In [ ]:

```
import numpy as np
import pandas as pd
from time import time
from sklearn.metrics import f1_score
ben_data=pd.read_csv('/content/drive/MyDrive/Train_Beneficiarydata-1542865627584.csv')
inp_data=pd.read_csv('/content/drive/MyDrive/Train_Inpatientdata-1542865627584.csv')
out_data=pd.read_csv('/content/drive/MyDrive/Train_Outpatientdata-1542865627584.csv')
train_data=pd.read_csv('/content/drive/MyDrive/Train-1542865627584.csv')
ben_data_test=pd.read_csv('/content/drive/MyDrive/Test_Beneficiarydata-1542969243754.csv')
inp_data_test=pd.read_csv('/content/drive/MyDrive/Test_Inpatientdata-1542969243754.csv')
out_data_test=pd.read_csv('/content/drive/MyDrive/Test_Outpatientdata-1542969243754.csv')
test_data=pd.read_csv('/content/drive/MyDrive/Test-1542969243754.csv')
```

In [ ]:

```
import pickle
X=[ben_data_test,inp_data_test,out_data_test,test_data]
def predict(X):
  cols=list(set(X[1].columns).intersection(set(X[2].columns)))
  patient=X[1].merge(X[2],how='outer',on=cols)
  patient_int=patient.merge(X[0],how='left',on='BeneID')
  X[3]=pd.DataFrame({'Provider':X[3].values})
  patient_fin=X[3].merge(patient_int,how='inner',on='Provider')
  cols=['ChronicCond_Alzheimer','ChronicCond_Heartfailure','ChronicCond_KidneyDisease',
      'ChronicCond_Cancer','ChronicCond_ObstrPulmonary','ChronicCond_Depression','ChronicCond_Diabetes',
      'ChronicCond_IschemicHeart','ChronicCond_Osteoporasis','ChronicCond_rheumatoidarthritis',
      'ChronicCond_stroke']
  for i in cols:
    patient_fin[i]=patient_fin[i].map({1:0,2:1})
  patient_fin.RenalDiseaseIndicator=patient_fin.RenalDiseaseIndicator.map({'0':0,'Y':1})
  patient_fin['ClaimStartDt']=pd.to_datetime(patient_fin.ClaimStartDt,format='%Y-%m-%d')
  patient_fin['ClaimEndDt']=pd.to_datetime(patient_fin.ClaimEndDt,format='%Y-%m-%d')
  patient_fin['settlement_days']=patient_fin['ClaimEndDt']-patient_fin['ClaimStartDt']
  patient_fin['settlement_days']=patient_fin['settlement_days'].dt.days
  patient_fin['AdmissionDt']=pd.to_datetime(patient_fin['AdmissionDt'],format='%Y-%m-%d')
  patient_fin['DischargeDt']=pd.to_datetime(patient_fin['DischargeDt'],format='%Y-%m-%d')
  patient_fin['Days_Admit']=patient_fin['DischargeDt']-patient_fin['AdmissionDt']
  patient_fin['Days_Admit']=patient_fin['Days_Admit'].dt.days
  inp_out=patient_fin.Days_Admit.isnull()
  inp_data=inp_out.map({True:1,False:0})
  patient_fin['inp_out']=inp_data
  patient_fin.Days_Admit.fillna(0,inplace=True)
  last_death='2009-12-01'
  patient_fin['Alive_or_not']=np.where(patient_fin['DOD'].isnull(),1,0)
  patient_fin['DOD']=np.where(patient_fin['DOD'].isnull(),last_death,patient_fin['DOD'])
  patient_fin['DOD']=pd.to_datetime(patient_fin['DOD'],format='%Y-%m-%d')
  patient_fin['DOB']=pd.to_datetime(patient_fin['DOB'],format='%Y-%m-%d')
  patient_fin=patient_fin.assign(Age=lambda x:(x['DOD']-x['DOB']))
  patient_fin['Age']=patient_fin['Age'].dt.days/365
  tot=0
  for i in cols:
    tot+=patient_fin[i]
  patient_fin['Total_disease']=tot
  patient_fin['Total_deductible']=patient_fin['OPAnnualDeductibleAmt']+patient_fin['IPAnnualDeductibleAmt']
  patient_fin['Total_reimbursible']=patient_fin['OPAnnualReimbursementAmt']+patient_fin['IPAnnualReimbursementAmt']
  patient_fin.OtherPhysician.fillna(0,inplace=True)
  patient_fin.AttendingPhysician.fillna(0,inplace=True)
  patient_fin.OperatingPhysician.fillna(0,inplace=True)
  val_op=[0 if i==0 else 1 for i in patient_fin.OtherPhysician]
  val_ap=[0 if i==0 else 1 for i in patient_fin.AttendingPhysician]
  val_opp=[0 if i==0 else 1 for i in patient_fin.OperatingPhysician]
  patient_fin['Num_physician_rq']=[sum(x) for x in zip(val_op,val_ap,val_opp)]
  num_claims=patient_fin.groupby('Provider')['ClaimID'].nunique()
  patient_fin.drop(['BeneID','DOB','DOD','ClaimID','ClaimStartDt','ClaimEndDt','AdmissionDt','DischargeDt'],axis=1,inpla
  patient_fin['Days_Admit'].fillna(0,inplace=True)
  col_nan=patient_fin.columns[patient_fin.isna().any()]
  for i in col_nan:
    patient_fin[i].fillna('Not_Available',inplace=True)
  def label_encoder(X):
    if patient_fin[X].dtype=='object' and X!='Provider':
      patient_fin[X]=patient_fin[X].astype('category')
      patient_fin[X]=patient_fin[X].cat.codes
      patient_fin[X]=patient_fin[X].astype('category')
  object_dtypes=patient_fin.select_dtypes(include='object').columns
  [label_encoder(i) for i in object_dtypes]
  Avg_clm_reimbursed=patient_fin.groupby('Provider')['InscClaimAmtReimbursed'].mean()
  Tot_clm_reimbursed=patient_fin.groupby('Provider')['InscClaimAmtReimbursed'].sum()
  Avg_inpclm_reimbursed=patient_fin.groupby('Provider')['IPAnnualReimbursementAmt'].mean()
  Tot_inpclm_reimbursed=patient_fin.groupby('Provider')['IPAnnualReimbursementAmt'].sum()
  Avg_inp_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].mean()
  Tot_inp_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].sum()
  Avg_out_reimbursement=patient_fin.groupby('Provider')['OPAnnualReimbursementAmt'].mean()
  Tot_out_deductible=patient_fin.groupby('Provider')['OPAnnualReimbursementAmt'].sum()
  Avg_out_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].mean()
  Tot_out_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].sum()
  Avg_age=patient_fin.groupby('Provider')['Age'].mean()
  Avg_settlement=patient_fin.groupby('Provider')['settlement_days'].mean()
  Freq_settlement=patient_fin.groupby('Provider')['settlement_days'].agg(lambda x:x.value_counts().index[0])
  Avg_days_admit=patient_fin.groupby('Provider')['Days_Admit'].mean()
  Num_phys_rq=patient_fin.groupby('Provider')['Num_physician_rq'].mean()
  Avg_total_deductible=patient_fin.groupby('Provider')['Total_deductible'].mean()
  Sum_total_deductible=patient_fin.groupby('Provider')['Total_deductible'].sum()
  Avg_total_reimbursible=patient_fin.groupby('Provider')['Total_reimbursible'].mean()
  Sum_total_reimbursible=patient_fin.groupby('Provider')['Total_reimbursible'].sum()
  Race_dummy=pd.get_dummies(patient_fin.Race)
  patient_fin=pd.concat([patient_fin,Race_dummy],axis=1)
  patient_fin.drop('Race',axis=1,inplace=True)
  Num_males=patient_fin.groupby('Provider')['Gender'].sum()
  Num_race1=patient_fin.groupby('Provider')[1].sum()
  Num_race2=patient_fin.groupby('Provider')[2].sum()
  Num_race3=patient_fin.groupby('Provider')[3].sum()
  Num_race5=patient_fin.groupby('Provider')[5].sum()
  Ren_disease=patient_fin.groupby('Provider')['RenalDiseaseIndicator'].sum()
  Month_sum=Month_sum=patient_fin.groupby('Provider')['NoOfMonths_PartACov'].sum()
  patient_fin.drop('NoOfMonths_PartACov',axis=1,inplace=True)
  alzheimer_cnt=patient_fin.groupby('Provider')['ChronicCond_Alzheimer'].sum()
  Hrt_failure=patient_fin.groupby('Provider')['ChronicCond_Heartfailure'].sum()
  kidney_dis=patient_fin.groupby('Provider')['ChronicCond_KidneyDisease'].sum()
  Cancer=patient_fin.groupby('Provider')['ChronicCond_Cancer'].sum()
  Pulmonary=patient_fin.groupby('Provider')['ChronicCond_ObstrPulmonary'].sum()
  Depression=patient_fin.groupby('Provider')['ChronicCond_Depression'].sum()
  Diabetes=patient_fin.groupby('Provider')['ChronicCond_Diabetes'].sum()
  chemicHeart=patient_fin.groupby('Provider')['ChronicCond_IschemicHeart'].sum()
  Osteoporasis=patient_fin.groupby('Provider')['ChronicCond_Osteoporasis'].sum()
  Rheumatoid=patient_fin.groupby('Provider')['ChronicCond_rheumatoidarthritis'].sum()
  Stroke=patient_fin.groupby('Provider')['ChronicCond_stroke'].sum()
  att_phys=patient_fin.groupby('Provider')['AttendingPhysician'].nunique()
  op_phys=patient_fin.groupby('Provider')['OperatingPhysician'].nunique()
  othr_phys=patient_fin.groupby('Provider')['OtherPhysician'].nunique()
  diagnosis_codes=patient_fin.groupby('Provider')['ClmAdmitDiagnosisCode'].nunique()
  diagnosis_grp_codes=patient_fin.groupby('Provider')['DiagnosisGroupCode'].nunique()
  diagnosis_code1=patient_fin.groupby('Provider')['ClmDiagnosisCode_1'].nunique()
  diagnosis_code2=patient_fin.groupby('Provider')['ClmDiagnosisCode_2'].nunique()
  diagnosis_code3=patient_fin.groupby('Provider')['ClmDiagnosisCode_3'].nunique()
  diagnosis_code4=patient_fin.groupby('Provider')['ClmDiagnosisCode_4'].nunique()
  diagnosis_code5=patient_fin.groupby('Provider')['ClmDiagnosisCode_5'].nunique()
  diagnosis_code6=patient_fin.groupby('Provider')['ClmDiagnosisCode_6'].nunique()
  diagnosis_code7=patient_fin.groupby('Provider')['ClmDiagnosisCode_7'].nunique()
  diagnosis_code8=patient_fin.groupby('Provider')['ClmDiagnosisCode_8'].nunique()
  diagnosis_code9=patient_fin.groupby('Provider')['ClmDiagnosisCode_9'].nunique()
  diagnosis_code10=patient_fin.groupby('Provider')['ClmDiagnosisCode_10'].nunique()
  procedure_code_1=patient_fin.groupby('Provider')['ClmProcedureCode_1'].nunique()
  procedure_code_2=patient_fin.groupby('Provider')['ClmProcedureCode_2'].nunique()
  procedure_code_3=patient_fin.groupby('Provider')['ClmProcedureCode_3'].nunique()
  procedure_code_4=patient_fin.groupby('Provider')['ClmProcedureCode_4'].nunique()
  procedure_code_5=patient_fin.groupby('Provider')['ClmProcedureCode_5'].nunique()
  procedure_code_6=patient_fin.groupby('Provider')['ClmProcedureCode_6'].nunique()
  num_inpatients=patient_fin.groupby('Provider')['inp_out'].sum()
  num_outpatients=patient_fin.groupby('Provider')['inp_out'].count()-patient_fin.groupby('Provider')['inp_out'].sum()
  unq_state=patient_fin.groupby('Provider')['State'].nunique()
  unq_county=patient_fin.groupby('Provider')['County'].nunique()
  patient_agg=pd.concat([procedure_code_1,procedure_code_2,procedure_code_3,
                         procedure_code_4,procedure_code_5,procedure_code_6,num_inpatients,
                         num_outpatients,diagnosis_codes,diagnosis_grp_codes,diagnosis_code1,
                         diagnosis_code2,diagnosis_code3,diagnosis_code4,diagnosis_code5,
                         diagnosis_code6,diagnosis_code7,diagnosis_code8,diagnosis_code9,diagnosis_code10,
                         att_phys,op_phys,othr_phys,alzheimer_cnt,Hrt_failure,
                         kidney_dis,Cancer,Pulmonary,Depression,Diabetes,chemicHeart,
                         Osteoporasis,Rheumatoid,Stroke,Month_sum,Num_males,Num_race1,
                         Num_race2,Num_race3,Num_race5,Ren_disease,Avg_clm_reimbursed,
                         Tot_clm_reimbursed,Avg_inpclm_reimbursed,Tot_inpclm_reimbursed,
                         Avg_inp_deductible,Tot_inp_deductible,Avg_out_reimbursement,
                         Tot_out_deductible,Avg_age,Avg_settlement,Freq_settlement,Avg_days_admit
                         ,Avg_total_deductible,Sum_total_deductible,Avg_total_reimbursible,
                         Sum_total_reimbursible,num_claims,unq_state,unq_county
                         ],axis=1)
  provider=patient_agg.merge(X[3],how='inner',on='Provider')
  drop_cols=['inp_out','DiagnosisGroupCode','ClmDiagnosisCode_3','ClmDiagnosisCode_4',
 'ClmDiagnosisCode_8', 'ChronicCond_Alzheimer','ChronicCond_Heartfailure','ChronicCond_KidneyDisease',
 'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary','IPAnnualDeductibleAmt',
 'ChronicCond_Depression', 'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart','ChronicCond_Osteoporasis',
 'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke', 'Gender',1,
 'InscClaimAmtReimbursed','IPAnnualReimbursementAmt',
 'settlement_days','Total_deductible','NoOfMonths_PartACov',
 'Total_reimbursible','ClaimID']
  for i in drop_cols:
    provider.drop(i,axis=1,inplace=True)
  provider.drop(['ClmProcedureCode_6', 'ClmProcedureCode_5','ClmProcedureCode_4','Provider'],axis=1,inplace=True)
  model = pickle.load(open('model.pkl', 'rb'))
  prediction=model.predict(provider.values)
  return prediction
```

In [ ]:
```
strt=time()
qr1=test_data.iloc[0]
qr2=inp_data_test[inp_data_test['Provider']==qr1.values[0]]
qr3=out_data_test[out_data_test['Provider']==qr1.values[0]]
X=[ben_data_test,qr2,qr3,qr1]
out=predict(X)
end=time()
```

In [ ]:
```
print('Time taken :',end-strt)
if out==0:
  print('Not fraud')
else:
  print('Fraud!!')
```

```
Time taken : 0.34317851066589355
Not fraud
```

In [ ]:
```
import pickle
```

```python
def score(X,y):
  cols=list(set(X[1].columns).intersection(set(X[2].columns)))
  patient=X[1].merge(X[2],how='outer',on=cols)
  patient_int=patient.merge(X[0],how='left',on='BeneID')
  patient_fin=X[3].merge(patient_int,how='inner',on='Provider')
  cols=['ChronicCond_Alzheimer','ChronicCond_Heartfailure','ChronicCond_KidneyDisease',
      'ChronicCond_Cancer','ChronicCond_ObstrPulmonary','ChronicCond_Depression','ChronicCond_Diabetes',
      'ChronicCond_IschemicHeart','ChronicCond_Osteoporasis','ChronicCond_rheumatoidarthritis',
      'ChronicCond_stroke']
  y=y.map({'Yes':1,'No':0})
  for i in cols:
    patient_fin[i]=patient_fin[i].map({1:0,2:1})
  patient_fin.RenalDiseaseIndicator=patient_fin.RenalDiseaseIndicator.map({'0':0,'Y':1})
  patient_fin['ClaimStartDt']=pd.to_datetime(patient_fin.ClaimStartDt,format='%Y-%m-%d')
  patient_fin['ClaimEndDt']=pd.to_datetime(patient_fin.ClaimEndDt,format='%Y-%m-%d')
  patient_fin['settlement_days']=patient_fin['ClaimEndDt']-patient_fin['ClaimStartDt']
  patient_fin['settlement_days']=patient_fin['settlement_days'].dt.days
  patient_fin['AdmissionDt']=pd.to_datetime(patient_fin['AdmissionDt'],format='%Y-%m-%d')
  patient_fin['DischargeDt']=pd.to_datetime(patient_fin['DischargeDt'],format='%Y-%m-%d')
  patient_fin['Days_Admit']=patient_fin['DischargeDt']-patient_fin['AdmissionDt']
  patient_fin['Days_Admit']=patient_fin['Days_Admit'].dt.days
  inp_out=patient_fin.Days_Admit.isnull()
  inp_data=inp_out.map({True:1,False:0})
  patient_fin['inp_out']=inp_data
  patient_fin.Days_Admit.fillna(0,inplace=True)
  last_death='2009-12-01'
  patient_fin['Alive_or_not']=np.where(patient_fin['DOD'].isnull(),1,0)
  patient_fin['DOD']=np.where(patient_fin['DOD'].isnull(),last_death,patient_fin['DOD'])
  patient_fin['DOD']=pd.to_datetime(patient_fin['DOD'],format='%Y-%m-%d')
  patient_fin['DOB']=pd.to_datetime(patient_fin['DOB'],format='%Y-%m-%d')
  patient_fin=patient_fin.assign(Age=lambda x:(x['DOD']-x['DOB']))
  patient_fin['Age']=patient_fin['Age'].dt.days/365
  tot=0
  for i in cols:
    tot+=patient_fin[i]
  patient_fin['Total_disease']=tot
  patient_fin['Total_deductible']=patient_fin['OPAnnualDeductibleAmt']+patient_fin['IPAnnualDeductibleAmt']
  patient_fin['Total_reimbursible']=patient_fin['OPAnnualReimbursementAmt']+patient_fin['IPAnnualReimbursementAmt']
  patient_fin.OtherPhysician.fillna(0,inplace=True)
  patient_fin.AttendingPhysician.fillna(0,inplace=True)
  patient_fin.OperatingPhysician.fillna(0,inplace=True)
  val_op=[0 if i==0 else 1 for i in patient_fin.OtherPhysician]
  val_ap=[0 if i==0 else 1 for i in patient_fin.AttendingPhysician]
  val_opp=[0 if i==0 else 1 for i in patient_fin.OperatingPhysician]
  patient_fin['Num_physician_rq']=[sum(x) for x in zip(val_op,val_ap,val_opp)]
  num_claims=patient_fin.groupby('Provider')['ClaimID'].nunique()
  patient_fin.drop(['BeneID','DOB','DOD','ClaimID','ClaimStartDt','ClaimEndDt','AdmissionDt','DischargeDt'],axis=1,inpla
  patient_fin['Days_Admit'].fillna(0,inplace=True)
  col_nan=patient_fin.columns[patient_fin.isna().any()]
  for i in col_nan:
    patient_fin[i].fillna('Not_Available',inplace=True)
  def label_encoder(X):
    if patient_fin[X].dtype=='object' and X!='Provider':
      patient_fin[X]=patient_fin[X].astype('category')
      patient_fin[X]=patient_fin[X].cat.codes
      patient_fin[X]=patient_fin[X].astype('category')
  object_dtypes=patient_fin.select_dtypes(include='object').columns
  [label_encoder(i) for i in object_dtypes]
  Avg_clm_reimbursed=patient_fin.groupby('Provider')['InscClaimAmtReimbursed'].mean()
  Tot_clm_reimbursed=patient_fin.groupby('Provider')['InscClaimAmtReimbursed'].sum()
  Avg_inpclm_reimbursed=patient_fin.groupby('Provider')['IPAnnualReimbursementAmt'].mean()
  Tot_inpclm_reimbursed=patient_fin.groupby('Provider')['IPAnnualReimbursementAmt'].sum()
  Avg_inp_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].mean()
  Tot_inp_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].sum()
  Avg_out_reimbursement=patient_fin.groupby('Provider')['OPAnnualReimbursementAmt'].mean()
  Tot_out_deductible=patient_fin.groupby('Provider')['OPAnnualReimbursementAmt'].sum()
  Avg_out_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].mean()
  Tot_out_deductible=patient_fin.groupby('Provider')['IPAnnualDeductibleAmt'].sum()
  Avg_age=patient_fin.groupby('Provider')['Age'].mean()
  Avg_settlement=patient_fin.groupby('Provider')['settlement_days'].mean()
  Freq_settlement=patient_fin.groupby('Provider')['settlement_days'].agg(lambda x:x.value_counts().index[0])
  Avg_days_admit=patient_fin.groupby('Provider')['Days_Admit'].mean()
  Num_phys_rq=patient_fin.groupby('Provider')['Num_physician_rq'].mean()
  Avg_total_deductible=patient_fin.groupby('Provider')['Total_deductible'].mean()
  Sum_total_deductible=patient_fin.groupby('Provider')['Total_deductible'].sum()
  Avg_total_reimbursible=patient_fin.groupby('Provider')['Total_reimbursible'].mean()
  Sum_total_reimbursible=patient_fin.groupby('Provider')['Total_reimbursible'].sum()
  Race_dummy=pd.get_dummies(patient_fin.Race)
  patient_fin=pd.concat([patient_fin,Race_dummy],axis=1)
  patient_fin.drop('Race',axis=1,inplace=True)
  Num_males=patient_fin.groupby('Provider')['Gender'].sum()
  Num_race1=patient_fin.groupby('Provider')[1].sum()
  Num_race2=patient_fin.groupby('Provider')[2].sum()
  Num_race3=patient_fin.groupby('Provider')[3].sum()
  Num_race5=patient_fin.groupby('Provider')[5].sum()
  Ren_disease=patient_fin.groupby('Provider')['RenalDiseaseIndicator'].sum()
  Month_sum=Month_sum=patient_fin.groupby('Provider')['NoOfMonths_PartACov'].sum()
  patient_fin.drop('NoOfMonths_PartACov',axis=1,inplace=True)
  alzheimer_cnt=patient_fin.groupby('Provider')['ChronicCond_Alzheimer'].sum()
  Hrt_failure=patient_fin.groupby('Provider')['ChronicCond_Heartfailure'].sum()
  kidney_dis=patient_fin.groupby('Provider')['ChronicCond_KidneyDisease'].sum()
  Cancer=patient_fin.groupby('Provider')['ChronicCond_Cancer'].sum()
  Pulmonary=patient_fin.groupby('Provider')['ChronicCond_ObstrPulmonary'].sum()
  Depression=patient_fin.groupby('Provider')['ChronicCond_Depression'].sum()
  Diabetes=patient_fin.groupby('Provider')['ChronicCond_Diabetes'].sum()
  chemicHeart=patient_fin.groupby('Provider')['ChronicCond_IschemicHeart'].sum()
```

```
    Osteoporasis=patient_fin.groupby('Provider')['ChronicCond_Osteoporasis'].sum()
    Rheumatoid=patient_fin.groupby('Provider')['ChronicCond_rheumatoidarthritis'].sum()
    Stroke=patient_fin.groupby('Provider')['ChronicCond_stroke'].sum()
    att_phys=patient_fin.groupby('Provider')['AttendingPhysician'].nunique()
    op_phys=patient_fin.groupby('Provider')['OperatingPhysician'].nunique()
    othr_phys=patient_fin.groupby('Provider')['OtherPhysician'].nunique()
    diagnosis_codes=patient_fin.groupby('Provider')['ClmAdmitDiagnosisCode'].nunique()
    diagnosis_grp_codes=patient_fin.groupby('Provider')['DiagnosisGroupCode'].nunique()
    diagnosis_code1=patient_fin.groupby('Provider')['ClmDiagnosisCode_1'].nunique()
    diagnosis_code2=patient_fin.groupby('Provider')['ClmDiagnosisCode_2'].nunique()
    diagnosis_code3=patient_fin.groupby('Provider')['ClmDiagnosisCode_3'].nunique()
    diagnosis_code4=patient_fin.groupby('Provider')['ClmDiagnosisCode_4'].nunique()
    diagnosis_code5=patient_fin.groupby('Provider')['ClmDiagnosisCode_5'].nunique()
    diagnosis_code6=patient_fin.groupby('Provider')['ClmDiagnosisCode_6'].nunique()
    diagnosis_code7=patient_fin.groupby('Provider')['ClmDiagnosisCode_7'].nunique()
    diagnosis_code8=patient_fin.groupby('Provider')['ClmDiagnosisCode_8'].nunique()
    diagnosis_code9=patient_fin.groupby('Provider')['ClmDiagnosisCode_9'].nunique()
    diagnosis_code10=patient_fin.groupby('Provider')['ClmDiagnosisCode_10'].nunique()
    procedure_code_1=patient_fin.groupby('Provider')['ClmProcedureCode_1'].nunique()
    procedure_code_2=patient_fin.groupby('Provider')['ClmProcedureCode_2'].nunique()
    procedure_code_3=patient_fin.groupby('Provider')['ClmProcedureCode_3'].nunique()
    procedure_code_4=patient_fin.groupby('Provider')['ClmProcedureCode_4'].nunique()
    procedure_code_5=patient_fin.groupby('Provider')['ClmProcedureCode_5'].nunique()
    procedure_code_6=patient_fin.groupby('Provider')['ClmProcedureCode_6'].nunique()
    num_inpatients=patient_fin.groupby('Provider')['inp_out'].sum()
    num_outpatients=patient_fin.groupby('Provider')['inp_out'].count()-patient_fin.groupby('Provider')['inp_out'].sum()
    unq_state=patient_fin.groupby('Provider')['State'].nunique()
    unq_county=patient_fin.groupby('Provider')['County'].nunique()
    patient_agg=pd.concat([procedure_code_1,procedure_code_2,procedure_code_3,
                         procedure_code_4,procedure_code_5,procedure_code_6,num_inpatients,
                         num_outpatients,diagnosis_codes,diagnosis_grp_codes,diagnosis_code1,
                         diagnosis_code2,diagnosis_code3,diagnosis_code4,diagnosis_code5,
                         diagnosis_code6,diagnosis_code7,diagnosis_code8,diagnosis_code9,diagnosis_code10,
                         att_phys,op_phys,othr_phys,alzheimer_cnt,Hrt_failure,
                         kidney_dis,Cancer,Pulmonary,Depression,Diabetes,chemicHeart,
                         Osteoporasis,Rheumatoid,Stroke,Month_sum,Num_males,Num_race1,
                         Num_race2,Num_race3,Num_race5,Ren_disease,Avg_clm_reimbursed,
                         Tot_clm_reimbursed,Avg_inpclm_reimbursed,Tot_inpclm_reimbursed,
                         Avg_inp_deductible,Tot_inp_deductible,Avg_out_reimbursement,
                         Tot_out_deductible,Avg_age,Avg_settlement,Freq_settlement,Avg_days_admit
                         ,Avg_total_deductible,Sum_total_deductible,Avg_total_reimbursible,
                         Sum_total_reimbursible,num_claims,unq_state,unq_county
                         ],axis=1)
    provider=patient_agg.merge(X[3],how='inner',on='Provider')
    drop_cols=['inp_out','DiagnosisGroupCode','ClmDiagnosisCode_3','ClmDiagnosisCode_4',
   'ClmDiagnosisCode_8', 'ChronicCond_Alzheimer','ChronicCond_Heartfailure','ChronicCond_KidneyDisease',
   'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary','IPAnnualDeductibleAmt',
   'ChronicCond_Depression', 'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart','ChronicCond_Osteoporasis',
   'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke', 'Gender',1,
   'InscClaimAmtReimbursed','IPAnnualReimbursementAmt',
   'settlement_days','Total_deductible','NoOfMonths_PartACov',
   'Total_reimbursible','ClaimID']
    for i in drop_cols:
      provider.drop(i,axis=1,inplace=True)
    provider.drop(['ClmProcedureCode_6', 'ClmProcedureCode_5','ClmProcedureCode_4','Provider'],axis=1,inplace=True)
    model = pickle.load(open('model.pkl', 'rb'))
    prediction=model.predict(provider.values)
    f1_scr=f1_score(y,prediction)

    return f1_scr
```

In [47]:
```
strt=time()
X=[ben_data,inp_data,out_data,train_data]
y=train_data['PotentialFraud']
train_data.drop('PotentialFraud',axis=1,inplace=True)
print('F1 score is ',score(X,y))
end=time()
print('Time taken : ',end-strt)
```

```
F1 score is  0.6896551724137931
Time taken :  24.777477025985718
```

In [ ]:

In [ ]: