In [1]:

```
# here we are importing the all necessary packages
from imblearn.under_sampling import RandomUnderSampler

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import os
import gc
import pandas as pd
import matplotlib.pyplot as plt
import time
import warnings
import numpy as np
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from scipy import stats
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from imblearn.under_sampling import RandomUnderSampler
import pickle
from sklearn.preprocessing import Normalizer
from sklearn.calibration import CalibratedClassifierCV
from scipy.sparse import hstack
from sklearn.preprocessing import OneHotEncoder
from sklearn import metrics
```

In [2]:

```
# reading the data using pandas
data=pd.read_csv('diabetic_data.csv')
print("Number of data points:",data.shape[0])
```

```
Number of data points: 101766
```

In [3]:

```
# splitting the data into train and test
train=data.sample(frac=0.67,random_state=200) #random state is a seed value
test=data.drop(train.index)
y_test=test['readmitted']
test=test.drop(columns=['readmitted'])
```

In [9]:

```
def function1(X_test):
    global train
    test=X_test
    # here for train and test fill the na with mean
    train=train.fillna(train.mean())
    test=test.fillna(test.mean())
    # drop encounter id
    train=train.drop(columns=['encounter_id'])
    test=test.drop(columns=['encounter_id'])
    condition = train['readmitted']=='<30'
    # convert the ouput to 1 and 0 based on condition
    train['readmitted'] = np.where(condition,1,0)
    y_train=train['readmitted']
    # using one hot encoding for categorical variables
    enc = OneHotEncoder(handle_unknown='ignore')
    enc.fit(train['race'].values.reshape(-1,1))
    train_race=enc.transform(train['race'].values.reshape(-1,1))
    test_race=enc.transform(test['race'].values.reshape(-1,1))
    enc.fit(train['gender'].values.reshape(-1,1))
    train_gender=enc.transform(train['gender'].values.reshape(-1,1))
    test_gender=enc.transform(test['gender'].values.reshape(-1,1))
    enc.fit(train['age'].values.reshape(-1,1))
    train_age=enc.transform(train['age'].values.reshape(-1,1))
    test_age=enc.transform(test['age'].values.reshape(-1,1))
    enc.fit(train['weight'].values.reshape(-1,1))
    train_weight=enc.transform(train['weight'].values.reshape(-1,1))
    test_weight=enc.transform(test['weight'].values.reshape(-1,1))
    enc.fit(train['payer_code'].values.reshape(-1,1))
    train_payer_code=enc.transform(train['payer_code'].values.reshape(-1,1))
    test_payer_code=enc.transform(test['payer_code'].values.reshape(-1,1))
    enc.fit(train['medical_specialty'].values.reshape(-1,1))
    train_medical_specialty=enc.transform(train['medical_specialty'].values.reshape(-1,1))
    test_medical_specialty=enc.transform(test['medical_specialty'].values.reshape(-1,1))
    enc.fit(train['diag_1'].values.reshape(-1,1))
    train_diag_1=enc.transform(train['diag_1'].values.reshape(-1,1))
    test_diag_1=enc.transform(test['diag_1'].values.reshape(-1,1))
    enc.fit(train['diag_2'].values.reshape(-1,1))
    train_diag_2=enc.transform(train['diag_2'].values.reshape(-1,1))
    test_diag_2=enc.transform(test['diag_2'].values.reshape(-1,1))
    enc.fit(train['diag_3'].values.reshape(-1,1))
    train_diag_3=enc.transform(train['diag_3'].values.reshape(-1,1))
    test_diag_3=enc.transform(test['diag_3'].values.reshape(-1,1))
    enc.fit(train['max_glu_serum'].values.reshape(-1,1))
    train_max_glu_serum=enc.transform(train['max_glu_serum'].values.reshape(-1,1))
    test_max_glu_serum=enc.transform(test['max_glu_serum'].values.reshape(-1,1))
    enc.fit(train['A1Cresult'].values.reshape(-1,1))
    train_A1Cresult=enc.transform(train['A1Cresult'].values.reshape(-1,1))
    test_A1Cresult=enc.transform(test['A1Cresult'].values.reshape(-1,1))
    enc.fit(train['metformin'].values.reshape(-1,1))
    train_metformin=enc.transform(train['metformin'].values.reshape(-1,1))
    test_metformin=enc.transform(test['metformin'].values.reshape(-1,1))
    enc.fit(train['metformin'].values.reshape(-1,1))
    train_metformin=enc.transform(train['metformin'].values.reshape(-1,1))
    test_metformin=enc.transform(test['metformin'].values.reshape(-1,1))
    enc.fit(train['repaglinide'].values.reshape(-1,1))
    train_repaglinide=enc.transform(train['repaglinide'].values.reshape(-1,1))
    test_repaglinide=enc.transform(test['repaglinide'].values.reshape(-1,1))
    enc.fit(train['nateglinide'].values.reshape(-1,1))
    train_nateglinide=enc.transform(train['nateglinide'].values.reshape(-1,1))
    test_nateglinide=enc.transform(test['nateglinide'].values.reshape(-1,1))
    enc.fit(train['chlorpropamide'].values.reshape(-1,1))
    train_chlorpropamide=enc.transform(train['chlorpropamide'].values.reshape(-1,1))
    test_chlorpropamide=enc.transform(test['chlorpropamide'].values.reshape(-1,1))
    enc.fit(train['glimepiride'].values.reshape(-1,1))
    train_glimepiride=enc.transform(train['glimepiride'].values.reshape(-1,1))
    test_glimepiride=enc.transform(test['glimepiride'].values.reshape(-1,1))
    enc.fit(train['acetohexamide'].values.reshape(-1,1))
    train_acetohexamide=enc.transform(train['acetohexamide'].values.reshape(-1,1))
    test_acetohexamide=enc.transform(test['acetohexamide'].values.reshape(-1,1))
    enc.fit(train['glipizide'].values.reshape(-1,1))
    train_glipizide=enc.transform(train['glipizide'].values.reshape(-1,1))
    test_glipizide=enc.transform(test['glipizide'].values.reshape(-1,1))
    enc.fit(train['glyburide'].values.reshape(-1,1))
    train_glyburide=enc.transform(train['glyburide'].values.reshape(-1,1))
    test_glyburide=enc.transform(test['glyburide'].values.reshape(-1,1))
    enc.fit(train['tolbutamide'].values.reshape(-1,1))
    train_tolbutamide=enc.transform(train['tolbutamide'].values.reshape(-1,1))
    test_tolbutamide=enc.transform(test['tolbutamide'].values.reshape(-1,1))
    enc.fit(train['pioglitazone'].values.reshape(-1,1))
    train_pioglitazone=enc.transform(train['pioglitazone'].values.reshape(-1,1))
    test_pioglitazone=enc.transform(test['pioglitazone'].values.reshape(-1,1))
    enc.fit(train['rosiglitazone'].values.reshape(-1,1))
    train_rosiglitazone=enc.transform(train['rosiglitazone'].values.reshape(-1,1))
    test_rosiglitazone=enc.transform(test['rosiglitazone'].values.reshape(-1,1))
    enc.fit(train['acarbose'].values.reshape(-1,1))
    train_acarbose=enc.transform(train['acarbose'].values.reshape(-1,1))
    test_acarbose=enc.transform(test['acarbose'].values.reshape(-1,1))
    enc.fit(train['miglitol'].values.reshape(-1,1))
    train_miglitol=enc.transform(train['miglitol'].values.reshape(-1,1))
    test_miglitol=enc.transform(test['miglitol'].values.reshape(-1,1))
    enc.fit(train['troglitazone'].values.reshape(-1,1))
    train_troglitazone=enc.transform(train['troglitazone'].values.reshape(-1,1))
    test_troglitazone=enc.transform(test['troglitazone'].values.reshape(-1,1))
    enc.fit(train['tolazamide'].values.reshape(-1,1))
    train_tolazamide=enc.transform(train['tolazamide'].values.reshape(-1,1))
    test_tolazamide=enc.transform(test['tolazamide'].values.reshape(-1,1))
    enc.fit(train['examide'].values.reshape(-1,1))
    train_examide=enc.transform(train['examide'].values.reshape(-1,1))
    test_examide=enc.transform(test['examide'].values.reshape(-1,1))
    enc.fit(train['citoglipton'].values.reshape(-1,1))
    train_citoglipton=enc.transform(train['citoglipton'].values.reshape(-1,1))
    test_citoglipton=enc.transform(test['citoglipton'].values.reshape(-1,1))
    enc.fit(train['insulin'].values.reshape(-1,1))
    train_insulin=enc.transform(train['insulin'].values.reshape(-1,1))
    test_insulin=enc.transform(test['insulin'].values.reshape(-1,1))
    enc.fit(train['glyburide-metformin'].values.reshape(-1,1))
    train_glyburide_metformin=enc.transform(train['glyburide-metformin'].values.reshape(-1,
    test_glyburide_metformin=enc.transform(test['glyburide-metformin'].values.reshape(-1,1)
    enc.fit(train['glipizide-metformin'].values.reshape(-1,1))
    train_glipizide_metformin=enc.transform(train['glipizide-metformin'].values.reshape(-1,
    test_glipizide_metformin=enc.transform(test['glipizide-metformin'].values.reshape(-1,1)
    enc.fit(train['metformin-rosiglitazone'].values.reshape(-1,1))
    train_metformin_rosiglitazone=enc.transform(train['metformin-rosiglitazone'].values.res
    test_metformin_rosiglitazone=enc.transform(test['metformin-rosiglitazone'].values.resha
    enc.fit(train['glimepiride-pioglitazone'].values.reshape(-1,1))
    train_glimepiride_pioglitazone=enc.transform(train['glimepiride-pioglitazone'].values.r
    test_glimepiride_pioglitazone=enc.transform(test['glimepiride-pioglitazone'].values.res
    enc.fit(train['metformin-rosiglitazone'].values.reshape(-1,1))
    train_metformin_rosiglitazone=enc.transform(train['metformin-rosiglitazone'].values.res
    test_metformin_rosiglitazone=enc.transform(test['metformin-rosiglitazone'].values.resha
    enc.fit(train['metformin-pioglitazone'].values.reshape(-1,1))
    train_metformin_pioglitazone=enc.transform(train['metformin-pioglitazone'].values.resha
    test_metformin_pioglitazone=enc.transform(test['metformin-pioglitazone'].values.reshape
    enc.fit(train['change'].values.reshape(-1,1))
    train_change=enc.transform(train['change'].values.reshape(-1,1))
    test_change=enc.transform(test['change'].values.reshape(-1,1))
    enc.fit(train['diabetesMed'].values.reshape(-1,1))
    train_diabetesMed=enc.transform(train['diabetesMed'].values.reshape(-1,1))
    test_diabetesMed=enc.transform(test['diabetesMed'].values.reshape(-1,1))
    enc.fit(train['admission_type_id'].values.reshape(-1,1))
    train_admission_type_id=enc.transform(train['admission_type_id'].values.reshape(-1,1))
    test_admission_type_id=enc.transform(test['admission_type_id'].values.reshape(-1,1))
    enc.fit(train['discharge_disposition_id'].values.reshape(-1,1))
    train_discharge_disposition_id=enc.transform(train['discharge_disposition_id'].values.r
    test_discharge_disposition_id=enc.transform(test['discharge_disposition_id'].values.res
    enc.fit(train['admission_source_id'].values.reshape(-1,1))
    train_admission_source_id=enc.transform(train['admission_source_id'].values.reshape(-1,
    test_admission_source_id=enc.transform(test['admission_source_id'].values.reshape(-1,1)
     # normalising the numerical data
    normalizer = Normalizer()
    normalizer.fit(train['patient_nbr'].values.reshape(1,-1))
    train_patient_nbr = normalizer.transform(train['patient_nbr'].values.reshape(1,-1))
    test_patient_nbr = normalizer.transform(test['patient_nbr'].values.reshape(1,-1))
    normalizer.fit(train['time_in_hospital'].values.reshape(1,-1))
    train_time_in_hospital = normalizer.transform(train['time_in_hospital'].values.reshape(
    test_time_in_hospital = normalizer.transform(test['time_in_hospital'].values.reshape(1,
    normalizer.fit(train['num_lab_procedures'].values.reshape(1,-1))
    train_num_lab_procedures = normalizer.transform(train['num_lab_procedures'].values.resh
    test_num_lab_procedures = normalizer.transform(test['num_lab_procedures'].values.reshap
    normalizer.fit(train['num_procedures'].values.reshape(1,-1))
    train_num_procedures = normalizer.transform(train['num_procedures'].values.reshape(1,-1
    test_num_procedures = normalizer.transform(test['num_procedures'].values.reshape(1,-1))
    normalizer.fit(train['num_medications'].values.reshape(1,-1))
    train_num_medications = normalizer.transform(train['num_medications'].values.reshape(1,
    test_num_medications = normalizer.transform(test['num_medications'].values.reshape(1,-1
    normalizer.fit(train['number_outpatient'].values.reshape(1,-1))
    train_number_outpatient = normalizer.transform(train['number_outpatient'].values.reshap
    test_number_outpatient = normalizer.transform(test['number_outpatient'].values.reshape(
    normalizer.fit(train['number_emergency'].values.reshape(1,-1))
    train_number_emergency = normalizer.transform(train['number_emergency'].values.reshape(
    test_number_emergency = normalizer.transform(test['number_emergency'].values.reshape(1,
    normalizer.fit(train['number_inpatient'].values.reshape(1,-1))
    train_number_inpatient = normalizer.transform(train['number_inpatient'].values.reshape(
    test_number_inpatient = normalizer.transform(test['number_inpatient'].values.reshape(1,
    normalizer.fit(train['number_diagnoses'].values.reshape(1,-1))
    train_number_diagnoses = normalizer.transform(train['number_diagnoses'].values.reshape(
    test_number_diagnoses = normalizer.transform(test['number_diagnoses'].values.reshape(1,
    train_patient_nbr =train_patient_nbr.reshape(-1,1)
    test_patient_nbr =test_patient_nbr.reshape(-1,1)
    train_time_in_hospital =train_time_in_hospital.reshape(-1,1)
    test_time_in_hospital =test_time_in_hospital.reshape(-1,1)
    train_num_lab_procedures = train_num_lab_procedures.reshape(-1,1)
    test_num_lab_procedures = test_num_lab_procedures.reshape(-1,1)
    train_num_procedures =train_num_procedures.reshape(-1,1)
    test_num_procedures = test_num_procedures.reshape(-1,1)
    train_num_medications = train_num_medications.reshape(-1,1)
    test_num_medications = test_num_medications.reshape(-1,1)
    train_number_outpatient =train_number_outpatient.reshape(-1,1)
    test_number_outpatient = test_number_outpatient.reshape(-1,1)
    train_number_emergency =train_number_emergency.reshape(-1,1)
    test_number_emergency =test_number_emergency.reshape(-1,1)
    train_number_inpatient =train_number_inpatient.reshape(-1,1)
    test_number_inpatient = test_number_inpatient.reshape(-1,1)
    train_number_diagnoses = train_number_diagnoses.reshape(-1,1)
    test_number_diagnoses =test_number_diagnoses.reshape(-1,1)
    X_train = hstack((train_race,train_gender,train_age,train_weight,train_payer_code,train
    X_test = hstack((test_race,test_gender,test_age,test_weight,test_payer_code,test_medica
        #  undersampling the train data
    under = RandomUnderSampler()
    X_train,y_train = under.fit_resample(X_train, y_train.ravel())
    X_train=X_train.toarray()
    X_test=X_test.toarray()
    scaler = StandardScaler()
    X_train=scaler.fit_transform(X_train)
    X_test=scaler.transform(X_test)
        # using  decision tree
    decision = DecisionTreeClassifier()
    param_grid = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 100, 500]}
    clf = GridSearchCV(decision, param_grid,scoring='roc_auc',cv=5,n_jobs=-1)
    clf.fit(X_train,y_train)
    decision =clf.best_estimator_
    sig_clf = CalibratedClassifierCV(decision, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    # using calibrated classifer to get the probability using decision tree
    predict_y_decision = sig_clf.predict_proba(X_test)
    # using random forest classifier
    rm = RandomForestClassifier()
    params={'n_estimators':[5,10,25,50,100,300,500],'n_estimators': [10, 25], 'max_features
     'max_depth': [10, 50,75,100, None], 'bootstrap': [True, False]}
    model_rf=GridSearchCV(rm,param_grid=params,cv=5,scoring='roc_auc',n_jobs=-1,verbose=1)
    model_rf.fit(X_train, y_train)
    rm =model_rf.best_estimator_
    sig_clf = CalibratedClassifierCV(rm, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    # used calibrated classifer to get the correct probability predicitons
    predict_y_rm = sig_clf.predict_proba(X_test)
    one,two=predict_y_decision,predict_y_rm
    z=[]
    # adding the probability of both the predictions to get the weighted average
    for i in range(len(one)):
        z.append([one[i][0]+two[i][0],one[i][1]+two[i][1]])
    predicted=[]
      # using argmax got the output and checked for the answer
    for i in range(len(z)):
        predicted.append(np.argmax(z[i]))
    return predicted
```

In [5]:

```
predicted=function1(test)
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   18.9s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:  3.3min finished
```

In [6]:

```
def function2(X_test,y_test):
    global train
    test=X_test
    # converting train and test output using condition as below
    test['readmitted']=y_test
    condition = test['readmitted']=='<30'
    test['readmitted'] = np.where(condition,1,0)
    y_test = test['readmitted'].values
    test=test.drop(columns=['readmitted'])
    y_train=train['readmitted']
    # fill na values based on the mean
    train=train.fillna(train.mean())
    test=test.fillna(test.mean())
    # using one hot encoding to get the categorical variables
    enc = OneHotEncoder(handle_unknown='ignore')
    enc.fit(train['race'].values.reshape(-1,1))
    train_race=enc.transform(train['race'].values.reshape(-1,1))
    test_race=enc.transform(test['race'].values.reshape(-1,1))
    enc.fit(train['gender'].values.reshape(-1,1))
    train_gender=enc.transform(train['gender'].values.reshape(-1,1))
    test_gender=enc.transform(test['gender'].values.reshape(-1,1))
    enc.fit(train['age'].values.reshape(-1,1))
    train_age=enc.transform(train['age'].values.reshape(-1,1))
    test_age=enc.transform(test['age'].values.reshape(-1,1))
    enc.fit(train['weight'].values.reshape(-1,1))
    train_weight=enc.transform(train['weight'].values.reshape(-1,1))
    test_weight=enc.transform(test['weight'].values.reshape(-1,1))
    enc.fit(train['payer_code'].values.reshape(-1,1))
    train_payer_code=enc.transform(train['payer_code'].values.reshape(-1,1))
    test_payer_code=enc.transform(test['payer_code'].values.reshape(-1,1))
    enc.fit(train['medical_specialty'].values.reshape(-1,1))
    train_medical_specialty=enc.transform(train['medical_specialty'].values.reshape(-1,1))
    test_medical_specialty=enc.transform(test['medical_specialty'].values.reshape(-1,1))
    enc.fit(train['diag_1'].values.reshape(-1,1))
    train_diag_1=enc.transform(train['diag_1'].values.reshape(-1,1))
    test_diag_1=enc.transform(test['diag_1'].values.reshape(-1,1))
    enc.fit(train['diag_2'].values.reshape(-1,1))
    train_diag_2=enc.transform(train['diag_2'].values.reshape(-1,1))
    test_diag_2=enc.transform(test['diag_2'].values.reshape(-1,1))
    enc.fit(train['diag_3'].values.reshape(-1,1))
    train_diag_3=enc.transform(train['diag_3'].values.reshape(-1,1))
    test_diag_3=enc.transform(test['diag_3'].values.reshape(-1,1))
    enc.fit(train['max_glu_serum'].values.reshape(-1,1))
    train_max_glu_serum=enc.transform(train['max_glu_serum'].values.reshape(-1,1))
    test_max_glu_serum=enc.transform(test['max_glu_serum'].values.reshape(-1,1))
    enc.fit(train['A1Cresult'].values.reshape(-1,1))
    train_A1Cresult=enc.transform(train['A1Cresult'].values.reshape(-1,1))
    test_A1Cresult=enc.transform(test['A1Cresult'].values.reshape(-1,1))
    enc.fit(train['metformin'].values.reshape(-1,1))
    train_metformin=enc.transform(train['metformin'].values.reshape(-1,1))
    test_metformin=enc.transform(test['metformin'].values.reshape(-1,1))
    enc.fit(train['metformin'].values.reshape(-1,1))
    train_metformin=enc.transform(train['metformin'].values.reshape(-1,1))
    test_metformin=enc.transform(test['metformin'].values.reshape(-1,1))
    enc.fit(train['repaglinide'].values.reshape(-1,1))
    train_repaglinide=enc.transform(train['repaglinide'].values.reshape(-1,1))
    test_repaglinide=enc.transform(test['repaglinide'].values.reshape(-1,1))
    enc.fit(train['nateglinide'].values.reshape(-1,1))
    train_nateglinide=enc.transform(train['nateglinide'].values.reshape(-1,1))
    test_nateglinide=enc.transform(test['nateglinide'].values.reshape(-1,1))
    enc.fit(train['chlorpropamide'].values.reshape(-1,1))
    train_chlorpropamide=enc.transform(train['chlorpropamide'].values.reshape(-1,1))
    test_chlorpropamide=enc.transform(test['chlorpropamide'].values.reshape(-1,1))
    enc.fit(train['glimepiride'].values.reshape(-1,1))
    train_glimepiride=enc.transform(train['glimepiride'].values.reshape(-1,1))
    test_glimepiride=enc.transform(test['glimepiride'].values.reshape(-1,1))
    enc.fit(train['acetohexamide'].values.reshape(-1,1))
    train_acetohexamide=enc.transform(train['acetohexamide'].values.reshape(-1,1))
    test_acetohexamide=enc.transform(test['acetohexamide'].values.reshape(-1,1))
    enc.fit(train['glipizide'].values.reshape(-1,1))
    train_glipizide=enc.transform(train['glipizide'].values.reshape(-1,1))
    test_glipizide=enc.transform(test['glipizide'].values.reshape(-1,1))
    enc.fit(train['glyburide'].values.reshape(-1,1))
    train_glyburide=enc.transform(train['glyburide'].values.reshape(-1,1))
    test_glyburide=enc.transform(test['glyburide'].values.reshape(-1,1))
    enc.fit(train['tolbutamide'].values.reshape(-1,1))
    train_tolbutamide=enc.transform(train['tolbutamide'].values.reshape(-1,1))
    test_tolbutamide=enc.transform(test['tolbutamide'].values.reshape(-1,1))
    enc.fit(train['pioglitazone'].values.reshape(-1,1))
    train_pioglitazone=enc.transform(train['pioglitazone'].values.reshape(-1,1))
    test_pioglitazone=enc.transform(test['pioglitazone'].values.reshape(-1,1))
    enc.fit(train['rosiglitazone'].values.reshape(-1,1))
    train_rosiglitazone=enc.transform(train['rosiglitazone'].values.reshape(-1,1))
    test_rosiglitazone=enc.transform(test['rosiglitazone'].values.reshape(-1,1))
    enc.fit(train['acarbose'].values.reshape(-1,1))
    train_acarbose=enc.transform(train['acarbose'].values.reshape(-1,1))
    test_acarbose=enc.transform(test['acarbose'].values.reshape(-1,1))
    enc.fit(train['miglitol'].values.reshape(-1,1))
    train_miglitol=enc.transform(train['miglitol'].values.reshape(-1,1))
    test_miglitol=enc.transform(test['miglitol'].values.reshape(-1,1))
    enc.fit(train['troglitazone'].values.reshape(-1,1))
    train_troglitazone=enc.transform(train['troglitazone'].values.reshape(-1,1))
    test_troglitazone=enc.transform(test['troglitazone'].values.reshape(-1,1))
    enc.fit(train['tolazamide'].values.reshape(-1,1))
    train_tolazamide=enc.transform(train['tolazamide'].values.reshape(-1,1))
    test_tolazamide=enc.transform(test['tolazamide'].values.reshape(-1,1))
    enc.fit(train['examide'].values.reshape(-1,1))
    train_examide=enc.transform(train['examide'].values.reshape(-1,1))
    test_examide=enc.transform(test['examide'].values.reshape(-1,1))
    enc.fit(train['citoglipton'].values.reshape(-1,1))
    train_citoglipton=enc.transform(train['citoglipton'].values.reshape(-1,1))
    test_citoglipton=enc.transform(test['citoglipton'].values.reshape(-1,1))
    enc.fit(train['insulin'].values.reshape(-1,1))
    train_insulin=enc.transform(train['insulin'].values.reshape(-1,1))
    test_insulin=enc.transform(test['insulin'].values.reshape(-1,1))
    enc.fit(train['glyburide-metformin'].values.reshape(-1,1))
    train_glyburide_metformin=enc.transform(train['glyburide-metformin'].values.reshape(-1,
    test_glyburide_metformin=enc.transform(test['glyburide-metformin'].values.reshape(-1,1)
    enc.fit(train['glipizide-metformin'].values.reshape(-1,1))
    train_glipizide_metformin=enc.transform(train['glipizide-metformin'].values.reshape(-1,
    test_glipizide_metformin=enc.transform(test['glipizide-metformin'].values.reshape(-1,1)
    enc.fit(train['metformin-rosiglitazone'].values.reshape(-1,1))
    train_metformin_rosiglitazone=enc.transform(train['metformin-rosiglitazone'].values.res
    test_metformin_rosiglitazone=enc.transform(test['metformin-rosiglitazone'].values.resha
    enc.fit(train['glimepiride-pioglitazone'].values.reshape(-1,1))
    train_glimepiride_pioglitazone=enc.transform(train['glimepiride-pioglitazone'].values.r
    test_glimepiride_pioglitazone=enc.transform(test['glimepiride-pioglitazone'].values.res
    enc.fit(train['metformin-rosiglitazone'].values.reshape(-1,1))
    train_metformin_rosiglitazone=enc.transform(train['metformin-rosiglitazone'].values.res
    test_metformin_rosiglitazone=enc.transform(test['metformin-rosiglitazone'].values.resha
    enc.fit(train['metformin-pioglitazone'].values.reshape(-1,1))
    train_metformin_pioglitazone=enc.transform(train['metformin-pioglitazone'].values.resha
    test_metformin_pioglitazone=enc.transform(test['metformin-pioglitazone'].values.reshape
    enc.fit(train['change'].values.reshape(-1,1))
    train_change=enc.transform(train['change'].values.reshape(-1,1))
    test_change=enc.transform(test['change'].values.reshape(-1,1))
    enc.fit(train['diabetesMed'].values.reshape(-1,1))
    train_diabetesMed=enc.transform(train['diabetesMed'].values.reshape(-1,1))
    test_diabetesMed=enc.transform(test['diabetesMed'].values.reshape(-1,1))
    enc.fit(train['admission_type_id'].values.reshape(-1,1))
    train_admission_type_id=enc.transform(train['admission_type_id'].values.reshape(-1,1))
    test_admission_type_id=enc.transform(test['admission_type_id'].values.reshape(-1,1))
    enc.fit(train['discharge_disposition_id'].values.reshape(-1,1))
    train_discharge_disposition_id=enc.transform(train['discharge_disposition_id'].values.r
    test_discharge_disposition_id=enc.transform(test['discharge_disposition_id'].values.res
    enc.fit(train['admission_source_id'].values.reshape(-1,1))
    train_admission_source_id=enc.transform(train['admission_source_id'].values.reshape(-1,
    test_admission_source_id=enc.transform(test['admission_source_id'].values.reshape(-1,1)
     # normalising the numerical data
    normalizer = Normalizer()
    normalizer.fit(train['patient_nbr'].values.reshape(1,-1))
    train_patient_nbr = normalizer.transform(train['patient_nbr'].values.reshape(1,-1))
    test_patient_nbr = normalizer.transform(test['patient_nbr'].values.reshape(1,-1))
    normalizer.fit(train['time_in_hospital'].values.reshape(1,-1))
    train_time_in_hospital = normalizer.transform(train['time_in_hospital'].values.reshape(
    test_time_in_hospital = normalizer.transform(test['time_in_hospital'].values.reshape(1,
    normalizer.fit(train['num_lab_procedures'].values.reshape(1,-1))
    train_num_lab_procedures = normalizer.transform(train['num_lab_procedures'].values.resh
    test_num_lab_procedures = normalizer.transform(test['num_lab_procedures'].values.reshap
    normalizer.fit(train['num_procedures'].values.reshape(1,-1))
    train_num_procedures = normalizer.transform(train['num_procedures'].values.reshape(1,-1
    test_num_procedures = normalizer.transform(test['num_procedures'].values.reshape(1,-1))
    normalizer.fit(train['num_medications'].values.reshape(1,-1))
    train_num_medications = normalizer.transform(train['num_medications'].values.reshape(1,
    test_num_medications = normalizer.transform(test['num_medications'].values.reshape(1,-1
    normalizer.fit(train['number_outpatient'].values.reshape(1,-1))
    train_number_outpatient = normalizer.transform(train['number_outpatient'].values.reshap
    test_number_outpatient = normalizer.transform(test['number_outpatient'].values.reshape(
    normalizer.fit(train['number_emergency'].values.reshape(1,-1))
    train_number_emergency = normalizer.transform(train['number_emergency'].values.reshape(
    test_number_emergency = normalizer.transform(test['number_emergency'].values.reshape(1,
    normalizer.fit(train['number_inpatient'].values.reshape(1,-1))
    train_number_inpatient = normalizer.transform(train['number_inpatient'].values.reshape(
    test_number_inpatient = normalizer.transform(test['number_inpatient'].values.reshape(1,
    normalizer.fit(train['number_diagnoses'].values.reshape(1,-1))
    train_number_diagnoses = normalizer.transform(train['number_diagnoses'].values.reshape(
    test_number_diagnoses = normalizer.transform(test['number_diagnoses'].values.reshape(1,
    train_patient_nbr =train_patient_nbr.reshape(-1,1)
    test_patient_nbr =test_patient_nbr.reshape(-1,1)
    train_time_in_hospital =train_time_in_hospital.reshape(-1,1)
    test_time_in_hospital =test_time_in_hospital.reshape(-1,1)
    train_num_lab_procedures = train_num_lab_procedures.reshape(-1,1)
    test_num_lab_procedures = test_num_lab_procedures.reshape(-1,1)
    train_num_procedures =train_num_procedures.reshape(-1,1)
    test_num_procedures = test_num_procedures.reshape(-1,1)
    train_num_medications = train_num_medications.reshape(-1,1)
    test_num_medications = test_num_medications.reshape(-1,1)
    train_number_outpatient =train_number_outpatient.reshape(-1,1)
    test_number_outpatient = test_number_outpatient.reshape(-1,1)
    train_number_emergency =train_number_emergency.reshape(-1,1)
    test_number_emergency =test_number_emergency.reshape(-1,1)
    train_number_inpatient =train_number_inpatient.reshape(-1,1)
    test_number_inpatient = test_number_inpatient.reshape(-1,1)
    train_number_diagnoses = train_number_diagnoses.reshape(-1,1)
    test_number_diagnoses =test_number_diagnoses.reshape(-1,1)
    X_train = hstack((train_race,train_gender,train_age,train_weight,train_payer_code,train
    X_test = hstack((test_race,test_gender,test_age,test_weight,test_payer_code,test_medica
        #  undersampling the train data
    under = RandomUnderSampler()
    X_train,y_train = under.fit_resample(X_train, y_train.ravel())
    X_train=X_train.toarray()
    X_test=X_test.toarray()
    scaler = StandardScaler()
    X_train=scaler.fit_transform(X_train)
    X_test=scaler.transform(X_test)
        # using decision tree
    decision = DecisionTreeClassifier()
    param_grid = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 100, 500]}
    clf = GridSearchCV(decision, param_grid,scoring='roc_auc',cv=5,n_jobs=-1)
    clf.fit(X_train,y_train)
    decision =clf.best_estimator_
    sig_clf = CalibratedClassifierCV(decision, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    # using calibrated classifer to get the probability using decision tree
    predict_y_decision = sig_clf.predict_proba(X_test)
    # using random forest classifier
    rm = RandomForestClassifier()
    params={'n_estimators':[5,10,25,50,100,300,500],'n_estimators': [10, 25], 'max_features
     'max_depth': [10, 50,75,100, None], 'bootstrap': [True, False]}
    model_rf=GridSearchCV(rm,param_grid=params,cv=5,scoring='roc_auc',n_jobs=-1,verbose=1)
    model_rf.fit(X_train, y_train)
    rm =model_rf.best_estimator_
    sig_clf = CalibratedClassifierCV(rm, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    # used calibrated classifer to get the correct probability predicitons
    predict_y_rm = sig_clf.predict_proba(X_test)
    one,two=predict_y_decision,predict_y_rm
    z=[]
    # added the probability of both the predictions
    for i in range(len(one)):
        z.append([one[i][0]+two[i][0],one[i][1]+two[i][1]])
    predicted=[]
      # using argmax got the output and checked for the answer
    for i in range(len(z)):
        predicted.append(np.argmax(z[i]))
    print("f1score is {0:.2f}".format(f1_score(y_test,predicted)))
```

In [7]:

```
function2(test,y_test)
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   15.7s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:  3.0min finished

f1score is 0.27
```