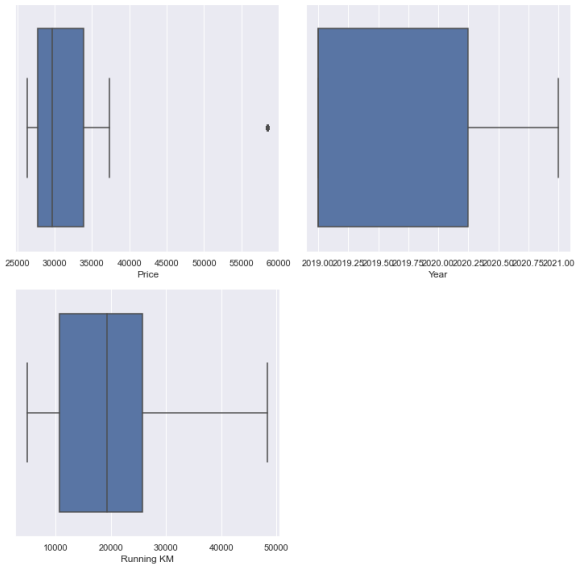
- **Visualizations**

In [12]:
```
df.hist(bins = 20, figsize = (10,10))
```

Out[12]:
```
array([[<AxesSubplot:title={'center':'Price'}>,
        <AxesSubplot:title={'center':'Year'}>],
       [<AxesSubplot:title={'center':'Running KM'}>, <AxesSubplot:>]],
      dtype=object)
```



```
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 :

In [46]:
```
from sklearn.model_selection import GridSearchCV
```

In [47]:
```
parameters={'criterion':['mse','mae'],
            'n_estimators':[40,60,80,100],
            'max_features':['auto','sqrt','log2']}
```

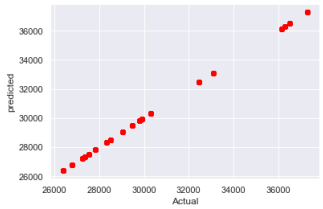
In [48]:
```
rfr=RandomForestRegressor()
grid=GridSearchCV(rfr,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_) #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

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

In [49]:
```
RF=RandomForestRegressor(random_state=48, n_estimators=40, criterion='mse', max_features='auto')
RF.fit(x_train,y_train)
pred=RF.predict(x_test)
print('r2_score: ',r2_score(y_test,pred))
```

```
r2_score:  1.0
```

In [50]:
```
plt.scatter(y_test,pred, color = 'red')      #Scatter Matrix for Actual VS predicted for the model
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



Finalizing the model :

In [51]:
```
rf_prediction=RF.predict(x)
print('Predictions of Random Forest Regressor: ',rf_prediction)
```

```
Predictions of Random Forest Regressor:  [35225.  35225.  35225.  ... 30592.5 30592.5 30592.5]
```

In [52]:
```
#Comparing actual and predicted values with the help of a dataframe
predictions=pd.DataFrame({'Original_price':y, 'Predicted_price':rf_prediction})
predictions
```

Out[52]:

	Original_price	Predicted_price
1	37300	35225.0
2	36500	35225.0
3	36300	35225.0
4	36150	35225.0
5	33100	35225.0
...	...	...
9975	27525	35225.0
9976	27350	35225.0
9977	27250	30592.5
9978	26775	30592.5
9979	26375	30592.5

9481 rows × 2 columns

Saving the model :

In [53]:
```
# Saving the model

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

Out[53]:
```
['Car_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.