Project Summary

Batch details	PGP - Data Science and Engineering. PGPDSE-FT Online Nov21
Team members	1. Varun M 2. E B Raghul 3. Aysha Siddika 4. Pradeep Joel X 5. Chilakapati Sadhana
Domain of Project	Financial and Risk Analytics
Proposed project title	Healthcare Provider Fraud Detection Analysis
Group Number	Team 02
Team Leader	Varun M
Mentor Name	Mr Animesh Tiwari

Date : 13.07.2022

Mr Animesh Tiwari

Varun M

Signature of the Mentor

Signature of the Team Leader

Table of Contents

Healthcare Provider Fraud Detection Analysis

What is Healthcare Fraud ?

Fraud is defined as any deliberate and dishonest act committed with the knowledge that it may result in an unauthorized benefit to the perpetrator or someone else who is similarly not entitled to the benefit. One type of fraud is healthcare fraud. In this section, we will analyse and detect "Healthcare Provider Fraud," which occurs when a provider fills out all of the details and submits a claim on behalf of the beneficiary.

One of the most serious issues confronting Medicare right now is provider fraud. Healthcare fraud is an organized crime in which peers of providers, physicians, and beneficiaries collaborate to file fraudulent claims. According to US law, an insurance company must pay a legitimate healthcare claim within 30 days. As a result, there is very little time to thoroughly investigate this. Insurance companies are the most vulnerable institutions as a result of these unethical practices. According to the government, total Medicare spending has increased exponentially as a result of Medicare claim fraud.

Healthcare fraud and abuse can take many different forms. Some of the most common types of provider fraud include: a) billing for services that were never provided. b) Submitting a claim for the same service twice. c) Falsifying the service provided. d) Charging for a more complicated or costly service than was actually provided. e) Billing for a covered service when the service was not actually provided.

Business Objective

Our goal is to predict whether a provider is potentially fraudulent or to assign a probability score to that provider's fraudulent activity, as well as to discover the reasons for it in order to avoid financial loss. Depending on the probability score and falsified reasons, the insurance company may accept or deny the claim or initiate an investigation into that provider. Discover the crucial characteristics that are the causes of potentially fraudulent providers. For example, if the claim amount is high for a patient with a low risk score, it is suspicious. Not only is financial loss a major concern, but so is protecting the healthcare system so that legitimate patients can receive quality and safe care.

Business Constraints

Cost of misclassification is very high. False Negative and False Positive should be as low as possible.If fraudulent providers are predicted as non-fraudulent (False Negative) it is a huge financial loss to the insurer and if legitimate providers are predicted as fraudulent (False Positive) it will cost for investigation and also it's a matter of reputation of the agency.

1. Model interpretability is very important because the agency or insurer should justify that fraudulent activity and may need to set up a manual investigation. It should not be a black box type prediction.
2. Insurer should pay the claim amount to the provider for legitimate claims within 30 days. So, there are no such strict latency constraints but it should not take more than a day because depending on the output of the model the agency may need to set up an investigation.

2. Objective

Identify fraudulent claims:

1. To protect the insurance providers from great loss due to this bad practice.
2. To protect the healthcare system so that they can provide quality and safe care to legitimate patients.
3. To reduce the cost of healthcare services.

Data Description

Train_Provider Data

It consists of provider numbers and corresponding whether this provider is potentially fraudulent. Provider ID is the primary key in that table.

Test_Provider Data

It consists of only the provider number. We need to predict whether these providers are potential fraud or not.

Outpatient Data (Train and Test)

It consists of the claim details for the patients who were not admitted into the hospital, who only visited there. Important columns are explained below.

BeneID: It contains the unique id of each beneficiary i.e. patients.

ClaimID: It contains the unique id of the claim submitted by the provider.

ClaimStartDt: It contains the date when the claim started in yyyy-mm-dd format.

ClaimEndDt: It contains the date when the claim ended in yyyy-mm-dd format.

Provider: It contains the unique id of the provider.

InscClaimAmtReimbursed: It contains the amount reimbursed for that particular claim.

AttendingPhysician: It contains the id of the Physician who attended the patient.

OperatingPhysician: It contains the id of the Physician who operated on the patient.

OtherPhysician: It contains the id of the Physician other than AttendingPhysician and Operating Physician who treated the patient.

ClmDiagnosisCode: It contains codes of the diagnosis performed by the provider on the patient for that claim.

ClmProcedureCode: It contains the codes of the procedures of the patient for treatment for that particular claim.

DeductibleAmtPaid: It consists of the amount by the patient. That is equal to Total_claim_amount — Reimbursed_amount.

Inpatient Data (Train and Test)

It consists of the claim details for the patients who were admitted into the hospital. So, it consists of 3 extra columns: Admission date, Discharge date, and Diagnosis Group code.

AdmissionDt: It contains the date on which the patient was admitted into the hospital in yyyy-mm-dd format.

DischargeDt: It contains the date on which the patient was discharged from the hospital in yyyy-mm-dd format.

DiagnosisGroupCode: It contains a group code for the diagnosis done on the patient.

Beneficiary Data (Train and Test)

This data contains beneficiary KYC details like DOB, DOD, Gender, Race, health conditions (Chronic disease if any), State, Country they belong to, etc. Columns of this dataset are explained below.

BeneID: It contains the unique id of the beneficiary.

DOB: It contains the Date of Birth of the beneficiary.

DOD: It contains the Date of Death of the beneficiary if the beneficiary id deal else null.

Gender, Race, State, Country: It contains the Gender, Race, State, Country of the beneficiary.

RenalDiseaseIndicator: It contains if the patient has existing kidney disease.

ChronicCond: The columns started with “ChronicCond_” indicates if the patient has existing that particular disease. Which also indicates the risk score of that patient.

IPAnnualReimbursementAmt: It consists of the maximum reimbursement amount for hospitalization annually.

IPAnnualDeductibleAmt: It consists of a premium paid by the patient for hospitalization annually.

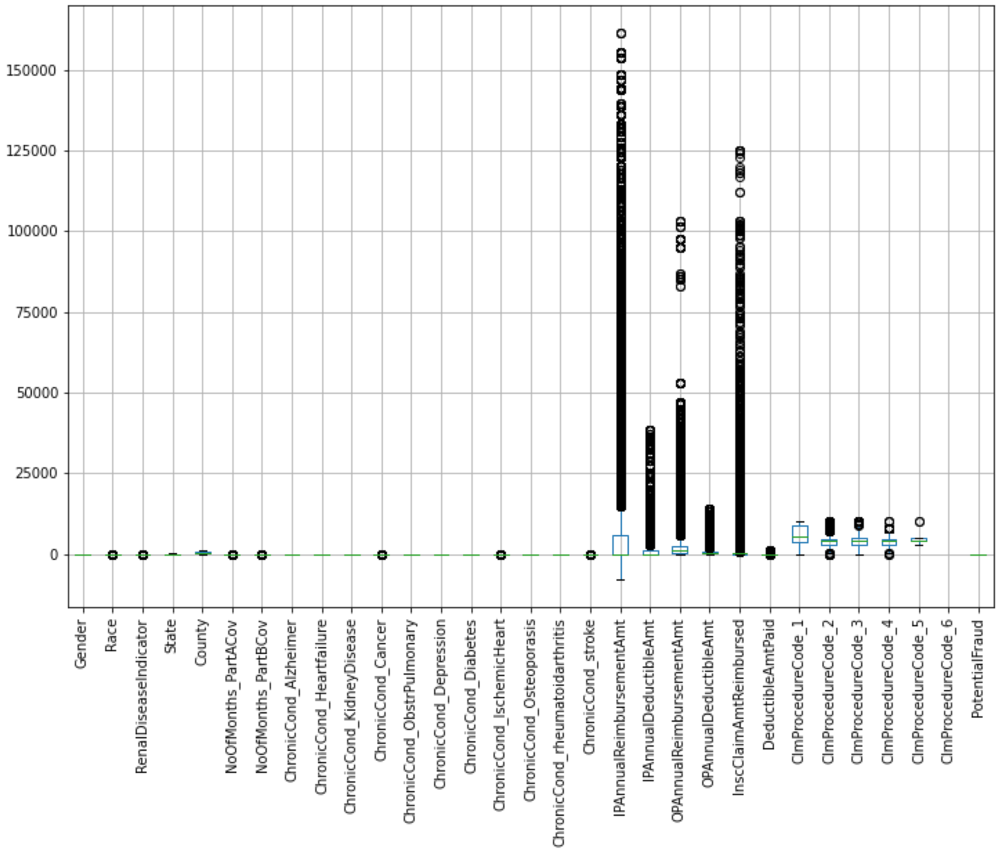
OPAnnualReimbursementAmt: It consists of the maximum reimbursement amount for outpatient visits annually.

OPAnnualDeductibleAmt: It consists of a premium paid by the patient for outpatient visits annually.

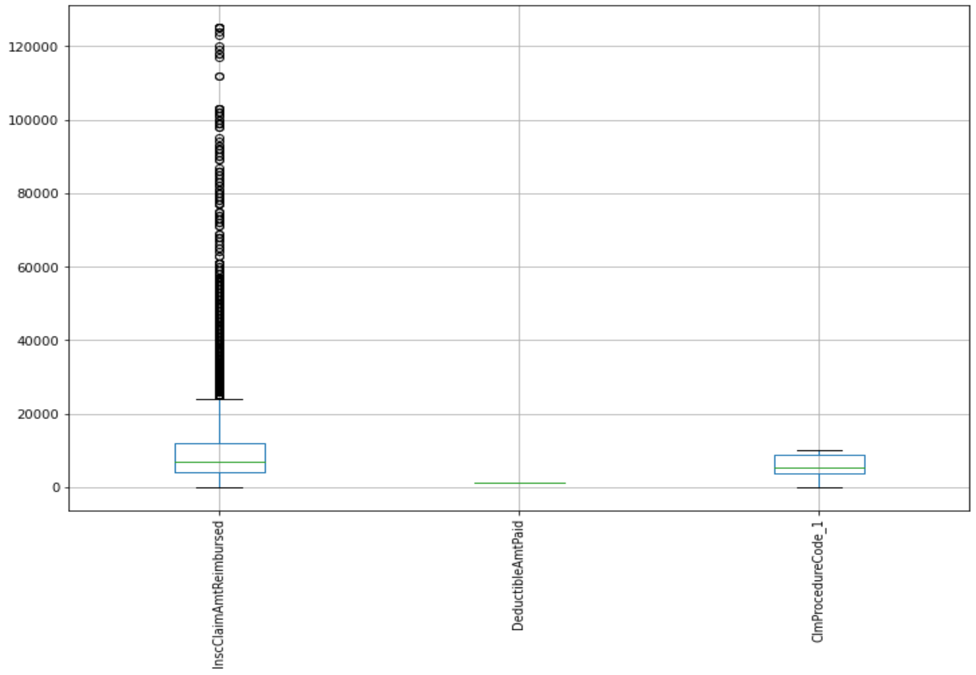
Data Pre-processing and Data Preparation

Handling Outliers (Outlier treatment):

1. In train beneficiary dataframe , outliers present in the following columns IPAnnualReimbursementAmt, IPAnnualDeductibleAmt, OPAnnualReimbursementAmt, OPAnnualDeductibleAmt were converted into missing values and was treated using MICE (Multiple Imputation by Chained Equations) algorithm.



2. In Inpatient dataframe, outliers present in the following columns InscClaimAmtReimbursed, DeductibleAmtPaid, ClmProcedureCode_1 were converted into missing values and was treated using MICE (Multiple Imputation by Chained Equations) algorithm.
3. In the Outpatient dataframe, Outliers present were converted into missing values.



Handling Missing Values:

1. In train beneficiary, Inpatient, Outpatient data frame the columns with missing values less than 80% are treated with MICE (Multiple Imputation by Chained Equations) algorithm.

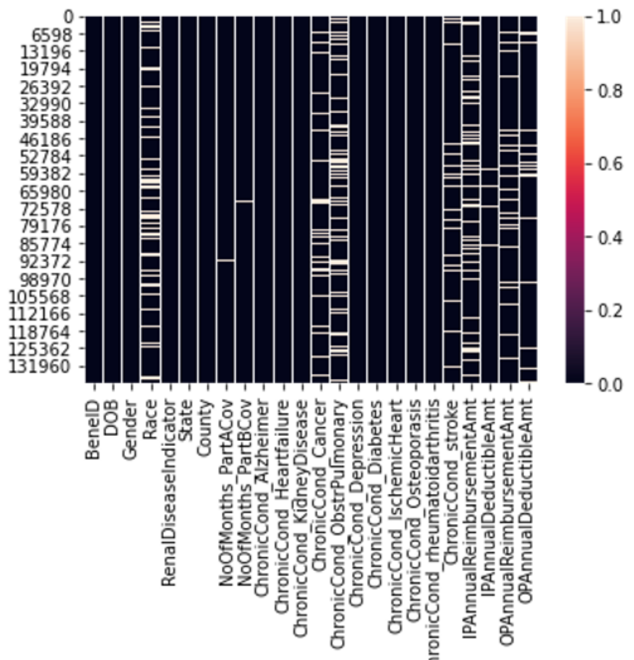
    - Multiple Imputation by Chained Equations (MICE) is a robust, informative method of dealing with missing data in datasets

2. In the train beneficiary dataframe, feature DOD has more than 95% of missing values hence it is dropped.
3. In Inpatient dataframe, feature dropped are ClmProcedureCode_2, ClmProcedureCode_3, ClmProcedureCode_4, ClmProcedureCode_5, ClmProcedureCode_6, OtherPhysician since it has more than 90% of missing values.
4. In Inpatient dataframe, AttendingPhysician, OperatingPhysician, ClmDiagnosisCode_1, ClmDiagnosisCode_2, ClmDiagnosisCode_3, ClmDiagnosisCode_4 ,ClmDiagnosisCode_5 , ClmDiagnosisCode_6, ClmDiagnosisCode_7, ClmDiagnosisCode_8, ClmDiagnosisCode_9 are

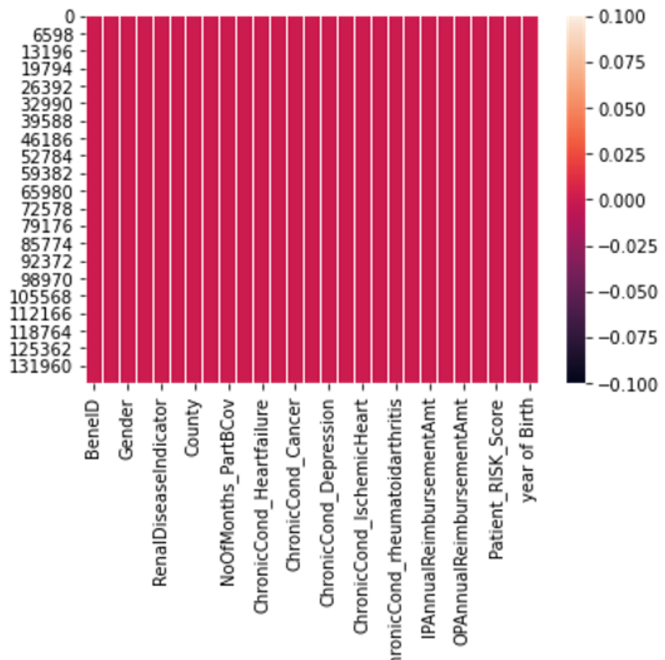
categorical columns and hence missing values detected were treated with most frequent observation(Mode).

5. Since it has more than 80% of missing values in Outpatient dataframe, feature dropped are ClmDiagnosisCode_5, ClmDiagnosisCode_6, ClmDiagnosisCode_7, ClmDiagnosisCode_8, ClmDiagnosisCode_9, ClmProcedureCode_1,ClmProcedureCode_2, ClmProcedureCode_3, ClmProcedureCode_4, ClmProcedureCode_5, ClmProcedureCode_6 , OtherPhysician.
6. In Outpatient dataframe, ClmAdmitDiagnosisCode , AttendingPhysician, OperatingPhysician, ClmDiagnosisCode_1, ClmDiagnosisCode_2, ClmDiagnosisCode_3, ClmDiagnosisCode_4 are categorical columns and hence missing values detected were treated with most frequent observation(Mode).

Before missing value treatment



After missing value treatment:



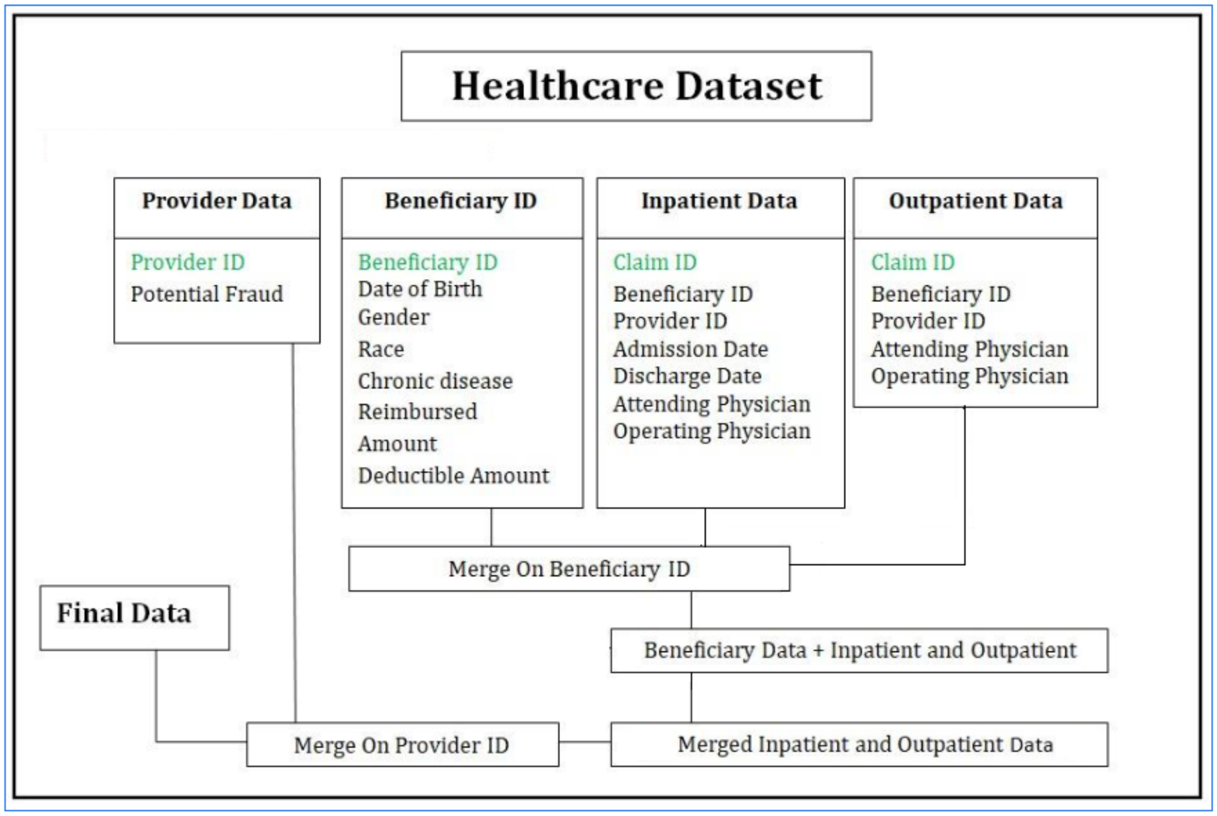
Scaling:

1. Because the numerical column in the dataset deals with capital, scaling our research may be hampered.
2. Because the category column in the dataset has several labels, scaling may not be appropriate.
3. If additional analysis or hypothesis are required, scaling procedures will be applied.

Merge of the Datasets:

We have 4 different datasets, which are interconnected by foreign keys. We need to merge them using the foreign keys to get an overall dataset. Below is a brief overview of the dataset.

1. Merge Inpatient and Outpatient data based on ClaimID
2. Merge beneficiary details with inpatient and outpatient data on BeneID.
3. Merge provider details with previously merged data on ProviderID.



Overall Representation of the Dataset

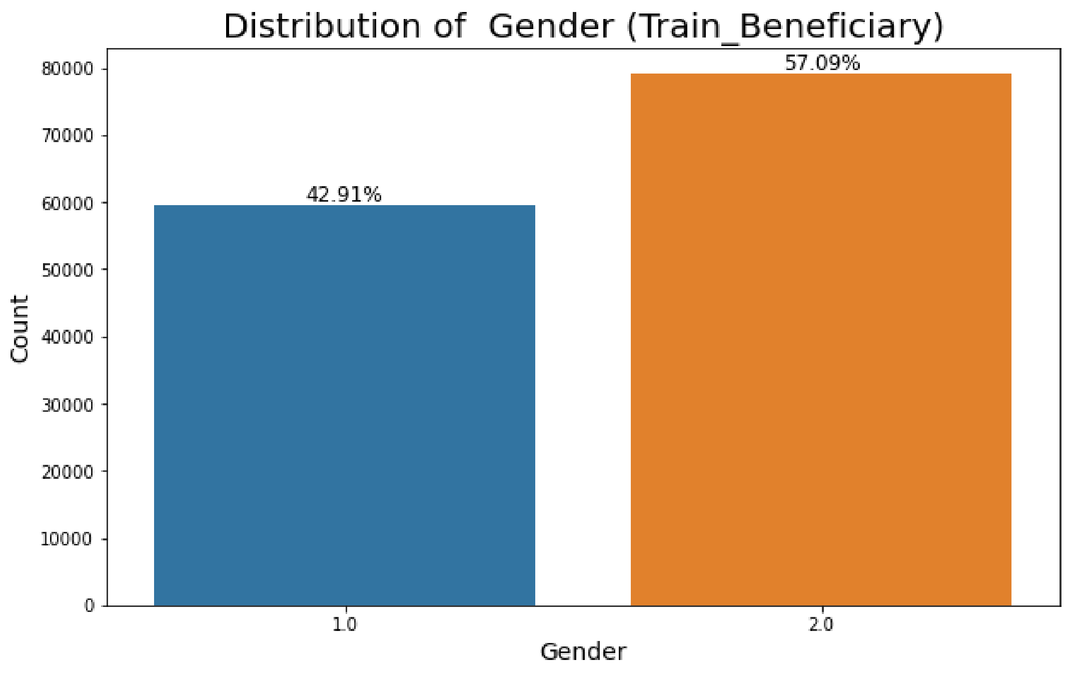
Feature Engineering

1. We are creating a new feature Patient Risk Score , year , Total AnnualReimbursementAmt in the beneficiary dataframe.
2. We are creating a new feature Hospitalization_Duration,Claim_Period, ExtraClaimDays ClaimStart_Year , ClaimStart_Month , ClaimEnd_Year, ClaimEnd_Month in Inpatient dataframe.
3. We are creating a new feature ClaimStart_Year , ClaimStart_Month , ClaimEnd_Year, ClaimEnd_Month in the Inpatient dataframe.

Exploratory Data Analysis & Business Insights

Univariate Analysis:

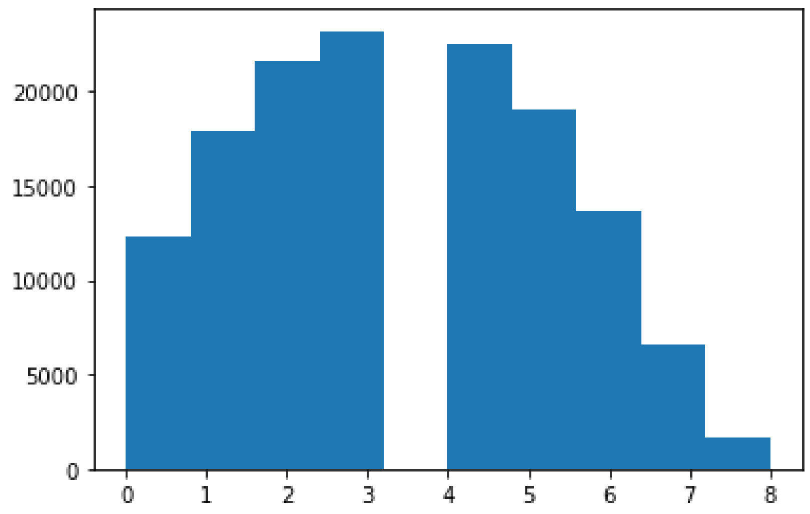
Distribution of Gender (Beneficiary Data)



Inference:

1. Gender 1.0 is 42.91% Gender 2.0 is 57.09%

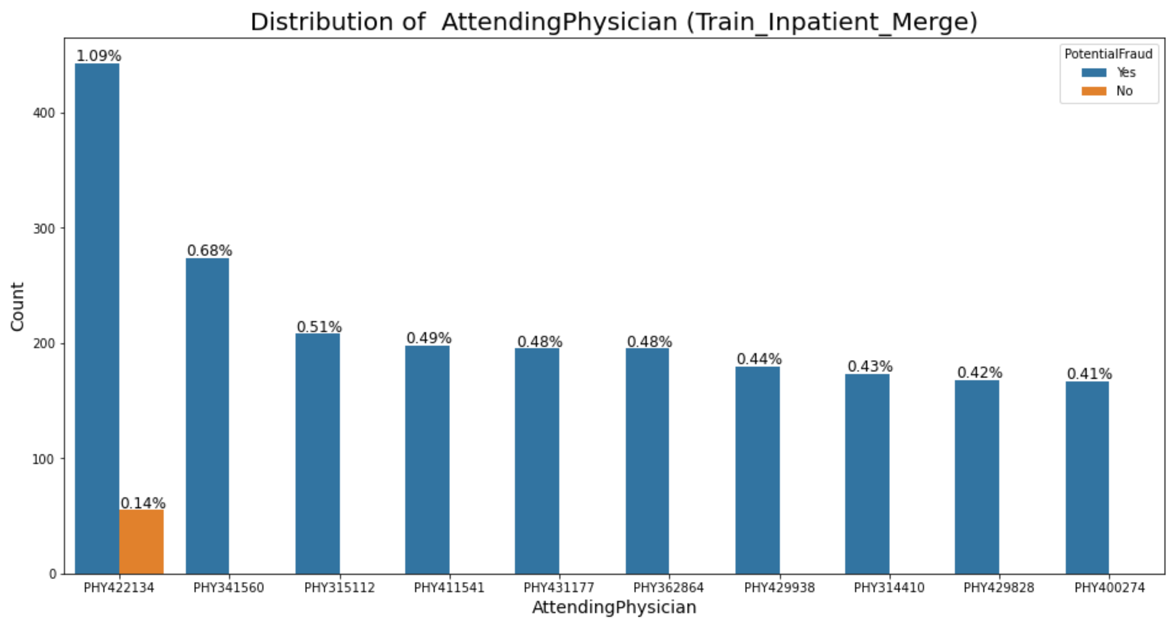
Distribution of Patient Risk Score (Beneficiary Data)



Inference:

1. Risk Score of 3 and 4 has more number of patients
2. There are 12278 people with no risk score which means they are healthy

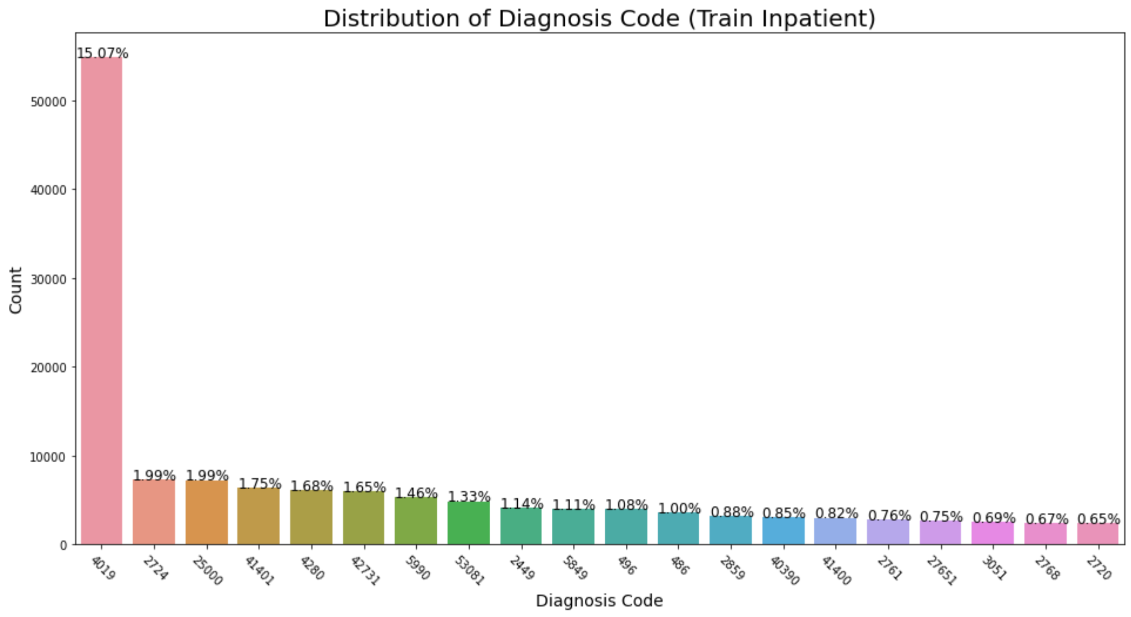
Distribution of Attending physician (Inpatient Data)



Inference:

1. Attending Physician PHY422134 has be noted with a Potential Fraud of 1.099%

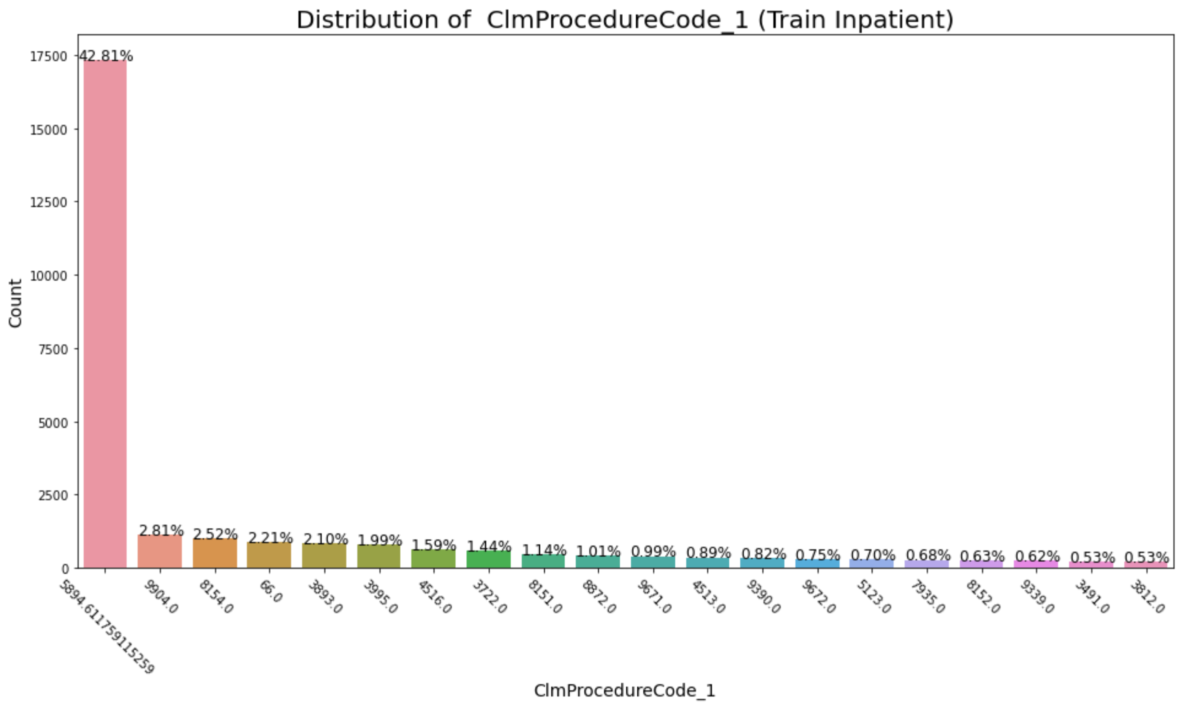
Distribution of Diagnosis Code (Inpatient Data)



Inference :

1. 4019, 2724, 25000, 41401, 4280 are the top 5 diagnosis codes in terms of number of diagnoses done.
2. The 4019 test is 15.07% of the total diagnosis.

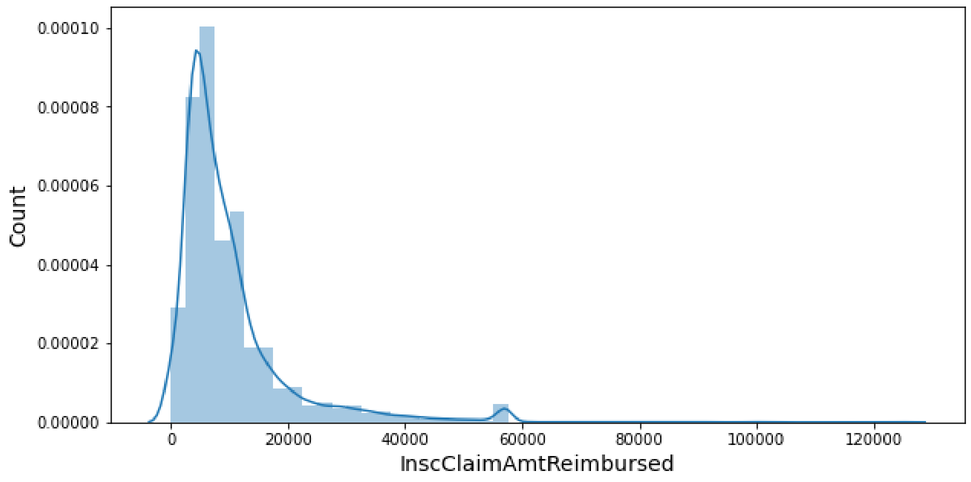
Distribution of ClmProcedureCode_1 (Inpatient Data)



Inference :

ClmProcedureCode_1 5894.6117 has the maximum PotentialFraud of 24.58%

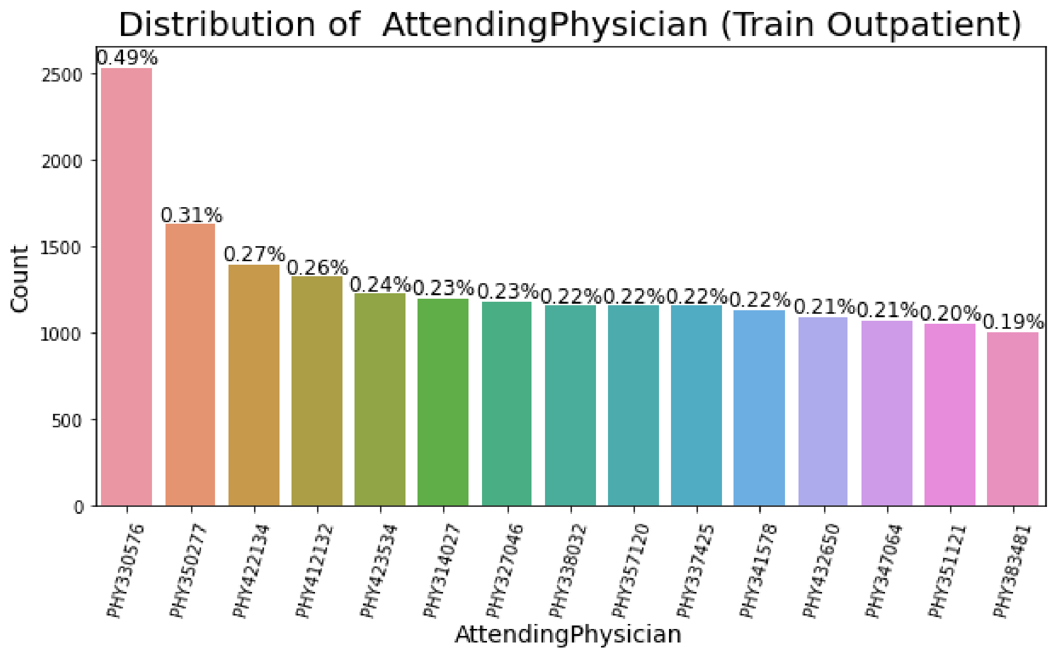
Plot of InscClaimAmtReimbursed (Outpatient Data)



Inference :

1. Total InscClaimAmtReimbursed for outpatient is 148246120
2. For very few claims InscClaimAmtReimbursed are very high.

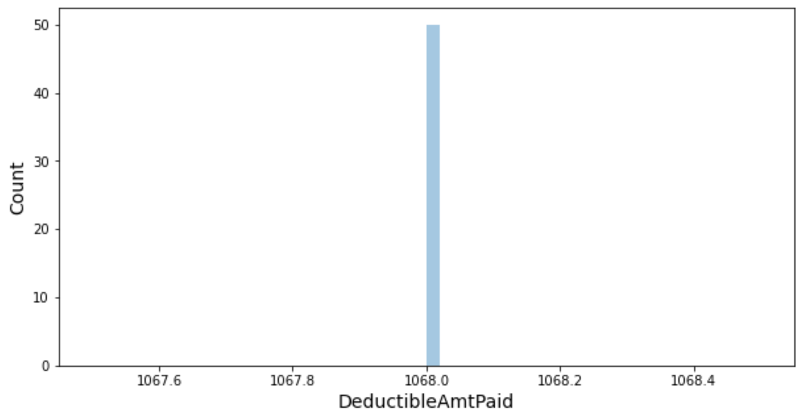
Plot of AttendingPhysician (Outpatient Data)



Inference:

1. PHY330576 attended 0.49% of the total patients.

Plot of Distribution of DeductibleAmtPaid (Outpatient Data)

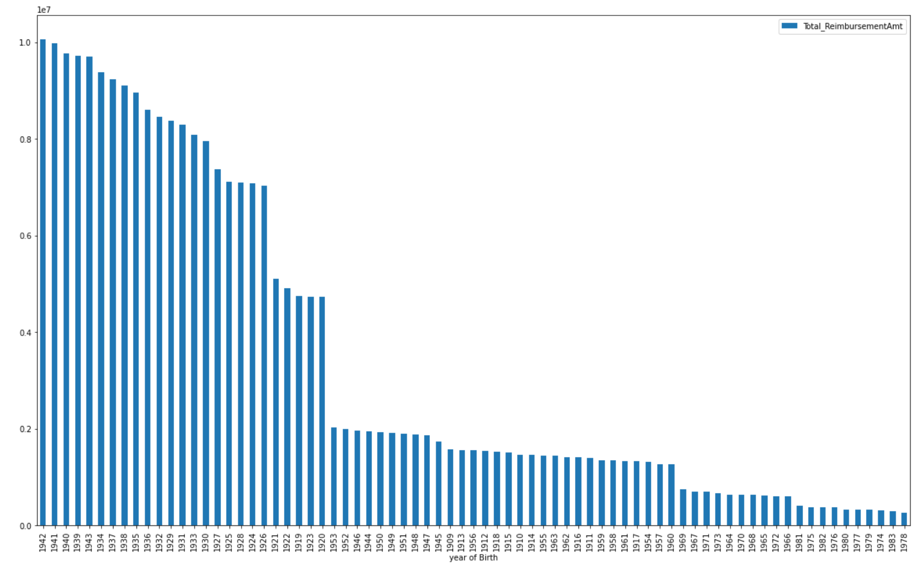


Inference :

1. Total DeductibleAmtPaid for outpatient is 1438912
2. DeductibleAmtPaid value is 1068 only

Bivariate Analysis:

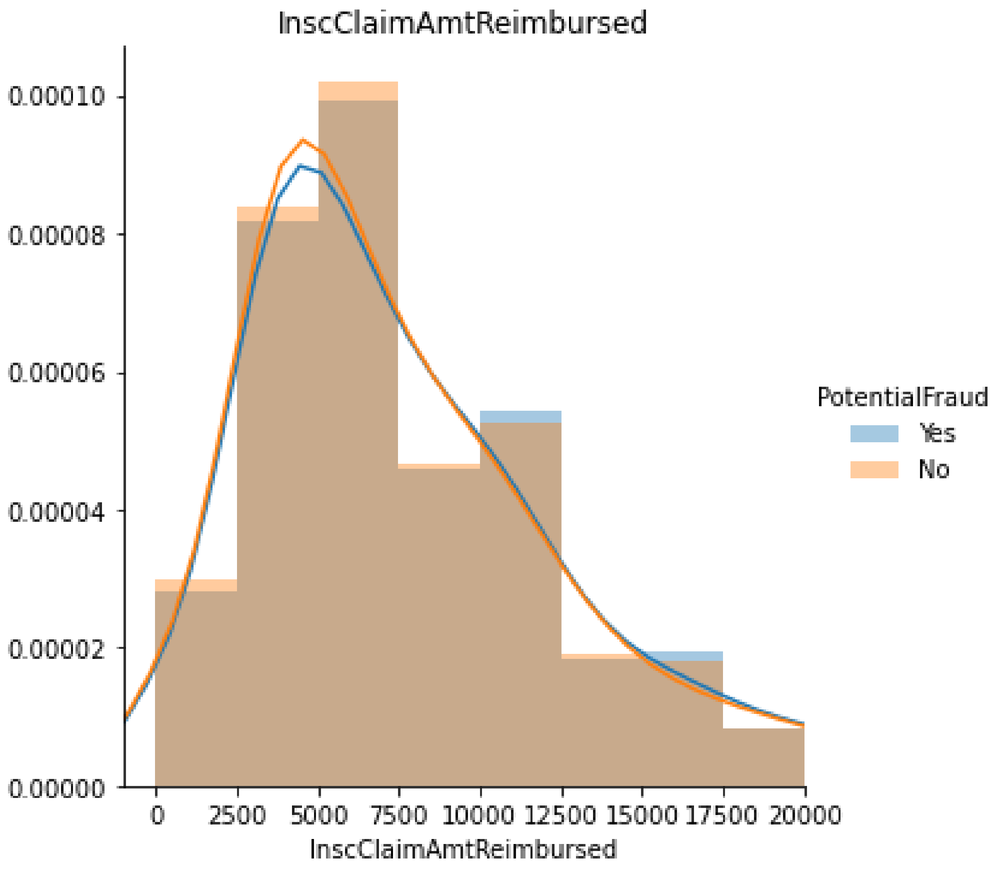
Plotting Year of birth Vs Total ReImbursementAmt (Beneficiary Data)



Inference:

1. The Beneficiaries born in the year 1942 has total reimbursement amount of 13091419 which is maximum out of all years of birth using pivot table found this inference

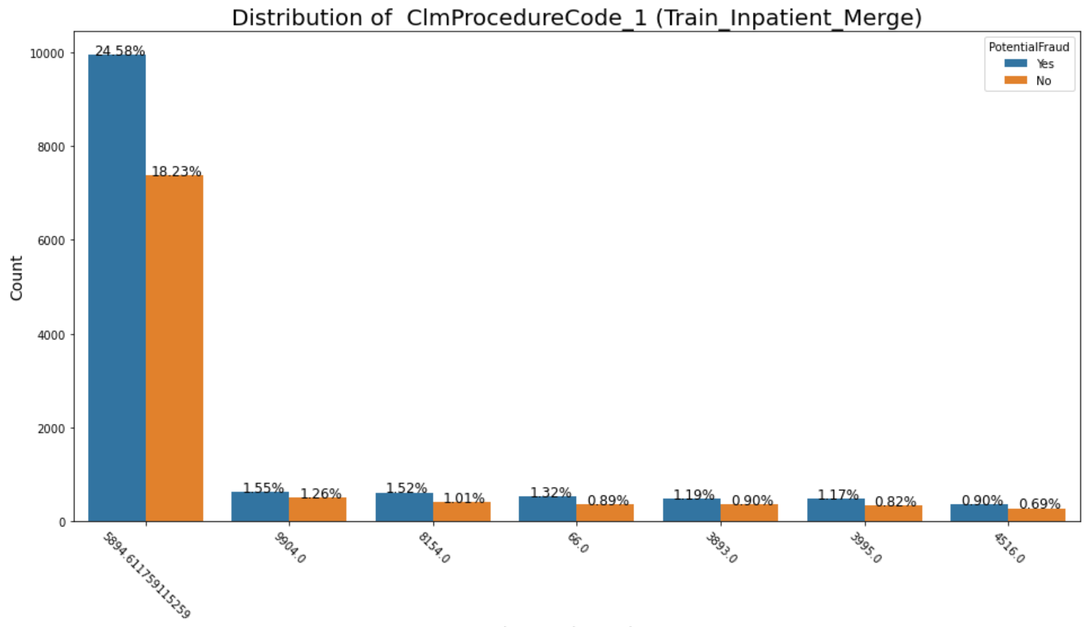
Plotting PotentialFraud Vs InscClaimAmtReimbursed (Inpatient Data)



Inference:

1. From the PDF and histogram of claim amount reimbursed we can observe that when the claim amount is less, the number of fraud claims are much higher compared to legitimate claims.
2. For higher claim amounts also fraudulent count is little higher.

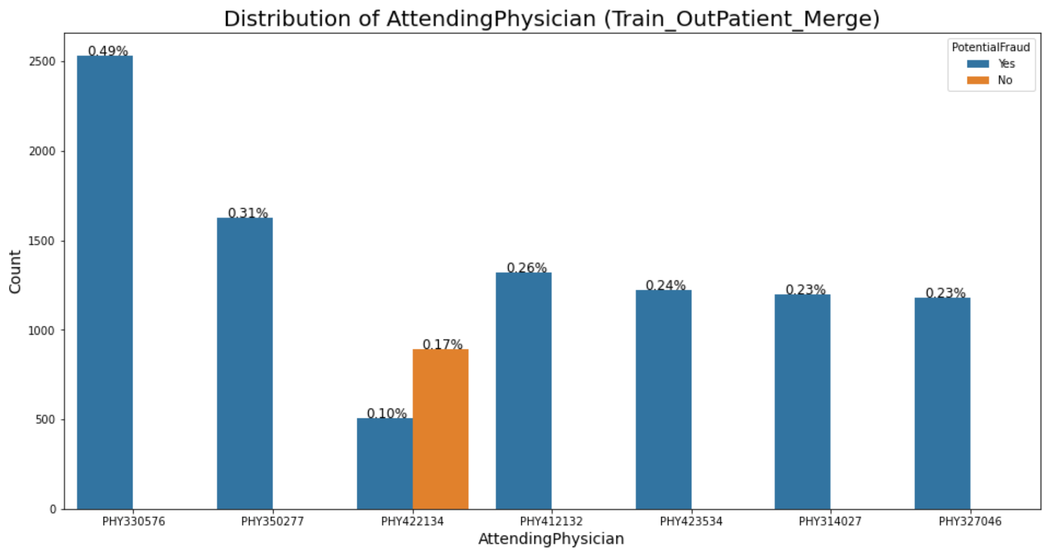
Countplot for ClmProcedureCode_1 and PotentialFraud (Inpatient Data)



Inference :

ClmProcedureCode_1 5894.6117 has the maximum PotentialFraud of 24.58%

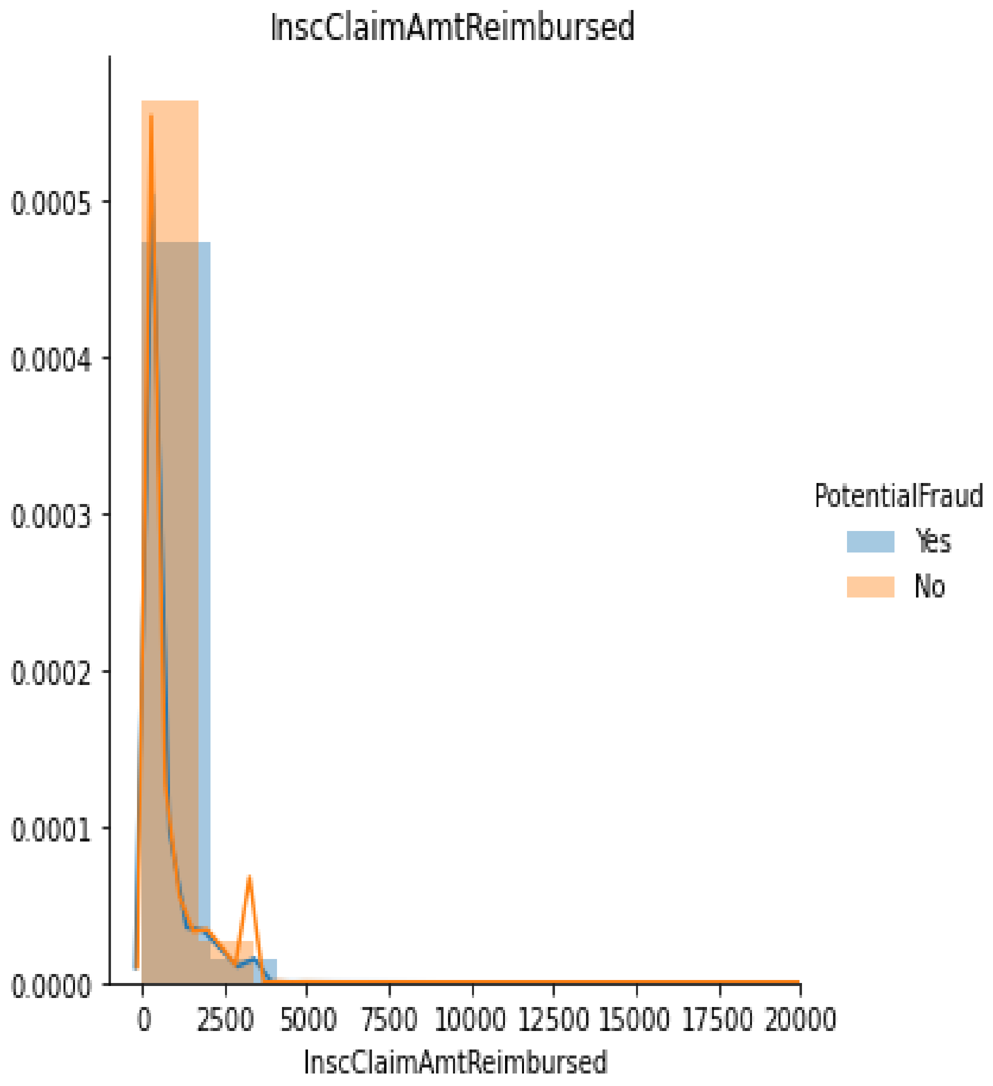
Plot of Distribution of AttendingPhysician Vs PotentialFraud (Outpatient Data)



Inference :

1.AttendingPhysician PHY330576 has be noted with a PotentialFraud of 0.49%

Plot of InscClaimAmtReimbursed (Outpatient Data)

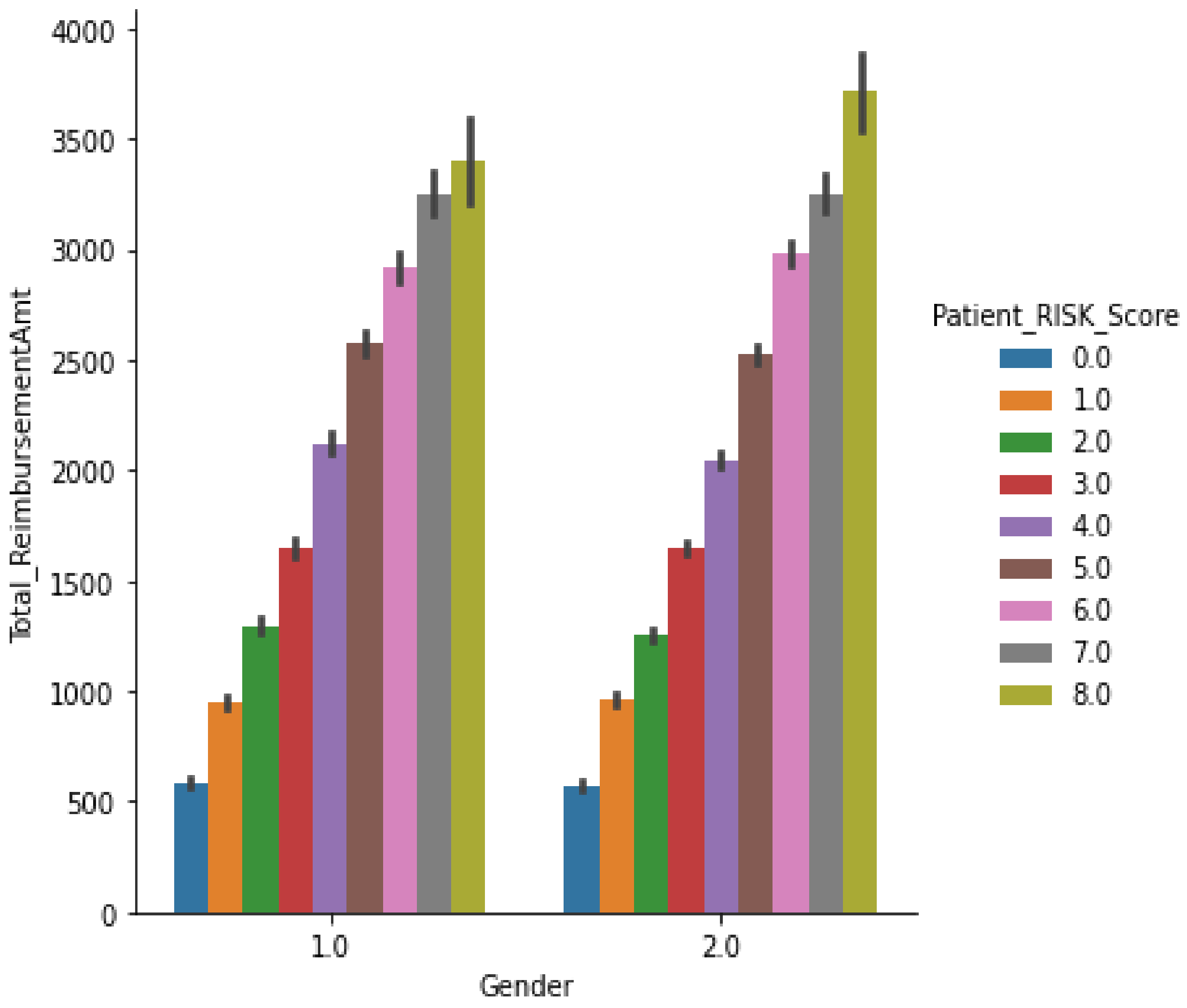


Inference:

1. From the PDF and histogram of claim amount reimbursed we can observe that when the claim amount is less, the number of fraud claims are much higher compared to legitimate claims.
2. For higher claim amounts also fraudulent count is little higher.

Multivariate Analysis:

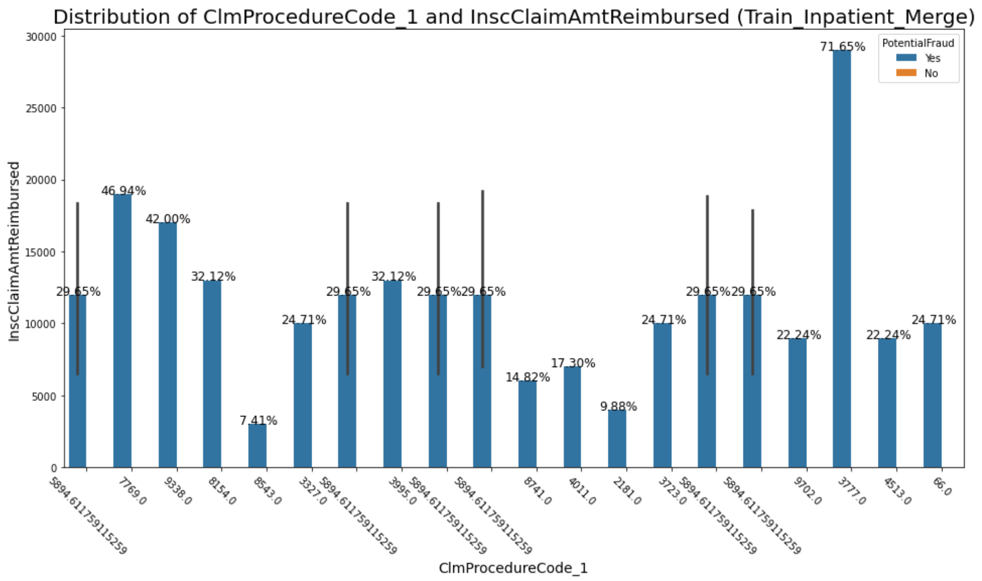
Bar Plot for Gender and Total ReImbursementAmt against **Patient risk scores** (**Beneficiary Data)**



Inference:

1. Gender 2 has more Total Reimbursement amt than Gender 1 for High risk score patients

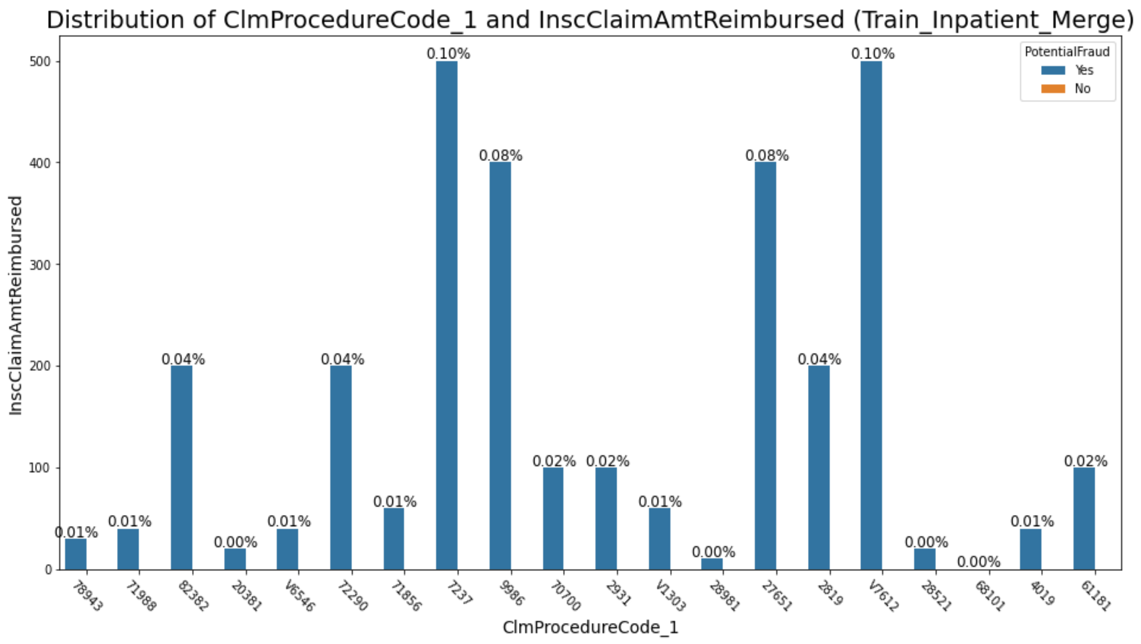
Barplot for ClmProcedureCode_1 and InscClaimAmtReimbursed against PotentialFraud (Inpatient Data)



Inference :

1. ClmProcedureCode_1 number 3777.0 has 71.65% of InscClaim Amount Reimbursed
2. ClmProcedureCode_1 number 5894.611 is the procedure done by most of the patients whereas it has only 29.65% of InscClaim Amount Reimbursed

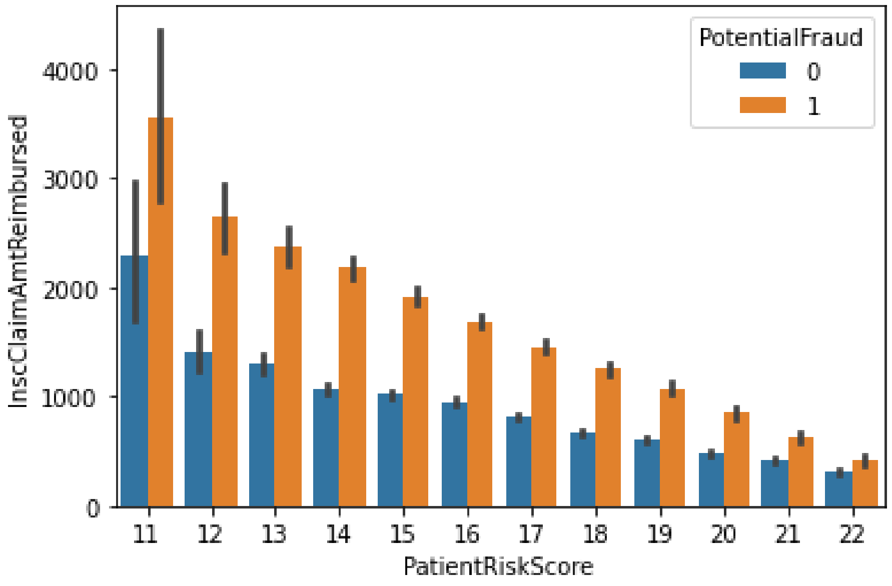
Plot of ClmProcedureCode_1 and InscClaimAmtReimbursed (Outpatient Data)



Inference:

1. ClmProcedureCode_1 number 7237 and V7612 has 0.10% of InscClaim Amount Reimbursement

Plot of PatientRiskScoreand InscClaimAmtReimbursed



Inference:

1. The Patient Risk Score 11-15 has the max potential fraud with respect to InscClaimAmtReimbursed

Business Insights

Misclassification has a high cost. False Negative and False Positive should be kept to a minimum. If fraudulent providers are predicted to be non-fraudulent (False Negative), the insurer suffers a significant financial loss, if legitimate providers are predicted to be fraudulent (False Positive), the investigation costs money, and the agency's reputation suffers.

Model interpretability is critical because the agency or insurer must justify the fraudulent activity and may be required to conduct a manual investigation. It must not be a black box prediction.

For legitimate claims, the insurer should pay the claim amount to the provider within 30 days. So there are no strict latency constraints, but it should not take more than a day because the agency may need to set up an investigation based on the model's output.

Machine Learning Modelling (Basic & Hyper parameters tuning)

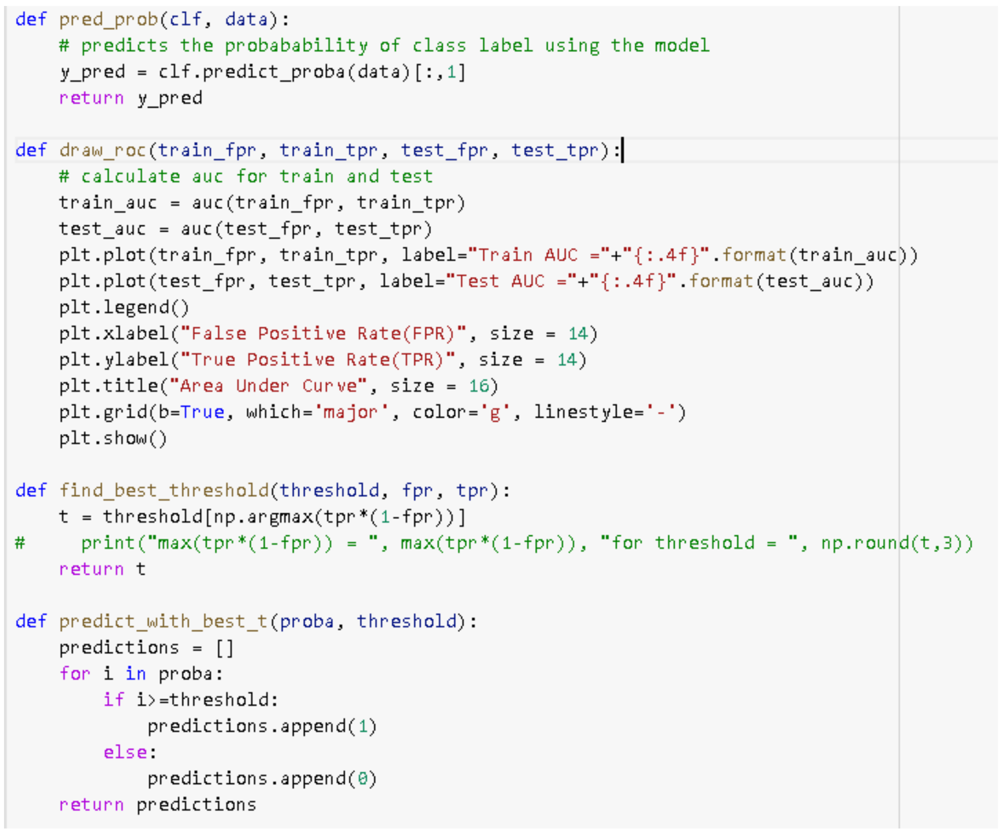
Modelling for fraud detection

Statistical modelling for fraud detection are generally classified into 2 categories.

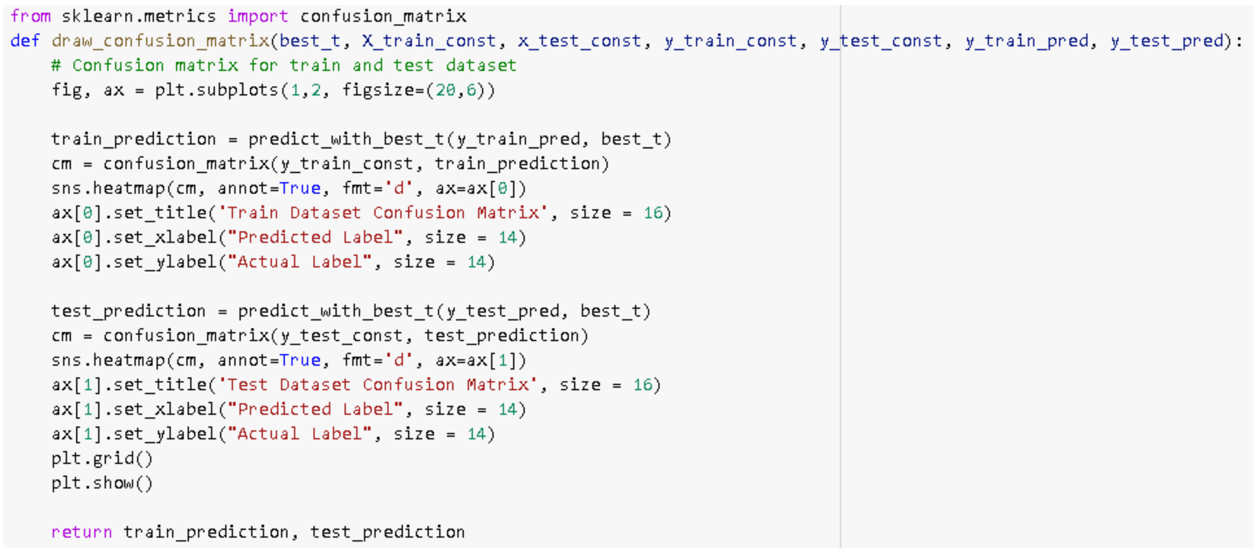
1. **Supervised methods:** This is used when labelled data (either fraud or legitimate) is available for training the model.
2. **Unsupervised methods:** This is used when we get only unlabeled data for training.

Now our dataset is ready. We need to try different models, first cut models along with ensemble models and validate the performance of every model. Based on the performance of the validation data we need to pick the best one for deployment.

- First, define some functions to validate every model, which will draw the ROC curve and confusion matrix.
- Then creating a generalized function to create a dataframe containing the scores for the models.

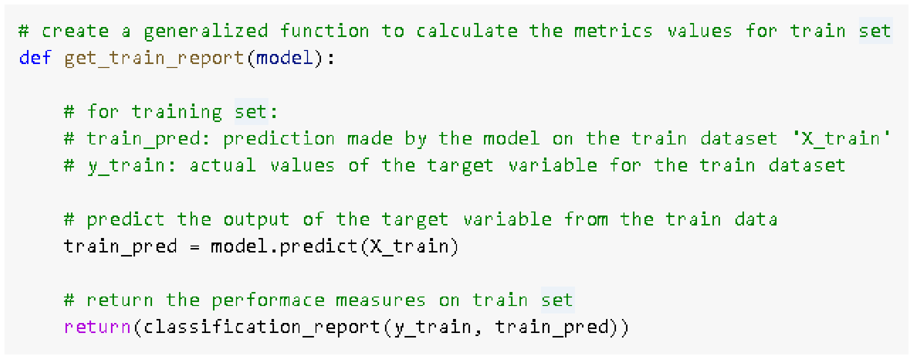


```
def pred_prob(clf, data):
    # predicts the probabability of class label using the model
    y_pred = clf.predict_proba(data)[:,1]
    return y_pred

def draw_roc(train_fpr, train_tpr, test_fpr, test_tpr):
    # calculate auc for train and test
    train_auc = auc(train_fpr, train_tpr)
    test_auc = auc(test_fpr, test_tpr)
    plt.plot(train_fpr, train_tpr, label="Train AUC ="+"{:.4f}".format(train_auc))
    plt.plot(test_fpr, test_tpr, label="Test AUC ="+"{:.4f}".format(test_auc))
    plt.legend()
    plt.xlabel("False Positive Rate(FPR)", size = 14)
    plt.ylabel("True Positive Rate(TPR)", size = 14)
    plt.title("Area Under Curve", size = 16)
    plt.grid(b=True, which='major', color='g', linestyle='-')
    plt.show()

def find_best_threshold(threshold, fpr, tpr):
    t = threshold[np.argmax(tpr*(1-fpr))]
#     print("max(tpr*(1-fpr)) = ", max(tpr*(1-fpr)), "for threshold = ", np.round(t,3))
    return t

def predict_with_best_t(proba, threshold):
    predictions = []
    for i in proba:
        if i>=threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

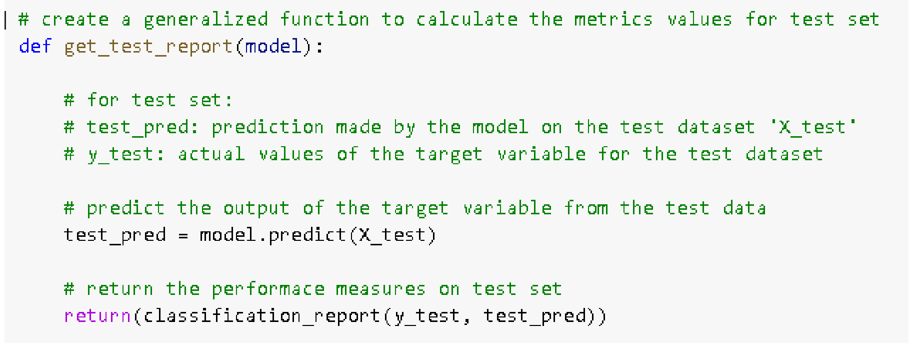


```python
from sklearn.metrics import confusion_matrix
def draw_confusion_matrix(best_t, X_train_const, x_test_const, y_train_const, y_test_const, y_train_pred, y_test_pred):
    # Confusion matrix for train and test dataset
    fig, ax = plt.subplots(1,2, figsize=(20,6))

    train_prediction = predict_with_best_t(y_train_pred, best_t)
    cm = confusion_matrix(y_train_const, train_prediction)
    sns.heatmap(cm, annot=True, fmt='d', ax=ax[0])
    ax[0].set_title('Train Dataset Confusion Matrix', size = 16)
    ax[0].set_xlabel("Predicted Label", size = 14)
    ax[0].set_ylabel("Actual Label", size = 14)

    test_prediction = predict_with_best_t(y_test_pred, best_t)
    cm = confusion_matrix(y_test_const, test_prediction)
    sns.heatmap(cm, annot=True, fmt='d', ax=ax[1])
    ax[1].set_title('Test Dataset Confusion Matrix', size = 16)
    ax[1].set_xlabel("Predicted Label", size = 14)
    ax[1].set_ylabel("Actual Label", size = 14)
    plt.grid()
    plt.show()

    return train_prediction, test_prediction
```

- Create a generalized function to calculate the metrics for the train and the test set.



```python
# create a generalized function to calculate the metrics values for train set
def get_train_report(model):

    # for training set:
    # train_pred: prediction made by the model on the train dataset 'X_train'
    # y_train: actual values of the target variable for the train dataset

    # predict the output of the target variable from the train data
    train_pred = model.predict(X_train)

    # return the performace measures on train set
    return(classification_report(y_train, train_pred))
```



```python
# create a generalized function to calculate the metrics values for test set
def get_test_report(model):

    # for test set:
    # test_pred: prediction made by the model on the test dataset 'X_test'
    # y_test: actual values of the target variable for the test dataset

    # predict the output of the target variable from the test data
    test_pred = model.predict(X_test)

    # return the performace measures on test set
    return(classification_report(y_test, test_pred))
```

ML Formulation:

Build a binary classification model based on the claims filled by the provider along with Inpatient data, Outpatient data, Beneficiary details to predict whether the provider is potentially fraudulent or not.

Below are the different approaches that we will follow in this study.

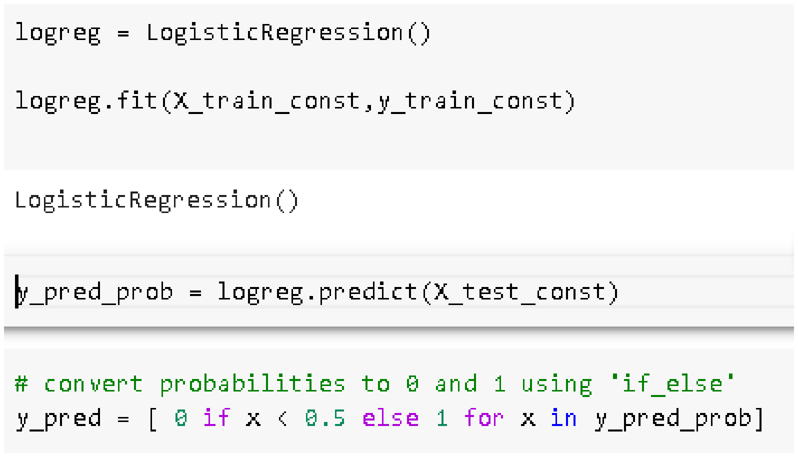
Approach 1:

a. Split the data into Train and Validation (80:20)

b. Use Logistic Regression, Decision Tree, Random Forest,Support Vector Classifier, and Naive Bayes on the datasets. Pick the best model based on the performance score.

Basic Logistic Full Model :

Initially we tried the base Logistics model which gives a reasonable Accuracy and F1 score.

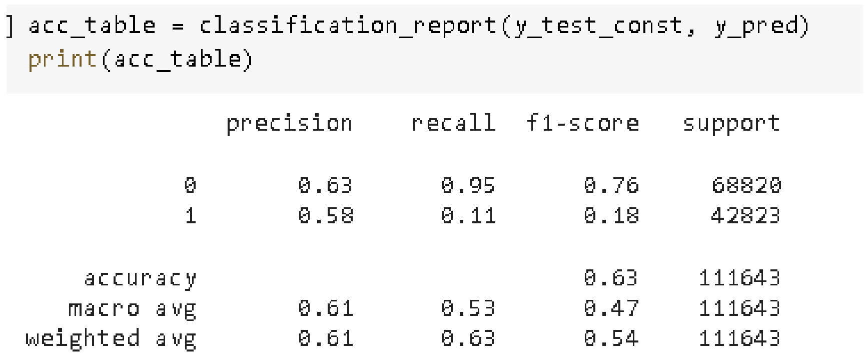


```
logreg = LogisticRegression()

logreg.fit(X_train_const,y_train_const)
```

```
LogisticRegression()
```

```
y_pred_prob = logreg.predict(X_test_const)
```

```
# convert probabilities to 0 and 1 using 'if_else'
y_pred = [ 0 if x < 0.5 else 1 for x in y_pred_prob]
```

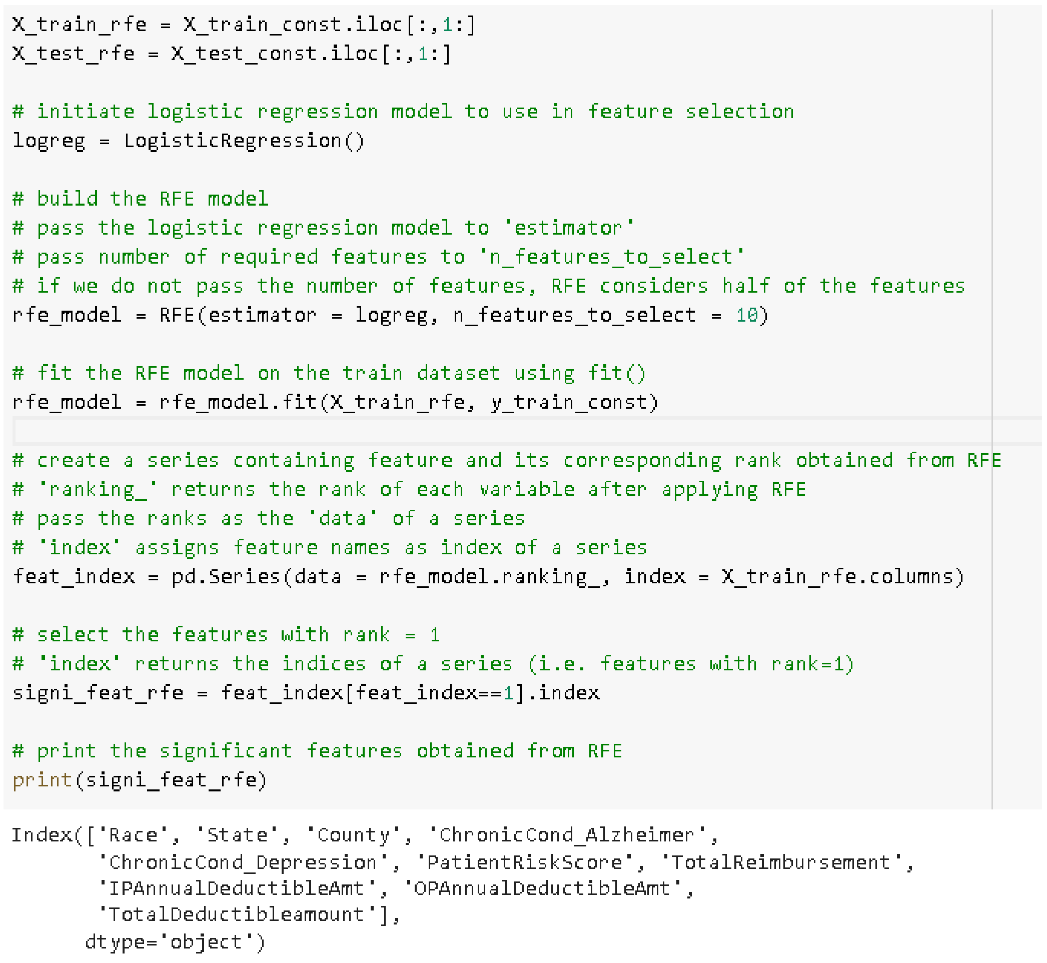


```
] acc_table = classification_report(y_test_const, y_pred)
  print(acc_table)
```

```
              precision    recall  f1-score   support

           0       0.63      0.95      0.76     68820
           1       0.58      0.11      0.18     42823

    accuracy                           0.63    111643
   macro avg       0.61      0.53      0.47    111643
weighted avg       0.61      0.63      0.54    111643
```

After the Data Preparation, Exploratory Data Analysis and Feature Engineering, Data Encoding and building our base models, we now further tune the model to optimize our key metrics which are Precision, i.e. the proportion of positive instances that were correctly predicted and Recall, i.e the proportion of actual positive cases that were correctly predicted, also sometimes called ‘True Positive Rate (TPR)’ or ‘Sensitivity’. We will do the Model Tuning in the following ways

Log Reg - Recursive Feature Elimination (RFE)



```
X_train_rfe = X_train_const.iloc[:,1:]
X_test_rfe = X_test_const.iloc[:,1:]

# initiate logistic regression model to use in feature selection
logreg = LogisticRegression()

# build the RFE model
# pass the logistic regression model to 'estimator'
# pass number of required features to 'n_features_to_select'
# if we do not pass the number of features, RFE considers half of the features
rfe_model = RFE(estimator = logreg, n_features_to_select = 10)

# fit the RFE model on the train dataset using fit()
rfe_model = rfe_model.fit(X_train_rfe, y_train_const)

# create a series containing feature and its corresponding rank obtained from RFE
# 'ranking_' returns the rank of each variable after applying RFE
# pass the ranks as the 'data' of a series
# 'index' assigns feature names as index of a series
feat_index = pd.Series(data = rfe_model.ranking_, index = X_train_rfe.columns)

# select the features with rank = 1
# 'index' returns the indices of a series (i.e. features with rank=1)
signi_feat_rfe = feat_index[feat_index==1].index

# print the significant features obtained from RFE
print(signi_feat_rfe)
```

```
Index(['Race', 'State', 'County', 'ChronicCond_Alzheimer',
       'ChronicCond_Depression', 'PatientRiskScore', 'TotalReimbursement',
       'IPAnnualDeductibleAmt', 'OPAnnualDeductibleAmt',
       'TotalDeductibleamount'],
      dtype='object')
```

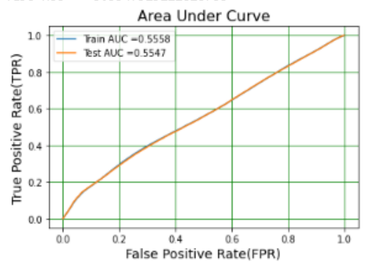
The above Features are obtained as Significant. It gives an accuracy of 62% and the F1 score as 76%.

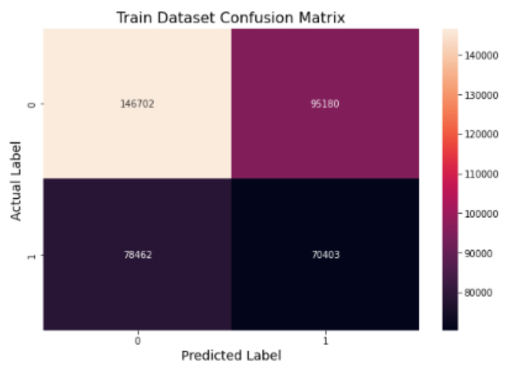
Log Reg - GridSearchCV - Hyper parameter tuning

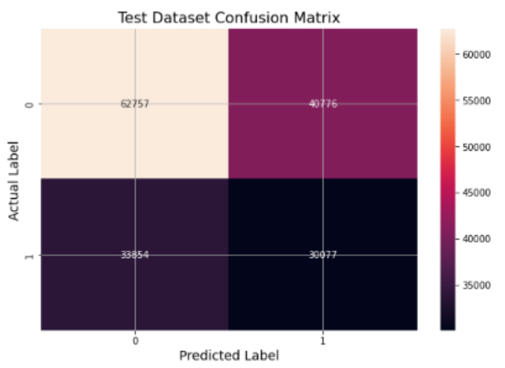
We try the Grid search strategy because the model performance is not very excellent. It is used to determine a model's ideal hyperparameters. Hyperparameters cannot be found in training data, unlike parameters. As a result, we build a model for each set of hyperparameters in order to determine the appropriate ones.

```python
# Validate LogisticRegression model
test_auc, test_f1_score, best_t = validate_model(log_reg, X_train_const, X_test_const, y_train_const, y_test_const)

print("Best Threshold = {:.4f}".format(best_t))
print("Model AUC is : {:.4f}".format(test_auc))
print("Model F1 Score is : {:.4f}".format(test_f1_score))
```

```
Train AUC =  0.555778277162794
Test AUC =  0.5547029221626768
```







```
Best Threshold = 0.3752
Model AUC is : 0.5547
Model F1 Score is : 0.4463
```

- Accuracy and F1 score are shown along with the ROC Curve and the confusion matrix.

The popular supervised methods which are used for healthcare fraud detection are Decision trees and Neural Networks.

Decision Trees:

Feature engineering can be followed by the usage of decision trees. It can produce rules based on the provided feature and class label. It can also manage any instances of missing values. Decision trees are particularly helpful for detecting healthcare fraud since they may be interpreted. The sole drawback is that if the dimension of the train dataset is big, too many rules are generated, leading to an overfitting tendency. It incorporates ensemble models like adaptive boosting to get rid of it. Using ensemble models, different classifiers are built sequentially, focusing on training instances that the preceding classifiers incorrectly classified.

Basic Decision Tree Model

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98    276595
           1       0.99      0.94      0.96    169973

    accuracy                           0.97    446568
   macro avg       0.98      0.97      0.97    446568
weighted avg       0.97      0.97      0.97    446568
```

Decision Tree - RandomizedSearchCV- Hyper parameter tuning

It appears that the traditional Decision Tree model is overfit, hence trying to resolve by using the RandomizedSearchCV model instead.

```
decisiontree = DecisionTreeClassifier() #class_weight = 'balanced'

parameters = {'criterion':['gini','entropy'], 'max_depth': [5, 10, 50, 100, 150, 200, 250, 500],
              'min_samples_split': [5, 10, 50, 100, 150, 200, 250, 500]}

decisiontree_cv = RandomizedSearchCV(decisiontree, parameters, cv=5, scoring='roc_auc', n_jobs=-1, return_train_score=True)
decisiontree_cv.fit(X_train, y_train)
```

```
RandomizedSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [5, 10, 50, 100, 150, 200,
                                                      250, 500],
                                        'min_samples_split': [5, 10, 50, 100,
                                                              150, 200, 250,
                                                              500]},
                   return_train_score=True, scoring='roc_auc')
```

It gives an accuracy of 77% and the F1 score as 82%.

Basic Random Forest Model

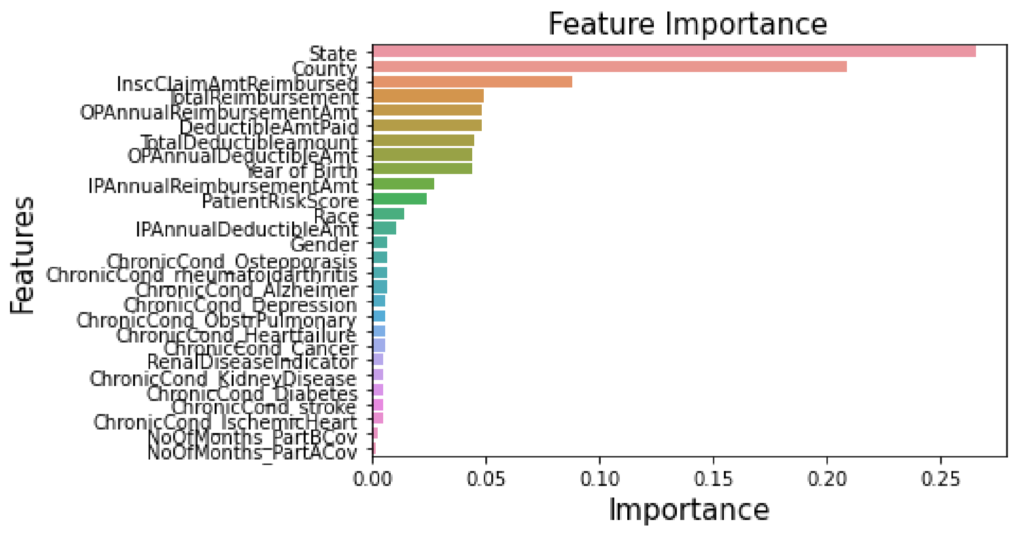
To determine whether the dataset responds well to the model, a simple Random Forest tuning was performed, and it eventually produced a favourable result.

```
rf_model = RandomForestClassifier(criterion = 'gini',
                                  max_depth = 50,
                                  min_samples_split = 30,
                                  max_leaf_nodes = 65,
                                  random_state = 10)
random_forest = rf_model.fit(X_train, y_train)

train_report = get_train_report(random_forest)
print('Train data:\n', train_report)

test_report = get_test_report(random_forest)
print('Test data:\n', test_report)
```

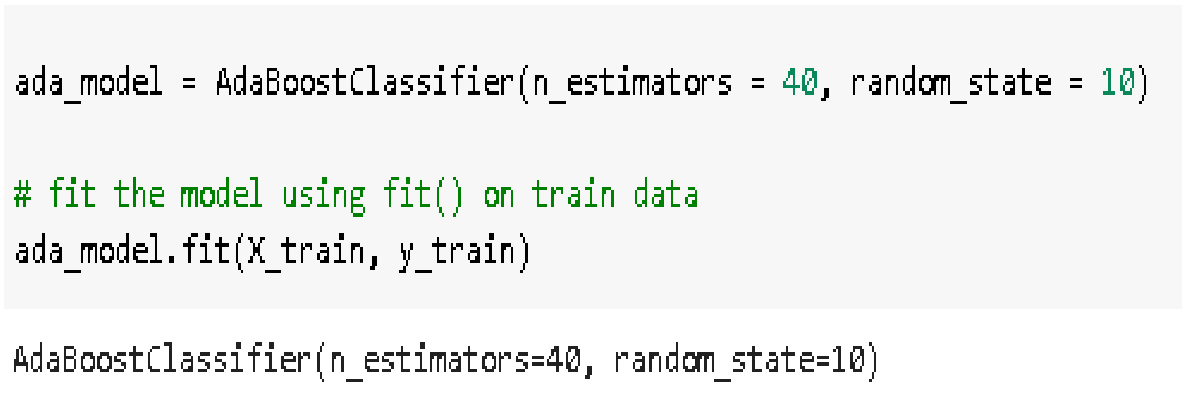
```
Train data:
               precision    recall  f1-score   support

           0       0.65      0.93      0.77    241882
           1       0.63      0.18      0.28    148865

    accuracy                           0.65    390747
   macro avg       0.64      0.56      0.52    390747
weighted avg       0.64      0.65      0.58    390747

Test data:
               precision    recall  f1-score   support

           0       0.65      0.94      0.77    103533
           1       0.63      0.18      0.28     63931

    accuracy                           0.65    167464
   macro avg       0.64      0.56      0.52    167464
weighted avg       0.64      0.65      0.58    167464
```

Random Forest with important Features

Now, in an effort to improve scores, we performed some parameter tuning using the model's key features.



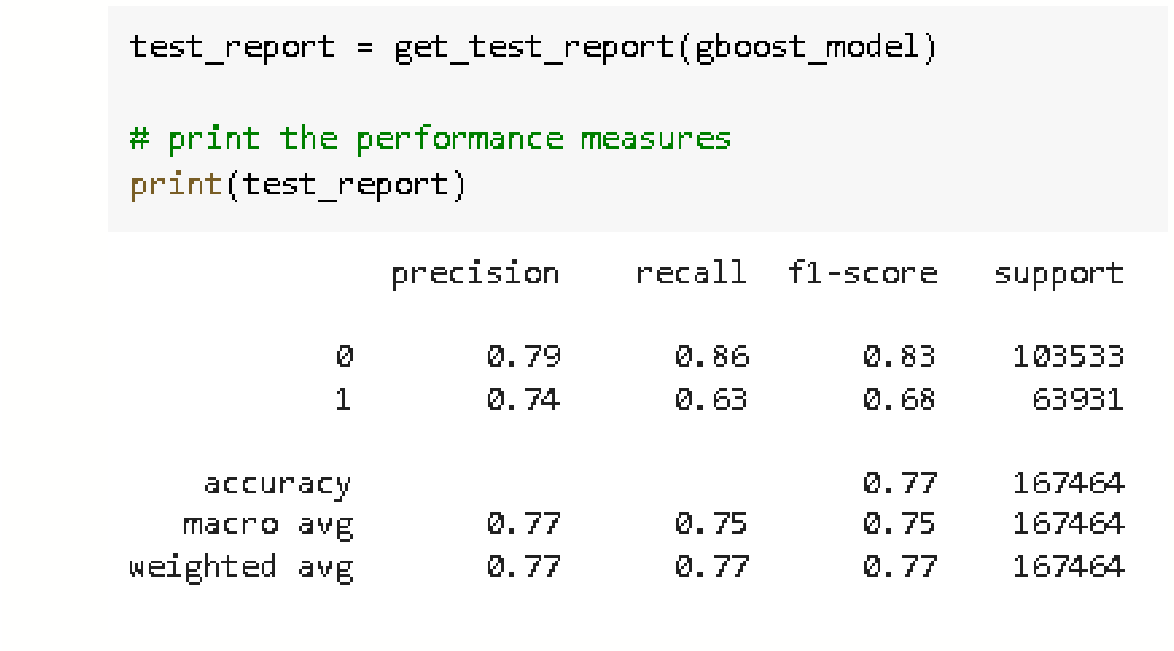
Basic AdaBoost Model



```
ada_model = AdaBoostClassifier(n_estimators = 40, random_state = 10)

# fit the model using fit() on train data
ada_model.fit(X_train, y_train)
```

```
AdaBoostClassifier(n_estimators=40, random_state=10)
```

Gradient Boosting

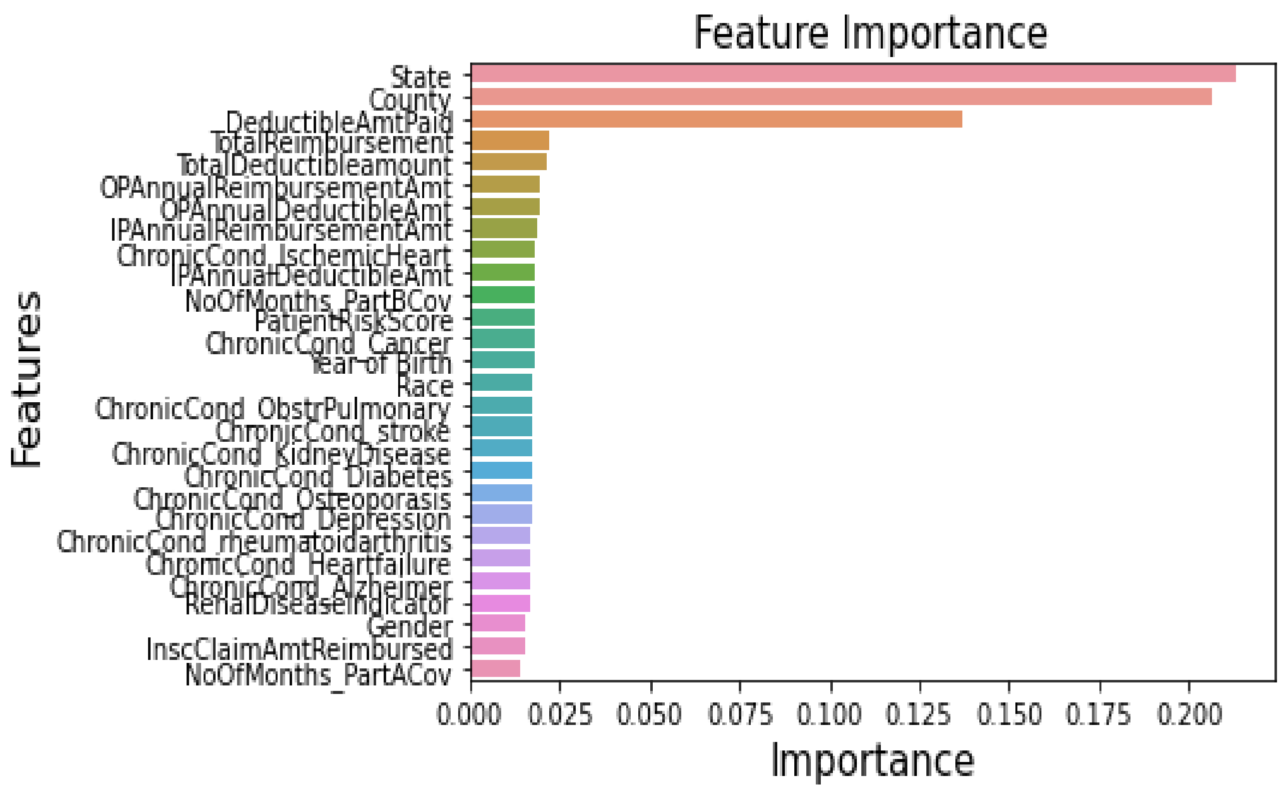


```
test_report = get_test_report(gboost_model)

# print the performance measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.83    103533
           1       0.74      0.63      0.68     63931

    accuracy                           0.77    167464
   macro avg       0.77      0.75      0.75    167464
weighted avg       0.77      0.77      0.77    167464
```

XGBoost

In comparison to gradient boosting machines, XGB has a variety of hyper-parameters that can be modified. XGBoost offers the ability to manage missing values right out of the box.



Basic Knn Model

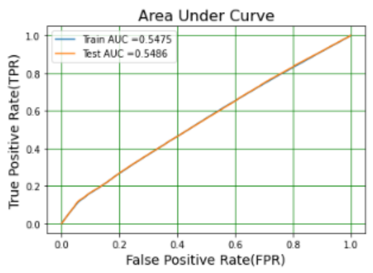
```
knn_classification = KNeighborsClassifier(n_neighbors = 3)

# fit the model using fit() on train data
knn_model = knn_classification.fit(X_train, y_train)
```

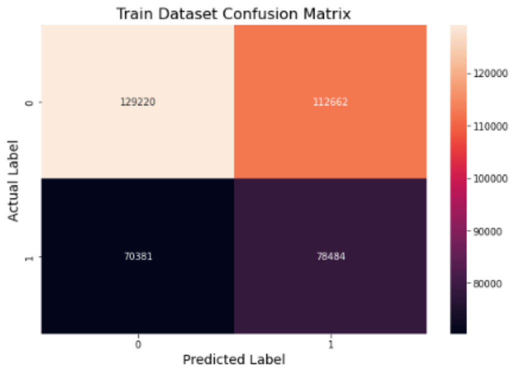
GaussianNB Classifier - Hyper parameter tuning

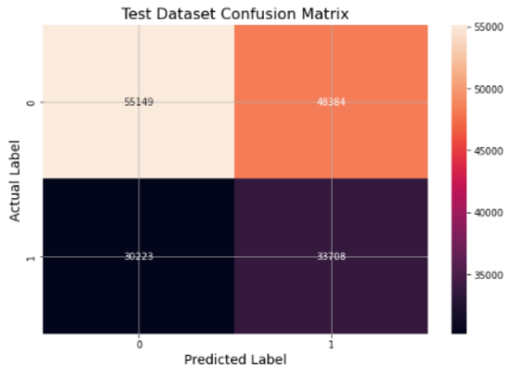
At the very end, the GaussianNB Classifier was tested, and it produced comparably poor results (60 percent accuracy and 50 percent F1 score).

```
test_auc, test_f1_score, best_t = validate_model(gaussian_nb, X_train, X_test, y_train, y_test)

print("Best Threshold = {:.4f}".format(best_t))
print("Model AUC is : {:.4f}".format(test_auc))
print("Model F1 Score is : {:.4f}".format(test_f1_score))
```

```
Train AUC =  0.5475360085349237
Test AUC =  0.5486402486238375
```







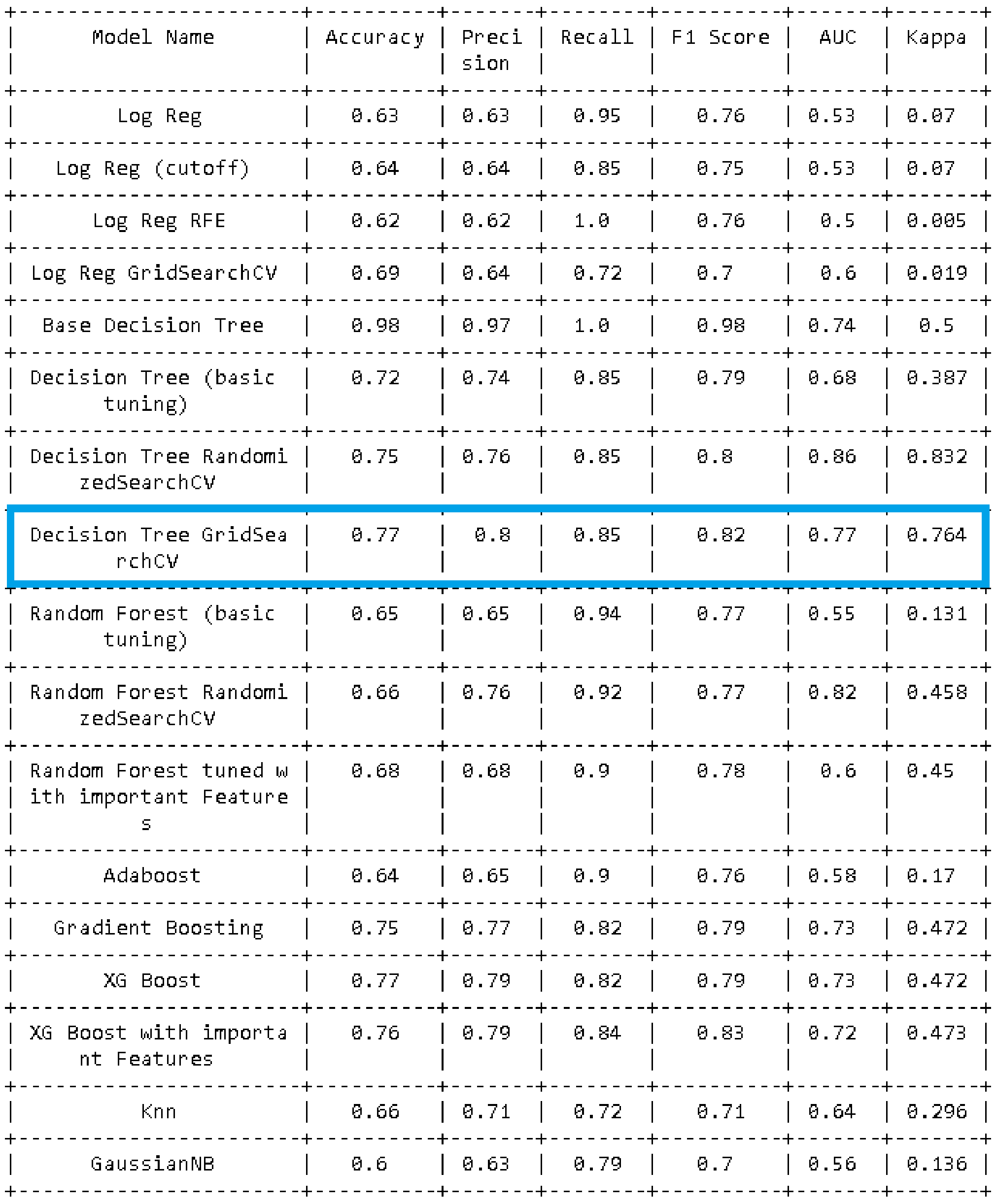
```
Best Threshold = 0.1588
Model AUC is : 0.5486
Model F1 Score is : 0.4617
```

Comparison and selection of model

We tried using Youden's index after the logistic regression base model produced a minimal outcome, and it produced a decent result with a 64 percent accuracy and 75 percent F1 score. Further research using RFE indicates that the logistic regression does not appear to function well, despite having a high recall value score. Finally, logistic regression was used to tune with GridsearchCV parameters, and the results were good, with an accuracy score of 69 percent, a recall value of 72 percent, and an F1 score of 70 percent.

Second, when the decision tree methods were used, the basic model was overfit. To address this overfitting problem, we first tried simple parameter adjustment, which produced acceptable results. Next, we tried randomised search CV and grid search CV to work more closely with the decision tree model. These two parameter tuning models yielded nearly identical scores, with the randomised search false behind the grid search scoring 0.5 and the great search scoring 0.7. The auc score and the recall numbers are also different from one another.

The random forest approaches are then used, where the same flow process is used to build a base model, a randomised search CV, and a random forest model with the key features. With an accuracy score of 74%, an F1 score of 79%, and a recall value of 81%, the random forest with randomised search CV appears to provide a decent outcome. The issue that we would like to address here is that despite having a large quantity of data, we were unable to effectively execute the random forest grid search CV, which may have resulted in a high score.



```
+------------------------+----------+-------+--------+----------+-------+-------+
|       Model Name       | Accuracy | Preci | Recall | F1 Score |  AUC  | Kappa |
|                        |          | sion  |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|        Log Reg         |   0.63   | 0.63  |  0.95  |   0.76   | 0.53  | 0.07  |
+------------------------+----------+-------+--------+----------+-------+-------+
|    Log Reg (cutoff)    |   0.64   | 0.64  |  0.85  |   0.75   | 0.53  | 0.07  |
+------------------------+----------+-------+--------+----------+-------+-------+
|      Log Reg RFE       |   0.62   | 0.62  |  1.0   |   0.76   |  0.5  | 0.005 |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Log Reg GridSearchCV  |   0.69   | 0.64  |  0.72  |   0.7    |  0.6  | 0.019 |
+------------------------+----------+-------+--------+----------+-------+-------+
|   Base Decision Tree   |   0.98   | 0.97  |  1.0   |   0.98   | 0.74  |  0.5  |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Decision Tree (basic  |   0.72   | 0.74  |  0.85  |   0.79   | 0.68  | 0.387 |
|        tuning)         |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Decision Tree Randomi |   0.75   | 0.76  |  0.85  |   0.8    | 0.86  | 0.832 |
|      zedSearchCV       |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Decision Tree GridSea |   0.77   |  0.8  |  0.85  |   0.82   | 0.77  | 0.764 |
|         rchCV          |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Random Forest (basic  |   0.65   | 0.65  |  0.94  |   0.77   | 0.55  | 0.131 |
|        tuning)         |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Random Forest Randomi |   0.66   | 0.76  |  0.92  |   0.77   | 0.82  | 0.458 |
|      zedSearchCV       |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|  Random Forest tuned w |   0.68   | 0.68  |  0.9   |   0.78   |  0.6  | 0.45  |
|  ith important Feature |          |       |        |          |       |       |
|           s            |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|        Adaboost        |   0.64   | 0.65  |  0.9   |   0.76   | 0.58  | 0.17  |
+------------------------+----------+-------+--------+----------+-------+-------+
|    Gradient Boosting   |   0.75   | 0.77  |  0.82  |   0.79   | 0.73  | 0.472 |
+------------------------+----------+-------+--------+----------+-------+-------+
|        XG Boost        |   0.77   | 0.79  |  0.82  |   0.79   | 0.73  | 0.472 |
+------------------------+----------+-------+--------+----------+-------+-------+
|  XG Boost with importa |   0.76   | 0.79  |  0.84  |   0.83   | 0.72  | 0.473 |
|      nt Features       |          |       |        |          |       |       |
+------------------------+----------+-------+--------+----------+-------+-------+
|          Knn           |   0.66   | 0.71  |  0.72  |   0.71   | 0.64  | 0.296 |
+------------------------+----------+-------+--------+----------+-------+-------+
|       GaussianNB       |   0.6    | 0.63  |  0.79  |   0.7    | 0.56  | 0.136 |
+------------------------+----------+-------+--------+----------+-------+-------+
```

Finally, we moved on to the procedures for boosting where the XG boost and add a boost were completed. A respectable accuracy score of 75% is provided by gradient boosting and extreme boosting in comparison. The attempt on XG boost with the key elements actually had an F1 score of 83 percent, a 76 percent accuracy score, and a 76 percent overall performance. A simple KNN and Gaussian NB model was tried, and it produced an accuracy value of roughly 65%.The test dataset is used to evaluate each of these scores.

Selection of Model

After evaluating all the models on the test dataset, we can conclude that the Decision Tree - GridsearchCV model works well on the data set with an accuracy score of 77 percent, recall value of 84 percent, F1 score of 82 percent, and kappa score of 74 percent.

Results & Discussion

In order to discuss the results in detail, the decision tree model with GridsearchCV, which has an accuracy score of 77%, was ultimately chosen. 77% of the predictions made by the model were accurate. A model is better when its accuracy value is relatively high.

The percentage of positive cases that were accurately anticipated is 80% when it comes to the precision value. Once more, our model's relatively greater precision value adds something to make it a superior model to the competition.

The F1 score is 76 percent, which is a harmonic mean of the precision and recall values for our classification model. The true positive rate for our model is 84 percent, which is comparatively higher than any other model's value. As the F1 score is a useful metric for balancing precision and recall, and as it is currently 82 percent, we can state with confidence that our model provides the highest F1 score.

The chosen model's kappa score is 0.746, which is very close to 0.8, and we can infer that the inter-rater reliability is high. The measure of separability between the classes in a target variable is 76 percent according to the auc score, which is around 76 percent. The auc score in the grid search CV is higher than the scores of the other models. One issue we would like to address is the fact that because the data set is so large, codes like hyper parameter tuning, selecting important features and even GridsearchCV often have trouble running or take too long to complete, which prevents us from testing new strategies constantly.

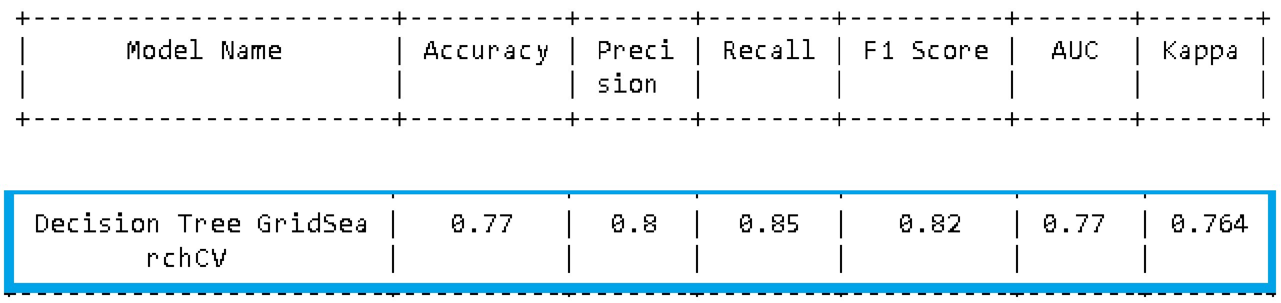
Whether the data is numerical, category, or boolean, decision trees can handle any form of data.

The Decision Tree does not require normalisation.One machine learning approach where we don't worry about feature scaling is the decision tree. Random woods are another. They are scale-invariant algorithms.

Description of Criterion

The model was originally trained with the following parameters: the maximum depth was selected from a range of values, and the minimum sample split was also given a range of values up to 500. The criteria on which these parameters were based were Gini and entropy combined. These parameters were added to the grid search CV, which used a decision tree as the model and included the parameters with a CV of 5 and ROC auc scoring. The model was then fitted using the aforementioned parameters. Gini, whose maximum depth is 250 and minimum sample split is also 250, has been proven to have the best parameters and score from this decision tree model. The model has supplied the above mentioned results as per the description.

Conclusion



```
+-----------------------+----------+-------+--------+----------+-------+-------+
|       Model Name      | Accuracy | Preci | Recall | F1 Score |  AUC  | Kappa |
|                       |          | sion  |        |          |       |       |
+-----------------------+----------+-------+--------+----------+-------+-------+

| Decision Tree GridSea |   0.77   |  0.8  |  0.85  |   0.82   |  0.77 | 0.764 |
|         rchCV         |          |       |        |          |       |       |
+-----------------------+----------+-------+--------+----------+-------+-------+
```

As previously discussed, the decision tree group search model produces a comparatively better output, with an accuracy score of 77% and an F1 score of 82%. The kapas score of 0.76 indicates that the model's inter rated reliability is significant.

We would like to improve our models by using the unsupervised clustering method to see how the data is distributed in clusters and then performing classification models based on the acquired information. Another issue we encountered was that because the data was so large, we were unable to test hyper parameter tuning and future importance on some models. Further advancements would include attempting to upsample the data in various combinations and testing the models.

Ref:
https://www.roselladb.com/healthcare-fraud-detection.htm#:~:text=Healthcare%20fraud%20detection%20involves%20account,feasible%20by%20any%20practical%20means.

Fraudulent activity can be identified by analysing statistics of patients, doctors and providers. Higher per-patient costs, excessive per-doctor patients, higher per-patient tests are suspicious.

1. The effect of fraudulent activities are more number of claims than usual as a result claim amount will be more per provider. So, we need to analyze each provider's total claim amount in a specific time period.
2. The average billing per patient will also be higher. So, average billing per patient needs to be analyzed for every provider.
3. Total number of patients per provider needs to be analyzed to check if this is suspicious.
4. Average visit per patient to be calculated to check if this is more than usual.
5. Average number of tests per patient needs to be analyzed.
6. Average medical tests cost per patient to check if this is high i.e. suspicious.
7. Average number of prescriptions per patient, this is also an indication of the number of hospital visits/checkups of the patient.
8. Doctors who treated patients more than usual are suspicious.
9. Providers administering far higher rates of tests than others.
10. Providers cost far more, per patient basis, than others.
11. Providers with a high ratio of distance patients. It is not natural that patients visit providers which are far away.
12. Providers prescribing certain drugs at a higher rate than others. Even though drugs with less price can be used for that disease.

Notes For Project Team

Sample Reference for Datasets (to be filled by team and mentor)

Original owner of data	Rohit Anand Gupta
Data set information	HEALTHCARE PROVIDER FRAUD DETECTION ANALYSIS
Any past relevant articles using the dataset	https://medium.com/analytics-vidhya/healthcare-provider-fraud-detection-analysis-using-machine-learning-81ebf09ed955
Reference	Kaggle
Link to web page	https://www.kaggle.com/datasets/rohitrox/healthcare-provider-fraud-detection-analysis

****