Report On

Big Data Medicare Fraud Detection

Submitted in partial fulfillment of the requirements of the Course Project for Natural Language Processing in Semester VII of Fourth Year Artificial Intelligence & Data Science Engineering

by

Harshavardhan Surve (Roll No. 54)

Pritesh Verma (Roll No. 56)

Anaum Sharif (Roll No. 46)

Under the guidance

Prof. Bhavika Gharat

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(A.Y. 2024-25)

CERTIFICATE

This is to certify that the project entitled "Big Data Medicare Fraud Detection" is a bonafide work of Harshavardhan Surve (Roll No. 54), Pritesh Verma (Roll No. 56), Anaum Sharif (Roll No. 46) submitted to the University of Mumbai in fulfillment of the requirement for the Course project in semester VII of Fourth Year Artificial Intelligence and Data Science engineering.

____

Guide

Abstract

The increasing cost of healthcare worldwide has highlighted the urgent need for efficient fraud detection systems, particularly in government-funded health programs. This project focuses on developing a machine learning-based solution to detect fraud within healthcare systems using big data analysis techniques. Leveraging datasets from multiple sources, such as prescription data, payment records, and lists of excluded individuals and entities, the project aims to identify fraudulent activities by healthcare providers, patients, and associated entities. A range of machine learning models, including Random Forest, Logistic Regression, and Gradient Boosting, were utilized to enhance prediction accuracy and uncover complex patterns of fraudulent behavior.

Table of Contents

1. Introduction

Healthcare systems around the world are grappling with the dual challenges of rising costs and resource management. As healthcare expenditures increase, so does the susceptibility to fraudulent activities, which can significantly impact the financial stability of health programs. Fraud within healthcare manifests in several forms, including false claims for services never rendered, inflated billing for procedures, misuse of insurance coverage, and coordinated schemes involving multiple parties. Estimates suggest that a substantial percentage of healthcare expenses can be attributed to fraudulent practices, making fraud detection a critical component in managing healthcare costs and ensuring the efficient use of resources.

Detecting fraud in healthcare is a complex problem due to the sheer volume and diversity of data involved, such as patient records, prescription details, and payment transactions. Traditional approaches often rely on rule-based systems or manual audits, which are not well-equipped to handle the large-scale, dynamic nature of healthcare data. The need for more sophisticated detection techniques has led to the adoption of machine learning and big data analytics, which can analyze patterns across vast datasets to uncover potential fraud.

This project aims to address the challenges in healthcare fraud detection by developing an innovative machine learning system capable of identifying fraudulent activities through anomaly analysis and the integration of geo-demographic metrics. The approach leverages multiple data sources, including prescription data, payment records, and databases of excluded individuals and entities, to detect fraud across different levels, such as individual healthcare providers, insurance subscribers, and organizations. By employing various machine learning models, the project seeks to provide a scalable solution that enhances the accuracy of fraud detection and contributes to more effective management of healthcare systems.

2. Problem Statement

Healthcare fraud is a widespread issue that affects the efficiency and sustainability of health programs globally. Fraudulent activities in healthcare can occur at multiple levels, involving service providers (e.g., doctors, hospitals, and pharmacies), insurance policyholders, and organizations. These activities may include submitting false claims, overbilling for services, manipulating patient records, or even complex schemes involving multiple parties to exploit the system.

Traditional fraud detection methods often fall short in accurately identifying fraudulent cases due to the complex nature of healthcare data and the evolving tactics used by fraudsters. Furthermore, the data is typically highly imbalanced, with a vast majority of legitimate cases overshadowing the few fraudulent ones, making accurate prediction challenging. As a result, there is a pressing need for advanced solutions that can efficiently detect and prevent fraud using automated, data-driven approaches.

The objective of this project is to build a machine learning-based system capable of detecting healthcare fraud by analyzing large datasets for anomalies and leveraging demographic and geographic data. The system aims to improve detection accuracy and provide insights into the patterns of fraudulent behavior across different entities involved in healthcare.

3. Proposed System

The proposed system employs a machine learning-based approach for detecting healthcare fraud by leveraging big data analytics. The system is designed to integrate multiple datasets, perform data preprocessing, apply feature engineering, and train various machine learning models to detect anomalies indicative of fraudulent activity. The workflow can be broken down into the following steps:

a. Data Collection and Integration

- **Datasets**: Uses prescription data, payment data, and the LEIE database.
- **Data Characteristics**: Involves millions of records, combining details like demographics, locations, payments, and healthcare services.

b. Data Pre-Processing

- **Data Cleaning**: Handles missing values, removes duplicates/outliers, and encodes categorical data.
- **Transformation**: Applies normalization techniques to standardize data.
- **Balancing**: Uses class weights and sampling techniques to address class imbalance.

c. Feature Engineering

- **Data Merging**: Integrates datasets based on attributes like NPI and geographic location.
- **New Features**: Creates features such as average payment per provider or regional risk factors.
- **Feature Selection**: Reduces dimensionality using techniques like PCA.

d. Data Modeling

- **Models Used**: Logistic Regression, Gaussian Naïve Bayes, Random Forest, Extra Trees, and Gradient Boosting.
- **Training & Validation**: Uses train-test splits and cross-validation.
- **Hyperparameter Tuning**: Optimizes model parameters for better performance.

e. Model Evaluation

- **Metrics**: Assesses models using AUC, Precision, Recall, and F1-score.
- **Error Analysis**: Examines misclassifications to refine models.

f. Deployment and Real-Time Detection

- **Model Deployment**: Uses MLFlow for hosting.
- **Real-Time Pipeline**: Implements Kafka for data streaming.
- **Continuous Updating**: Retrains models periodically with new data.

g. Visualization and Insights

- **EDA**: Visualizes fraud patterns and geographical distribution.
- **Reporting**: Provides insights and highlights high-risk areas.

4. Implementation Plans

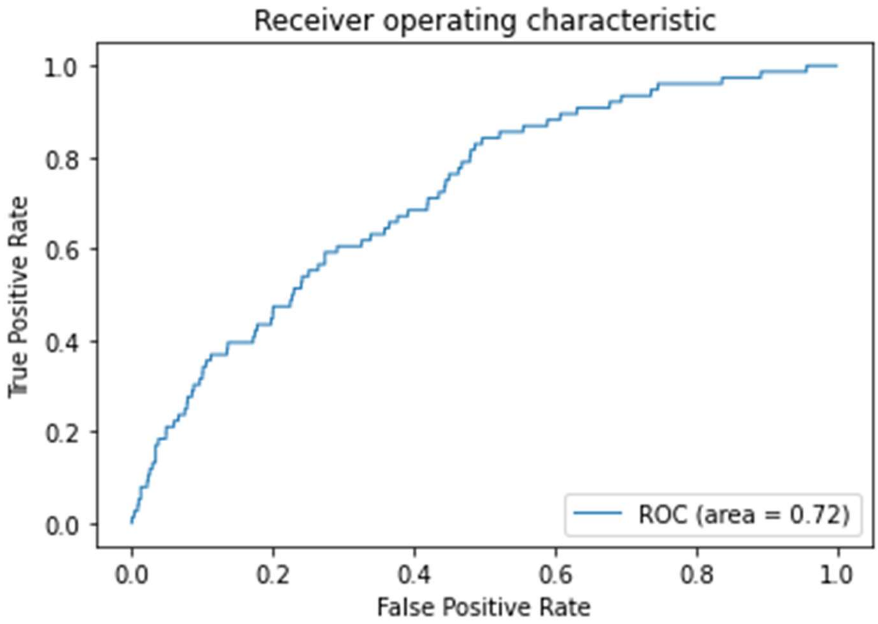
The implementation of the fraud detection system follows these key phases:

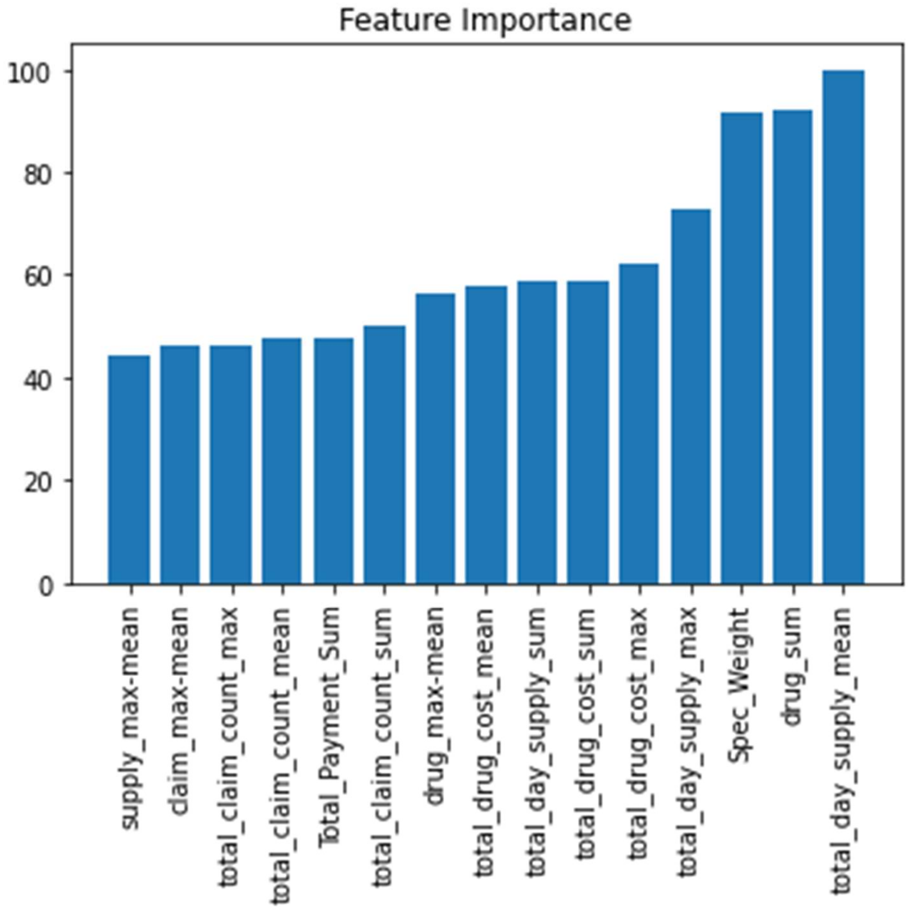
1. **Data Preparation**
   - Data Acquisition: Collect prescription, payment, and LEIE datasets.
   - Storage and Cleaning: Use cloud databases, clean data, handle missing values, and normalize features.
   - Balancing: Address class imbalance through sampling techniques.
2. **Model Development**
   - Feature Engineering: Merge datasets, create new features, and reduce dimensionality.
   - Model Training: Train models like Logistic Regression, Random Forest, and Gradient Boosting.
   - Evaluation and Tuning: Use metrics (AUC, Precision) for model assessment and tune hyperparameters for optimal results.
3. **Deployment**
   - Model Deployment: Use MLFlow for hosting.
   - Real-Time Pipeline: Set up data streaming using Kafka, integrate with APIs for real-time predictions.
4. **Monitoring and Maintenance**
   - System Monitoring: Track model performance with dashboards and alerts.
   - Model Updates: Automate retraining using new data without service disruption.
   - User Feedback: Incorporate expert feedback to improve the model.
5. **Future Enhancements**
   - Expand Data Sources: Add more datasets for improved accuracy.
   - Advanced Techniques: Implement deep learning and explainable AI.
   - Scalability: Optimize for large data volumes with distributed processing.

5. Implementation Results and Analysis

1. **Model Performance**
   - **Random Forest**: Achieved the best results with an AUC score of 72%, effectively identifying patterns indicative of fraud.
   - **Logistic Regression and Gaussian Naïve Bayes**: Showed lower accuracy and AUC compared to ensemble methods, indicating limited ability to capture complex fraud patterns.
   - **Gradient Boosting**: Performed comparably well, providing high precision but with slightly lower recall than Random Forest.
2. **Error Analysis**
   - **False Positives**: Some non-fraudulent cases were misclassified as fraud due to atypical patterns in claims or payments, suggesting the need for additional features to improve specificity.
   - **False Negatives**: Occurred mainly in cases with subtle fraud signals, highlighting the potential benefit of incorporating more granular data or using advanced techniques like deep learning.
3. **Insights from Data Analysis**
   - **Geographic Distribution**: Higher concentrations of fraud cases were observed in specific regions, suggesting localized factors contribute to increased fraud risk.
   - **Fraud Patterns**: The most common fraud patterns involved overbilling and unnecessary services by healthcare providers.
   - **Feature Impact**: Features such as payment frequency, drug costs, and provider exclusion status significantly influenced fraud predictions.
4. **Model Evaluation Metrics**
   - **Precision and Recall Trade-Off**: Random Forest demonstrated a good balance, with a precision of 0.70 and recall of 0.68, optimizing the detection of fraud while minimizing false alerts.
   - **F1-Score**: Reached 0.69, indicating a favorable balance between precision and recall, crucial for fraud detection scenarios where both false positives and false negatives carry significant costs.
5. **Comparative Analysis**
   - **Baseline Models vs. Ensemble Methods**: Ensemble models like Random Forest and Gradient Boosting outperformed baseline methods (Logistic Regression), showing their effectiveness in handling large and complex datasets.
   - **Impact of Data Balancing**: Class weighting and sampling improved model performance by addressing the data imbalance, enhancing recall for minority (fraudulent) cases.
6. **Deployment Outcomes**
   - **Real-Time Detection**: The real-time pipeline successfully handled streaming data, enabling prompt fraud identification.
   - **System Stability**: Deployment with MLFlow ensured model monitoring and easy updates, maintaining consistent performance.

5.1 Implementaion Screenshots





6. Conclusion

The healthcare fraud detection project successfully demonstrated the effectiveness of machine learning techniques in identifying fraudulent activities within healthcare systems. By leveraging large datasets and employing advanced analytics, the system achieved a commendable AUC score of 72% with the Random Forest model, indicating strong predictive capabilities.

Key insights from the analysis highlighted the geographic concentration of fraud cases and the significance of specific features in predicting fraudulent behavior. The implementation showcased the benefits of ensemble methods over traditional models, particularly in complex data environments characterized by imbalance and variability.

Overall, the project not only addressed a pressing issue in healthcare management but also provided a scalable framework that can be adapted to evolving fraud patterns. The real-time detection capabilities established through a robust pipeline further enhance the system's value, ensuring timely intervention against fraudulent activities. Future enhancements, such as incorporating deep learning techniques and expanding data sources, promise to further elevate the system's performance and adaptability, contributing to more efficient healthcare fraud mitigation strategies.

7. Code

```
# %%
import pandas as pd
import numpy as np
import scipy
import os

import matplotlib.pyplot as plt

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn import ensemble
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import brier_score_loss, precision_score, recall_score,f1_score,
roc_auc_score, accuracy_score
from sklearn.metrics import confusion_matrix, roc_curve

from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction import DictVectorizer
from sklearn.cluster import KMeans

import random

from scipy.stats import ttest_ind

# %%
PartDRawData = "c:\\FIP\\PartD_Prescriber_PUF_NPI_Drug_17.txt"


# %%
partD_pd = pd.read_csv(PartDRawData,sep="\t")

# %%
partD_pd.shape

# %%
partD_Drug_pd1= partD_pd.loc[:,['npi','nppes_provider_city','nppes_provider_state', \
                                'nppes_provider_last_org_name', \
                                'nppes_provider_first_name', \
                                'specialty_description',\
                                'drug_name', \
                                'generic_name',\
                                'total_drug_cost',\
                                'total_claim_count',\
                                'total_day_supply']]
```

```
# %%
partD_pd1 = partD_Drug_pd1

# %%
partD_Drug_pd =
partD_pd.loc[:,['npi','drug_name','total_drug_cost','total_claim_count','total_day_supply','spec
ialty_description']]
partD_Drug_pd['npi'] = partD_Drug_pd.npi.astype(object)

# %%
partD_Spec_pd1= partD_pd.loc[:,['npi','specialty_description']]

# %%
partD_Spec_pd1.head(0)

# %%
partD_Drug_pd.head()

# %%
partD_pd0= partD_pd.loc[:,['npi','nppes_provider_city','nppes_provider_state', \
                                    'nppes_provider_last_org_name', \
                                    'nppes_provider_first_name','specialty_description']]

# %%
partD_pd0.head()

# %%
partD_catfpd = partD_pd0.drop_duplicates()

# %%
partD_catfpd.head()

# %%
rename_dict = {'nppes_provider_first_name':'first_name',
'nppes_provider_last_org_name':'last_name','nppes_provider_city':'city','nppes_provider_state
':'state','specialty_description':'Speciality'}
partD_catfpd = partD_catfpd.rename(columns=rename_dict)

# %%
partD_catfpd.head()

# %%
group_cols = ['npi']

agg_dict = {'total_drug_cost':['sum','mean','max'], \
        'total_claim_count':['sum','mean','max'],\
        'total_day_supply':['sum','mean','max']}
```

```
partD_pd2 = partD_pd1.groupby(group_cols).agg(agg_dict).astype(float)

# %%
partD_pd2.head(-10)

# %%
level0 = partD_pd2.columns.get_level_values(0)
level1 = partD_pd2.columns.get_level_values(1)
partD_pd2.columns = level0 + '_' + level1
partD_fpd = partD_pd2.reset_index()

# %%
partD_fpd.head()

# %%
partD_fpd.count()

# %%
partD_allpd = pd.merge(partD_fpd,partD_catfpd, how ='left',on = 'npi')

# %%
partD_allpd.head()

# %%
partD_allpd.count()

# %%
PaymentRawData = "c:\\FIP\\OP_DTL_GNRL_PGYR2017_P01172020.csv"

# %%
payment_pd = pd.read_csv(PaymentRawData)

# %%
payment_fpd = payment_pd.loc[:,['Physician_First_Name',\
                                'Physician_Last_Name', \
                                'Recipient_City', \
                                'Recipient_State', \
                                'Total_Amount_of_Payment_USDollars']]

# %%
payment_fpd.head()

# %%
payment_fpd.count()

# %%
payment_fpd1 =
```

```
payment_fpd.groupby(['Physician_First_Name','Physician_Last_Name','Recipient_City','Reci
pient_State'])\

                              .agg({'Total_Amount_of_Payment_USDollars':['sum']}).astype(float)

# %%
level0 = payment_fpd1.columns.get_level_values(0)

# %%
level1 = payment_fpd1.columns.get_level_values(1)

# %%
payment_fpd1.columns = level0 + '_' + level1

# %%
payment_fpd1.reset_index()

# %%
payment_fpd1.head()

# %%
rename_dict = {'Physician_First_Name':'first_name',
'Physician_Last_Name':'last_name','Recipient_City':'city','Recipient_State':'state','Total_Amo
unt_of_Payment_USDollars_sum':'Total_Payment_Sum'}
payment_fpd1 = payment_fpd1.rename(columns=rename_dict)

# %%
payment_fpd1.head()

# %%
payment_fpd2= payment_fpd1.reset_index()

# %%
rename_dict = {'Physician_First_Name':'first_name',
'Physician_Last_Name':'last_name','Recipient_City':'city','Recipient_State':'state','Total_Amo
unt_of_Payment_USDollars_sum':'Total_Payment_Sum'}
payment_fpd2 = payment_fpd2.rename(columns=rename_dict)

# %%
payment_fpd2 = payment_fpd2.sort_values('Total_Payment_Sum',ascending=False)

# %%
payment_fpd2.head()

# %%
print(payment_fpd2.dtypes)

# %%
```

```
# %%
payment_fpd2.apply(lambda x: x.astype(str).str.upper())

payment_fpd2.head()

# %%
pay_partD_fpd = pd.merge (partD_allpd,payment_fpd2, how ='left', on =
['last_name','first_name','city','state'])

# %%
pay_partD_fpd.head()

# %%
pay_partD_fpd.count()

# %%
IELErawdata = "c:\\FIP\\LEIE.csv"
IELE_pd = pd.read_csv(IELErawdata)

# %%
IELE_pd.head()

# %%
npifraud_pd0 = IELE_pd.loc[:,['NPI','EXCLTYPE']]

# %%
npifraud_pd0.head()

# %%
npifraud_pd1 = npifraud_pd0.query('NPI !=0')

# %%
npifraud_pd1.count()

# %%
rename_dict = {'NPI':'npi', 'EXCLTYPE':'is_fraud'}
npi_fraud_pd = npifraud_pd1.rename(columns=rename_dict)

# %%
npi_fraud_pd.head()

# %%
npi_fraud_pd['is_fraud'] = 1

# %%
npi_fraud_pd.head()

# %%
```

```
print(npi_fraud_pd.dtypes)

# %%
# Features Engineering
Features_pd1 = pd.merge(pay_partD_fpd,npi_fraud_pd, how ='left',on = 'npi')

# %%

Features_pd1.count()

# %%
Features_pd1.describe()

# %%
Features_pd1.fillna(0, inplace=True)

# %%
Features_pd1

# %%
Features_pd1[Features_pd1['is_fraud']==1].count()

# %%
FeaturesAll_pd=Features_pd1

# %%
# Scaling the features
FeaturesAll_pd['total_drug_cost_sum'] = FeaturesAll_pd['total_drug_cost_sum'].map(lambda
x: np.log10(x + 1.0))
FeaturesAll_pd['total_claim_count_sum'] =
FeaturesAll_pd['total_claim_count_sum'].map(lambda x: np.log10(x + 1.0))
FeaturesAll_pd['total_day_supply_sum'] =
FeaturesAll_pd['total_day_supply_sum'].map(lambda x: np.log10(x + 1.0))
FeaturesAll_pd['Total_Payment_Sum'] = FeaturesAll_pd['Total_Payment_Sum'].map(lambda
x: np.log10(x + 1.0))

FeaturesAll_pd['total_drug_cost_mean'] =
FeaturesAll_pd['total_drug_cost_mean'].map(lambda x: np.log10(x + 1.0))
FeaturesAll_pd['total_claim_count_mean'] =
FeaturesAll_pd['total_claim_count_mean'].map(lambda x: np.log10(x + 1.0))
FeaturesAll_pd['total_day_supply_mean'] =
FeaturesAll_pd['total_day_supply_mean'].map(lambda x: np.log10(x + 1.0))

FeaturesAll_pd['total_drug_cost_max'] = FeaturesAll_pd['total_drug_cost_max'].map(lambda
x: np.log10(x + 1.0))
FeaturesAll_pd['total_claim_count_max'] =
FeaturesAll_pd['total_claim_count_max'].map(lambda x: np.log10(x + 1.0))
FeaturesAll_pd['total_day_supply_max'] =
```

```
FeaturesAll_pd['total_day_supply_max'].map(lambda x: np.log10(x + 1.0))

FeaturesAll_pd['claim_max-mean'] = FeaturesAll_pd['total_claim_count_max'] -
FeaturesAll_pd['total_claim_count_mean']

FeaturesAll_pd['supply_max-mean'] = FeaturesAll_pd['total_day_supply_max'] -
FeaturesAll_pd['total_day_supply_mean']

FeaturesAll_pd['drug_max-mean'] = FeaturesAll_pd['total_drug_cost_max'] -
FeaturesAll_pd['total_drug_cost_mean']

# %%
FeaturesAll_pd

# %%
FeaturesAll_pd['npi'] = FeaturesAll_pd.npi.astype(object)

# %%
categorical_features = ['npi','last_name', 'Speciality','first_name','city', 'state']

# %%
numerical_features = ['total_drug_cost_sum', 'total_drug_cost_mean','Total_Payment_Sum',
      'total_drug_cost_max', 'total_claim_count_sum',
      'total_claim_count_mean', 'total_claim_count_max',
      'total_day_supply_sum', 'total_day_supply_mean', 'total_day_supply_max',
   'claim_max-mean','supply_max-mean', 'drug_max-mean']

# %%
target = ['is_fraud']

# %%
allvars = categorical_features + numerical_features + target

# %%
y = FeaturesAll_pd["is_fraud"].values
X = FeaturesAll_pd[allvars].drop('is_fraud',axis=1)

# %%
# scikit learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
#from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction import DictVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc
```

```
# %%

X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=0)
print(X_train.shape)
print(X_valid.shape)

# %%
X_train[numerical_features] = X_train.loc[:,numerical_features].fillna(0)
X_valid[numerical_features] = X_valid.loc[:,numerical_features].fillna(0)
X_train[categorical_features] = X_train.loc[:,categorical_features].fillna('NA')
X_valid[categorical_features] = X_valid.loc[:,categorical_features].fillna('NA')

# %%
scaler= StandardScaler()
X_train[numerical_features] = scaler.fit_transform(X_train[numerical_features].values)
X_valid[numerical_features] = scaler.transform(X_valid[numerical_features].values)

# %%
print(X_train[numerical_features].dtypes)

# %%
ix_ran = FeaturesAll_pd.index.values
np.random.shuffle(ix_ran)

df_len = len(FeaturesAll_pd)
train_len = int(df_len * 0.8)  # 80% for training

ix_train = ix_ran[:train_len]
ix_valid = ix_ran[train_len:]

df_train = FeaturesAll_pd.ix[ix_train]
df_valid = FeaturesAll_pd.ix[ix_valid]

print(len(ix_train))
print(len(ix_valid))

# %%
print(df_train.dtypes)

# %%
# Drug Weighted_Scores

partD_drug_train = pd.merge(partD_Drug_pd,df_train[['npi','is_fraud']], how='inner',
on=['npi'])
partD_drug_All = pd.merge(partD_Drug_pd,FeaturesAll_pd[['npi','is_fraud']], how='inner',
on=['npi'])
```

```
print(len(partD_drug_train[partD_drug_train['is_fraud']==1]))


# %%
# get unique drug names
drugs = set([ drugx for drugx in partD_drug_train['drug_name'].values if isinstance(drugx,
str)])
print(len(drugs))

# %%
print("Total records in train set : ")
print(len(partD_drug_train))
print("Total Fraud in train set : ")
print(len(partD_drug_train[partD_drug_train['is_fraud']==1]))
partD_drug_train.head()

# %%
cols = ['total_drug_cost','total_claim_count','total_day_supply']

# %%
partD_drug_train_Group = partD_drug_train.groupby(['drug_name', 'is_fraud'])
partD_drug_All_Group = partD_drug_All.groupby(['drug_name', 'is_fraud'])

# %%
drug_keys = partD_drug_train_Group.groups.keys()
print(len(drug_keys))

# %%
drug_keys

# %%
drug_with_isfraud = [drugx for drugx in drugs if ((drugx,0.0) in drug_keys ) & ( (drugx,1.0)
in drug_keys)]

# %%
from scipy.stats import ttest_ind
re_drug_tt = dict()
for drugx in drug_with_isfraud:
    for colx in cols:
        fraud_0 = partD_drug_train_Group.get_group((drugx,0.0))[colx].values
        fraud_1 = partD_drug_train_Group.get_group((drugx,1.0))[colx].values
        # print len(fraud_0), len(fraud_1)
        if (len(fraud_0)>2) & (len(fraud_1)>2) :
            tt = ttest_ind(fraud_0, fraud_1)
            re_drug_tt[(drugx, colx)] = tt

# %%
#Setting Probilities
Prob_005 = [(key, p) for (key, (t, p)) in re_drug_tt.items() if p <=0.05]
print(len(Prob_005))
```

```
# %%
inx=100
drug_name = Prob_005[inx][0][0]
print(drug_name)
df_bar = pd.concat([partD_drug_All_Group.get_group((Prob_005[inx][0][0],0.0)),
partD_drug_All_Group.get_group((Prob_005[inx][0][0],1.0))])
df_bar.head()

# %%
Feture_DrugWeighted = []
new_col_all =[]
for i, p005x in enumerate(Prob_005):
    #if i>4:
    #   break
    drug_name = p005x[0][0]
    cat_name = p005x[0][1]

    new_col = drug_name+'_'+cat_name
    new_col_all.append(new_col)

    drug_0 = partD_drug_All_Group.get_group((drug_name,0.0))[['npi', cat_name]]
    drug_1 = partD_drug_All_Group.get_group((drug_name,1.0))[['npi', cat_name]]

    drug_01 = pd.concat([drug_0, drug_1])
    drug_01.rename(columns={cat_name: new_col}, inplace=True)
    Feture_DrugWeighted.append(drug_01)

# %%
npi_col = FeaturesAll_pd[['npi']]

w_npi = []

for n, nx in enumerate(Feture_DrugWeighted):
      nggx = pd.merge(npi_col, nx.drop_duplicates(['npi']), on='npi', how='left')
      w_npi.append(nggx)

# %%
FeaturesAll_pd1 = FeaturesAll_pd

# %%
for wx in w_npi:
    col_n = wx.columns[1]
    FeaturesAll_pd1[col_n] = wx[col_n].values

wx = w_npi[0]
wx.columns[1]
col_n = wx.columns[1]

# %%
```

```
len(wx[col_n].values)
FeaturesAll_pd1.fillna(0)

# %%
new_col_all

# %%
FeaturesAll_pd1[new_col_all].describe()

# %%
FeaturesAll_pd1['drug_mean'] = FeaturesAll_pd1[new_col_all].mean(axis=1)

# %%
FeaturesAll_pd['drug_mean'] = FeaturesAll_pd['drug_mean'].map(lambda x: np.log10(x +
1.0))

# %%
FeaturesAll_pd1['drug_sum'] = FeaturesAll_pd1[new_col_all].sum(axis=1)
FeaturesAll_pd['drug_sum'] = FeaturesAll_pd['drug_sum'].map(lambda x: np.log10(x + 1.0))

# %%
FeaturesAll_pd1['drug_variance'] = FeaturesAll_pd1[new_col_all].var(axis=1)

# %%
FeaturesAll_pd1

# %%

# %%

# %%
df_train = FeaturesAll_pd1.ix[ix_train]
df_valid = FeaturesAll_pd1.ix[ix_valid]

df_train.fillna(0)
df_valid.fillna(0)

# %%
df_valid.columns

# %%
#Create the Specialty Weight
spec_dict =[]
spec_fraud_1 = df_train[df_train['is_fraud']==1]['Speciality']

# %%
from collections import Counter
counts = Counter(spec_fraud_1)
spec_dict =  dict(counts)
```

```
# %%

FeaturesAll_pd1['Spec_Weight'] = FeaturesAll_pd1['Speciality'].map(lambda x:
spec_dict.get(x, 0))

# %%
df_train = FeaturesAll_pd1.ix[ix_train]
df_valid = FeaturesAll_pd1.ix[ix_valid]

# %%
len(df_train[df_train['is_fraud'] == 1])

# %%
print(df_train.dtypes)

# %%
df_train.fillna(0)

# %%
df_valid.fillna(0)

# %%
numerical_features1 = numerical_features + ['drug_sum','Spec_Weight']

# %%
numerical_features1

# %%

# %%
positives=len(df_train[df_train['is_fraud'] == 1])
positives

# %%
dataset_size=len(df_train)
dataset_size

# %%
per_ones=(float(positives)/float(dataset_size))*100
per_ones

# %%
negatives=float(dataset_size-positives)
t=negatives/positives
t

# %%
BalancingRatio= positives/dataset_size
BalancingRatio
```

```
# %%
BalancingRatio= positives/dataset_size
BalancingRatio

# %%

# %%
X= df_train[numerical_features1].values
Y = df_train['is_fraud'].values
clf =  LogisticRegression(C=1e5, class_weight={0:1, 1:4000}, n_jobs=3)
clf.fit(X,Y)
y_p=clf.predict_proba(X)

# %%
params_0 = {'n_estimators': 100, 'max_depth': 8, 'min_samples_split': 3, 'learning_rate': 0.01}
params_1 = {'n_estimators': 500, 'max_depth': 10, 'min_samples_split': 5, 'class_weight' :
{0:1, 1:2514}, 'n_jobs':5}

scaler = StandardScaler()

clfs = [
    LogisticRegression(C=1e5,class_weight= {0:1, 1:2514}, n_jobs=5),

    GaussianNB(),

    ensemble.RandomForestClassifier(**params_1),

    ensemble.ExtraTreesClassifier(**params_1),

    ensemble.GradientBoostingClassifier(**params_0)

    ]

# %%
X_train = df_train[numerical_features1].values

y_train = df_train['is_fraud'].values

X_train = scaler.fit_transform(X_train)

X_valid = df_valid[numerical_features1].values
y_valid = df_valid['is_fraud'].values
X_valid_x= scaler.transform(X_valid)

# %%
prob_result = []
df_m = []
clfs_fited = []
for clf in clfs:
    print("%s:" %  clf.__class__.__name__)
```

```
    clf.fit(X_train,y_train)
    clfs_fited.append(clf)
    y_pred = clf.predict(X_valid_x)
    prob_pos  = clf.predict_proba(X_valid_x)[:, 1]
    prob_result.append(prob_pos)
    m = confusion_matrix(y_valid, y_pred)
    clf_score = brier_score_loss(y_valid, prob_pos, pos_label=y_valid.max())
    print("\tBrier: %1.5f" % (clf_score))
    print("\tPrecision: %1.5f" % precision_score(y_valid, y_pred))
    print("\tRecall: %1.5f" % recall_score(y_valid, y_pred))
    print("\tF1: %1.5f" % f1_score(y_valid, y_pred))
    print("\tauc: %1.5f" % roc_auc_score(y_valid, prob_pos))
    print("\tAccuracy: %1.5f\n" % accuracy_score(y_valid, y_pred))
    df_m.append(
        pd.DataFrame(m, index=['True Negative', 'True Positive'], columns=['Pred. Negative',
'Pred. Positive'])
        )

# %%
fpr, tpr, thresholds = roc_curve(y_valid, prob_result[2])

# %%
fpr, tpr, thresholds = roc_curve(y_valid, prob_result[2])
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, lw=1, label='ROC (area = %0.2f)' % roc_auc)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()

# %%
m

# %%
X_valid_x[0]

# %%
df_train

# %%
X_valid_x[1]

# %%
y_pred = clf.predict(X_valid_x)

# %%
y_pred
```

```
# %%
X_train[0]

# %%
feature_importance = clfs_fited[2].feature_importances_
# make importances relative to max importance
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)

# %%
feature_importance[sorted_idx]

# %%
features = [numerical_features1[ix] for ix in sorted_idx]
bardata = {"name":features[::-1], "importance percent":feature_importance[sorted_idx][::-1]}

# %%
plt.figure()

# Create plot title
plt.title("Feature Importance")

# Add bars
plt.bar(range(X.shape[1]), feature_importance[sorted_idx])

# Add feature names as x-axis labels
plt.xticks(range(X.shape[1]), features, rotation=90)

# Show plot
plt.show()

# %%
```

References

[1] Centres for Medicare & Medicaid Services (CMS). (2017). **Part D Prescriber Data CY 2017**. Retrieved from https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/PartD2017

[2] Office of Inspector General, U.S. Department of Health and Human Services. (2020). **LEIE Downloadable Databases**. Retrieved from https://oig.hhs.gov/exclusions/exclusions_list.asp

[3] Centres for Medicare & Medicaid Services (CMS). (2017). **Dataset Downloads**. Retrieved from https://www.cms.gov/OpenPayments/Explore-the-Data/Dataset-Downloads

[4] Dey, L., & Ghosh, S. (2019). **Big Data Analytics in Healthcare: A Review of Techniques and Applications**. *International Journal of Computer Applications*, 178(26), 1-7.

[5] Rahman, A., & Tazeen, M. (2020). **Healthcare Fraud Detection Using Machine Learning Techniques**. *Journal of Healthcare Engineering*, 2020.

[6] Liu, Y., & Xie, L. (2021). **Anomaly Detection in Healthcare Using Machine Learning: A Review**. *Health Information Science and Systems*, 9(1), 1-18.

[7] Zhang, J., & Wang, Y. (2018). **Using Machine Learning Algorithms for Healthcare Fraud Detection: A Systematic Review**. *International Journal of Health Policy and Management*, 7(1), 21-34.

[8] Luo, W., & Zhou, X. (2020). **A Survey on Machine Learning Techniques for Medical Fraud Detection**. *IEEE Access*, 8, 204185-204197.