FLIP ROBO

CAR PRICE PREDICTION

Submitted by:

SAYAN KUMAR BHUNIA

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion of the project.

- https://towardsdatascience.com/
- https://anshikaaxena.medium.com/
- https://medium.com/https://medium.com/

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. So, they are looking for new machine learning models from new data. We have to make car price valuation model.

➢ Conceptual Background of the Domain Problem

This project contains two phase--

Data Collection Phase- I have to scrape used cars data. I need to scrape the data of used cars from websites (Olx, cardekho, Cars24 etc.) I need web scraping for this. I have to fetch data for different locations. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometers, fuel, number of owners, location and at last target variable Price of the car. I can make changes to it, I can add or I can remove some columns, it completely depends on the website from which I am fetching the data. Try to include all types of cars in my data for example- SUV, Sedans, Coupe, minivan, Hatchback. Model Building Phase

After collecting the data, I need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Model Building Phase-

After collecting the data, I need to build a machine learning model. Before model building do all

data pre-processing steps. Trying different models with different hyper parameters and select the best

model.

Follow the complete life cycle of data science. Include all the steps like.

1. Data Cleaning

2. Exploratory Data Analysis

3. Data Pre-processing

4. Model Building

5. Model Evaluation

6. Selecting the best model

.

➢ Motivation for the Problem Undertaken

In this project I have to scrap data first from different different website and after that I haved to make a model to predict price.This is really interesting to work with selenium.

Analytical Problem Framing

➢ Mathematical/ Analytical Modelling of the Problem

We shall build a supervised Regression model to predict the price of a car based on given features.

Now, when we talk about building a supervised regression catering to certain use-case, following three things come into our minds:

- **Data** appropriate to the business requirement or use-case we are trying to solve
- A **regression model** which we think (or, rather assess) to be the best for our solution.
- **Optimize** the chosen model to ensure best performance.

➢ Data Sources and their formats

I am using CSV (comma-separated values) format file which we have scraped with selenium. which is having 2211rows × 10 columns.

```
In [4]: df
Out[4]:
```

	Brand	Model	Variant	Manufacturing Year	Driven KMs	Fuel Type	Number Of Owners	Price	Location	Website
0	Renault	Renault Kwid	RXL Manual	2019.0	21,521	Petrol	1st Owner	3,48,000	Jaipur	Cars24
1	Honda	Honda Amaze	1.2 SMT I VTEC Manual	2018.0	46,809	Petrol	1st Owner	5,27,000	Jaipur	Cars24
2	Maruti	Maruti Swift	LXI OPT Manual	2016.0	90,859	Petrol	1st Owner	4,35,000	Jaipur	Cars24
3	Mahindra	Mahindra XUV500	W8 FWD Manual	2015.0	95,615	Diesel	2nd Owner	7,10,000	Jaipur	Cars24
4	Hyundai	Hyundai i10	MAGNA 1.1 IRDE2 Manual	2014.0	32,716	Petrol	2nd Owner	3,12,000	Jaipur	Cars24
...	...	...	...	...	...	...	...	...	...	...
2206	Maruti	Maruti Baleno	ZETA 1.2 K12 AMT Automatic	2017.0	22,390	Petrol	2nd Owner	6,86,000	Bangalore	Cars24
2207	Maruti	Maruti Alto K10	VXI Manual	2014.0	38,841	Petrol	2nd Owner	2,81,000	Bangalore	Cars24
2208	Tata	Tata NEXON	XZ+ 1.2 Manual	2020.0	25,946	Petrol	2nd Owner	9,78,000	Bangalore	Cars24
2209	Hyundai	Hyundai Creta	1.6 SX (O) VTVT Manual	2019.0	18,051	Petrol	1st Owner	14,44,000	Bangalore	Cars24
2210	Maruti	Maruti Ritz	VXI Manual	2013.0	66,671	Petrol	1st Owner	4,04,000	Bangalore	Cars24

2211 rows × 10 columns

➢ Data Pre-processing

Following steps have been performed on the data.

- **checking missing values**-
  - If there is any missing value present in your data set then for a better and correct accuracy you have to impute it.
  - If missing data present in object type column, then you have to take most frequent value for your missing data.
  - If missing data present in int or float type column then use mean/median for missing value.

  In the following case no missing value present:

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

Brand                     21
Model                     21
Variant                   56
Manufacturing Year        21
Driven KMs                17
Fuel Type                 17
Number Of Owners        1200
Price                     17
Location                   0
Website                    0
dtype: int64
```

- **Encoding categorical variables** -as we can see there are object data type columns present so we will encode it into (int) format.

  In the following case

```
[27]: import sklearn
      from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

```
[28]: le=LabelEncoder()
      list1=['Brand','Model','Variant','Fuel Type','Number Of Owners','Location','Website']
      for i in list1:
          df[i]=le.fit_transform(df[i].astype(str))
      df
```

[28]:

	Brand	Model	Variant	Manufacturing Year	Driven KMs	Fuel Type	Number Of Owners	Price	Location	Website
0	82	204	787	2019	21521	9	0	348000	3	0
1	23	87	63	2018	46809	9	0	527000	3	0
2	55	161	723	2016	90859	9	0	435000	3	0
3	47	143	1013	2015	95615	4	1	710000	3	0

  Label Encoding is used.

- **Feature scaling-** Feature Scaling ensures that all features will get equal importance in supervised regressor models. Standard scaler was used to scale all features in the data.

```
51]: from sklearn.preprocessing import StandardScaler

52]: sc=StandardScaler()
     x=sc.fit_transform(x)
     x

52]: array([[ 1.46484779,  0.95835615,  0.45008391, ..., -0.33138244,
              0.05768019, -1.05000598],
            [-0.99650011, -0.65441481, -1.70751082, ..., -0.33138244,
              0.05768019, -1.05000598],
            [ 0.33846824,  0.36562836,  0.2593573 , ..., -0.33138244,
              0.05768019, -1.05000598],
            ...,
            [ 1.59000107,  1.38567153,  1.22491074, ...,  3.01766138,
             -1.46974967, -1.05000598],
            [-0.87134683, -0.5303555 , -1.34095813, ..., -0.33138244,
             -1.46974967, -1.05000598],
            [ 0.33846824,  0.32427526,  1.04014434, ..., -0.33138244,
```

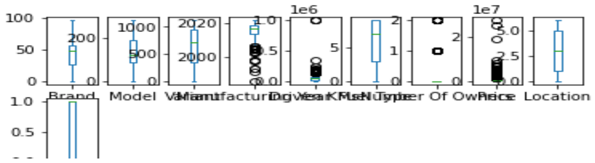
- **Reducing dimension of the data-** Sklearn's pca can be used to apply principal component analysis on the data. This helped in finding the vectors of maximal variance in the data.

- **Outliers detection-** In simple words, an outlier is an observation that diverges from an overall pattern on a sample.

  In the following case box plot is used to detect outliers.

```
df.plot(kind='box',subplots=True,layout=(3,9))

Brand                     AxesSubplot(0.125,0.657941;0.0731132x0.222059)
Model                  AxesSubplot(0.212736,0.657941;0.0731132x0.222059)
Variant                AxesSubplot(0.300472,0.657941;0.0731132x0.222059)
Manufacturing Year     AxesSubplot(0.388208,0.657941;0.0731132x0.222059)
Driven KMs             AxesSubplot(0.475943,0.657941;0.0731132x0.222059)
Fuel Type              AxesSubplot(0.563679,0.657941;0.0731132x0.222059)
Number Of Owners       AxesSubplot(0.651415,0.657941;0.0731132x0.222059)
Price                  AxesSubplot(0.739151,0.657941;0.0731132x0.222059)
Location               AxesSubplot(0.826887,0.657941;0.0731132x0.222059)
Website                   AxesSubplot(0.125,0.391471;0.0731132x0.222059)
dtype: object
```



There are many types of outlier detection techniques such as Z-Score or Extreme Value Analysis, Probabilistic and Statistical Modelling, Information Theory Models, Standard Deviation etc.

➢ Outliers Removal

In our dataset, we observed variations in the relation between values of some attributes.

So that these types of rows are dropped from the dataset.

```
[35]: from scipy.stats import zscore
      z=np.abs(zscore(df))
      z
```

```
36]: threshold=3
     print(np.where(z>3))

(array([  19,   39,   56,  152,  175,  369,  373,  546,  559,  839,  872,
         875,  892,  915,  923,  926,  947,  948, 1010, 1031, 1158, 1175,
        1195, 1219, 1220, 1228, 1239, 1246, 1269, 1295, 1358, 1384, 1398,
        1444, 1450, 1503, 1505, 1506, 1539, 1540, 1543, 1557, 1612, 1616,
        1655, 1708, 1737, 1759, 1794, 1802, 1840, 1916, 1920, 1953, 1973,
        2033, 2037, 2088, 2101, 2106], dtype=int64), array([6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 4, 3, 3, 4, 7, 7, 7, 7,
        4, 7, 7, 3, 7, 3, 4, 3, 7, 7, 7, 7, 3, 3, 4, 7, 3, 4, 4, 4, 7, 3,
```

```
52]: df_new=df[(z<3).all(axis=1)]
     df_new
```

52]:

	Brand	Model	Variant	Manufacturing Year	Driven KMs	Fuel Type	Number Of Owners	Price	Location	Website
0	82	204	787	2019	21521	9	0	348000	3	0
1	23	87	63	2018	46809	9	0	527000	3	0
2	55	161	723	2016	90859	9	0	435000	3	0
3	47	143	1013	2015	95615	4	1	710000	3	0

➢ Software Requirements and library Used

```
[1]: import pandas
     import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     import warnings
     warnings.filterwarnings('ignore')
```

```
10]: import sklearn
     from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

- **NumPy**

  NumPy is a popular Python library for multi-dimensional array and matrix processing because it can be used to perform a great variety of mathematical operations. Its capability to handle linear algebra, Fourier transform, and more, makes NumPy ideal for machine learning and artificial intelligence (AI) projects, allowing users to manipulate the matrix to easily improve machine learning performance. NumPy is faster and easier to use than most other Python libraries.

- **Scikit-learn**
  Scikit-learn is a very popular machine learning library that is built on NumPy and SciPy. It supports most of the classic supervised and unsupervised learning algorithms, and it can also be used for data mining, modelling, and analysis.

- **Seaborn**
  Seaborn is another open-source Python library, one that is based on Matplotlib (which focuses on plotting and data visualization) but features Pandas' data structures. Seaborn is often used in ML projects because it can generate plots of learning data. Of all the Python libraries, it produces the most aesthetically pleasing graphs and plots, making it an effective choice if you'll also use it for marketing and data analysis.

- **Pandas**
  Pandas is another Python library that is built on top of NumPy, responsible for preparing high-level data sets for machine learning and training. It relies on two types of data structures, one-dimensional (series) and two-dimensional (Data Frame). This allows Pandas to be applicable in a variety of industries including finance, engineering, and statistics. Unlike the slow-moving animals themselves, the Pandas library is quick, compliant, and flexible.

➢ Class imbalance problem

The first challenge we hit upon exploring the data, is class imbalanced problem. Imbalance data will lead to a bad accuracy of a model. To achieve better accuracy, we'll balance the data by using

Smote Over Sampling or under sampling Method .But in this project data is balanced so ewe are not using it.

Model/s Development and Evaluation

➢ Run and evaluate selected models

Let's select our regression model for this project:

- LinearRegression
- RandomForestRegressor
- KNeighborsRegressor
- SVR
- DecisionTreeRegressor
-

➢ Testing of Identified Approaches (Algorithms)

```
[68]: from sklearn.metrics import r2_score
      from sklearn.linear_model import LinearRegression
      from sklearn.ensemble import RandomForestRegressor
      from sklearn.neighbors import KNeighborsRegressor
      from sklearn.svm import SVR
      from sklearn.tree import DecisionTreeRegressor
      from sklearn.metrics import mean_squared_error,mean_absolute_error
      from sklearn.model_selection import train_test_split
```

```
[75]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=50)
```

```
[82]: lr=LinearRegression()
knr=KNeighborsRegressor()
dtr=DecisionTreeRegressor()
svr=SVR()
rf=RandomForestRegressor(n_estimators=400,random_state=50)
model=[lr,knr,dtr,svr,rf]
for m in model:
    m.fit(x_train,y_train)
    predm=m.predict(x_test)
    print("predicted price:",m,predm)
    print("actual price:",m,y_test)
    print('r2_score:', r2_score(y_test,predm))
    print('error:')
    print('mean absolute error:',m,mean_absolute_error(y_test,predm))
    print('mean squared error:',m,mean_squared_error(y_test,predm))
    print('root mean squarred error:',m,np.sqrt(mean_squared_error(y_test,predm)))
```

```
  694180.6975     372682.5       529762.5       204832.2725
 1296894.495      585294.9975    731857.365     320620.01
  309462.4975     482967.5       893797.52     1147100.02
  332352.485      281744.765     605152.5       565980.
  342463.33666667 560348.61      652577.5      1299522.4775
 1432927.455      370892.59      574447.5      1067100.82833333
```

```
predrf=rf.predict(x_test)
from sklearn.model_selection import cross_val_score
for j in range(4,9):
    rfscore=cross_val_score(rf,x,y,cv=j)
    rfcv=rfscore.mean()
    print('at cv:-',j)
    print('crossvalidation score:',rfcv*100)
    print('r2_score:', r2_score(y_test,predrf))
    print('\n')
```

```
at cv:- 4
crossvalidation score: 58.61370482057766
r2_score: 0.7029994896204437


at cv:- 5
```

```
from sklearn. linear_model import Lasso,LassoCV,Ridge,RidgeCV
```

```
lasscv=LassoCV(alphas=None,cv=10,normalize=True)
lasscv.fit(x_train,y_train)
```

```
▾ LassoCV
LassoCV(cv=10, normalize=True)
```

```
alpha=lasscv.alpha_
alpha
```

```
36.43915041267054
```

```
lasso_reg=Lasso(alpha)
lasso_reg.fit(x_train,y_train)
```

```
▾ Lasso
Lasso(alpha=36.43915041267054)
```

```
lasso_reg.fit(x_train,y_train)
```

```
▾ Lasso
Lasso(alpha=36.43915041267054)
```

```
lasso_reg.score(x_test,y_test)
```

```
]: alphas=np.random.uniform(low=0,high=10,size=(50,))
   ridgecv=RidgeCV(alphas=alphas,cv=10,normalize=True)
   ridgecv.fit(x_train,y_train)
]:                         RidgeCV
        4.65149868, 0.66048062, 1.35006998, 8.22329692, 2.29051879,
        0.39108899, 0.92311271, 8.79736568, 6.79834872, 8.61511689,
        2.4110645 , 4.3864994 , 5.07178347, 1.99919938, 1.89931676,
        3.50803468, 9.35667196, 9.46383061, 2.00288774, 9.99380043,
        4.8406096 , 3.17272645, 7.80635745, 5.90596003, 5.16054   ,
        0.6830677 , 6.39574326, 5.14439787, 3.84461414, 0.59534833,
        6.897924  , 7.93153376, 1.40686601, 1.07758772, 0.07696937,
        8.1991404 , 8.47064566, 4.29815756, 4.15022337, 1.79834582,
        6.70388314, 0.7248519 , 2.67250668, 7.10640323, 7.07634784]),
         cv=10, normalize=True)

]: ridgecv.alpha_
]: 0.07696936814929556

]: ridge_reg=Ridge(alpha=ridgecv.alpha_)
   ridge_reg.fit(x_train,y_train)
]:            Ridge
   Ridge(alpha=0.07696936814929556)

]: ridge_reg.score(x_test,y_test)
]: 0.2748714368873968
```

➢ Key Metrics for success in solving problem under consideration

Selection of a model requires evaluation and evaluation requires a good metric. This is indeed important. If we optimize a model based on incorrect metric, then, our model might not be suitable for the business goals.

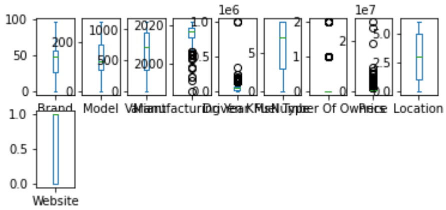
```
print( error: )
print('mean absolute error:',m,mean_absolute_error(y_test,predm))
print('mean squared error:',m,mean_squared_error(y_test,predm))
print('root mean squarred error:',m,np.sqrt(mean_squared_error(y_test,predm)))
```

➢ Visualizations

For better understanding of outliers, I have used boxplot.

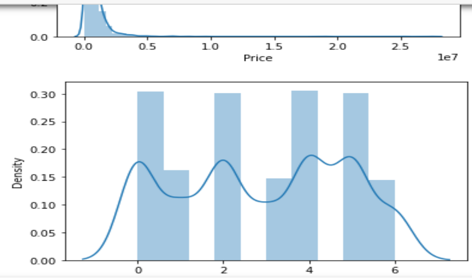
```
In [31]: df.plot(kind='box',subplots=True,layout=(3,9))
```

```
Out[31]: Brand                  AxesSubplot(0.125,0.657941;0.0731132x0.222059)
         Model               AxesSubplot(0.212736,0.657941;0.0731132x0.222059)
         Variant             AxesSubplot(0.300472,0.657941;0.0731132x0.222059)
         Manufacturing Year  AxesSubplot(0.388208,0.657941;0.0731132x0.222059)
         Driven KMs          AxesSubplot(0.475943,0.657941;0.0731132x0.222059)
         Fuel Type           AxesSubplot(0.563679,0.657941;0.0731132x0.222059)
         Number Of Owners    AxesSubplot(0.651415,0.657941;0.0731132x0.222059)
         Price               AxesSubplot(0.739151,0.657941;0.0731132x0.222059)
         Location            AxesSubplot(0.826887,0.657941;0.0731132x0.222059)
         Website                AxesSubplot(0.125,0.391471;0.0731132x0.222059)
         dtype: object
```



For better understanding of skewness, I have used distribution plot.

```
In [32]: for i in df.columns:
             plt.figure()
             sns.distplot(df[i])
```



CONCLUSION

Key aspects of building successful regressor are:

- Selecting correct data according to the purpose or problem statement.
- Proper processing and understanding of the data
- Selecting the model and optimizing the model.

In this project I have dealt with outliers and using z score I removed those outliers for better accuracy.

I have used label encoder for encoding object data type into int datatype as machine doesn’t understand object type data.

I have used three regression model and found Random Forest Regression to be the best fit. It is giving good accuracy.