FLIP ROBO

CAR PRICE PREDICTION

Prepared by: **SME Name:**

NASIM PATIL   Ms.KHUSHBOO GARG

ACKNOWLEDGMENT

INTRODUCTION

1. Business Problem Framing

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, theyare looking for new machine learning models from new data. We haveto make car price valuation model. This project contains two phases:

a. **Data Collection Phase**
b. **Model Building Phase**

2. Conceptual Background of the Domain Problem

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 someare not in demand hence cheaper. So, one of our clients works with small traders, who sell used cars due to covid 19 impact, our client is facing problemswith their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data.

3. Review of Literature

We have to made car price valuation model. This project contains twophases:

c. **Data Collection Phase**: We have scraped more than 6000 used cars data from websites: Olx and cardekho. We have fetched data for different locations. All types of cars are present in data for example- SUV, Sedans, Coupe, minivan, Hatchback.
d. **Model Building Phase**: After collecting the data, built a machine learning model. Before model building have done all data pre- processing steps. Tried different models with different hyper parameters and selected the best model. Followed 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

4. **Motivation for the Problem Undertaken:** 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.

➕ **Analytical Problem Framing**

1. Mathematical/ Analytical Modelling of the Problem

1) Scrapped Data from websites: Cardekho, OLX and Cars24
2) Used Panda's Library to save data into csv file
3) Descriptive Statistics
4) Analysed correlation
5) Detected Outliers and removed
6) Detected Skewness and removed
7) Scaled data using Standard Scaler
8) Removed Multicollinearity

2. Data Sources and their formats

Scraped Data from websites: Cardekho, OLX and Cars24and used Panda's Library to save data into csv file: **car_price.csv**. Target and Features variables of this dataset are:

Target:

☐ **Car_Price:** Price of the used cars

Features:

- ☐ **Brand**
- ☐ **Model**
- ☐ **Variant**
- ☐ **Manufacturing_Year**
- ☐ **Driven_KiloMeters**
- ☐ **Fuel**
- ☐ **Number_of_Owners**
- ☐ **Location**

3. Data Pre-processing Done:

a) Checked Total Numbers of Rows and Column

```
car.shape
```

```
(5616, 10)
```

b) Checked All Column Name

```
car.columns
```

```
Index(['Unnamed: 0', 'Brand', 'Model', 'Variant', 'Manufacturing_Year',
       'Driven_KiloMeters', 'Fuel', 'Location', 'Car_Price',
       'Number_of_Owners'],
      dtype='object')
```

c) Checked Data Type of All Data

```
car.dtypes
```

```
Unnamed: 0            int64
Brand               object
Model               object
Variant             object
Manufacturing_Year   int64
Driven_KiloMeters   object
Fuel                object
Location            object
Car_Price           object
Number_of_Owners    object
dtype: object
```

d) Checked for Null Values

```
car.isnull().sum()
```

```
Unnamed: 0              0
Brand                  0
Model                  0
Variant              127
Manufacturing_Year     0
Driven_KiloMeters      0
Fuel                   0
Location               0
Car_Price              0
Number_of_Owners    3863
dtype: int64
```

There is null value in the dataset.

e) Information about Data

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5483 entries, 0 to 5482
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Brand               5483 non-null   object
 1   Model               5483 non-null   object
 2   Variant             5483 non-null   object
 3   Manufacturing_Year  5483 non-null   int64
 4   Driven_KiloMeters   5483 non-null   object
 5   Fuel                5483 non-null   object
 6   Location            5483 non-null   object
 7   Car_Price           5483 non-null   float64
dtypes: float64(1), int64(1), object(6)
memory usage: 342.8+ KB
```

f) Checked total number of unique values

```
car.nunique()
```

```
Unnamed: 0            5616
Brand                  29
Model                 243
Variant              1237
Manufacturing_Year     23
Driven_KiloMeters    2885
Fuel                    9
Location               73
Car_Price            2322
Number_of_Owners        3
dtype: int64
```

g) Data cleaning

- Column "Number_of_Owners" contains missing value(3863) which is more than 50%. So, dropped this column.

- Column 'Unnamed: 0' contains serial no, so dropped thiscolumn.

```
#Droping column "Number_of_Owners" as it contains most missing value(3863) which is more than 50%.
car.drop(columns=['Number_of_Owners'],inplace=True)
```

```
#Droping column "Unnamed: 0"
car.drop(columns=['Unnamed: 0'],inplace=True)
```

- Column "Variant" contains missing value (127) which isvery less, less than 5%. So, we will keep this column and will drop those rows.

```
#We cannot fill any value in "Variant" as it is unique for each car model,
car.dropna(inplace = True)
```

- Checked all values of each column and replacedirrelevant value or data.

Observation:

1. Brand column:
   - Brand "BMW" and "Bmw" both is same brand so we will replacing "Bmw" with "BMW"
   - Brand "KIA" and "Kia" both is same brand so we will replacing "Kia" with "KIA"
2. Driven_KiloMeters column:
   - Replacing "KM","kms", "km", "," and ".0" as these are not required and column name itself defines these are KM values
3. Fuel column:
   - Fuel "Petrol" and "PETROL" both are same so will replace "Petrol" with "PETROL"
   - Fuel "Diesel" and "DIESEL" both are same so will replace "Diesel" with "DIESEL"
4. Location column:
   - Location "Delhi" and "Delhi NCR" is same so will replace "Delhi NCR" with "Delhi"
5. Car_Price column:
   - Removing "₹", "." and ","
   - Replacing value "Lakh" with "000"
6. Car_Price is object type so wil convert it's datatype from object to integer datatype

```
#Brand column:
car["Brand"]= car["Brand"].str.replace('Bmw', 'BMW')
car["Brand"]= car["Brand"].str.replace('Kia', 'KIA')
```

```
#Driven_KiloMeters column:
car["Driven_KiloMeters"]= car["Driven_KiloMeters"].str.replace('KM', ' ')
```

```
car["Driven_KiloMeters"]= car["Driven_KiloMeters"].str.replace('kms', '')
```

```
car["Driven_KiloMeters"]= car["Driven_KiloMeters"].str.replace('km', '')
```

```
car["Driven_KiloMeters"]= car["Driven_KiloMeters"].str.replace(',', '')
```

```
car["Driven_KiloMeters"]= car["Driven_KiloMeters"].str.replace('.0', '')
```

```
#Fuel column:
car["Fuel"].replace('Petrol', 'PETROL',inplace=True)
```

```
#Fuel column:
car["Fuel"].replace('Diesel', 'DIESEL',inplace=True)
```

```
#Location column:
car["Location"].replace('Delhi NCR', 'Delhi',inplace=True)
```

```
#Car_Price column:
car["Car_Price"]= car["Car_Price"].str.replace('₹', '')
car["Car_Price"]= car["Car_Price"].str.replace('.', '')
car["Car_Price"]= car["Car_Price"].str.replace(',', '')
car["Car_Price"]= car["Car_Price"].str.replace(' Lakh', '000')
```

- Converted Data Type of column "Car_Price" as thiscolumn contains numeric values but it's datatype isobject.

```
car['Car_Price']=pd.to_numeric(car['Car_Price'],errors='coerce')
```

```
car.dtypes
```

```
Brand                  object
Model                  object
Variant                object
Manufacturing_Year      int64
Driven_KiloMeters      object
Fuel                   object
Location               object
Car_Price             float64
dtype: object
```

```
#Checking Null value again after conversion of datatype
car.isnull().sum()
```

```
Brand                0
Model                0
Variant              0
Manufacturing_Year   0
Driven_KiloMeters    0
Fuel                 0
Location             0
Car_Price            6
dtype: int64
```

- After conversion of Datatype, we seen Car_Price column contains null values. So, dropped those rowsonly as total missing value is 6.

```
car.dropna(inplace =
                          car.shape
car.reset_index(inpla  (5483, 8)        rue)

car.isnull().sum()
Brand                   0
Model                   0
Variant                 0
Manufacturing_Year      0
Driven_KiloMeters       0
Fuel                    0
Location                0
Car_Price               0
dtype: int64
```

Now, there is no null value in our dataset.

- Checked again total no of rows and column

h) Data Visualization

i. Univariate Analysis
➢ Used Countplot

ii. Bivariate Analysis
(For comparison between each feature with target)
➢ Used Catplot and Scatterplot

iii. Multivariate Analysis
(For comparison between all feature with target)
➢ Used Pairplot

4. Data Inputs- Logic- Output Relationships

i) Checking Correlation

```
car.corr()
```

	Brand	Model	Variant	Manufacturing_Year	Driven_KiloMeters	Fuel	Location	Car_Price
Brand	1.000000	0.984207	0.123862	0.129157	-0.028763	0.015162	0.000459	-0.003022
Model	0.984207	1.000000	0.106493	0.088589	-0.023588	-0.013179	0.044661	0.002470
Variant	0.123862	0.106493	1.000000	0.005068	-0.072386	0.169128	-0.073757	-0.086074
Manufacturing_Year	0.129157	0.088589	0.005068	1.000000	-0.160688	-0.005101	-0.151555	0.231122
Driven_KiloMeters	-0.028763	-0.023588	-0.072386	-0.160688	1.000000	-0.166733	-0.019620	-0.085451
Fuel	0.015162	-0.013179	0.169128	-0.005101	-0.166733	1.000000	-0.122324	-0.309870
Location	0.000459	0.044661	-0.073757	-0.151555	-0.019620	-0.122324	1.000000	0.032826
Car_Price	-0.003022	0.002470	-0.086074	0.231122	-0.085451	-0.309870	0.032826	1.000000

This gives the correlation between the denpendent and independent variables.

```
car.corr()["Car_Price"].sort_values()

Fuel                  -0.309870
Variant               -0.086074
Driven_KiloMeters     -0.085451
Brand                 -0.003022
Model                  0.002470
Location               0.032826
Manufacturing_Year     0.231122
Car_Price              1.000000
Name: Car_Price, dtype: float64
```

We can observe :

- All columns are sorted in ascending order showing least to strong correlation with target column.
- 3 columns are negatively correlated and 4 columns are positively correlated.
- Column 'Fuel' is highly positively correlated with Target column and Column 'Brand' is highly negatively correlated with Target column

Checking correlation with heatmap

```
plt.figure(figsize=(20,10))
sns.heatmap(car.corr(),annot=True,annot_kws= {"size": 10}, linewidth=0.5, linecolor='white', fmt='.2f')
```

Outcome of Correlation

- **Brand** has **-0 percent** correlation with the target column which can be considered as No correlation and is negatively correlated.
- **Model** has **-6 percent** correlation with the target column which can be considered as good correlation and negatively correlated.
- **Variant** has **10 percent** correlation with the target column which can be considered as good correlation and positively correlated.
- **Manufacturing_Year** has **29 percent** correlation with the target column which can be considered as good correlation and positively correlated.
- **Driven_KiloMeters** has **4 percent** correlation with the target column which can be considered as weak correlation and positively correlated.
- **Fuel** has **33 percent** correlation with the target column which can be considered as strong correlation and positively correlated.
- **Location** has **-21 percent** correlation with the target column which can be considered as weak correlation and negatively correlated.
  - Max correlation is with **Fuel**
 - Min correlation is with **Brand**

Checking corrRlation with barplot

```python
plt.figure(figsize=(15,7))
car.corr()['Car_Price'].sort_values(ascending=True).drop(['Car_Price']).plot(kind='bar',color='c')
plt.xlabel('Feature',fontsize=18)
plt.ylabel('Target',fontsize=14)
plt.title('Correlation',fontsize=18)
plt.show()
```
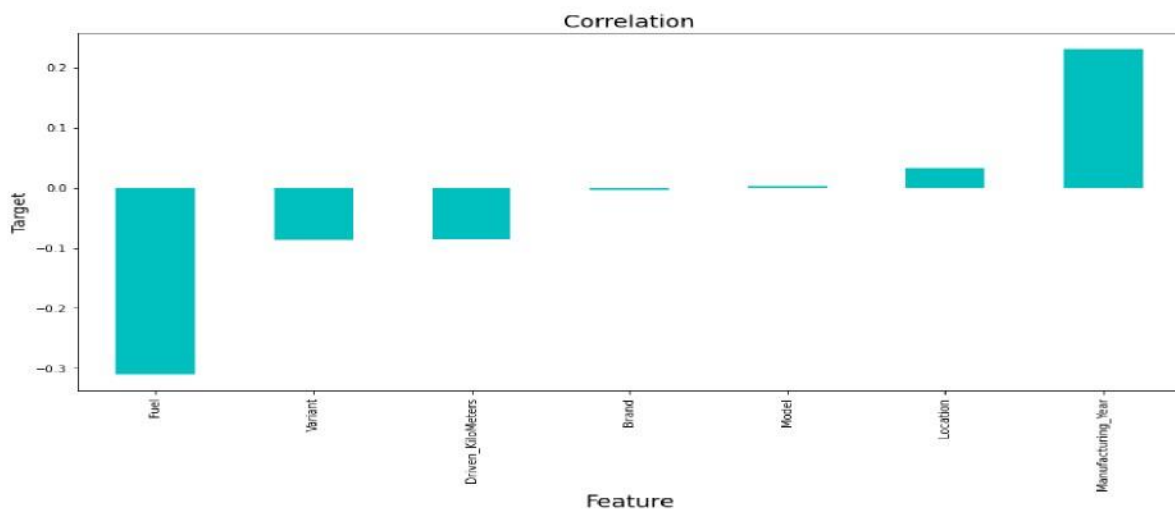
CHECKING OUTLIERS

```
collist=car.columns.values
ncol=8
nrows=7
plt.figure(figsize=(ncol,3*ncol))
for i in range(0,len(collist)):
    plt.subplot(nrows,ncol,i+1)
    sns.boxplot(data=car[collist[i]],color='green',orient='v')
    plt.tight_layout()
```
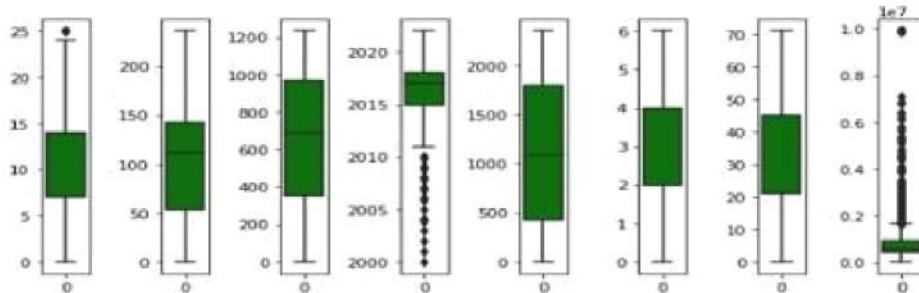

Observation:

- Outliers present in columns: "6rand', 'l.1arufacturino year' and "Car Price'
- But we will not remove Outliers -row ' Brancl" column as it is categorical cc1rimr and free "Car Ffice" coJu mn as it is a target coJumn
- **Outliers not present in columns:** 'Model', 'Variant','Driven_KiloMeters', 'Fuel' and 'Location'.

REMOVING OUTLIERS

- *Outliers are removed only from continuous features and not fi om target*

- Checking two methods and compare between them which is give less percentage loss and then using that method for further process.
1. Zscore method using Scipy
z. IQR (Inter Quartile Range) method

Zscore method using Scipy

```
# Outliers will be removed only from Continuous column i.e; "Manufacturing_Year".
# We will not remove outliers from Categorical column i.e; "Brand".

variable = car[['Manufacturing_Year']]

z=np.abs(zscore(variable))

# Creating new dataframe
car_price = car[(z<3).all(axis=1)]
```

Comparing shape of old and new DataFrame after outliers removal

```
print("Old DataFrame data in Rows and Column:",car.shape)
print("New DataFrame data in Rows and Column:",car_price.shape)
print("Total Dropped rows:",car.shape[0]-car_price.shape[0])

Old DataFrame data in Rows and Column: (5483, 8)
New DataFrame data in Rows and Column: (5413, 8)
Total Dropped rows: 70
```

2.1 IQR (Inter Quantile Range) method

```
#1st quantile
Q1=variable.quantile(0.25)

# 3rd quantile
Q3=variable.quantile(0.75)

#IQR
IQR=Q3 - Q1
car_price_pred=car[~((car < (Q1 - 1.5 * IQR)) |(car > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Comparing shape of old and new DataFrame after outliers removal

```
print("Old DataFrame data in Rows and Column:",car.shape)
print("\nNew DataFrame data in Rows and Column:",car_price_pred.shape)
print("\nTotal Dropped rows:",car.shape[0]-car_price_pred.shape[0])
```

```
Old DataFrame data in Rows and Column: (5483, 8)

New DataFrame data in Rows and Column: (5027, 8)

Total Dropped rows: 456
```

Comparing Data Loss Using both Method after Outlier Removal

1.2 Percentage Data Loss using Zscore

```
loss_percent=(5489-5419)/5489*100
print("loss_percent= ",loss_percent,"%")
```

```
loss_percent=  1.275277828384041 %
```

2.2 Percentage Data Loss using IQR

```
loss_perc = (5489-5033)/5489*100
print("loss_percent= ",loss_perc,"%")
```

```
loss_percent=  8.307524139187466 %
```

We can check by using IQR method there is large data loss in comparision to Zscore method. So, we will consider Zscore method.
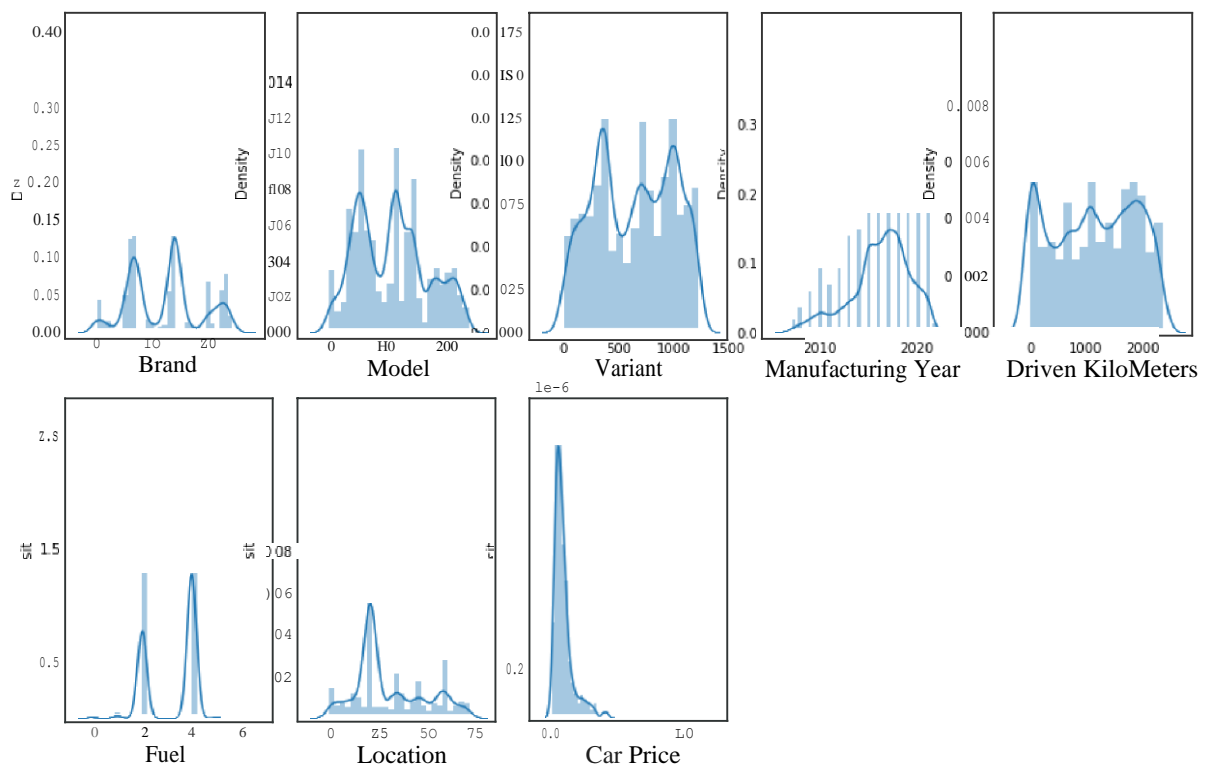
CHECKING SKEWNESS

```
car_price.skew()
```

Brand	0.181949
Model	0.283178
Variant	-0.052374
Manufacturing_Year	-0.703623
Driven_KiloMeters	-0.068305
Fuel	-0.441923
Location	0.640418
Car_Price	5.656679

Checking skweness through Data Visualization also

```python
plt.figure(figsize=(l5,l5). facecolor='ivhite')
plotnumBen = 1


for column in car_price:
    if plotnumber<=i5:
        ax = plt. subplot(3.5,plotnumBec)
        sns.distplot(cac_pcice[column])
        plt.xlabel(column,fontsize=15)
    plotnumber+=1
plt.shoe()
```


Observation:

- Skewness threshold taken is +/-0.2S
- All the columns are not nornallly distributed, they are skewed.
- Columns which are having Skewness: 'Brand', 'Model', 'Manufacturing Year'. 'Fuel'. 'Car Price'.
- The 'Fuel' column data is negatively highly skewed and 'Location' is positively highly sker/ed
- Since 'Brand'. 'Model'. 'Fuel' are categorical column so we will not remove ske 'mess and 'Car Price" is Target Column so we can not remove skewness.
- So we will remove ske\vness from Manufacturing_Year column as it contains continuous data

REMOVING SKEWNESS

Using yeo-johnson method

```
collist=['Manufacturing_Year']
car_price[collist]=power_transform(car_price[collist],method='yeo-johnson')
car_price[collist]
```

	Manufacturing_Year
0	1.642232
1	0.922537
2	0.574776
3	-0.097454
4	1.278296
...	...
5478	-0.739757
5479	-0.097454
5480	-1.050095
5481	0.234837
5482	-0.097454

5413 rows × 1 columns
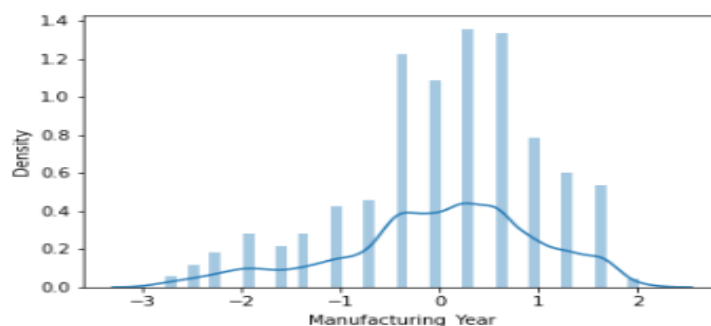
CHECKING SKEWNESS AFTER REMOVAL

```
car_price.skew()
```

```
Brand                 0.181949
Model                 0.283178
Variant              -0.052374
Manufacturing_Year   -0.521735
Driven_KiloMeters    -0.068305
Fuel                 -0.441923
Location              0.640418
Car_Price             5.656679
dtype: float64
```

Still we can see skewness is present but from earlier it is removed.

Checking through Visualization

```
sns.distplot(car_price['Manufacturing_Year'])
```

```
<AxesSubplot:xlabel='Manufacturing_Year', ylabel='Density'>
```

Data preprocessing

Spliting data into Target and Features

```
x.head()
```

	Brand	Model	Variant	Manufacturing_Year	Driven_KiloMetera	Fuel	Location
0	7	71	625	J.642232	T5	4	2
1	10	9g	639	0.922537	332	4	2
2	20	176	75J	0.574776	804	4	2
3	6	46	121J	-0.097454	549	4	2
4	3	28	9J7	J.278296	483	4	2

```
y.head()
```

```
0    2300000.0
1    1553000.0
2     711000.0
3     733000.0
4     381000.0
Name: Car_Price, dtype: float64
```

```
x.shape, y.shape
```

```
((54*3, 7), (54*3,))
```

Oca ling clata using OtancJard Oca ter

```
scaler = StandardScaler()
x = pd.OataFcame(scaler.fit_transform(x),          x.columns)

x.head()
```

	Brand	Model	Variant	k1anufacturing_Year	Driven_Kila¥deters	Fuel	Location
0	-0.810184	-0.610846	-0.039TB9	1.642232	-J.413423	0.723823	
1	-0.32&Z58	-0.2BB392	0.000313	0.92253T	-4.062234	0.723B23	-1.601946
'	1.278161	1.171435	0.321125	0.5T4TT6	-0.417250	0.723B23	-1.601946
3	-D.97D&ZB	-1.D35427	1.63-BT46	-0.09745-4	-D.7B57O5	D.723&Z3	-1.BD1946
4	-1.452752	-1.340609	0.796614	1.2TB296	-0.B55B94	0.7Z3&Z3	-1.601946

Checking for Multicolinearity

VIF (Variance Inflation factor)

```
vif = pd.DataFrame()
gif['VIF valuez']= [variance_inflation_factor(r.values,i) for i in range(len(r.coLumns))j
gif['Frature='] = x.columns
gif
```

	VU-valueB	Features
0	36.S8743-B	Brand
1	@.1W661	Model
2	1.04T921	Variant
3	1.144527	Manufacturinq_Year
4	1.075751	Drivefi_KiloMete+s
5	1.090563	Fuel
6	1.096371	Location

*The VU value is more tha*n 10 in the columns '8rancf; '/I4odel'. But column 'Brand" *is having highest VII* va/ue. So, \+'e wi// *drop column 'Brand'.*

```
' x = r.drop(['brand'],axis=2)
```

```
#Checking again Multicolinearity using VIF
gif = pd.DataFrame()
vif['UIF values']= [variance inflation factor(x.values,i) for i in range(len{x.columns))]
gif['Fraiures'] = xcolsmns
EU
```

	VUvalueB	Features
0	1.035566	Model
1	1.043016	Variant
2	1.090540	ManufacturingVear
3	1.073187	Driven_KiloMeters
4	1.8715T7	fuel
5	1.052502	Localior

How, we can check Multicolineantf is removed from the cmumns as VII *valve of all columns are lens than 10.*

Variance Threshold Method

It removes all features which variance doesn't meet some threshold. By default, it removes all zero-variance features.

```
var_threshold = VarianceThreshold(threshold=0)
var_threshold.fit(x)
```

```
VarianceThreshold(threshold=0)
```

```
var_threshold.get_support()
```

```
array([ True,  True,  True,  True,  True,  True])
```

```
x.columns[var_threshold.get_support()]
```

```
Index(['Model', 'Variant', 'Manufacturing_Year', 'Driven_KiloMeters', 'Fuel',
       'Location'],
      dtype='object')
```

```
# taking  out  all  the  constant  columns
cons_columns = [column for column in x.columns
                if column not in x.columns[var_threshold.get_support()]]
print(len(cons_columns))
```

```
0
```

So we can see that, with the help of variance threshold method, we got to know all the features here are important. So, we will create model now.

5. State the set of assumptions (if any) related to the problem under consideration

- By observing Target Variable "label" it is already assumed that it is a Regression Problem and to understand it have to use Regression model.
- Also, it was observed that there is one column "Unnamed 0" which is irrelevant column as it contains serial no so have to drop this column.
- Have to convert datatype of "Car_Price" column.

6. Hardware and Software Requirements and Tools Used

- **Hardware used:**
- **Processor**: 11th Gen Intel(R) Core(TM) i3-1125G4 @2.00GHz 2.00 GHz
- **System Type**: 64-bit OS
- **Software used:**
- **Anaconda** for 64-bit OS
- **Jupyter** notebook

- **Tools, Libraries and Packages used:**

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from scipy.stats import zscore
from sklearn.preprocessing import power_transform, StandardScaler, LabelEncoder
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_classif
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split, GridSearchCV,cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import roc_curve, auc, roc_auc_score, plot_roc_curve, r2_score, classification_report, mean_absolute_error, mean_squared_error
from sklearn.metrics import confusion_matrix, mean_absolute_error, mean_squared_error
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR

import pickle
```

Model/s Development and Evaluation

1. Identification of possible problem-solving approaches(methods)

   In this project, we want to predict the micro-credit defaulter and for thiswe have used these approaches:

- Checked Total Numbers of Rows and Column
- Checked All Column Name
- Checked Data Type of All Data
- Checked for Null Values
- Checked for special character present in dataset or not
- Checked total number of unique values
- Dropped irrelevant Features
- Replaced duplicate values, special characters and irrelevant data
- Checked all features through visualization.
- Information about Data
- Checked correlation of features with target
- Detected Outliers and removed
- Checked skewness and removed
- Scaled data using Standard Scaler
- Checked Multicollinearity
- Used Feature Selection Method: Variance threshold method

Testing of Identified Approaches (Algorithms)
1. Linear Regression
2. Random Forest Regressor
3. KNN Regressor
4. Gradient Boosting Regressor
5. Decision Tree Regressor

2. Run and evaluate selected models

Creating Model

Finding the best random state among all the models

On the basis of target column as it contains continuous data, we will understand this by Regression Problem

```
maxAcc = 0
maxRS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = .20, random_state = i)
    modDTR =  DecisionTreeRegressor()
    modDTR.fit(x_train,y_train)
    pred = modDTR.predict(x_test)
    acc  = r2_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")
```

Best Accuracy is: 0.920262170460058 on random_state: 19

Creating train-test-split

```
# creating new train test split using the random state.
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = .25, random_state = maxRS)
```

Regression Algorithm

1. Linear Regression

```python
# Checking r2score for Linear Regression
LR = LinearRegression()
LR.fit(x_train,y_train)

# Brett std on
predLR=LR.peed ict(x_test )
print('R2 score:',r2_score(y_test,predLR))
print('Mran ab:e++'or:',mean_absolute_error(y_test, predLR))
print('Mran squa+'ed rror:',mean_squared error(y_test, predLR))
print("Root Mean Squdrrd Error: ", np.sqrt(mean_squared_error(y_test,predLR)))

R2_score: 0.16172401078532514
Mean abs error: a25238.7765592ag3
Mean squared error: 797760765690.3414
Soot Mean Squared Error:  69317a.5437988623
```

2. Random forest Regression Model

```python
# Checking R2 score for Random Forest Regressor
RFR=RandomForestRegressor(n_estimators=C0•B, random_state=maxRS)
RFR.fit(x_train,y_train)

# prediction
predRFR=RFR.predict(x_test)
print('R2_Score:',r2_score(y_test,predRFR))
print('Mran abs error:',mean_absolute_error(y_test, predRFR))
print('Mran squared rrror:',mean_squared_error(y_test, predRFR))
print("Root .Mean Squarrd Error: ", np.sqrt(mean_squared_error(y_test,predRFR)))

R2_Sc ore : 8 . 9 514161523616257
Mean abs error: Bt944.57991BB2557
Mean squared error: 46235712332.01829
Root f9ean Squared Error:  21502d.91&95688957
```

3. KNN Regressor

```
# Checking R2 score for KNN regressor
knn=KNeighborsRegres5or(n_neighbor5=B )
knn.fit(x_train,y_train)

#prediction
predknn=knn.predict(x_test)
pmind('R2_Scowe:', r2  scome {y _6es6, pred knn))
p mind('f•1ean abs ei- no :', me an_abso1u6e_e wr or y_4es6,  p wedknn})
p mind('f•1can  s qu aimed c zoo :,' mean_squawe d_er row (yokest   pm edknn})
print("Root .Mean Squarrd Error: ", np.sqrt(m>ean_squared_error(y_test,predknn)))

lR2_Score: 0.63366240#9265553
mean abs error:  251330.278552ZB745
mean squared error: 3  631911335. 91406
Root Clean Squared Error:  59B45B.6fflB250Z784
```

4. Gradient boosting Regressor

```
# Checking R2 score for GBR
Gfi= GradientBoostingRegressor(n_estimators=400, random_state=maxRS, learning_rate=0.1, max_depth=3)
GD.fit(x_tnain,y train)

Up.•edi ct t on
predGb-6b . pr'ed ict ( x_test )
print ('81 Score: ', r2 score(y_test, predGb ) )
pr1nt( l'.ean,   d s emor: , nean_absoltite error(y_test,  pned6b ) }
print ( 'l".eai1 Squared error: ', mean squared error(y test, pr'edGb) )
pr:int ( "Boot mean S§uai•ed Er ror: ", np . sqr't (mean_squared_error(y_test , predGL) ) )

R2 Scoe: 0.950817155983341
)ean abs error: 102938.44305000136
Plan sqaedenon: 42747311647.7256a
ROot Mean Squared Error: 206754.23006005381
```

5. Decision Tree Regressor

```
# Checking R2 score for GBR
DTR= DecisionTreeRegressor()
DTR.fit(x train,y trains

predDTR=DTR. predict (x_Le st )
print ('R2 Score:', r2   score (y £es£, predDTR))
print ( 'l.eais a frs e rror :,' mean_absOlufe_errOr ( y_tesf, predDTR ))
print('H.ean squared error:',mean_squared_error(y_test, predDTR))
print("9oot Clear. Squared Error: ", np.sqrt(mean squared error(y test,predDTR))}

R 2_S cone: B . 877369466468728
f'1ean a bs error : 1B9852. 2B9 512 55539
Mean squared error: 1167607032?2.8é07
Root Mean Squared Error:  341782.65327165765
```

Cross Validation Score for all the model

```
#CV Score for Linear Regress ion
print('CV score fOr Linear eegnession: ',cross val score(LR,x,y,cv=5).mean(})

#CP Score for R0ndom Forest Pegression
print('CV score fOr Random forest Regression: ',cross_val_score(RFR,x,y,cv=5).mean())

#CV Score /or KNN Regr'es s i on
pnint('CV score for ANN eegnession: ',cross_val_score(knn,x,y,cv=5).mean())

#CV Score for Grodient 8oosting Pegression
print('CV score fOr Gradient BDosting Regression: ',cross_val_score(Gb,x,y,cv=S}.mean())

PfP Score for Deci sion Tree Regression
print('CV score for Decision Tree Regression: ',cross_val_score{DTA,x,y,cv=5).mean())

CY score for Linear Regression:  0.1S7072IS392i15464
CV score for Random forest Regression: 8.74649434372859S2
CV score for KNB Regression:  0.321[9277110194466
CV score for tradient Boosting Regression:  0.7seg483B31060308
CV scone for Decision Tree Regression:  B.5074589918030621
```

Hyper Parameter **Tuning**

The Gradient boosting regressor with GridsearchCV

```
parameter ={'n estimators':[180,200,300,400],
          'learning_rate':[8.1,0.01,9.B0l,l],
          'subsample': [B.1,8.Z,8.3,0.5,1],
          'max_depth':[1,2,3,4],
          'alpha':[0.1,0.Bl,0.8B1,1])

CV GBR   GridSearchCV(GradientBoostingRegressor(),parameter,cv=6,n jobs   3,verbose  2)

rV_GBR.fit(x_train,y_train)

Fitting 6 folds for each of 1280 candidates, totalling 7680 fits

GridSearchCV(cv=6, estInaton=GradientBoostingRegr'essor () , n jobs=3,
          par'am_gr1d=( ' alpha ' : [6. 1, 8.B1, B. 061, 1,]
                     learning_rate': [0.t, 0.B1, 0.001, 1],
                    'max_depth': [1, 2, 3, 4],
                    'n_estimators': [100, 200, 3B0, 480},
                    'subsample': [0.1, 0.2, 0.3, 0.5, 1]},
          ver'bose=2)




CV  GBR . be sly pa rams

{'alpha':  B.0B1,
  learnim   rate' : 0. 1,
  " max _ dB P* h ":  4
  "n_e  s t:1maca as " :   4OB,
  ' su b s amP*   ' :    - * l
```

## Creating Regressor Model with Gradient Boosting Regressor

```python
GBR = GradientBoostingRegressor(n_estimators=400, alpha=0.001,learning_rate= 0.1, max_depth= 4, subsample = 0.5)
GBR.fit(x_train, y_train)
```

```
GradientBoostingRegressor(alpha=0.001, max_depth=4, n_estimators=400,
                          subsample=0.5)
```

```python
#prediction
GBRpred = GBR.predict(x_test)
#R2 score
acc = r2_score(y_test,GBRpred)
print(acc*100)
```

```
95.60799200909335
```

```python
#Verifying the final  performance of the model by graph
plt.figure(figsize=(10,10))
sns.scatterplot(x=y_test,y=GBRpred,palette='Set2')
sns.kdeplot(x=y_test,y=GBRpred, cmap='Set1')
plt.plot(y_test,y_test,color='g')
#Verifying the performance of the model by graph
plt.xlabel("Actual Car Price by GBR")
plt.ylabel("Predicted Car Price by GBR")
plt.title(" Final  metrics of  accuracy  from Hypertuning of Gradient boosting regressor")
plt.show()
```
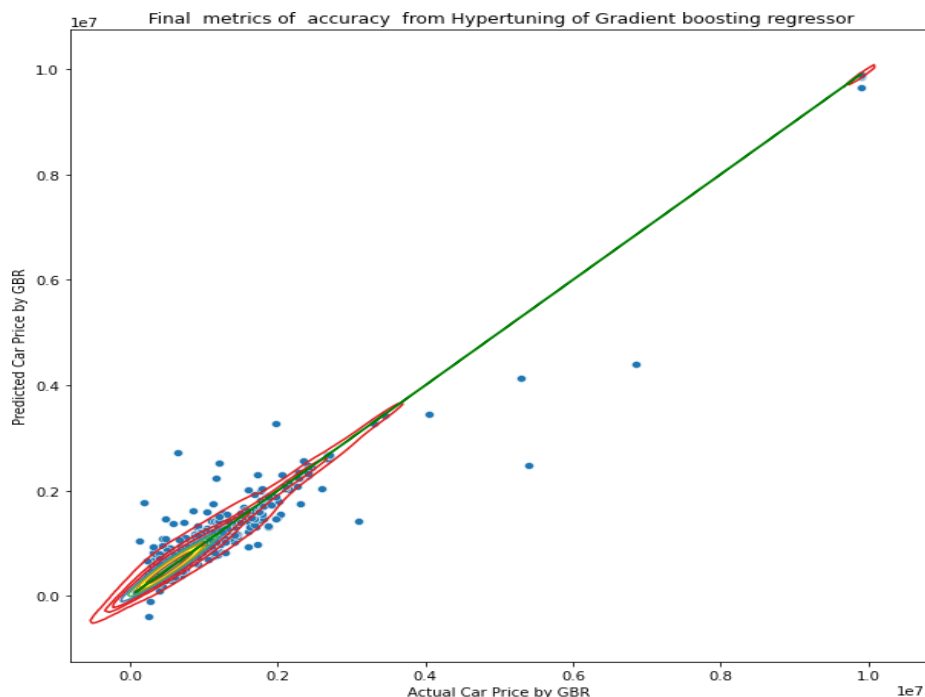


```python
#saving the model at local file system
filename='Car_price_prediction.pickle'
pickle.dump(CV_GBR,open(filename,'wb'))
#prediction using the saved model
loaded_model = pickle.load(open(filename, 'rb'))
loaded_model.predict(x_test)
```
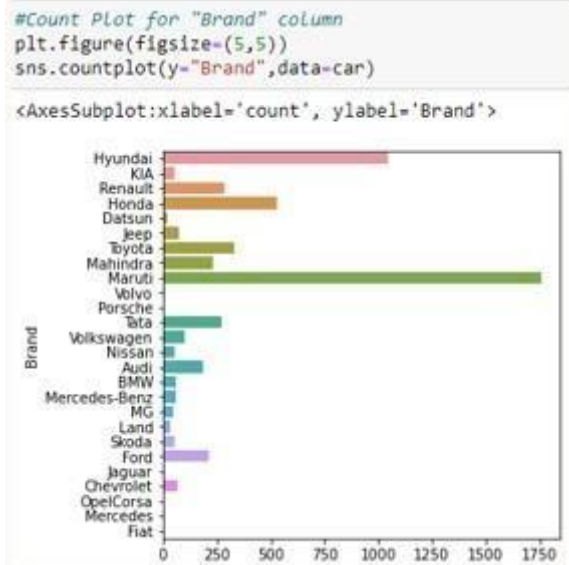
Saving The Predictive Model

1. Key Metrics for success in solving problem underconsideration
- R2 Score, Mean abs error, Mean squared error, Root MeanSquared Error, CV score are used for success.
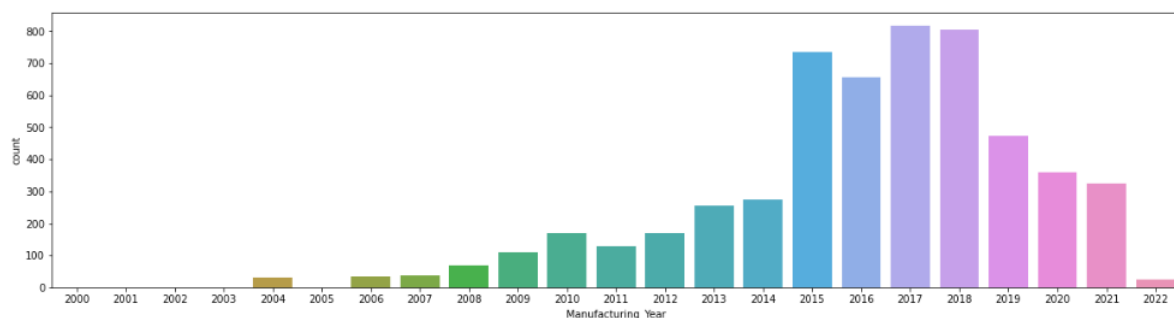 Visualization

- Univariate Analysis
- Using Countplot

```
#Count Plot for "Brand" column
plt.figure(figsize=(5,5))
sns.countplot(y="Brand",data=car)
```

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


```
#Count Plot for "Manufacturing_Year" column
plt.figure(figsize=(20,5))
sns.countplot(x="Manufacturing_Year",data=car)
```
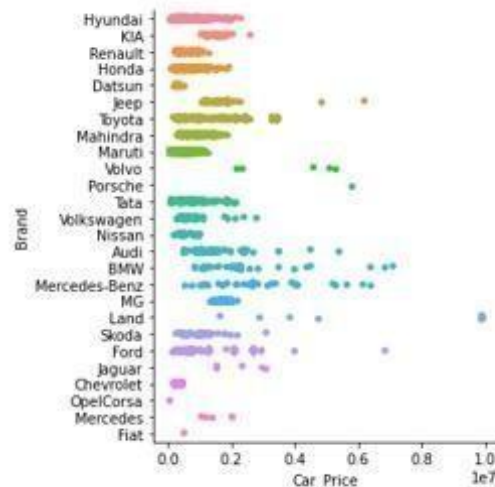
`<AxesSubplot:xlabel='Manufacturing_Year', ylabel='count'>`

- Bivariate Analysis
  (For comparison between each feature with target)
- Using Catplot

```
plt.figure(figsize=(25,20))
sns.catplot(x='Car_Price', y= 'Brand', data = car)
plt.show()

<Figure size 1800x1440 with 0 Axes>
```
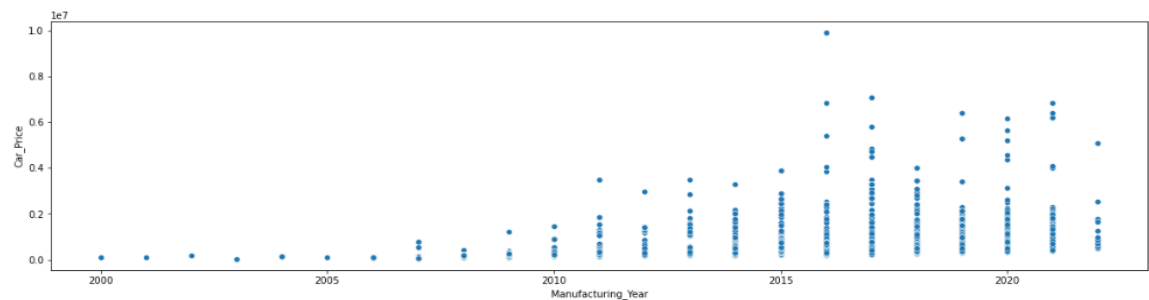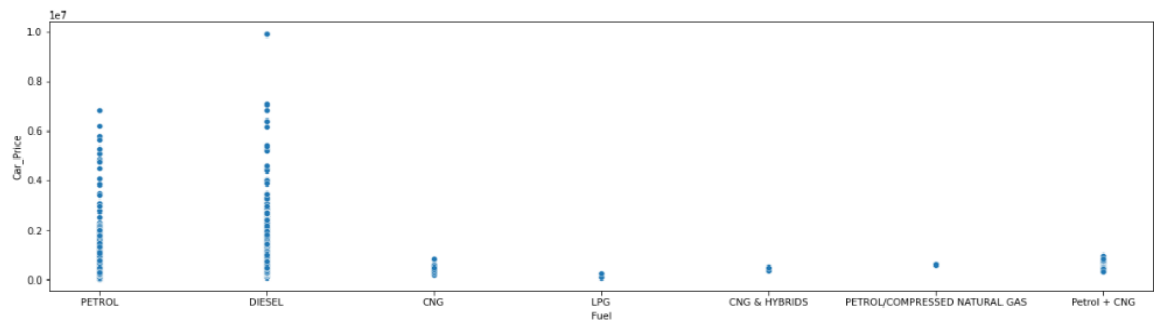

- Using Scatterplot

```
#scatterplot for comparision between "Manufacturing_Year" and "Car_Price" column
plt.figure(figsize=(20,5))
sns.scatterplot(x="Manufacturing_Year",data=car, y='Car_Price')

<AxesSubplot:xlabel='Manufacturing_Year', ylabel='Car_Price'>
```



```
#scatterplot for comparision between "Fuel" and "Car_Price" column
plt.figure(figsize=(20,5))
sns.scatterplot(x="Fuel",data=car, y='Car_Price')

<AxesSubplot:xlabel='Fuel', ylabel='Car_Price'>
```

- Multivariate Analysis
  (For comparison between all feature with target)
- ➢ Using Pairplot

Interpretation of the Results

- Through Visualization it is interpretated that Data is skewed dueto presence of outliers in Dataset.

- Through Pre-processing it is interpretated that outliers & skewness was present in dataset, data was improper scaled, multicollinearity was present.

- By creating/building model we get best model: Gradient Boosting Regressor.

CONCLUSION

6. Key Findings and Conclusions of the Study

Here we have made a new car price valuation model as due to covid19 impact previous car price valuation machine learning models is not working well because some cars are in demand hence making them costly and some are not in demand hence cheaper.

For new car price valuation model, we have done prediction on basis of Data using EDA, Data Cleaning, Data Visualization, Data Pre- processing, Checked Correlation, removed irrelevant features,

Removed Outliers, Removed Skewness and at last train our data by splitting our data through train-test split process.

Built our model using 5 models and finally selected best model which was giving best accuracy that is Gradient Boosting Regressor. Then tunned our model through Hyper Tunning using GridSearchCV. And at last compared our predicted and Actual Price of Car. Thus, our project is completed.

7. **Learning Outcomes of the Study in respect of DataScience**

This project has demonstrated the importance of scrapping data then converting that data of those data into csv and then using that csv file bult a model to predict on data.
Through different powerful tools of visualization, we were able to analyse and interpret different hidden insights about the data.
Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

The few challenges while working on this project were: -
Improper scaling: scaled it to a single scale using Standard Scaler
Too many features: 10 features were present in the dataset, after data cleaning 2 features were reduced due to no relation with target variable and last after removing multicollinearity, we were able to reduce 1 more feature.
Converted datatype of target column
Replaced irrelevant values or data from features
Skewed data due to outliers: Removed using power transformer 'yeo-johnson' method and outliers was removed through zscore.

8. Limitations of this work and Scope for FutureWork

While we couldn't reach out goal of 100% accuracy in detecting defaulter but created a system that made data get very close to that goal. This project allows multiple algorithms to be integrated together to combine modules and their results to increase theaccuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others which will make modules easy to add as done in the code.