LINEAR MODEL ASSIGNMENT

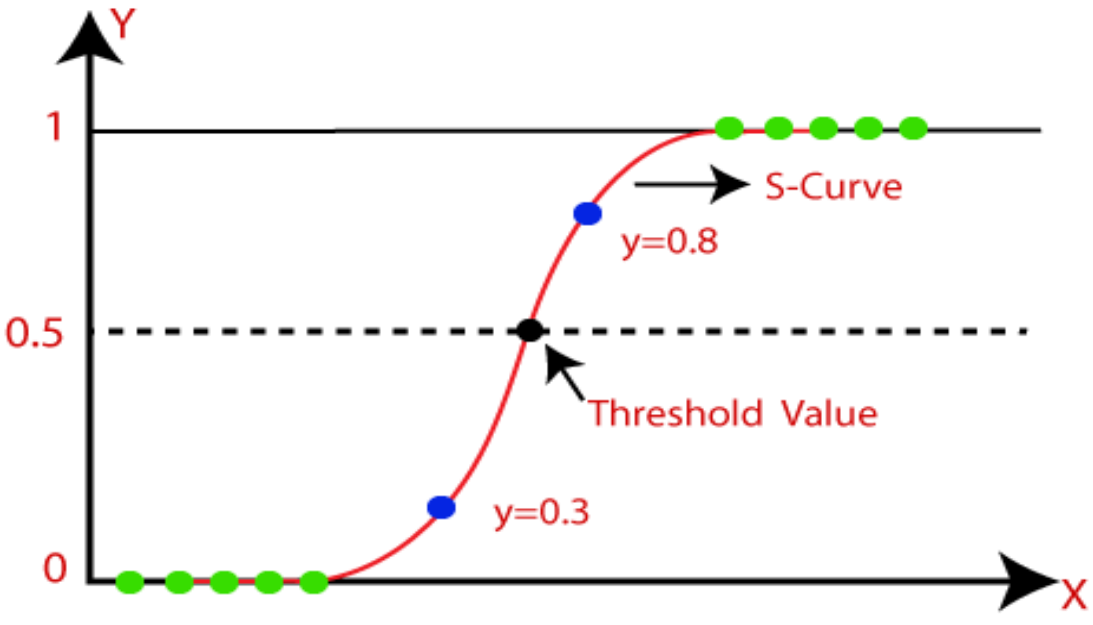
Name: Vedangi Duse (22060641036)

Name: Prajakta Sawant (22060641038)

Name: Gauri Salunke (22060641048)

1. Logistic Regression

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

For example,

To predict whether an email is spam (1) or (0):

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Introduction to data:

The Small Business Administration (SBA) was founded in 1953 to assist small businesses in obtaining loans. Small businesses have been the primary source of employment in the United States. Helping small businesses help with job creation, which reduces unemployment. Small business growth also promotes economic growth. One of the ways the SBA helps small businesses is by guaranteeing bank loans. This guarantee reduces the risk to banks and encourages them to lend to small businesses. If the loan defaults, the SBA covers the amount guaranteed, and the bank suffers a loss for the remaining balance.

There have been several small business success stories like FedEx and Apple. However, the rate of default is very high. Many economists believe the banking market works better without the assistance of the SBA. Supporter claim that the social benefits and job creation outweigh any financial costs to the government in defaulted loans.

The Data Set:

The original data set is from the U.S.SBA loan database, which includes historical data from 1987 through 2014 (899,164 observations) with 27 variables. The data set includes information on whether the loan was paid off in full or if the SMA had to charge off any amount and how much that amount was. The data set used is a subset of the original set. It contains loans about the Real Estate and Rental and Leasing industry in California. This file has 2,090 observations and 35 variables. The column Default is an integer of 1 or zero, and I had to change this column to a factor.

The data set includes information on whether the loan was paid off in full or if the SMA had to charge off any amount and how much that amount was. The data set used is a subset of the original set. It contains loans about the Real Estate and Rental and Leasing industry in California. This file has 2,090 observations and 35 variables. The column Default is an integer of 1 or zero, and I had to change this column to a factor.

Introduction to data:

Variable name	Data type	Description of variable
Loan NrChkDgt	Text	Identifier Primary key
Name	Text	Borrower name
City	Text	Borrower city
State	Text	Borrower state
Zip	Text	Borrower zip code
Bank	Text	Bank name
BankState	Text	Bank state
NAICS	Text	North American industry classification system code
ApprovalDate	Date/Time	Date SBA commitment issued
ApprovalFY	Text	Fiscal year of commitment
Term	Number	Loan term in months
NoEmp	Number	Number of business employees
NewExist	Text	1 = Existing business, 2 = New business
Create Job	Number	Number of jobs created
Retained Job	Number	Number of jobs retained
Franchise Code	Text	Franchise code, (00000 or 00001) = No franchise
Urban Rural	Text	1 = Urban, 2 = rural, 0 = undefined
RevLineCr	Text	Revolving line of credit: Y = Yes, N = No
LowDoc	Text	LowDoc Loan Program: Y = Yes, N = No
ChgOffDate	Date/Time	The date when a loan is declared to be in default
DisbursementDate	Date/Time	Disbursement date
DisbursementGross	Currency	Amount disbursed
BalanceGross	Currency	Gross amount outstanding
MIS_Status	Text	Loan status charged off = CHGOFF, Paid in full = PIF
ChgOffPrinGr	Currency	Charged-off amount
GrAppv	Currency	Gross amount of loan approved by bank
SBA_Appv	Currency	SBA's guaranteed amount of approved loan
New	Number	=1 if NewExist=2 (New Business), =0 if NewExist=1 (Existing Business)
Portion	Number	Proportion of gross amount guaranteed by SBA
RealEstate	Number	=1 if loan is backed by real estate, =0 otherwise
Recession	Number	=1 if loan is backed by real estate, =0 otherwise
Selected	Number	=1 if the data are selected as training data to build model for assignment, =0 if the data are selected as testing data to validate model
Default	Number	=1 if MIS_Status=CHGOFF, =0 if MIS_Status=PIF
Daysterm	Number	How many Days were in the Loan Terms
xx	Number	Amount of Default if Any

Analysis of the Data:

Importing the Dataset:

```
import pandas as pd
import numpy as np
import seaborn as sb
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('SBA.csv')
df
```

	LoanNr_ChkDgt	Name	City	Bank	BankState	NAICS	ApprovalDate	ApprovalFY	Term	NoEmp
0	1004285007	SIMPLEX OFFICE SOLUTIONS	ANAHEIM	CALIFORNIA BANK & TRUST	CA	532420	08-04-1941	2001	36	1
1	1004535010	DREAM HOME REALTY	TORRANCE	CALIFORNIA BANK & TRUST	CA	531210	03-06-1941	2001	56	1
2	1005005006	Winset, Inc. dba Bankers Hill	SAN DIEGO	CALIFORNIA BANK & TRUST	CA	531210	31-07-1941	2001	36	10
3	1005535001	Shiva Management	SAN DIEGO	CALIFORNIA BANK & TRUST	CA	531312	13-01-1943	2003	36	6
4	1005996006	GOLD CROWN HOME	LOS ANGELES	SBA - EDF ENFORCEMENT ACTION	CO	531390	07-02-1946	2006	240	65

Looking for rows and columns:

```
df.shape
```

```
(2090, 28)
```

```
df.duplicated().sum()
```

```
0
```

Looking for null values:

df.isnull().sum()

```
LoanNr_ChkDgt         0
Name                  0
City                  0
Bank                  0
BankState             0
NAICS                 0
ApprovalDate          0
ApprovalFY            0
Term                  0
NoEmp                 0
NewExist              0
CreateJob             0
RetainedJob           0
UrbanRural            0
RevLineCr             0
LowDoc                0
DisbursementDate      0
DisbursementGross     0
MIS_Status            0
GrAppv                0
SBA_Appv              0
New                   0
RealEstate            0
Portion               0
Recession             0
daysterm              0
xx                    0
Default               0
dtype: int64
```

print(df.dtypes)

```
LoanNr_ChkDgt          int64
Name                  object
City                  object
Bank                  object
BankState             object
NAICS                  int64
ApprovalDate          object
ApprovalFY             int64
Term                   int64
NoEmp                  int64
NewExist               int64
CreateJob              int64
RetainedJob            int64
UrbanRural             int64
RevLineCr             object
LowDoc                object
DisbursementDate      object
DisbursementGross      int64
MIS_Status            object
GrAppv                 int64
SBA_Appv               int64
New                    int64
RealEstate             int64
Portion              float64
Recession              int64
daysterm               int64
xx                     int64
Default                int64
dtype: object
```

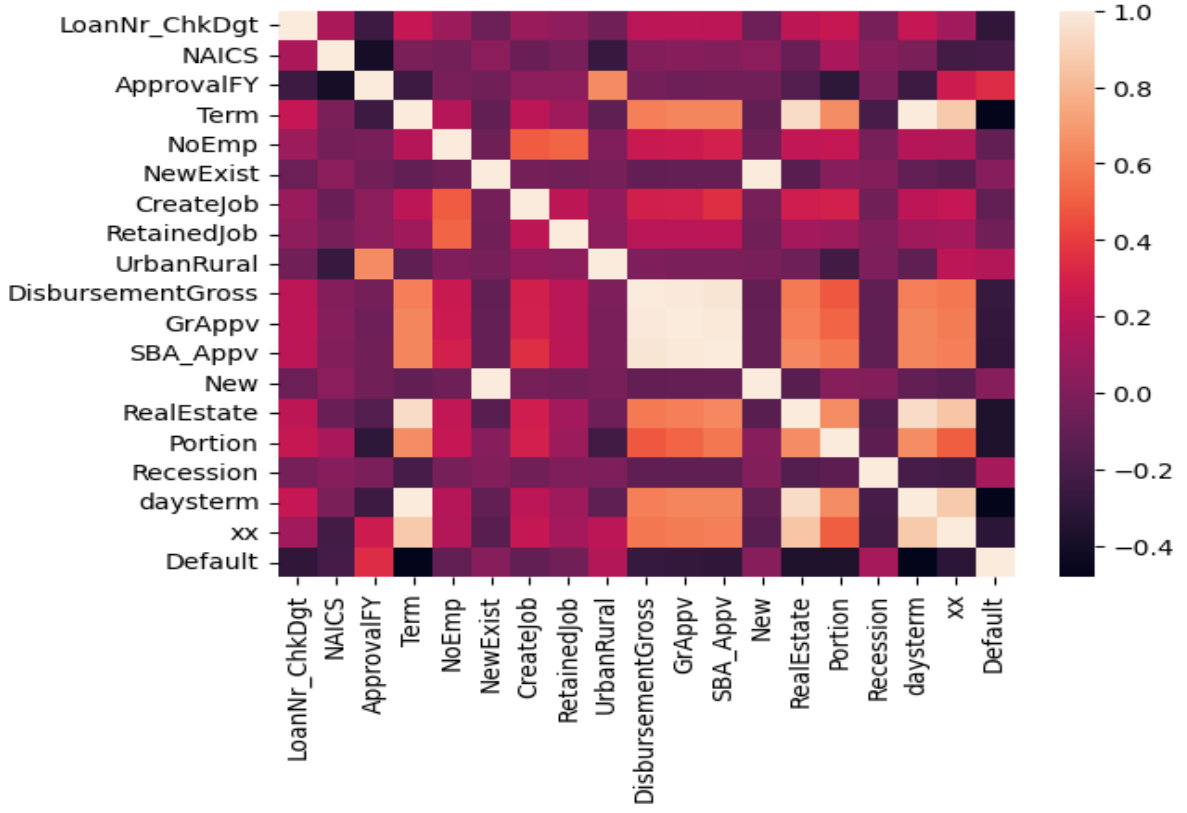
Correlation Matrix:

corr=df.corr()

corr

	LoanNr_ChkDgt	NAICS	ApprovalFY	Term	NoEmp	NewExist	CreateJob	RetainedJob	UrbanRural
LoanNr_ChkDgt	1.000000	0.141887	-0.243475	0.231238	0.091456	-0.075637	0.084100	0.047015	-0.054789
NAICS	0.141887	1.000000	-0.397422	-0.018198	-0.045724	0.033774	-0.081839	-0.032816	-0.266772
ApprovalFY	-0.243475	-0.397422	1.000000	-0.241592	-0.030008	-0.052479	0.036896	0.039695	0.646476
Term	0.231238	-0.018198	-0.241592	1.000000	0.181741	-0.110053	0.206312	0.101471	-0.121960
NoEmp	0.091456	-0.045724	-0.030008	0.181741	1.000000	-0.070615	0.492182	0.517613	-0.005620
NewExist	-0.075637	0.033774	-0.052479	-0.110053	-0.070615	1.000000	-0.038431	-0.056031	-0.031185
CreateJob	0.084100	-0.081839	0.036896	0.206312	0.492182	-0.038431	1.000000	0.208189	0.056399
RetainedJob	0.047015	-0.032816	0.039695	0.101471	0.517613	-0.056031	0.208189	1.000000	0.033306
UrbanRural	-0.054789	-0.266772	0.646476	-0.121960	-0.005620	-0.031185	0.056399	0.033306	1.000000
DisbursementGross	0.196746	0.009826	-0.041184	0.603152	0.251832	-0.107216	0.287368	0.192056	-0.009334
GrAppv	0.201355	0.020359	-0.061343	0.624433	0.254544	-0.098504	0.292668	0.190004	-0.022941
SBA_Appv	0.198756	0.007835	-0.052364	0.626550	0.287908	-0.098353	0.349665	0.199041	-0.020642
New	-0.074641	0.032788	-0.052957	-0.108724	-0.070675	0.998173	-0.036800	-0.055735	-0.031105
RealEstate	0.209947	-0.081488	-0.162812	0.939537	0.219079	-0.146379	0.271005	0.116295	-0.064026
Portion	0.237895	0.136527	-0.307470	0.651305	0.231418	0.016977	0.290646	0.091841	-0.228091
Recession	-0.033710	0.015309	-0.022271	-0.206338	-0.031226	0.004611	-0.052811	-0.004122	-0.009751
daysterm	0.231238	-0.018198	-0.241592	1.000000	0.181741	-0.110053	0.206312	0.101471	-0.121960
xx	0.112032	-0.219794	0.260455	0.872017	0.171524	-0.137277	0.233046	0.123033	0.206279
Default	-0.292461	-0.215068	0.341245	-0.483012	-0.110195	0.016314	-0.109024	-0.049389	0.166517

corr=sb.heatmap((df.corr()))



Fitting Logistic Regression Model:

X=df[['Term']]

Y=df[['Default']]

We split data into train and test:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = None)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
explained_variance
```

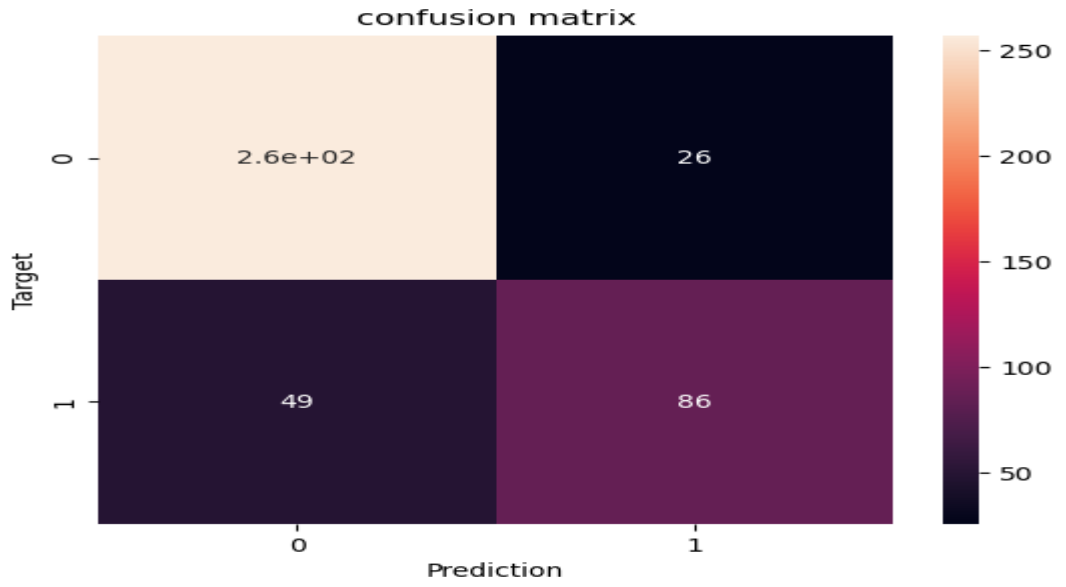
```
array([1.])
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, max_iter=1000)
classifier.fit(X_train, y_train)
```

Confusion Matrix:

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
cm = confusion_matrix(y_test, y_pred)
tp, fn, fp, tn = confusion_matrix(y_test, y_pred, labels=[1,0]).reshape(-1)

print('outcome values testing data:\n\n')
print('True positive',tp)
print('false negative',fn)
print('false postive',fp)
print('true negative',tn)
print(sep='\n')

plt.figure()
sns.heatmap(cm, annot= True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('confusion matrix')
```

```
outcome values testing data:


True positive 86
false negative 49
false postive 26
true negative 257

Text(0.5, 1.0, 'confusion matrix')
```



Interpretation:

1. True Positive: Actual & Predicted value are same/positive.
2. True Negative: Actual & Predicted value are same/negative.
3. False Positive: Actual values are negative & Predicted values are positive.
4. False Negative: Actual values are positive & Predicted values are negative.

Train Test Model Accuracy:

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 0.8205741626794258
Precision: 0.7678571428571429
Recall: 0.6370370370370371
F1 score: 0.6963562753036437
```

Interpretation:

1. Accuracy: Fitting of the model is 82% better.
2. Precision: Fitting of the predicted model is 76% better.
3. Recall: Model is 63% good at correctly predicted value.
4. F1 score: Average of precision & recall.

OLS Model:

Theory:

Ordinary Least Squares (OLS) method is widely used to estimate the parameter of a linear regression model. The goal is minimizing the differences between the collected observations in some arbitrary dataset and the responses predicted by the linear approximation of the data.

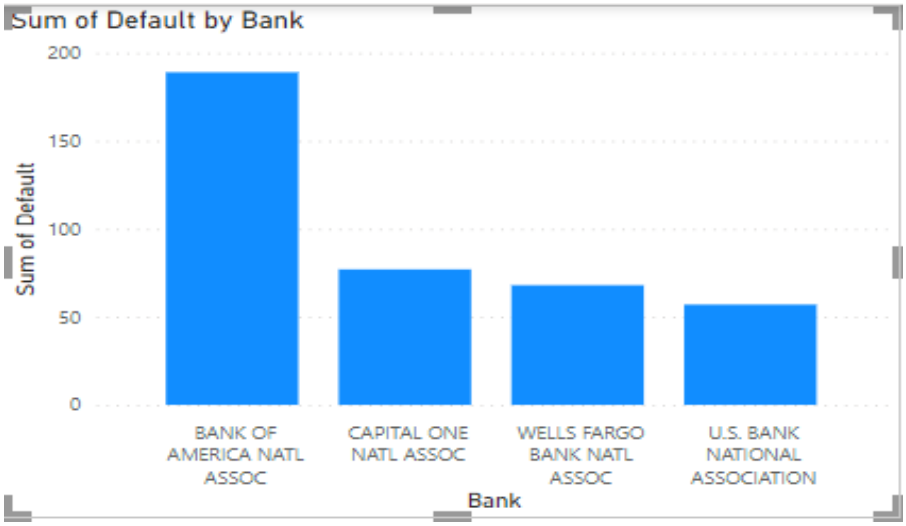
```
get_ipython().system('pip install statsmodel')
import scipy.signal.signaltools

def _centered(arr, newsize):
    # Return the center newsize portion of the array.
    newsize = np.asarray(newsize)
    currsize = np.array(arr.shape)
    startind = (currsize - newsize)  // 2
    endind = startind + newsize
    myslice = [slice(startind[k], endind[k]) for k in range(len(endind))]
    return arr[tuple(myslice)]
scipy.signal.signaltools._centered = _centered
import statsmodels.api as sm
import pandas as pd
x= sm.add_constant(X)
result = sm.OLS(Y,X).fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================================
Dep. Variable:                 Default   R-squared (uncentered):                   0.051
Model:                             OLS   Adj. R-squared (uncentered):              0.050
Method:                  Least Squares   F-statistic:                              111.6
Date:                 Sat, 29 Apr 2023   Prob (F-statistic):                    1.87e-25
Time:                         19:40:01   Log-Likelihood:                         -1745.5
No. Observations:                2090   AIC:                                      3493.
Df Residuals:                    2089   BIC:                                      3499.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Term           0.0008   7.72e-05     10.565      0.000       0.001       0.001
==============================================================================
Omnibus:                    17039.154   Durbin-Watson:                   1.377
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              348.212
Skew:                           0.692   Prob(JB):                     2.44e-76
Kurtosis:                       1.556   Cond. No.                         1.00
==============================================================================
```

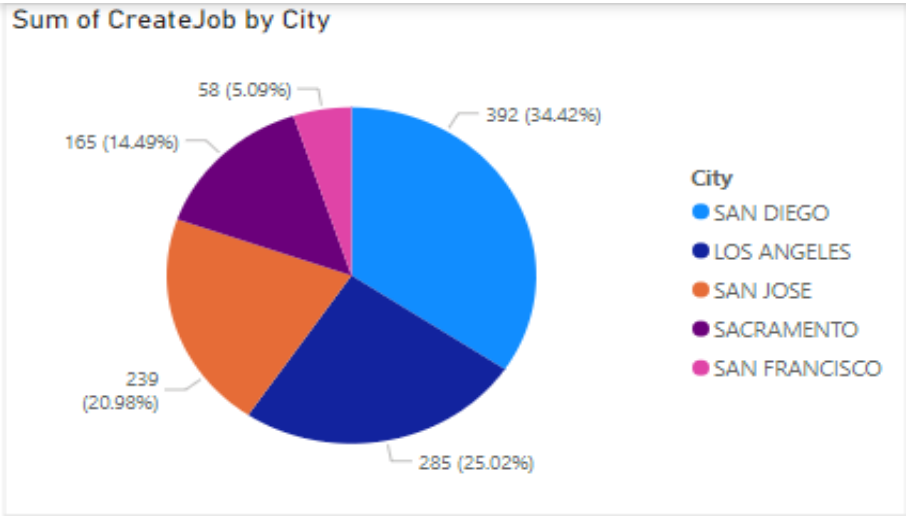
Interpretation:

1. Cond. no. is 1 means there is weak collinearity.
2. Value of Durbin-watson is less than 1.5 means there is presence of positive autocorrelation.
3. Jarque-Bera test value is far from zero, this indicates that data do not have a normal distribution.

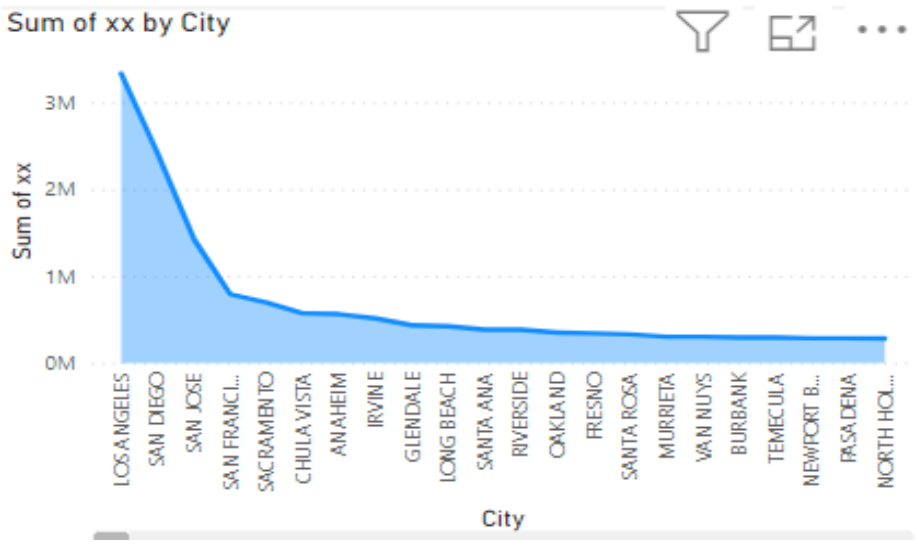
2.Graphical Representation Using Power BI:



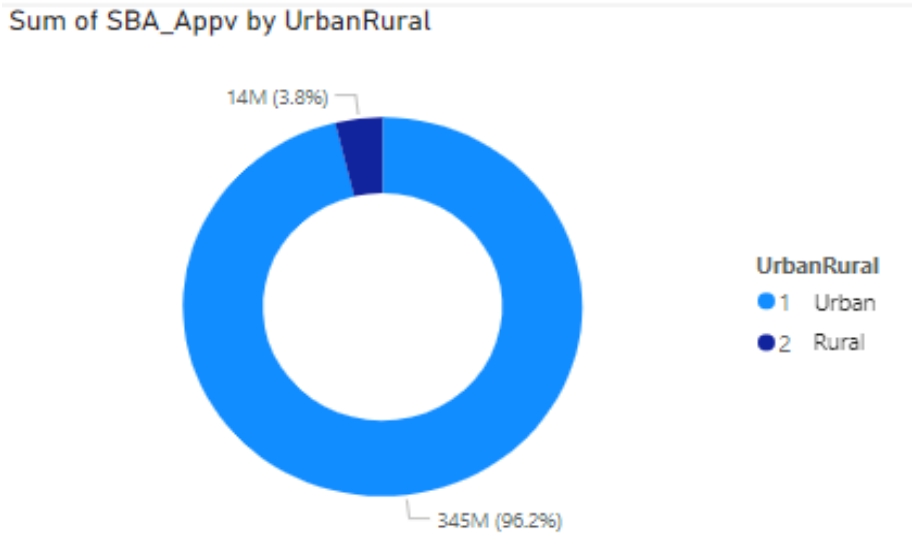
Interpretation: From this graph we conclude that among all the banks listed BANK of AMERICA NAIL ASSOC has greater no of defaulter around 189.



Interpretation: The Greater number of Job created is observe in the SAN DIEGO City by obtain the loans from SBA.



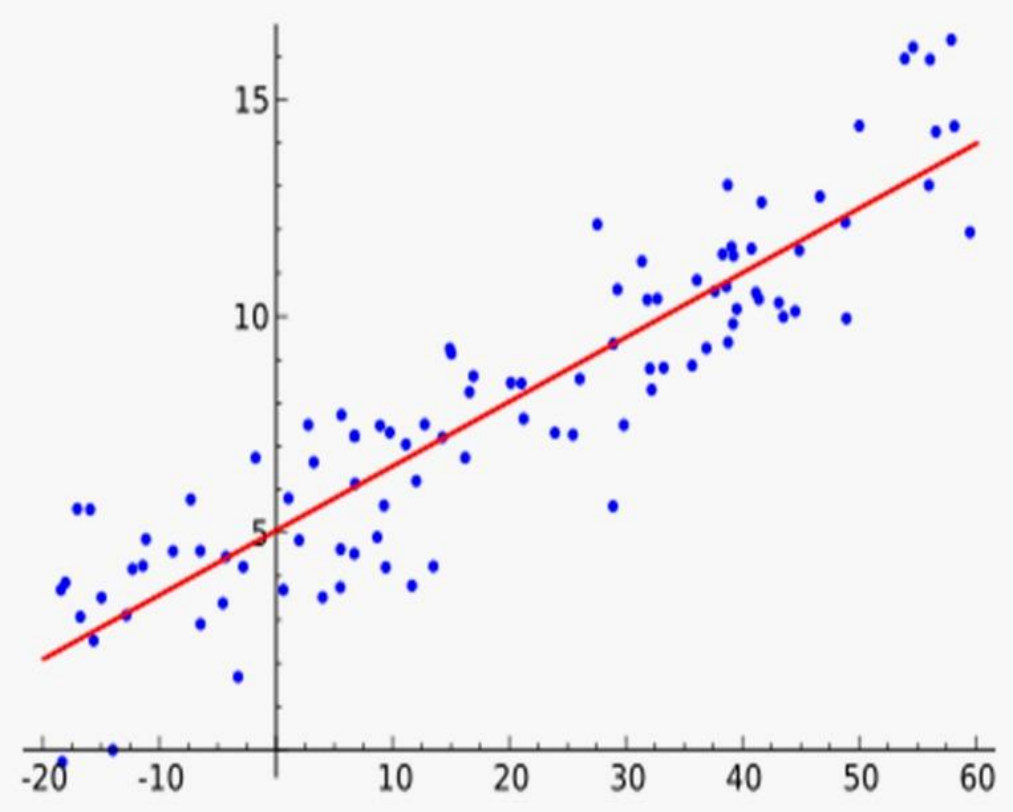
Interpretation: LOS ANGELES has a large amount of default around $33,35,894 among all the city .



Interpretation: SBA's guaranteed amount of approved loan has greater proportion in Urban area than the Rural area.

3. Multiple Linear Regression Model

- Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.
- Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. MLR tries to fit a regression line through a multidimensional space of data-points.
- For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.



Assumptions for Multiple Linear Regression:

- A linear relationship should exist between the Target and predictor variables.
- The regression residuals must be normally distributed.
- MLR assumes little or no multicollinearity (correlation between the independent variable) in data.

Multiple linear Regression Equation:

In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables x1, x2, x3, ...,xn. Since it is an enhancement of Simple Linear Regression, so the same is applied for the multiple linear regression equation, the equation becomes:

Where,

Y = Output/Response variable
b0, b1, b2, b3 , bn.... = Coefficients of the model.
x1, x2, x3, x4,.. = Various Independent/feature variable

For example,

Real estate example:

You're a real estate employee who wants to create a model to help predict the best time to sell homes. You hope to sell homes at the maximum sales price, but multiple factors can affect the sales price. These variables include the age of the house, the value of other homes in the neighbourhood, quantitative measurements of the public school system regarding student performance and the number of nearby parks, among other factors.

You can build a prediction model off these four independent variables to predict the maximum sales price of homes. You can adjust the variables if any of these factors change in terms of their coefficient values.

Introduction to Data:

The dataset is about the Commodity Stock Prices of the United States. The aim of this study is to analyze and examine the dataset of major commodity prices to get the better knowledge of factors that affects the prices of commodities over the years.

The commodities that are listed above are GOLD, PALLADIUM, NICKEL, BRENT OIL, NATURAL GAS, WHEAT as these major commodities affect the markets and economies of the countries around the globe. So, the price analysis is important for predicting the future price and volume of commodity that is available.

The analysis is carried by utilizing the data by looking at the prices of each commodities on each day of the year from 2000-2022. And the price closing and opening on each day with volume traded.

Analysis of the Data:

Importing the Dataset:

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
import warnings
from sklearn.linear_model import LinearRegression
warnings.filterwarnings("ignore")
data=pd.read_excel("commodity.xlsx")
data
```

	Symbol	Date	Open	High	Low	Close	Volume
0	Gold	2000-01-04	281.00	281.00	281.00	282.70	4
1	Gold	2000-01-05	283.20	283.20	283.20	281.10	16
2	Gold	2000-01-06	281.40	281.40	281.40	281.40	0
3	Gold	2000-01-07	281.90	281.90	281.90	281.90	0
4	Gold	2000-01-10	281.70	281.70	281.70	281.70	0
...	...	...	...	...	...	...	...
29930	US Wheat	2022-04-04	989.50	1014.50	984.75	1010.25	0
29931	US Wheat	2022-04-05	1020.50	1074.00	1020.00	1045.25	0
29932	US Wheat	2022-04-06	1049.40	1056.40	1024.25	1041.12	0
29933	US Wheat	2022-04-07	1040.25	1044.50	1017.75	1020.00	0
29934	US Wheat	2022-04-08	1020.00	1054.25	1017.00	1051.50	0

29935 rows × 7 columns

Looking for rows and columns:

```
data.shape
(29935, 7)
```

Looking for duplicate and null values:

```
data.duplicated().sum()
```

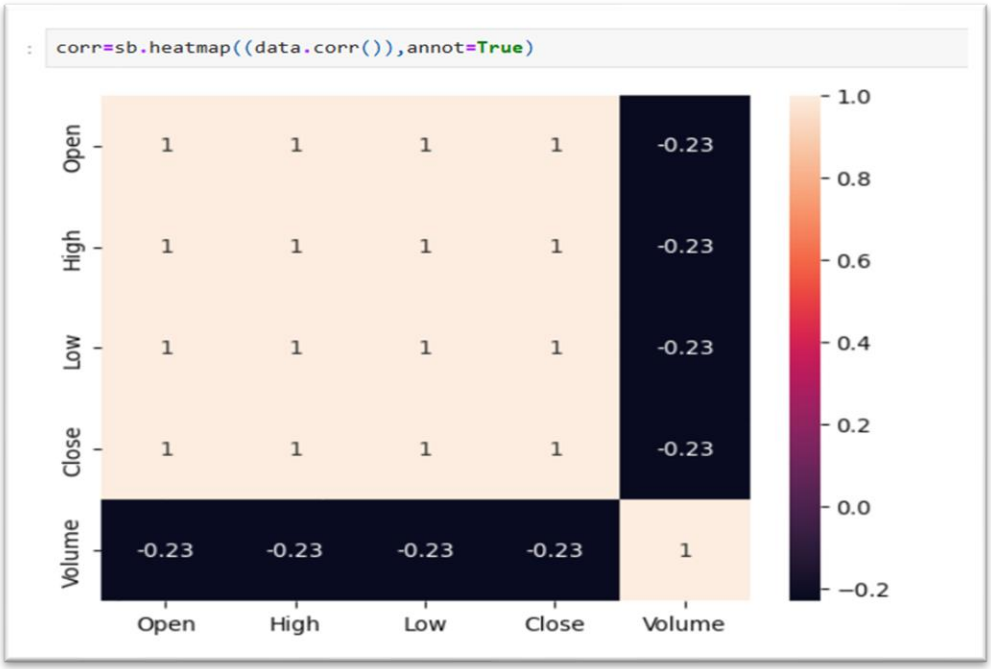
```
0
```

```
data.isnull().sum()
```

```
Symbol    0
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

Correlation Matrix:

```
corr=sb.heatmap((data.corr()),annot=True)
```



Interpretation: From Correlation matrix we can see that, there is high correlation between our Dependent variable Close Price and Independent variable High Price, Low Price, Open Price, and Volume.

Fitting Multiple Linear Regression Model:

```
X=data[['High','Low','Volume','Open']]
Y=data['Close']
```

```
Reg=linear_model.LinearRegression()
Reg.fit(X,Y)
```

```
LinearRegression()
```

```
Reg.coef_                    ## for + ind increase dependent increase and for - independent incease deoendent decreases
```

```
array([ 5.75643652e-01,  6.47555781e-01, -1.12926346e-05, -2.23200828e-01])
```

```
Reg.intercept_               ## the mean for Y when all of the x takse value 0
```

```
2.3704579980303606
```

```
print("R square =",Reg.score(X,Y))     ## how well the regression model explains observed data
```

```
R square = 0.9998609980890555
```

Interpretation: The R-square value is 0.99 which means our model is overfit.

Train Test for model accuracy:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
print("X_train")
```

```
X_train
```

```
#View shapes of splitted data
print("X_train:",X_train.shape)
print("X_test:",X_test.shape)
print("Y_train:",Y_train.shape)
print("Y_test:",Y_test.shape)
```

```
X_train: (23948, 4)
X_test: (5987, 4)
Y_train: (23948,)
Y_test: (5987,)
```

```
Reg = LinearRegression()
Reg.fit(X_train,Y_train)
```

```
LinearRegression()
```

```
Y_pred=Reg.predict(X_test)
Y_pred
```

```
array([  68.30574842,    4.57625881,    3.8431176 , ..., 1766.79653495,
        921.83457802, 1226.4095991 ])
```

```
Accuracy=r2_score(Y_test,Y_pred)*100
print("Accuracy of the model is %.2f" %Accuracy)
```

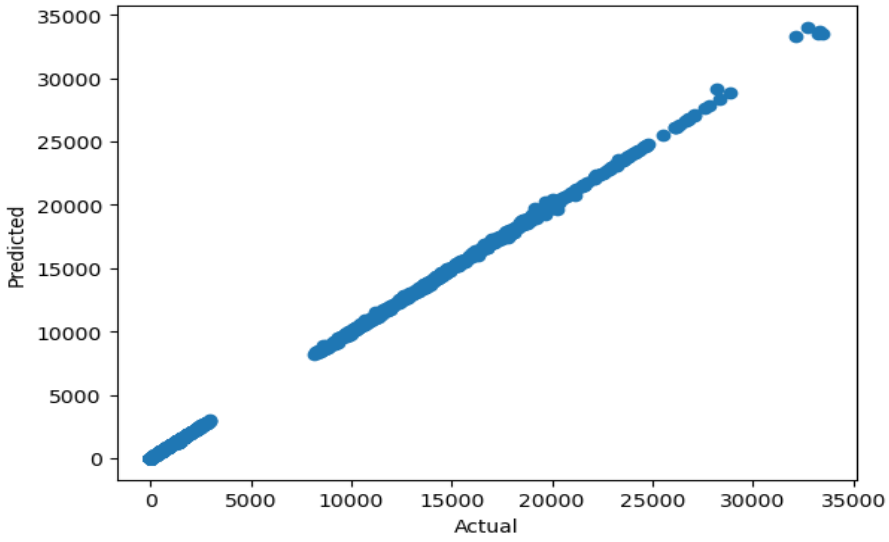
```
Accuracy of the model is 99.99
```

Interpretation: The Accuracy of the Model is 99.99, which means our actual and predicted values are 99.99% accurate.

Graphical Representation of Actual versus Predicted value:

```
plt.scatter(Y_test,Y_pred)
plt.xlabel("Actual")
plt.ylabel("Predicted")
```

```
Text(0, 0.5, 'Predicted')
```

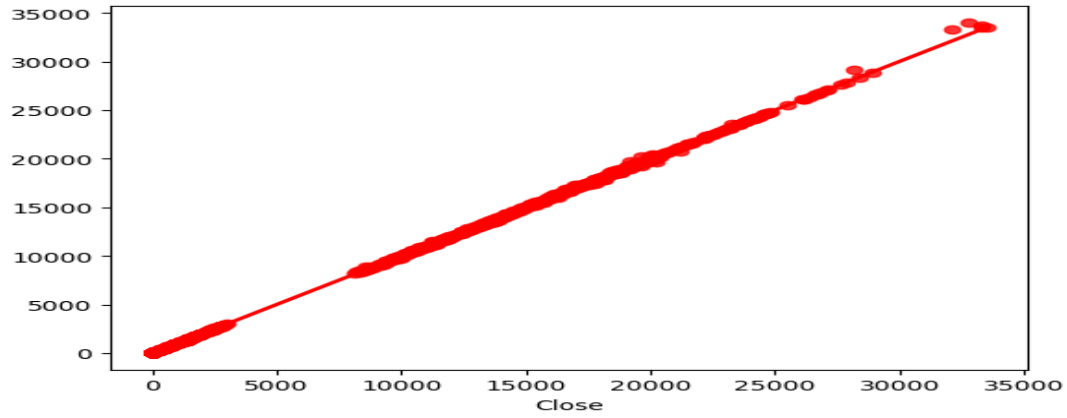


Interpretation: There are very few outliers in the model. Almost all the values fall on the line of best fit.

Regression Plot for the dependent variable Close Price:

```
sb.regplot(x=Y_test,y=Y_pred,ci=None,color='red')
```

```
<AxesSubplot:xlabel='Close'>
```



Interpretation: There are very few outliers in the model. Almost all the values fall on the line of best fit.

OLS Model:

Theory:

Ordinary Least Squares (OLS) method is widely used to estimate the parameter of a linear regression model. The goal is minimizing the differences between the collected observations in some arbitrary dataset and the responses predicted by the linear approximation of the data.

OLS Regression Results

Dep. Variable:	Close	R-squared (uncentered):	1.000
Model:	OLS	Adj. R-squared (uncentered):	1.000
Method:	Least Squares	F-statistic:	6.397e+07
Date:	Sat, 29 Apr 2023	Prob (F-statistic):	0.00
Time:	11:21:37	Log-Likelihood:	-1.6485e+05
No. Observations:	29935	AIC:	3.297e+05
Df Residuals:	29931	BIC:	3.297e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
High	0.5767	0.005	109.245	0.000	0.566	0.587
Low	0.6459	0.006	109.772	0.000	0.634	0.657
Volume	1.335e-06	3.51e-06	0.380	0.704	-5.55e-06	8.22e-06
Open	-0.2224	0.008	-27.355	0.000	-0.238	-0.207

Omnibus:	75704.857	Durbin-Watson:	2.177
Prob(Omnibus):	0.000	Jarque-Bera (JB):	17738316869.303
Skew:	26.917	Prob(JB):	0.00
Kurtosis:	3773.752	Cond. No.	2.84e+03

Interpretation:
1. The condition number is large, 2.84e+03. This might indicate that there are strong multicollinearity or other numerical problems.
2. Durbin-Watson value is greater than 2 this indicates that the negative autocorrelation.