Name: Akanksha Dewangan **Roll No. : MT19049**

Assignment -3

1.Running code is uploaded, where each line is commented for understanding.
2.Analyse the dataset :
-Dataset
-It is mixed up of categorical(object), integer and floating point values.

data informtaion

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3998 entries, 0 to 3997
Data columns (total 34 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     3998 non-null   int64
 1   Gender                 3998 non-null   object
 2   DOB                    3998 non-null   datetime64[ns]
 3   10percentage           3998 non-null   float64
 4   10board                3998 non-null   object
 5   12graduation           3998 non-null   int64
 6   12percentage           3998 non-null   float64
 7   12board                3998 non-null   object
 8   CollegeID              3998 non-null   int64
 9   CollegeTier            3998 non-null   int64
 10  Degree                 3998 non-null   object
 11  Specialization         3998 non-null   object
 12  collegeGPA             3998 non-null   float64
 13  CollegeCityID          3998 non-null   int64
 14  CollegeCityTier        3998 non-null   int64
 15  CollegeState           3998 non-null   object
 16  GraduationYear         3998 non-null   int64
 17  English                3998 non-null   int64
 18  Logical                3998 non-null   int64
 19  Quant                  3998 non-null   int64
 20  Domain                 3998 non-null   float64
 21  ComputerProgramming    3998 non-null   int64
```

```
df.head(4)
```

	ID	Gender	DOB	10percentage	10board	12graduation	12percentage	12board	CollegeID	CollegeTier	Degree	Specialization	cc
0	203097	f	1990-02-19	84.3	board ofsecondary education,ap	2007	95.8	board of intermediate education,ap	1141	2	B.Tech/B.E.	computer engineering	
1	579905	m	1989-10-04	85.4	cbse	2007	85.0	cbse	5807	2	B.Tech/B.E.	electronics and communication engineering	
2	810601	f	1992-08-03	85.0	cbse	2010	68.2	cbse	64	2	B.Tech/B.E.	information technology	
3	267447	m	1989-12-05	85.6	cbse	2007	83.6	cbse	6920	1	B.Tech/B.E.	computer engineering	

-shape and size:

Dataset having 3998 rows and 34 columns included target column that is High-Salary.

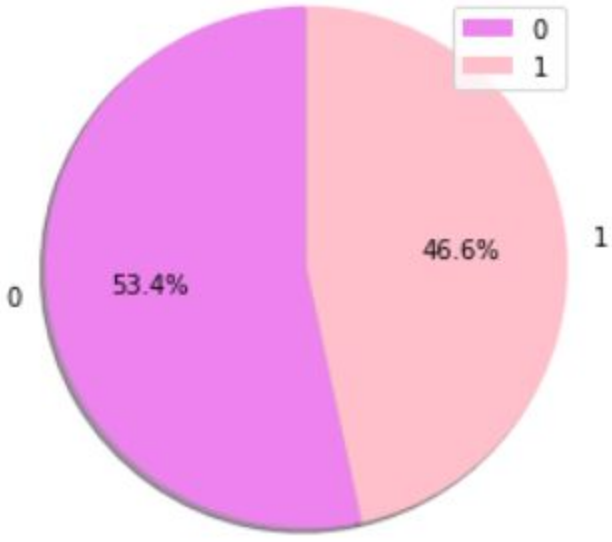
```
print("shape of data: ",df.shape)
```

```
shape of data:  (3998, 34)
```

-Checking class imbalance:

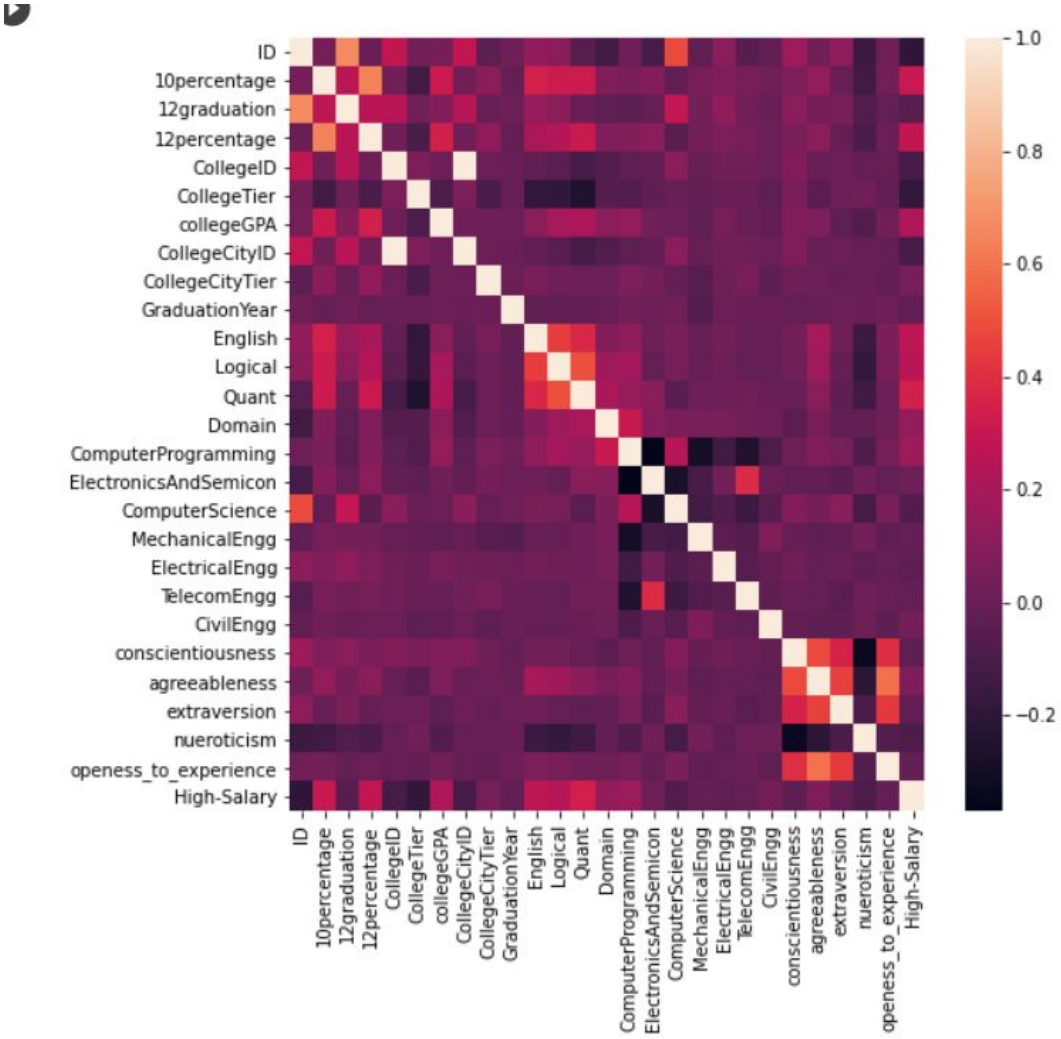
Here both classes 0 and 1 denotes non-high, high salary respectively are in balanced ration of 53:46 so their will be no biasing toward any of the class while training the model. Hence class is balanced.

```
1    2135
0    1863
Name: High-Salary, dtype: int64
```



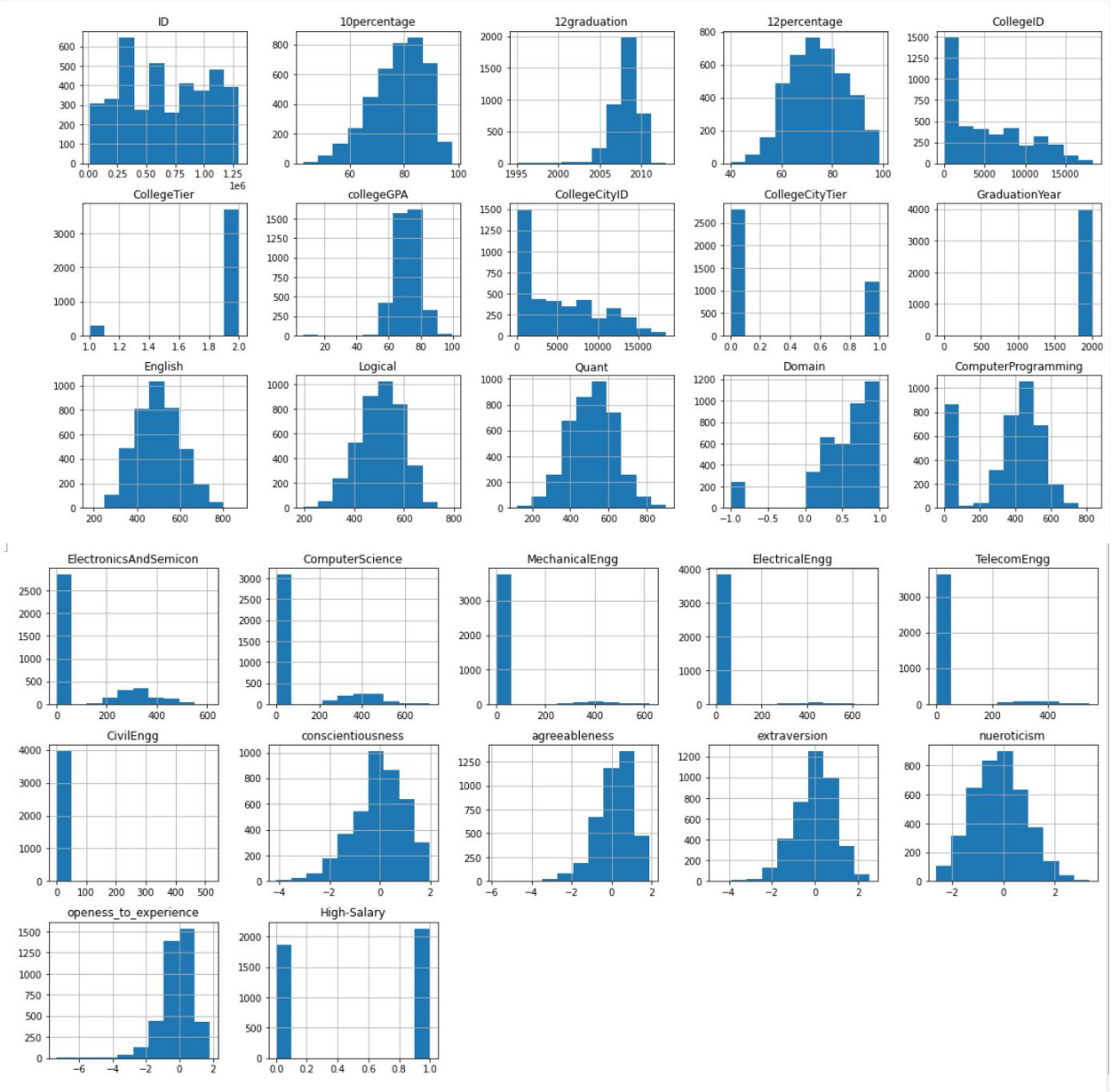
-Checking correlation between the columns:

Here we check whether is their any columns which are similar to each other so we can drop if two column are same. But in below heatmap no one are correlated to each other as no one is having light color map which conclude that no two columns are correlate to drop, not even they relate to 90%



-Histogram plot for every attribute.

A histogram divides the values within a numerical variable into “bins”, and counts the number of observations that fall into each bin. By visualizing these binned counts in a columnar fashion, we can obtain a very immediate and intuitive sense of the distribution of values within a variable.



3.Preprocessing:

-step1:

-Removing target column from the dataset and put it in one variable so can use furthur.

```
[1023] df_label=df['High-Salary']
       df=df.drop(['High-Salary'], axis=1)
```

```
[1024] print("Column names are:",list(df.columns))
       print("Total Columns currently are after removing target column: ",len(df.columns))
```

```
Column names are: ['ID', 'Gender', 'DOB', '10percentage', '10board', '12graduation', '12percentage', '12board', 'CollegeID', 'CollegeTier', 'Deg
Total Columns currently are after removing target column:  33
```

-step2:

-Encoding of catagory attributes.

Here I encoded all the catagorical columns into the number form by assigning numbers to the values as per they are uniquely present. Encoded columns are: Gender, 10borad, Degree, Specialization, CollegeState, 12board.

```
[1025] df['Gender']=df['Gender'].astype('category').cat.codes
       # df['DOB']=df['DOB'].astype('category')
       df['10board']=df['10board'].astype('category').cat.codes
       df['Degree']=df['Degree'].astype('category').cat.codes
       df['Specialization']=df['Specialization'].astype('category').cat.codes
       df['CollegeState']=df['CollegeState'].astype('category').cat.codes
       df['12board']=df['12board'].astype('category').cat.codes
```

-step 3:

-Here I tried to take only years of date of birth and then label encode the DOB column.:

```
[1026] df['DOB'] =[int(dt.split('-')[0]) for dt in df['DOB'].dt.strftime('%Y-%m-%d')]
```

```
[1027] df['DOB']=df['DOB'].astype('category').cat.codes
```

-step 4:

-These is Final data after above preprocessing. -But after applying them on logistic regression accuracy is not increasing thats why we are dropping few column like '10board','12board','DOB', becuase their presence doesnt affect the accuracy as much.

```
[1028] df.head()
```

	ID	Gender	DOB	10percentage	10board	12graduation	12percentage	12board	CollegeID	CollegeTier	Degree	Specialization	collegeGPA	Co
0	203097	0	10	84.3	43	2007	95.8	47	1141	2	0	10	78.00	
1	579905	1	9	85.4	60	2007	85.0	77	5807	2	0	21	70.06	
2	810601	0	12	85.0	60	2010	68.2	77	64	2	0	33	70.00	
3	267447	1	9	85.6	60	2007	83.6	77	6920	1	0	10	74.64	
4	343523	1	11	78.0	60	2008	76.8	77	11368	2	0	21	73.90	

-Drop '10board','12board','DOB' columns:

```
[1029] df=df.drop(['10board','12board','DOB'],axis=1)
```

-step5: Scaling with min_max scaling.

With Scaling

-Here I have done minmax sacling which transformed the values in between 0 to 1, and then futhur we can apply feature selection.

```
[1030] scaler = MinMaxScaler().fit(df)
       df=scaler.transform(df)
```

-step6: Feature selection

-select k best feature selection is applied with which 20 features are taken out.

+ Code + Text

```
[1031] df=SelectKBest(score_func=f_regression,k=20).fit_transform(df,df_label)
```

-step7: split into train_test data with observations with 60:40, 70:30 ratios.

60:40, 70:30

```
1032] X_train, X_test, y_train, y_test = train_test_split(df, df_label, test_size=0.4, random_state=10)
```

4.Model:

-I had applied logistic regression, and checked the perfect model by applying parameter tuning for that i used gridsearch cv so that i can get best parameters which will give me best accuracy.

-Also cross validation with 10 folds is used to train a model perfectly without overfitting or underfitting the model and also checking the cross validation score.

-Here for tunning the parameters of logistic regression, we used gridsearchcv

```
[1034] grid={"C":[0.5,1,1.5,2,10,10.5,11,11.5], "penalty":["l1","l2"]}
       logreg= LogisticRegression(random_state=10,solver='liblinear')
       logreg_cv=GridSearchCV(logreg,grid,cv=10)
       logreg_cv.fit(X_train,y_train)
       pre=logreg_cv.predict(X_test)
```

```
[1035] print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)

       tuned hpyerparameters :(best parameters)  {'C': 10, 'penalty': 'l1'}
```

OBSERVATIONS:

5.Normalised Accuracies

6.Confusion matrix.

A.WITH MINMAX SCALING OF DATA:

CASE 1:60:40 (test train split)

-Model parameters:

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
```

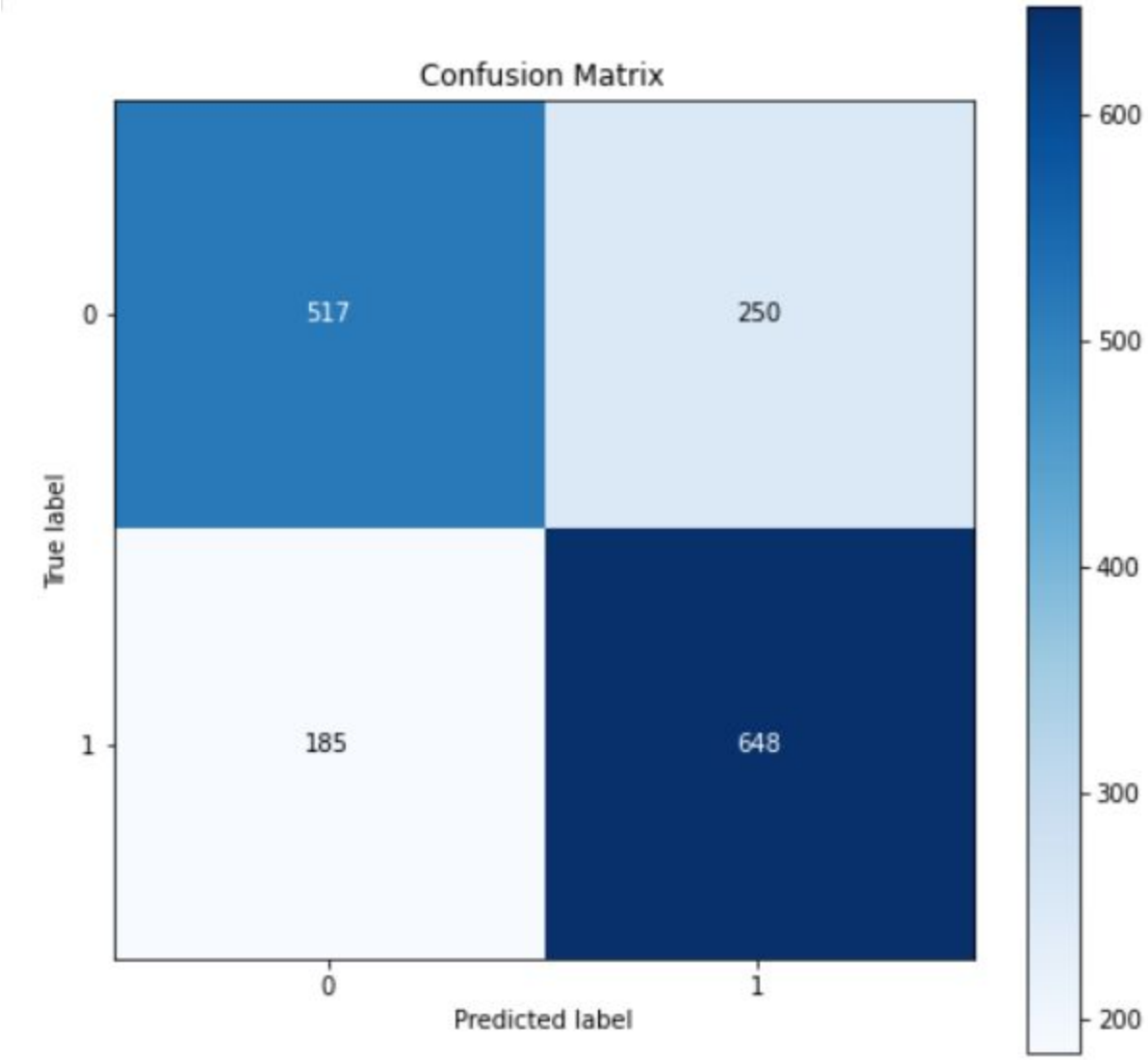
```
tuned hpyerparameters :(best parameters)  {'C': 10, 'penalty': 'l1'}
```

```
print("crossvalidation score : ",accuracy_score(y_train, logreg_cv.predict(X_train)))
```

```
crossvalidation score :  0.7231025854879066
```

```
print("Test data Accuracy: ",accuracy_score(y_test, pre))
```

```
Test data Accuracy:  0.728125
```



Class wise accuracy:

```
cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
print("Class 0 accurcay :",cm.diagonal()[0], ", Class  1 accurcay: ",cm.diagonal()[1])
```

```
Class 0 accurcay : 0.6740547588005215 , Class  1 accurcay:  0.7779111644657863
```

```
--------------analysis on test data------------
Accuracy:  0.728125
Sensitivity:  0.7779111644657863
Specificity: 0.6740547588005215
```

CASE 2:70:30 (test train split)

-Model parameters

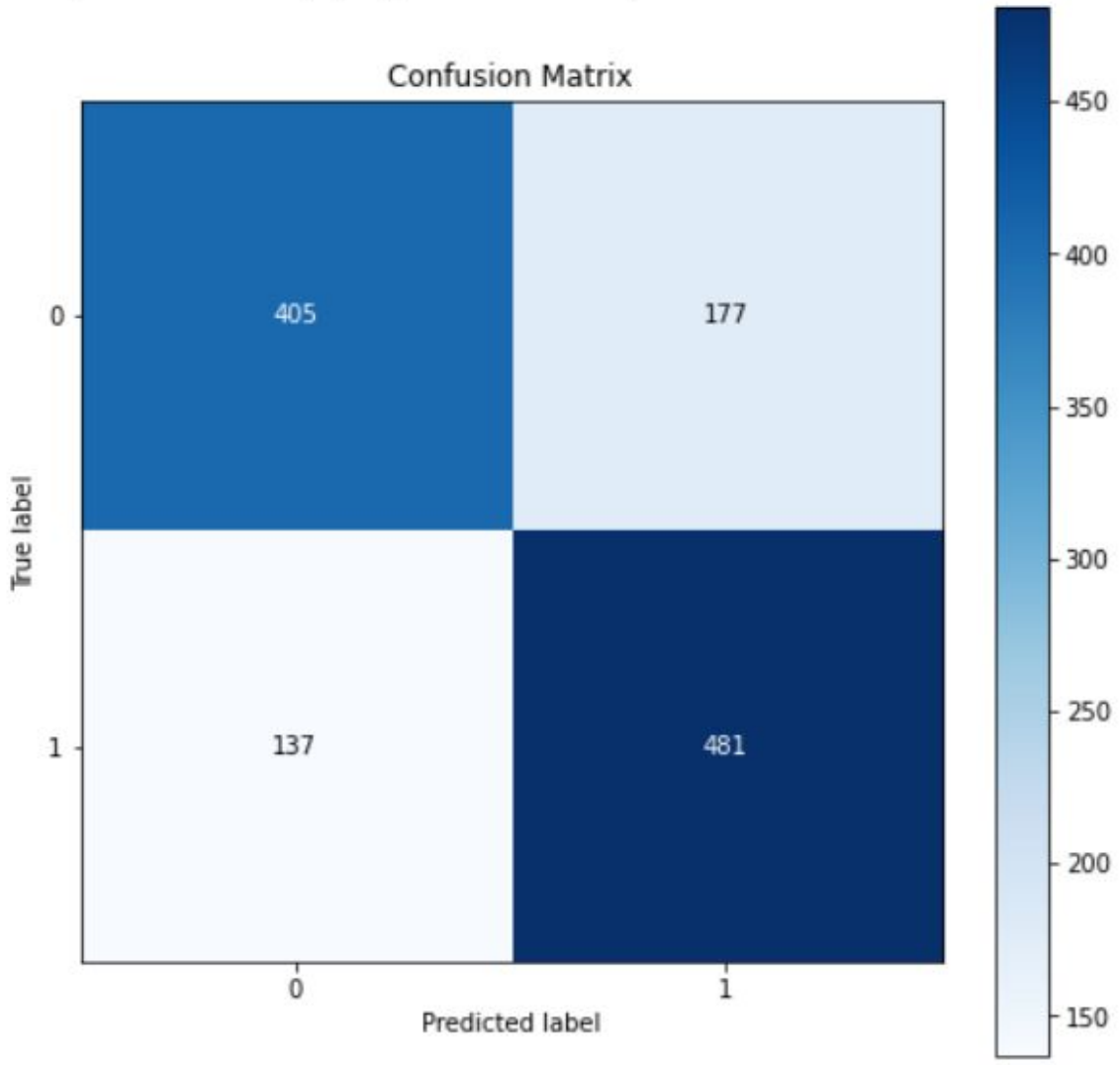
```
tuned hpyerparameters :(best parameters)  {'C': 1, 'penalty': 'l1'}
```

```
] print("crossvalidation score : ",accuracy_score(y_train, logreg_cv.predict(X_train)))
```

```
crossvalidation score :  0.718012866333095
```

```
] print("Test data Accuracy: ",accuracy_score(y_test, pre))
```

```
Test data Accuracy:  0.7383333333333333
```



Class wise accuracy:

```
[27]
     cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
     print("Class 0 accurcay :",cm.diagonal()[0], ", Class  1 accurcay: ",cm.diagonal()[1])
```

```
     Class 0 accurcay : 0.6958762886597938 , Class  1 accurcay:  0.7783171521035599
```

```
--------------analysis on test data------------
Accuracy:  0.7383333333333333
Sensitivity:  0.7783171521035599
Specificity: 0.6958762886597938
```

CASE 3:90:10 (test train split)
-Model parameters

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
```

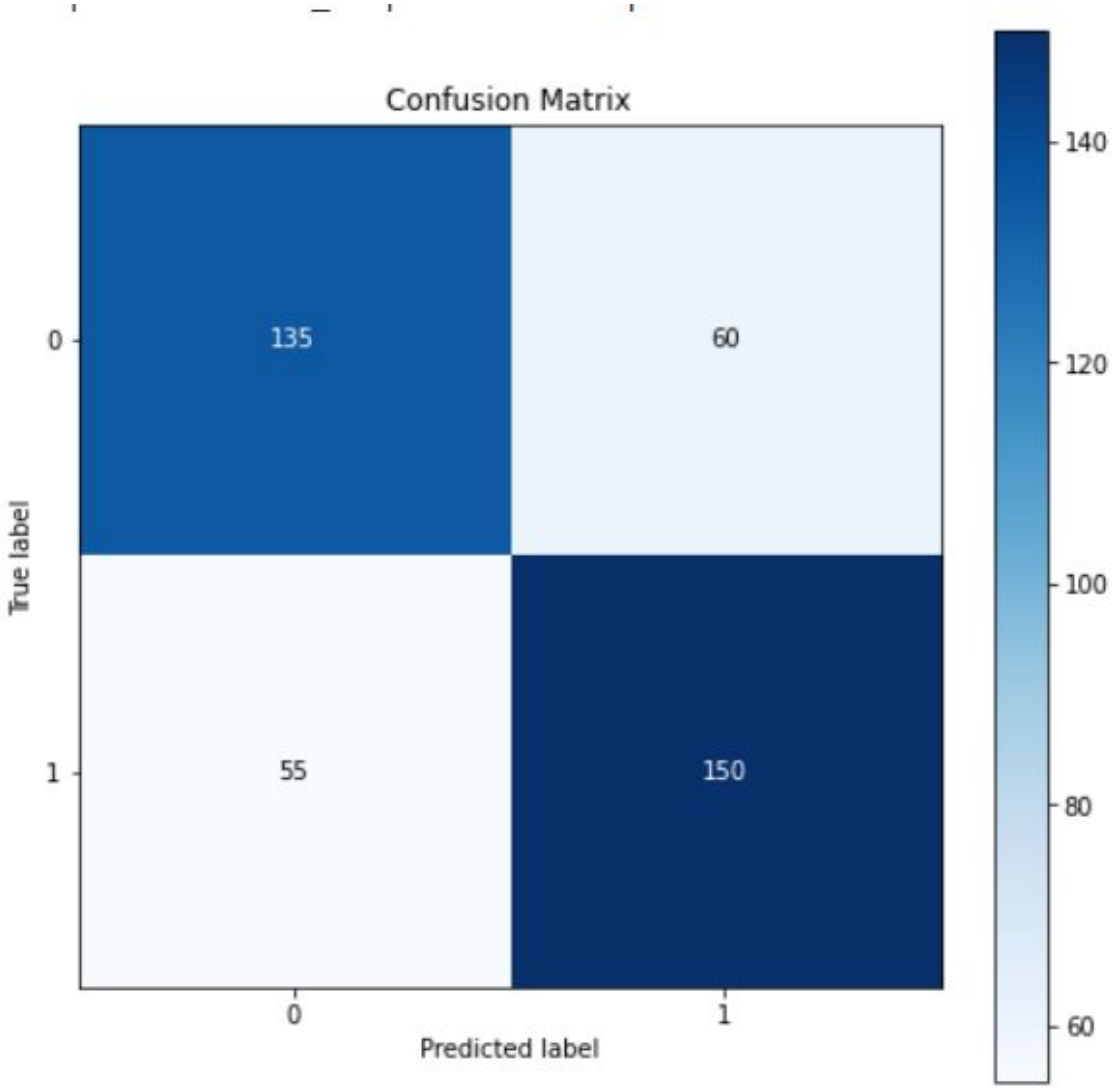
```
tuned hpyerparameters :(best parameters)  {'C': 10, 'penalty': 'l2'}
```

```
print("crossvalidation score : ",accuracy_score(y_train, logreg_cv.predict(X_train)))
```

```
crossvalidation score :  0.7281823235130628
```

```
print("Test data Accuracy: ",accuracy_score(y_test, pre))
```

```
Test data Accuracy:  0.7125
```



Class wise accuracy:

```
[27]
     cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
     print("Class 0 accurcay :",cm.diagonal()[0], ", Class  1 accurcay: ",cm.diagonal()[1])

     Class 0 accurcay : 0.6923076923076923 , Class  1 accurcay:  0.7317073170731707
```

```
--------------analysis on test data------------
Accuracy:  0.7125
Sensitivity:  0.7317073170731707
Specificity: 0.6923076923076923
```

B.WITHOUT MINMAX SCALING OF DATA:

CASE 1:60:40 (test train split)
-Model parameters

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
```

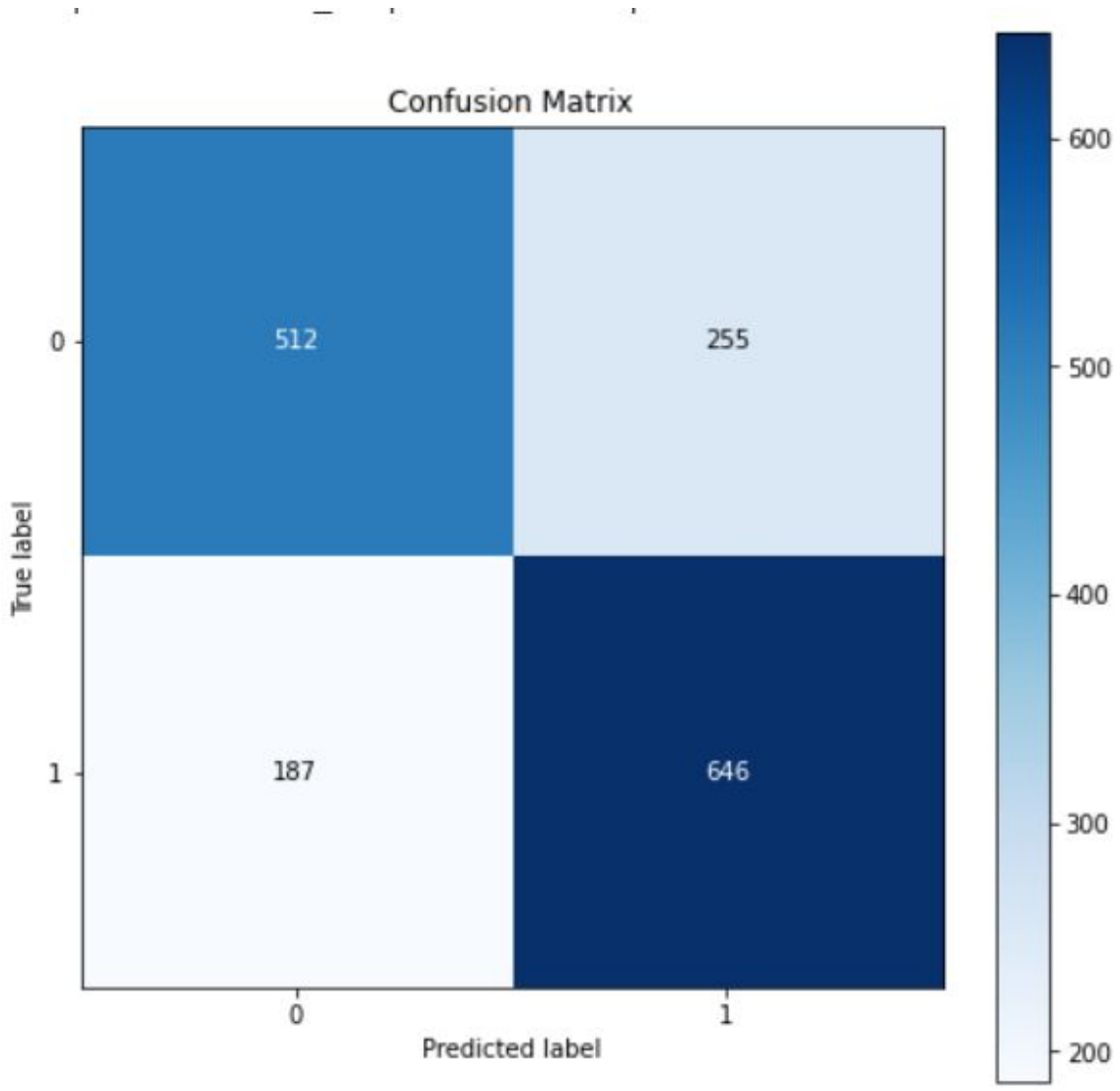
```
tuned hpyerparameters :(best parameters)  {'C': 11, 'penalty': 'l2'}
```

```
print("crossvalidation score : ",accuracy_score(y_train, logreg_cv.predict(X_train)))
```

```
crossvalidation score :  0.7164303586321935
```

```
print("Test data Accuracy: ",accuracy_score(y_test, pre))
```

```
Test data Accuracy:  0.72375
```




```
cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
print("Class 0 accurcay :",cm.diagonal()[0], ", Class  1 accurcay: ",cm.diagonal()[1])
```

```
Class 0 accurcay : 0.6675358539765319 , Class  1 accurcay:  0.7755102040816326
```

```
--------------analysis on test data------------
Accuracy:  0.72375
Sensitivity:  0.7755102040816326
Specificity: 0.6675358539765319
```

CASE 2:70:30 (test train split)

-Model parameters

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
```

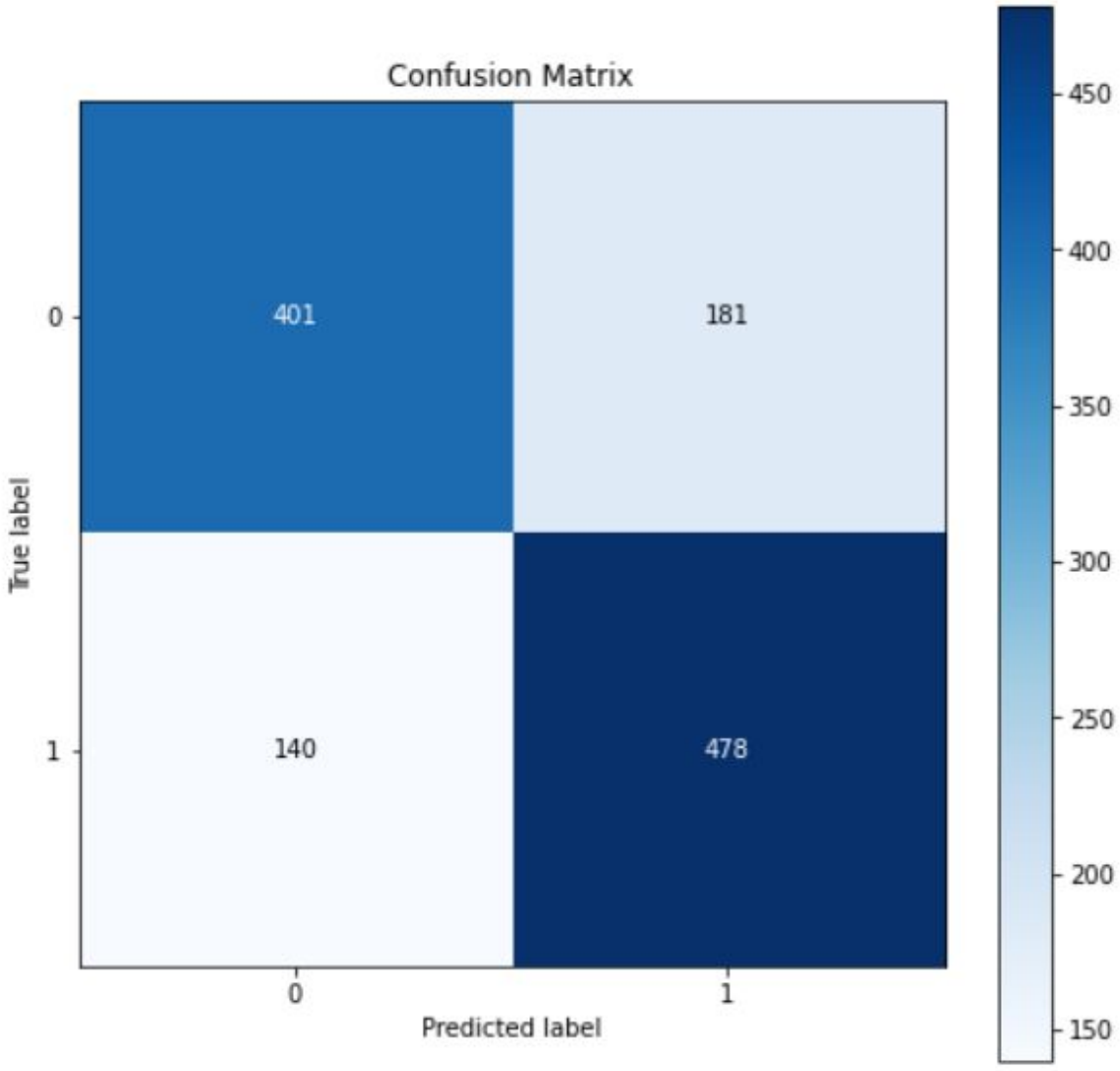
```
tuned hpyerparameters :(best parameters)  {'C': 0.5, 'penalty': 'l1'}
```

```
print("crossvalidation score : ",accuracy_score(y_train, logreg_cv.predict(X_train)))
```

```
crossvalidation score :  0.7190850607576841
```

```
print("Test data Accuracy: ",accuracy_score(y_test, pre))
```

```
Test data Accuracy:  0.7325
```



Class wise accuracy:

```
[27]
     cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
     print("Class 0 accurcay :",cm.diagonal()[0], ", Class  1 accurcay: ",cm.diagonal()[1])
```

```
     Class 0 accurcay : 0.6890034364261168 , Class  1 accurcay:  0.7734627831715211
```

```
--------------analysis on test data------------
Accuracy:  0.7325
Sensitivity:  0.7734627831715211
Specificity: 0.6890034364261168
```

CASE 3:90:10 (test train split)

-Model parameters

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
```

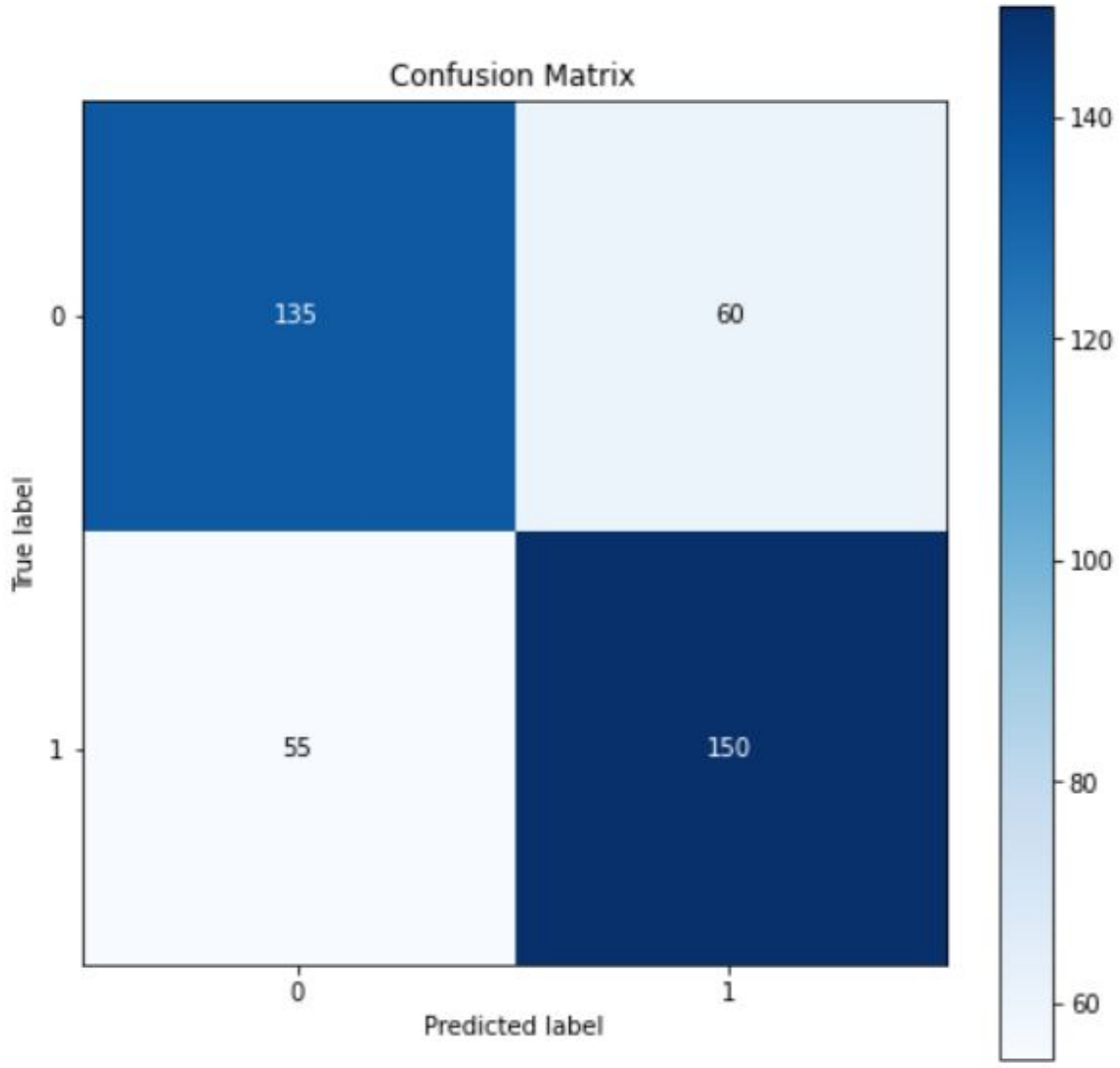
```
tuned hpyerparameters :(best parameters)  {'C': 0.5, 'penalty': 'l1'}
```

```
print("crossvalidation score : ",accuracy_score(y_train, logreg_cv.predict(X_train)))
```

```
crossvalidation score :  0.725958866036687
```

```
print("Test data Accuracy: ",accuracy_score(y_test, pre))
```

```
Test data Accuracy:  0.7125
```



Class wise accuracy:

```
[27]
     cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
     print("Class 0 accurcay :",cm.diagonal()[0], ", Class  1 accurcay: ",cm.diagonal()[1])
```

```
Class 0 accurcay : 0.6923076923076923 , Class  1 accurcay:  0.7317073170731707
```

```
--------------analysis on test data------------
Accuracy:  0.7125
Sensitivity:  0.7317073170731707
Specificity: 0.6923076923076923
```

References:

1.https://mode.com/example-gallery/python_histogram/