Python coding :-

```
        ##**************"Missing Values"*******************##

import os

import pandas as pd

import matplotlib as mlt

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import chi2_contingency


os.chdir("E:\EDWISOR\Data project1")


data = pd.read_csv("train.csv",sep = ',')


missing_val = pd.DataFrame(data.isnull().sum())


missing_val.head(5)


missing_val = missing_val.rename(columns ={0:'missing'})


missing_val = missing_val.sort_values('missing',ascending = False)
```

```
#NO Missing value found in the data

##**************** Feature Selection ********************##

f, ax = plt.subplots(figsize = (15,12))

corr = data.corr()

sns.heatmap(corr,mask=np.zeros_like(corr,
dtype=np.bool),cmap=sns.diverging_palette(220,10, as_cmap=True), square = True, ax=ax)

# NO dependency of variables on each other, can't omit any variable.

##**********************************************************************##

##**********************************************************************##

##***************** Decision Tree *********************##

import os
import pandas as pd

import matplotlib as mlt

import numpy as np


os.chdir("E:\EDWISOR\Data project1")


os.getcwd()


data = pd.read_csv("train.csv",sep = ',')


data.shape


data.head(9)


data['ID_code'] = data.ID_code.str.replace('train_','').astype(float)


data['Target'] = (data['target'])


data.drop(["target"],axis = 1, inplace = True)


data['Target'] = data['Target'].replace(1,'Yes')

data['Target'] = data['Target'].replace(0,'No')


import sklearn as sk
```

```
from sklearn.model_selection import train_test_split


x = data.values[:,0:201]

y = data.values[:,201]

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2)


from sklearn import tree


clf = tree.DecisionTreeClassifier(criterion = 'entropy').fit(x_train,y_train)


y_pred = clf.predict(x_test)


from sklearn.metrics import confusion_matrix

CM = confusion_matrix(y_test, y_pred)


CM = pd.crosstab(y_test,y_pred)


TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]
```

```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_pred)*100


=>  83.3725


#False negative rate

(FN*100)/(FN+TP)


=> 80.73982737361283




#Recall

(TP*100)/(TP+FN)


=> 18.726777309883577




##********************************************************************************##

##********************************************************************************##


                    ##**********KNN Model ****************##


from sklearn.neighbors import KNeighborsClassifier
```

```
KNN_Model = KNeighborsClassifier(n_neighbors = 1).fit(x_train,y_train)

KNN_Predictions = KNN_Model.predict(x_test)

CM = pd.crosstab(y_test,KNN_Predictions)

TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

((TP+TN)*100/(TP+TN+FP+FN))


#False negative rate
(FN*100)/(FN+TP)

##****************************************************************************##
##****************************************************************************##

        ##************** Logistic Regression **************##


Sample_Index = np.random.rand(len(data)) < .80
```

```
train = data[Sample_Index]
test = data[~Sample_Index]

train_cols = train.columns[1:201]

import statsmodels.api as sm

logit = sm.Logit(train['Target'], train[train_cols]).fit()

logit.summary()

test['Actual_prob'] = logit.predict(test[train_cols])

test['ActualVal'] = 1
test.loc[test.Actual_prob < 0.5, 'ActualVal'] = 0

test.loc[test.Actual_prob < 0.5, 'ActualVal']

CM = pd.crosstab(test['Target'], test['ActualVal'])
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

```
((TP+TN)*100/(TP+TN+FP+FN))


=> 91.4948969381629


#False negative rate

(FN*100)/(FN+TP)


=>72.65234765234766


#Recall

(TP*100)/(TP+FN)


27.347652347652346


          ##*****************************************##

           ##***********************************##


## Now we will work on the Test data which is being provided


test1 = pd.read_csv("test.csv",sep = ',')


test1['ID_code'] = test1.ID_code.str.replace('test_','').astype(float)
```

```
test_cols = test1.columns[1:201]

test1['prob'] = logit.predict(test1[test_cols])

test1['Target'] = 1

test1.loc[test1.prob < 0.5, 'Target'] = 0

test1.head(3)

test1['Target'].value_counts()
```

```
0    193845
1      6155
Name: Target, dtype: int64
```