FRAUD DETECTION ANALYSIS IN HEALTH CARE

Post Graduate Program in Data Science Engineering

Location: **BLR** Batch: **PGP-DSEFT_Capstone Group 1**

Submitted by

Durjoy Ghosh, Swarnadip Majumder, V.Satish Kumar Raju, Chiranjeevi B, Sathya Swaroop V R, Aathiragul

Mentored by

Srikar Muppidi

Table of Contents

ACKNOWLEDGEMENT

Any endeavour in a specific field requires the guidance and support of many people for successful completion. The sense of achievement on completing anything remains incomplete if the people who were instrumental in its execution are not properly acknowledged. We would like to take this opportunity to verbalize our deepest sense of indebtedness to our project mentor, Mr. Srikar Muppidi, who was a constant pillar of support and continually provided us with valuable insights to improve upon our project and make it a success. Further, we would like to thank our parents for encouraging us and providing us a platform wherein we got an opportunity to design our own project.

Date: 24th Mar, 2022

Place: Bangalore

CERTIFICATE OF COMPLETITION

This is to certify that the work titled Fraud Detection Analysis in Health Care by Appliances submitted by Capstone Group 1 of PGP- DSE of Great Lakes Institute of Management, Gurgaon has been carried out under my supervision. This work has not been submitted partially or wholly to any other university or institute for the award of any other degree or diploma.

Name of the Mentor: Mr. Srikar Muppidi

Signature of the Mentor:

Date: 24th Mar, 2022

Place: Bangalore

DECLARATION

We hereby declare, that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 24th Mar, 2022

Place: Bangalore

LIST OF ABBREVIATIONS

S. No.	Abbreviation	Meaning
1.	CRISP-DM	Cross-industry process for data mining
2.	OOP	Out of Pocket
3.	Claim	Claim by Healthcare Provider
4.	DT	Decision Tree
5.	RF	Random Forest
6.	RFE	Recursive factor elimination
7.	XGB	Extreme Gradient Boosting
8.	CAT Boost	Category Boosting
9.	GBM	Gradient boosting machine
10.	CV	Cross Validation

1. Abstract

It is estimated that approximately 10% of healthcare system expenditures are wasted due to medical/healthcare fraud and abuse.

Medicare provider Fraud is one of the biggest issues in the healthcare domain. According to the government, the total Medicare spending increased exponentially due to frauds in Medicare claims. Due to these malpractices, the Healthcare Insurance companies suffer a lot and their businesses are impacted extensively. As a result, they are increasing the insurance premiums day by day, resulting in healthcare becoming much more expensive.

2. Introduction

As in the medical area, the combination of thousands of drugs and diseases make the supervision of health care more difficult, oftentimes these organized crimes involve **peers of healthcare providers, physicians**, and **beneficiaries acting together with the physicians** leading to fraudulent claims.

Some of the potential Healthcare frauds and abuse happens due to underlining reasons:

- Claim a bill using ambiguous diagnosis code to conduct the costliest treatment and therefore charge for an expensive treatment than it was really needed.
- Billing for services that were not provided.
- Submitting the duplicate copy of the claim several times for the same service.
- Claim for a covered service, when the service provided was not covered.

2.1.Domain and Feature Review

Potential Fraud detection in healthcare services will not only reduce the overall Medicare expenditure but will also help to reduce the burden of the Health Insurance providers as businesses.

It can have the following real-time impacts if the business problem is solved thoroughly.

- In the future, any potential fraud can be predicted beforehand in real-time, so intern the loss can be minimized.
- We can discover the most important features, that help detect the behavior of potential Fraud Healthcare providers.
- we can further study the fraudulent patterns in the Healthcare provider's claims to understand the future behavior of such providers.
- Health insurance can be provided at a reasonable/cheaper cost.
- More customers can avail the benefit of such schemes.

2.2.Dataset Information

From source we get datasets of Inpatient, Outpatient, Beneficiary and Provider Details.

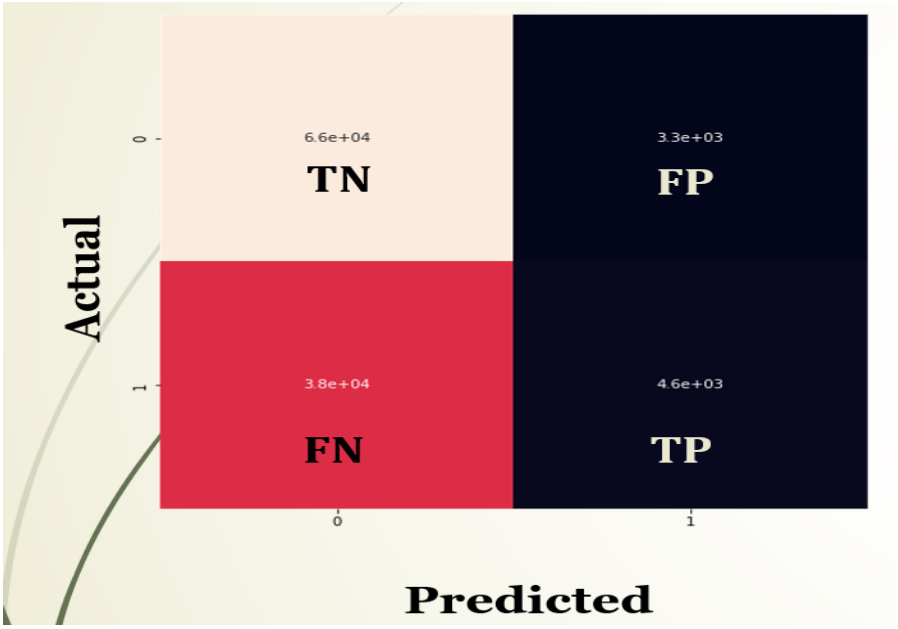
First we outer merge the inpatient and outpatient data to get all the Patient's details. Then we merge all patient data with the beneficiary data on BeneID. Lastly we merge the allpatient_benificiary data with the provider details to get the final data to work on.

Dataset has 558211 instances (rows) and 55 attributes (columns).

2.3.Problem Statement

Fraud Detection in Healthcare services using Classification

- Here our objective is to correctly classify the Potential Fraud, which is our Target Variable.
- In the Business Problem:
  - **FPR:** In actual the claims are Legit but model says its Fraud
  - **FNR:** In actual the claims fraud, but the model says its Legit
- Here our aim is to keep the **FN** as minimum as possible so that the business can minimize its loss.



2.4.Variable Categorization with Description

Categorical Variable: 26, Numerical Variable: 29

Name Of Features	Description
BeneID	ID of the Beneficiary (Patients) registered for the healthcare insurance scheme
ClaimID	ID of the claim submitted by healthcare providers
ClaimStartDt	The date on which the claim is initiated/submitted
ClaimEndDt	The date on which the claim has settled
Provider	ID of the Healthcare provider
InscClaimAmtReimbursed	the amount reimbursed as per the claim for the bill by the Healthcare provider. Here the highest amount reimbursed is 125000.
AttendingPhysician	ID of the physician attending the patient as per the Claim
OperatingPhysician	ID of the physician conducting the operation procedure as per the Claim
OtherPhysician	ID of another physician that the beneficiary has consulted as per the claim
AdmissionDt	Date of admission of the beneficiary(patient) at the hospital as per the claim (Applicable only for inpatient services)
ClmAdmitDiagnosisCode	It contains codes of the diagnosis performed by the healthcare provider on the patient as per the claim(inpatient)
DeductibleAmtPaid	Amount claimant/patient has to pay before insurance company starts paying up, insurance company liable to pay only when bill exceeds deductible, deductible can be zero also meaning entire bill to paid by the insurance provider. (Total claim amount – reimbursed amount)
DischargeDt	Date of discharge of the beneficiary(patient) from the hospital as per the claim (Applicable only for inpatient services)

DiagnosisGroupCode	It contains a group code for the diagnosis done on the patient. (For inpatient services)
ClmDiagnosisCode_1	As per the claim it contains the codes of certain diagnosis performed by the provider to the patient. As per the International Classification of Diseases (ICD). Codes are updated in every 10-15 years. Healthcare provider can attach multiple codes for a single claim.
ClmDiagnosisCode_2	Do
ClmDiagnosisCode_3	Do
ClmDiagnosisCode_4	Do
ClmDiagnosisCode_5	Do
ClmDiagnosisCode_6	Do
ClmDiagnosisCode_7	Do
ClmDiagnosisCode_8	Do
ClmDiagnosisCode_9	Do
ClmDiagnosisCode_10	Do
ClmProcedureCode_1	"Procedure" code is a catch-all term for codes used to identify what was done to or given to a patient in terms of medical procedures and services (surgeries, durable medical equipment, medications, laboratory, radiology etc.) This indicates the medical services provided during the period covered by the institutional claim. As per the Current Procedural Terminology (CPT). Codes are updated every 5 years.
ClmProcedureCode_2	Do

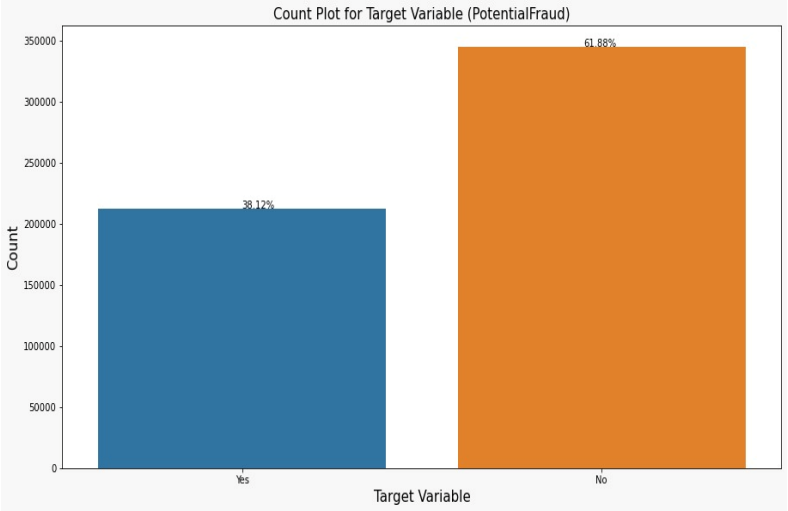
ClmProcedureCode_3	Do
ClmProcedureCode_4	Do
ClmProcedureCode_5	Do
ClmProcedureCode_6	Do
DOB	Date of Birth of the Beneficiary
DOD	Date of Death of the Beneficiary
Gender	Gender of the Beneficiary
Race	Race of the Beneficiary
Renal Disease Indicator	It indicates whether the Beneficiary has pre-existing kidney disease or not. It indicates the risk score of the patient
State	State(living) of the Beneficiary
County	County of the Beneficiary
NoOfMonths_PartACov	No of months does the beneficiary gets part A Coverage. It's like Impatient coverage for the beneficiary who wants to Admit into Hospital.
NoOfMonths_PartBCov	No of months does the beneficiary gets part B Coverage. It's like outpatient coverage for the beneficiary, who wants to come, treat and go.
ChronicCond_Alzheimer	Pre-existing disease of the Beneficiary

ChronicCond_Heartfailure	Pre-existing disease of the Beneficiary
ChronicCond_KidneyDisease	Pre-existing disease of the Beneficiary
ChronicCond_Cancer	Pre-existing disease of the Beneficiary
ChronicCond_ObstrPulmonary	Pre-existing disease of the Beneficiary
ChronicCond_Depression	Pre-existing disease of the Beneficiary
ChronicCond_Diabetes	Pre-existing disease of the Beneficiary
ChronicCond_IschemicHeart	Pre-existing disease of the Beneficiary
ChronicCond_Osteoporasis	Pre-existing disease of the Beneficiary
ChronicCond_rheumatoidarthritis	Pre-existing disease of the Beneficiary
ChronicCond_stroke	Pre-existing disease of the Beneficiary
IPAnnualReimbursementAmt	It indicates maximum reimbursement amount/year for In Patient services(hospitalized) to the patient
IPAnnualDeductibleAmt	The amount that the patient pays as premium annually before insurance company starts paying up for In Patient (hospitalization) services
OP Annual Reimbursement Amt	It indicates maximum reimbursement amount/year for Out Patient services to the patient
OP Annual Deductible Amt	The amount that the patient pays as premium annually before insurance company starts paying up for Out Patient services

PotentialFraud	Target Variable containing Yes = Potential Fraud, No = Legit Claim

2.5.Target Variable:

Here, our target variable is PotentialFraud. Our target variable is binomial in nature. Below, we have shown the distribution of the two classes 'Yes' and 'No' present in the target variable.



Interpretation: The plot shows that there is not that much imbalance in the target variable. So, no class-imbalance treatment is required.

3. Exploratory Data Analysis

3.1. Check for Data Types and Change it accordingly

1. From our knowledge, we can say that the independent variables 'ClaimStartDt', 'ClaimEndDt', 'AdmissionDt', 'DischargeDt', 'DOB', 'DOD' are date-time object in nature. But here, they are defined as object. So, we will change the datatype of those columns so that it will be easy for us to gather more data or more features in our further analysis.

2. All the Claim Procedure Codes are object by definition. But here it is interpreted as float64. So, we will change the datatype of all the ClaimProcedureCodes.

3. After checking the datatypes of the variables, we see that the data type of 'Gender', 'Race', 'State', 'County' are 'int64'. But according to our understanding, 'Gender', 'Race', 'State', 'County' is

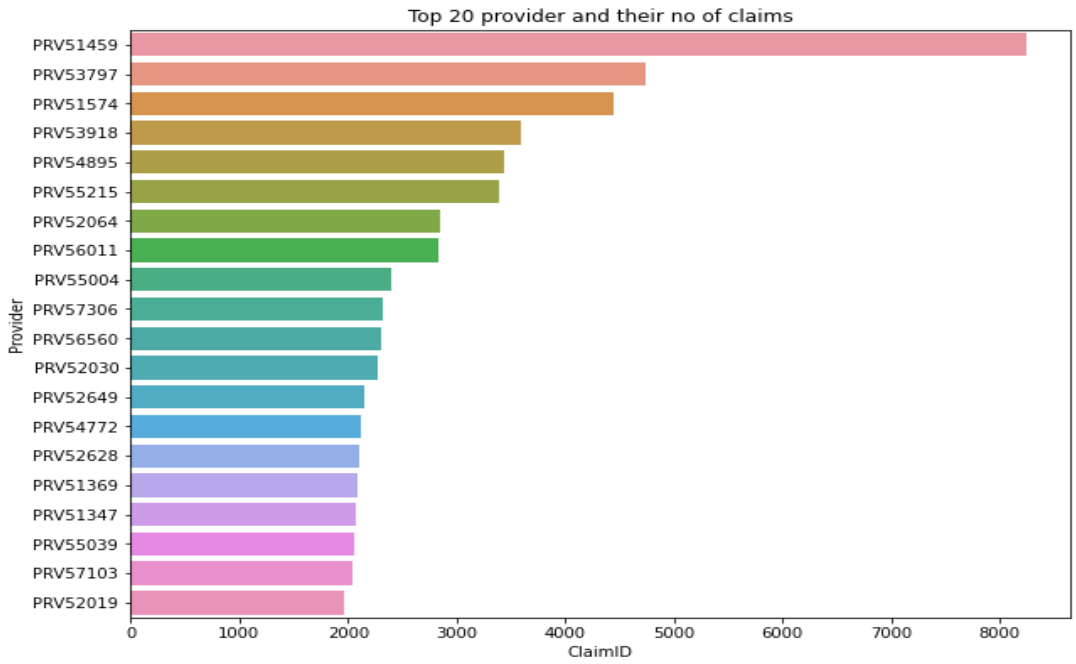
categorical variables, which is wrongly interpreted as 'int64', so we will convert these variables data type to 'object'.

4. By nature, All the Chronic Coditions are categorical vairables as they represent only 'Yes' or 'No' values. So, we will type cast those vairables.

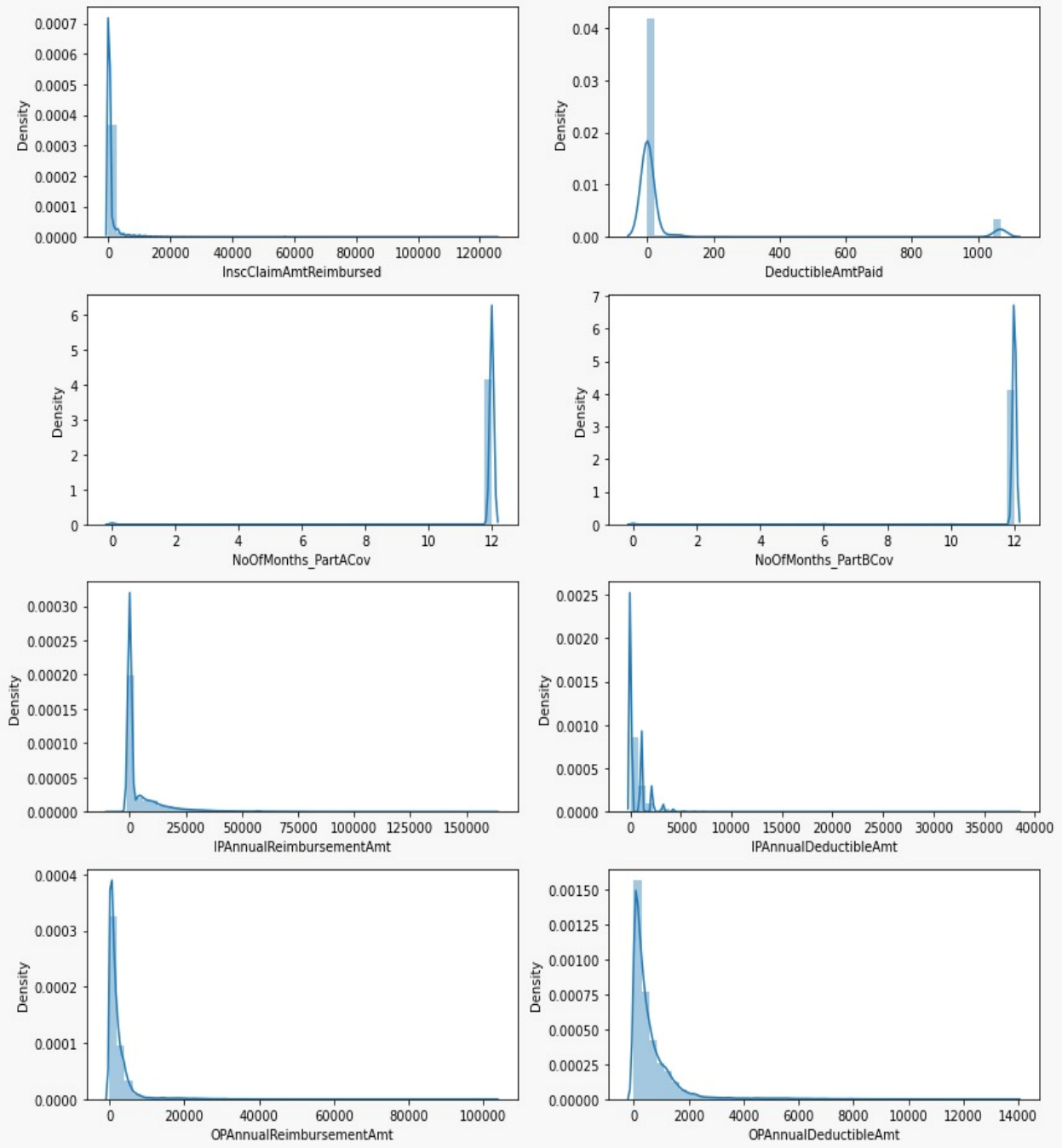
3.2. Uni Variate Analysis:

As 'BeneID', 'ClaimID', 'Provider', 'County' are basically representing the beneficiary id, claim id and provider id respectively. We are not going to keep those variables for our univariate analysis.

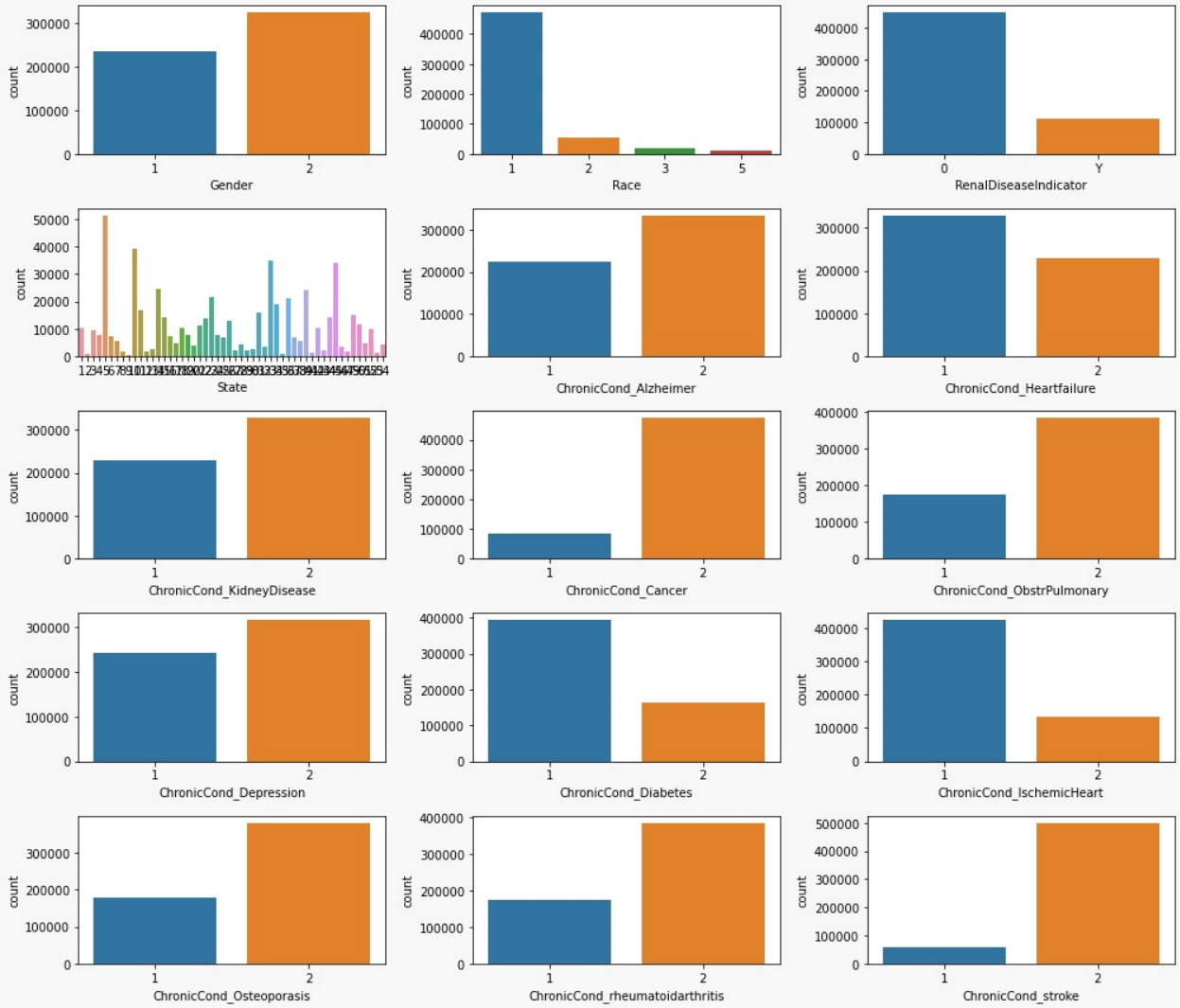
Also, We will extract 'Gender','Race','RenalDiseaseIndicator','State' and all the chronic conditions from the categorical variables to do our univariate analysis as other variables have high number of categories present in them, so it will be hard to interpret from such congested graphs. But, We will still visualize some of the important features who have high number of categories.



Interpretation : In the above plot, we can visualize the top 20 providers based on their claim count.



Interpretation: The plot indicates that all the variables are either right or left skewed. We might need to do scaling or transformation later.



For categorical variables:

Gender:

- **Males** are in between 2 lakh to 3lakh.
- **Females** are more than 3lakhs.

Race:

- Race-1 having data more than 4 lakhs.
- Rest Others having less than 1 lakh.

- People having **chroniccond_Kidney Disease** are around 2 lakhs. Rest of them not affected.
- People having **chroniccond_Alzheimer** are around 2 lakhs. Rest of them not affected.

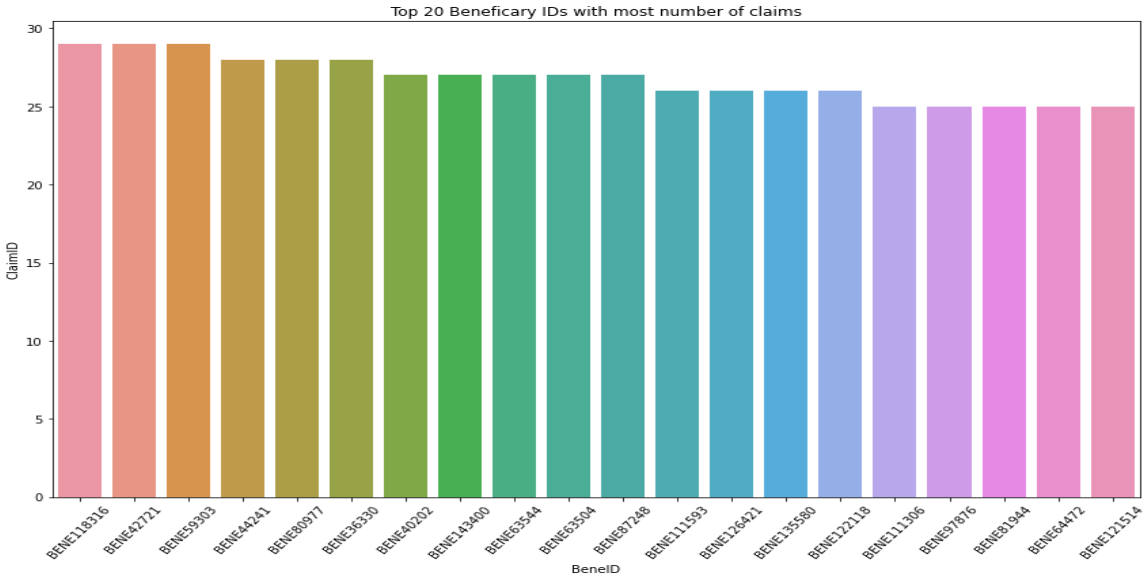
- People having **chroniccond_depression** are around 2 lakhs. Rest of them not affected.
- People having **chroniccond_Osteoporasis** are nearly close to 2 lakhs. Rest of them not affected.
- People having **chroniccond_rheumatoidarthitis** are close to 2 lakhs. Rest of them not affected.
- People having **chroniccond_Diabetes** are close to 4 lakhs. Rest of them not affected.
- People having **chroniccond_Cancer** are less than 1 lakh. Rest of them not affected.
- People having **chroniccond_Stroke** are less than 1 lakh. Rest of them not affected.
- People having **chroniccond_IschemicHeart** are around 4 lakhs. Rest of them not affected.

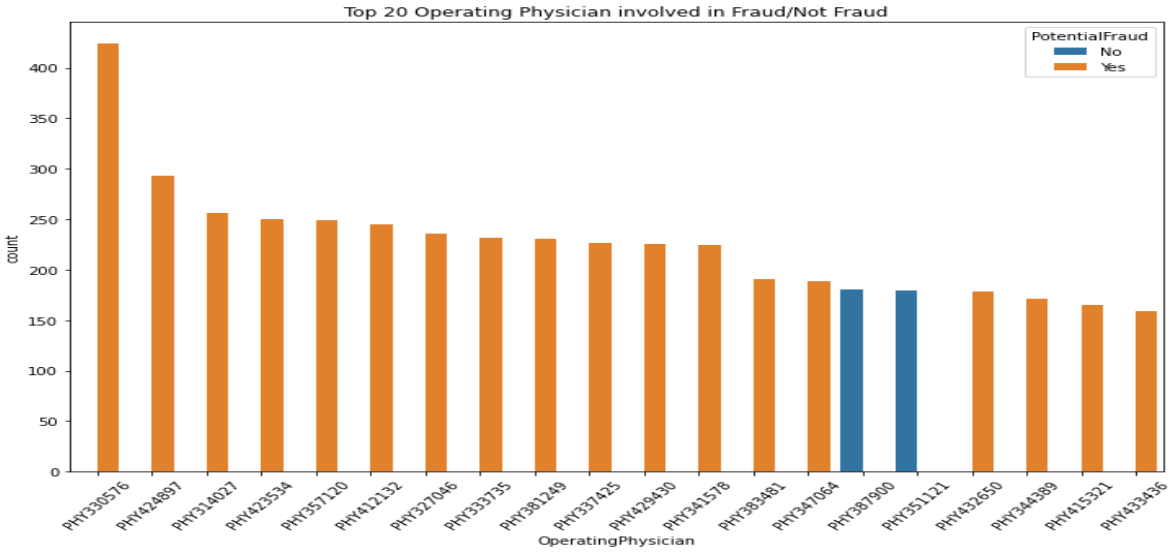
3.3. Bi Variate Analysis

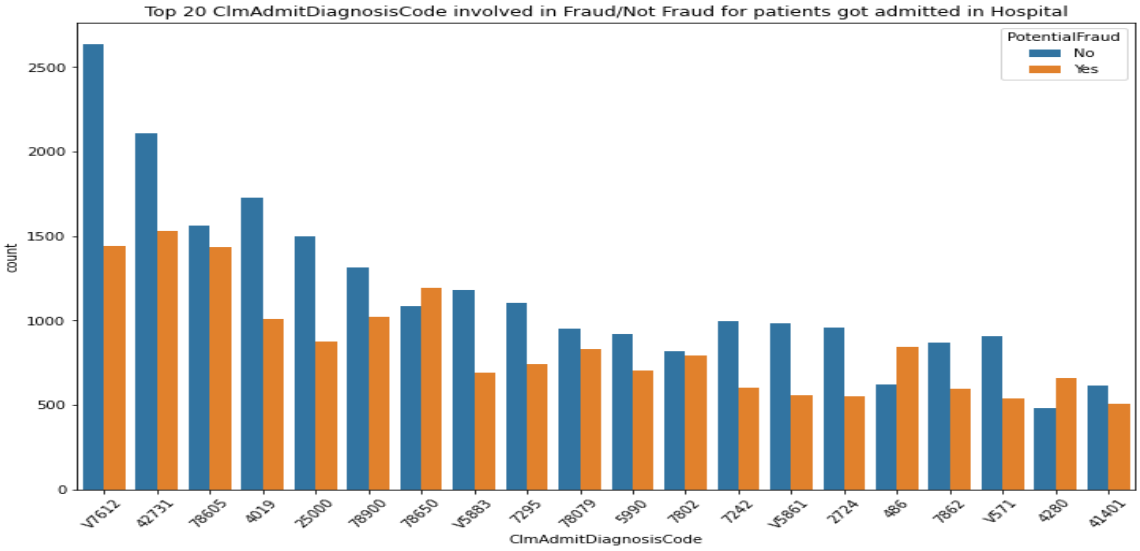
. Bivariate Analysis:

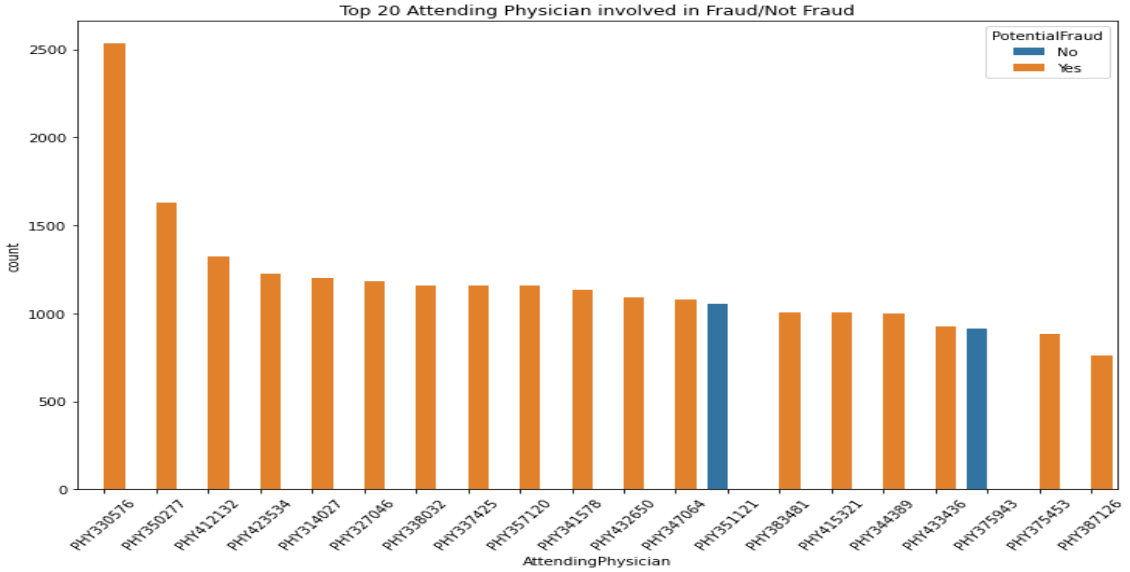
- Out of Top 20 **claim-id's** There are top-5 **Provider-id** whose claims are crossed like around 3700 claims for each provider id.
- Out of top 20 **Operating physician** who involved in fraud, there is physician id PHY330576 who done with more than 400 frauds. From Rest of 19 physicians , 17 Physicians who intends to do fraud are on an average of 200 frauds. Only 2 physicians are not doing any kind of Fraudulent activities.
- Out of top 20 **Attending physician** who involved in fraud, There is physician id PHY330576 who done with more than 2500 frauds. From Rest of 19 physicians, 17 Physicians who intends to do fraud are on an average of 1300 frauds. Only 2 physicians are not doing any kind of Fraudulent activities.
- In **ClmAdmitDiagnosisCode** we can see that Out of Top 20 ClmAdmitDiagnosisCode There are more number of codes that which are not impacting of doing frauds. Doing frauds by using this ClmAdmitDiagnosisCode are less than 1600 I can say.
- Each **Bene-Id** having on an average of 25 claims each.
- **ClmProcedureCode-1** is performing potential Frauds of maximum above 600 out of top 20 ClmProcedureCode-1.
- Op annual Reimbursement amnt, IP annual reimbursement amnt, Ip annual deductible amnt, OP Annual deductible amnt, No of months partA Cov, No of months PartB cov are performing More or less around similar kind fraud detection for both having fraud and not having fraud.
- **InscClaimAmntReimbursed** are having with fraud are more than 1250.without Fraud are 750.
- **DeductibleAmntPaid** are having more than 100 frauds and around 50 who not done frauds.

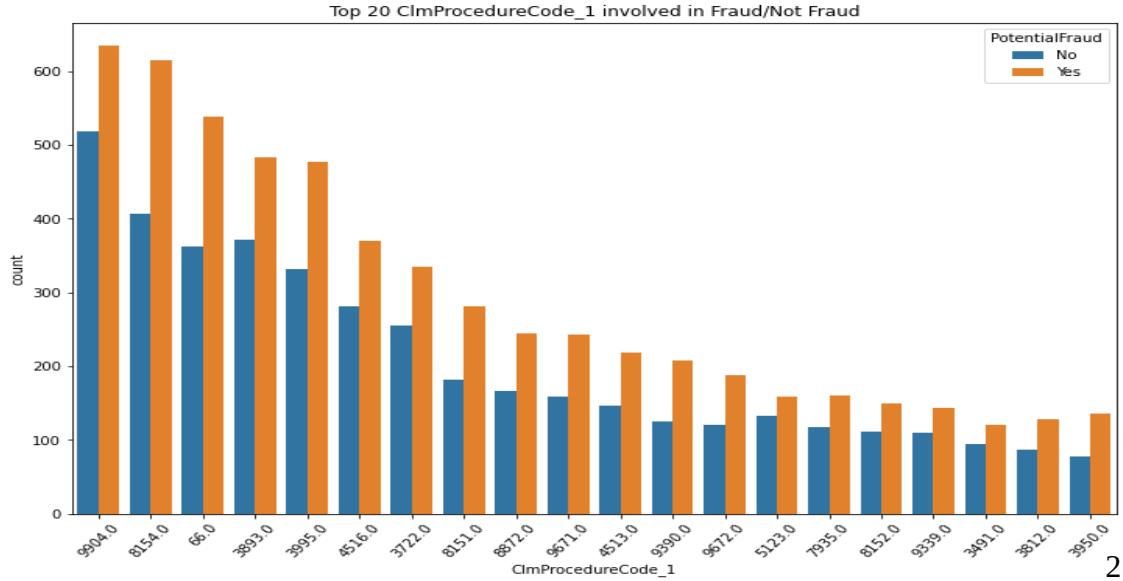
- Chroniccond_stroke, chroniccond_ischemic heart ,chroniccond_cancer, chroniccond_Kidney Disease , chroniccond_Alzheimer , chroniccond_depression ,chroniccond_ObstrPulmonary ,are not having any disease but performing more fraud.(around 2 lakh Frauds happening).
- Rest other Features also performing frauds but not to that level as above mentioned.





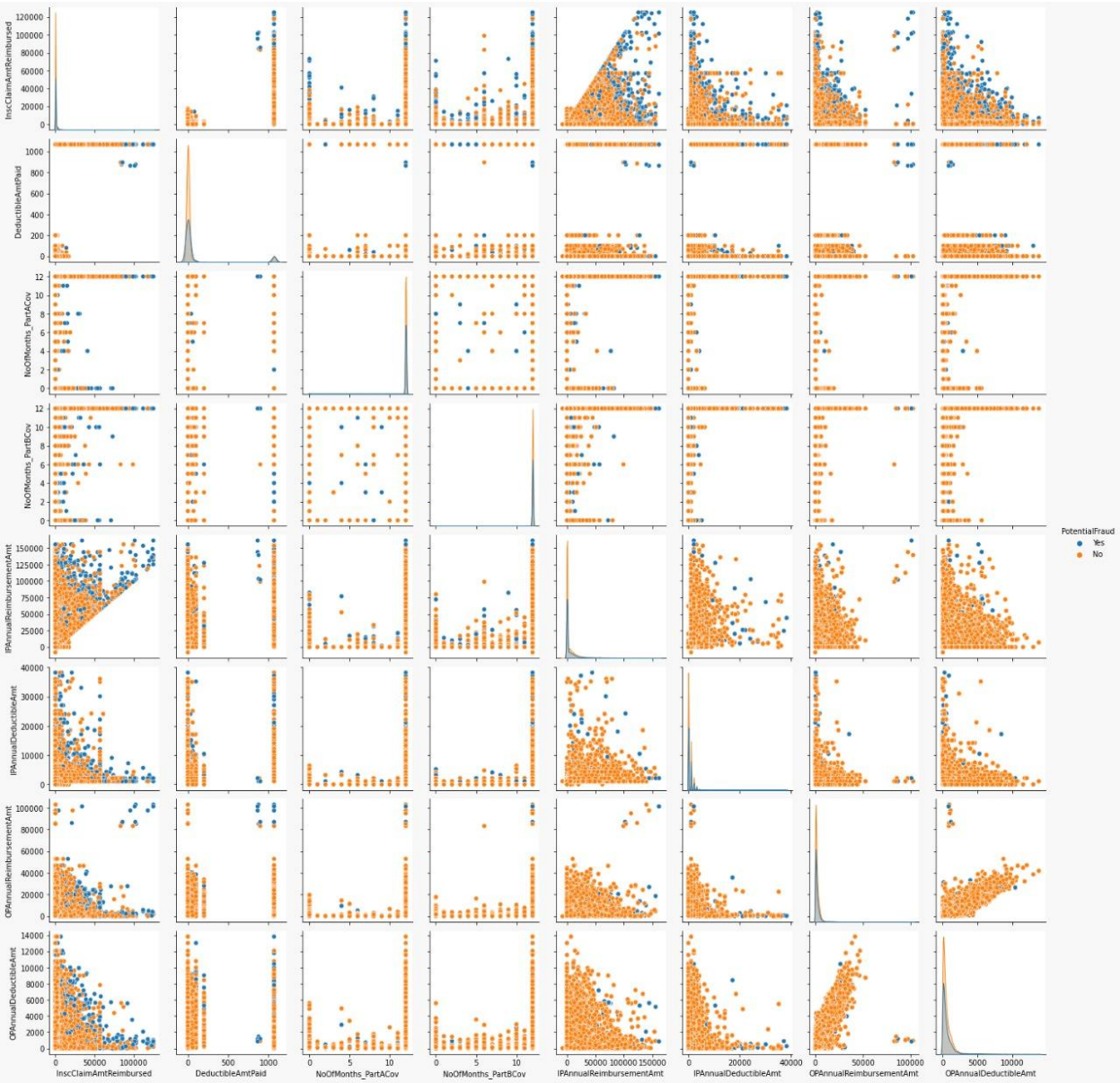

Top 20 ClmAdmitDiagnosisCode involved in Fraud/Not Fraud for patients got admitted in Hospital
PotentialFraud
No
Yes
count
0
500
1000
1500
2000
2500
V7612
42731
78605
4019
25000
78900
78650
V5883
7295
78079
5990
7802
7242
V5861
2724
486
7862
V571
4280
41401
ClmAdmitDiagnosisCode


Top 20 Attending Physician involved in Fraud/Not Fraud
PotentialFraud
No
Yes
count
0
500
1000
1500
2000
2500
PHY330576
PHY350277
PHY412132
PHY423534
PHY314027
PHY327046
PHY338032
PHY337425
PHY357120
PHY341578
PHY432650
PHY347064
PHY351121
PHY383481
PHY415321
PHY344389
PHY433436
PHY375943
PHY375453
PHY387126
AttendingPhysician

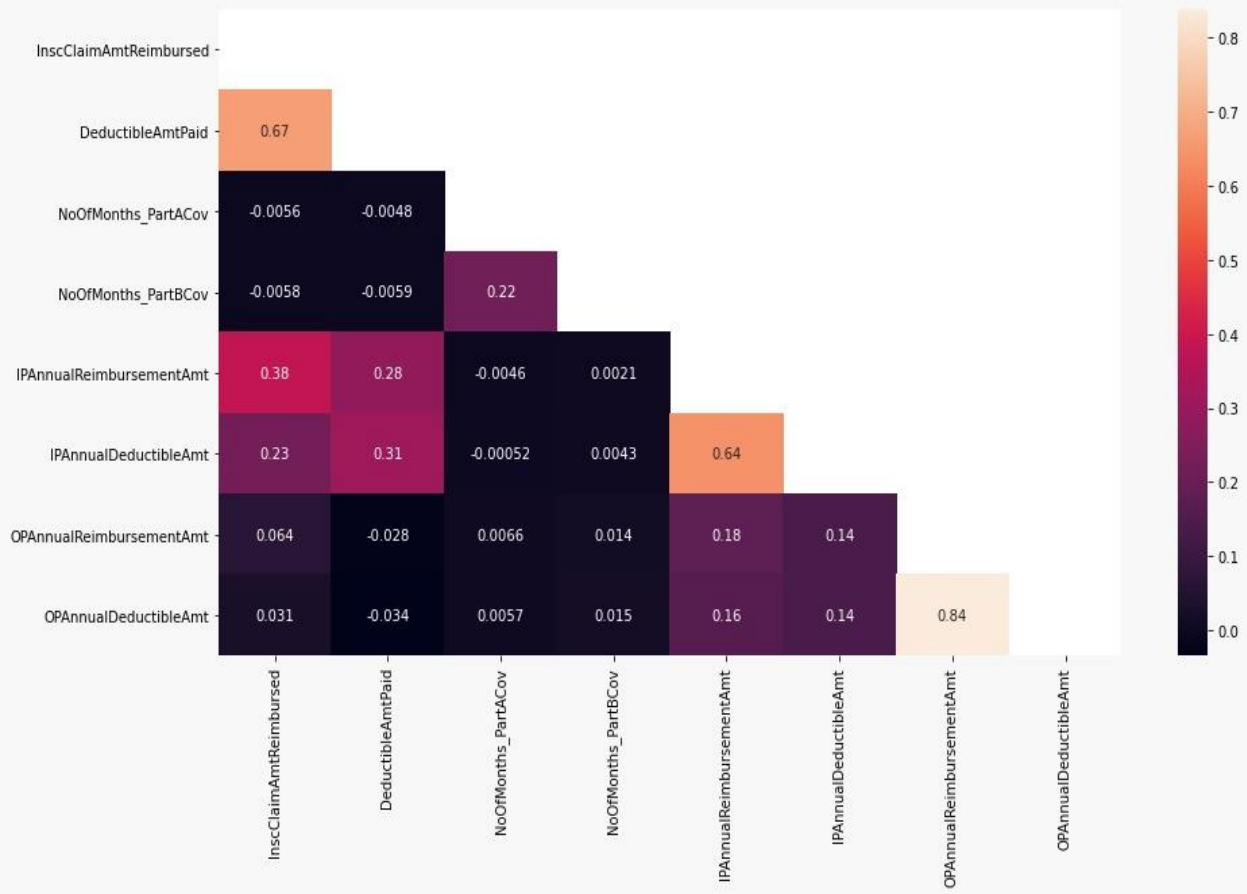

Top 20 ClmProcedureCode_1 involved in Fraud/Not Fraud
PotentialFraud
No
Yes
count
0
100
200
300
400
500
600
9904.0
8154.0
66.0
3893.0
3995.0
4516.0
3722.0
8151.0
8872.0
9671.0
4513.0
9390.0
9672.0
5123.0
7935.0
8152.0
9339.0
3491.0
3812.0
3950.0
ClmProcedureCode_1

3.4. Multi Variate Analysis

- For **Op Annual Deductible Amnt** and **Op Annual Reimbursed Amnt** there is high correlation that we find from the below pairplot.
- For **Op Annual Reimbursed Amnt** and **Op Annual Deductible Amnt** there is high correlation that we find from the below pairplot.
- Rest of the variables are having Homoskedasticity like same constant variance I can observe.
- Off diagonals distribution is not even visible, In the sense there is large amount of skewness in the data.



3.5. Numerical Features Corelation

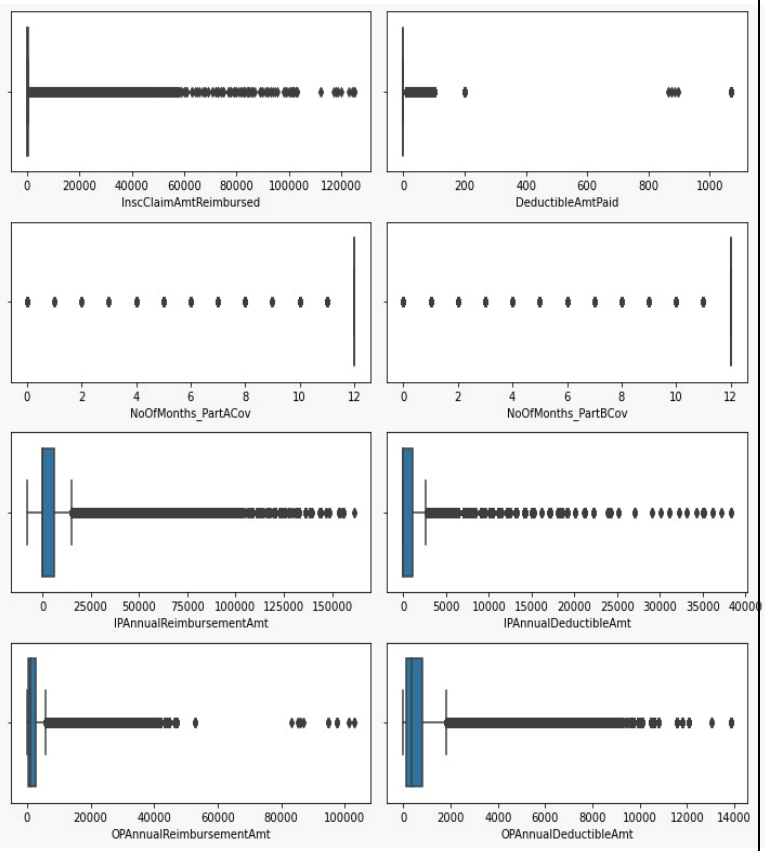


- ❖ OP Annual Deductible Amount and OP Annual Reimbursement Amount are highly corelated (0.84)
- ❖ IP Annual Deductible Amount and IP Annual Reimbursement Amount are moderately corelated (0.64)
- ❖ Deductiable Amt. Paid and Inc Claim Amt.Reimbursed are moderately corelated (0.67)

4. Data Pre-processing

4.1.Check For Outliers:

All the numerical variables have outliers.



4.2.Missing Values Treatment and Feature Engineering

First, we will run a check for the presence of missing values and their percentage for each column. Then choose the right approach to treat them. We have sorted the percentage of missing values in descending order and also checked for duplicates in each column and created a data frame with Total Missing Values, Percentage Of Missing Values, Duplicated. We will only show those columns which have missing values below.

	Total Missing Values	Percentage of Missing Values	Duplicated
ClmProcedureCode_6	558211	100	TRUE
ClmProcedureCode_5	558202	99.99839	TRUE
ClmProcedureCode_4	558093	99.97886	TRUE
ClmProcedureCode_3	557242	99.82641	TRUE
DOD	554080	99.25996	TRUE
ClmDiagnosisCode_10	553201	99.10249	TRUE
ClmProcedureCode_2	552721	99.0165	TRUE
ClmProcedureCode_1	534901	95.82416	TRUE

AdmissionDt	517737	92.74934	TRUE
DischargeDt	517737	92.74934	TRUE
DiagnosisGroupCode	517737	92.74934	TRUE
ClmDiagnosisCode_9	516396	92.5091	TRUE
ClmDiagnosisCode_8	504767	90.42584	TRUE
ClmDiagnosisCode_7	492034	88.14481	TRUE
ClmDiagnosisCode_6	473819	84.8817	TRUE
ClmDiagnosisCode_5	446287	79.94952	TRUE
OperatingPhysician	443764	79.49754	TRUE
ClmAdmitDiagnosisCode	412312	73.86311	TRUE
ClmDiagnosisCode_4	393675	70.52441	TRUE
OtherPhysician	358475	64.21855	TRUE
ClmDiagnosisCode_3	315156	56.45822	TRUE
ClmDiagnosisCode_2	195606	35.04159	TRUE
ClmDiagnosisCode_1	10453	1.872589	TRUE
AttendingPhysician	1508	0.270149	TRUE
DeductibleAmtPaid	899	0.16105	TRUE

Interpretation :

- We can see that there are missing values present in 'AttendingPhysician', 'OperatingPhysician', 'OtherPhysician'. But, instead of treating those missing values, we will count the total number of physicians for each claim records and create a new feature named 'TotalPhysicianCount'.

- Also, we will count the total number of unique physicians for each claim records and store it in a new feature named 'UniquePhysicianCount'. Now, we will substract the 'TotalPhysicianCount' and 'UniquePhysicianCount' and store it in a new variable called 'RepeatedPhysician'. Now, we will check if a claim record is having at least one repeated physician or not for different procedures and store it in the same column named 'RepeatedPhysician'. Basically, 'RepeatedPhysician' will store 'Yes' and 'No' values.

- If the RepeatedPhysician count is 0, then it will take 'no'(Here, we are representing 'no' with 0) and if the RepeatedPhysician count is more than 0, then it will take 'yes'(Here, we are representing 'Yes' with 1)

- Now, we have retained only two features 'TotalPhysicianCount' and 'RepeatedPhysician' out of all the features we have worked with and dropped the rest of them.

- All the Claim Diagnosis Codes and the Claim Procedure Codes are basically the diagnosis codes and procedure codes each patient has gone through for each claim records. So, we are going to count the Diagnosis Codes for each claim and store it in a new variable 'ClmDiagnosisCount' and also count the Procedure Codes for each claim and store it in a new variable 'ClmProcedureCount'. After that, we have dropped all the Diagnosis and Procedure codes.

- Diagnosis Group Code is also present in Inpatient Data only. So for other outpatient data, we can impute that feature with 0.

- There are null values present in the 'DOD' column. It means that the patient is alive till the dataset was recorded. So we will create a new columns 'Age' after substracting 'DOD' and 'DOB' and count the total days and divide it by 365 to get that person's age. There will be null values in the Age column because most of the patients are alive. So, we will fill those null values after calculating the age from their 'DOB' and max value of 'DOD'(max value of 'DOD' is almost similar to the last record of the dataset)

- Also, based on the null values in 'DOD', we will be creating a new feature named 'Alive' which will take two values '0'(no, the person is not alive) and '1'(yes, the person is alive).

- We can see that the features 'AdmissinDt' and 'DischargeDt' has 92 % of missing values. 'AdmissionDt' and 'DischargeDt' is basically available for inpatient data. So based on that, we will be creating a new feature 'PatientVisitType' which will take only two values 1 (Inpatient/The beneficiary admitted in the hospital) and 0 (Outpatient/The beneficiary came for outdoor check-up).

- We will also be creating a new feature 'DaysStayedAtHospital' taking the difference in days between 'AdmissionDt' and 'DischargeDt'. There will be null values for those patients who didn't admit in the hospital(outpatients). We will fill those null values with 0 as they haven't stayed at the hospital. Then we will drop those two columns 'AdmissionDt' and 'DischargeDt'.

- We can see that there are 0.161% missing values present in the feature 'DeductibleAmtPaid'. We will fill the null values with 0 as some of the beneficiaries' medical expense is so low that they don't reach till the deductible amount level.

- We will also be creating a new feature 'DaysToSettleClaim' taking the difference in days between 'ClaimStartDt' and 'ClaimEndDt' and also drop those two columns.

4.3.Encoding of Categorical Variable

- Interpretation: After checking all the value counts of the renal disease and chronic conditions, There are redundant values we can say. We will encode them properly. 'Y'(yes) in 'RenalDiseaseIndicator' will be encoded as 1. Also, practically, if we see the value counts in the chronic conditions, we can see that there are 2 values 1 and 2. Basically, 2 is representing 0 (Chronic Condition not present) and we will keep 1(yes chronic condition present) as it is.

- Now, we will encode the target variable 'PotentialFraud' as needed for our future model evaluation. Basically, there are two values present in the target variable 'Yes' and 'No'. We will encode them with 1 and 0 respectively**.**

- Now we will change the data types of some object variables 'Gender', 'Race', 'State', 'County' as they are already encoded.

4.4.Transformation Techniques:

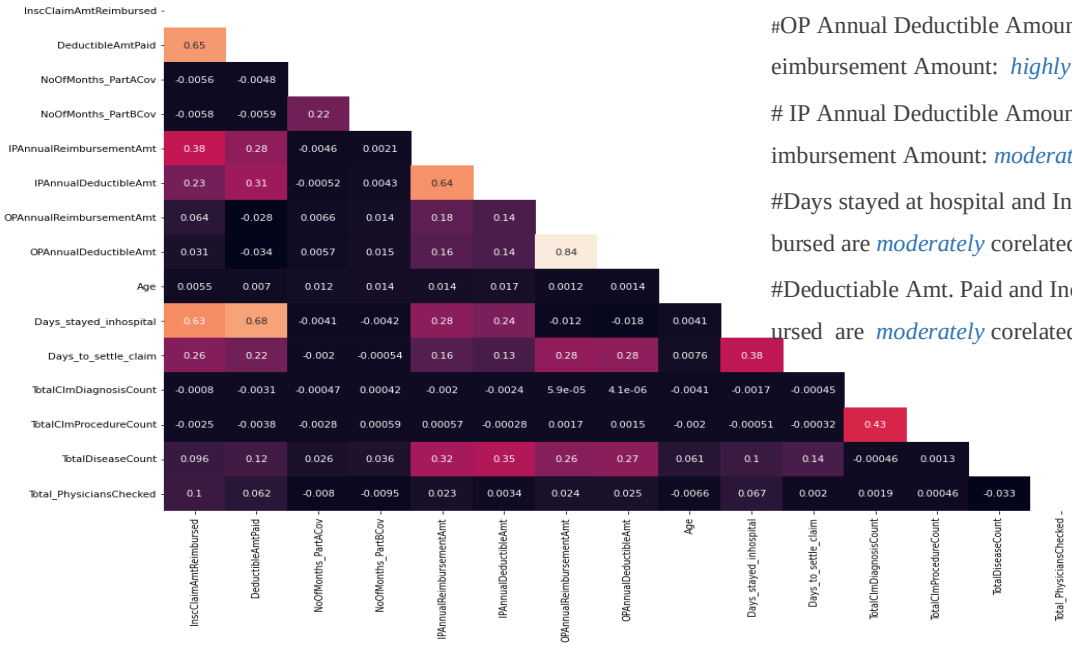
The step will be performed once we are done with our Baseline Model and improve the accuracy with other techniques.

4.5.Dropping Of Columns:

We can say that the feature 'DiagnosisGroupCode' is representing only an ID for the diagnosis done on the inpatient data. We have already extracted required features from the diagnosis done on the patients. So, we will drop it.

We can say that the features 'BeneID','ClaimID','Provider' is working as ID only. We have already created new features based on those columns. So we will drop those features for model building now and will add if needed for later.

4.6.Checking the Corelation between Numerical Features after Feature Engineering:



#OP Annual Deductible Amount and OP Annual Reimbursement Amount: *highly* corelated (0.84)

IP Annual Deductible Amount and IP Annual Reimbursement Amount: *moderately* corelated (0.64)

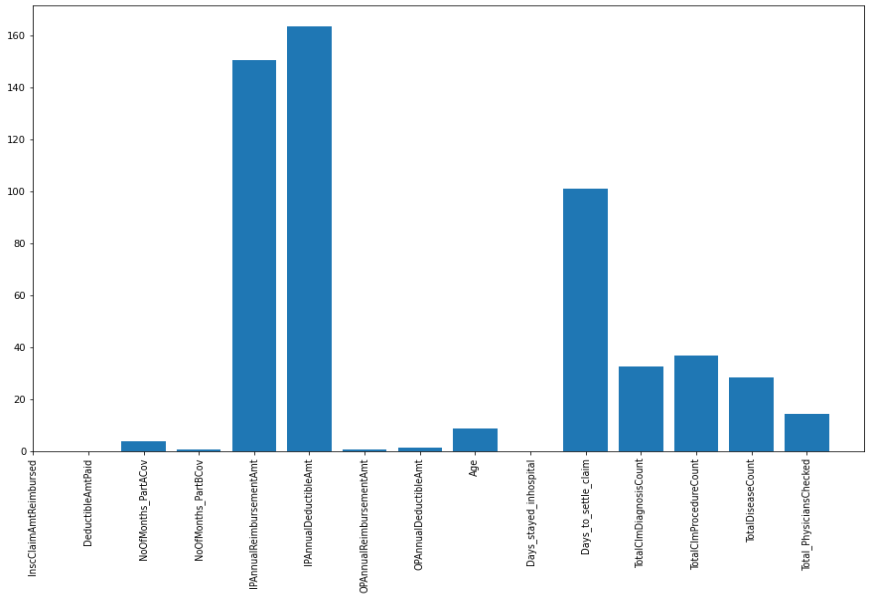
#Days stayed at hospital and Inc Claim Amt.Reimbursed are *moderately* corelated (0.63)

#Deductiable Amt. Paid and Inc Claim Amt.Reimbursed are *moderately* corelated(0.65)

5. Feature selection Techniques:

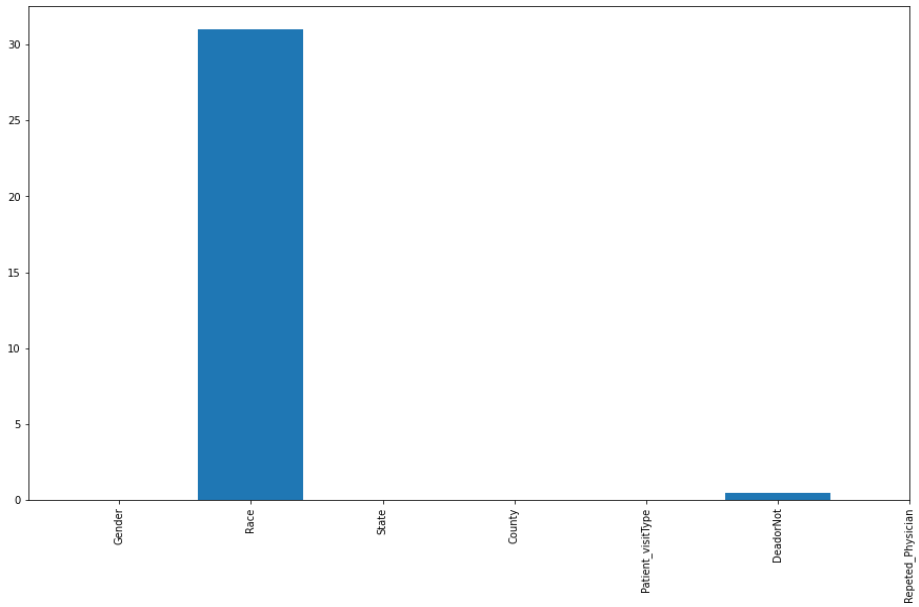
4.7. Using Select KBest

4.7.1. Performing f-one_way ANOVA using Select KBest for Continuous variable



Here we can find the top five continuous independent variables are IPAnnualDeductibleAmount, IPAnnualReimb.Amount, DaystoSettleClaim, Total Claim procedure count, TotalClaimDiagonisisCount

4.7.2. Performing Chi-Square Test using Select KBest for Categorical variable



Here we can find the important categorical features are Race and DeadorNot

4.8. Using Recursive Feature Elimination (RFE)

- RFE is a wrapper-type feature selection algorithm. This means that a different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. This is in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score.
- Technically, RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally.
- RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains. This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.

Here Top 8 features are chosen though RFE

```
from sklearn.feature_selection import RFE

rf = RandomForestClassifier(random_state= 8)

rfe_model = RFE(estimator=rf, n_features_to_select = 8)

rfe_model = rfe_model.fit(xtrain, ytrain)

feat_index = pd.Series(data = rfe_model.ranking_, index = xtrain.columns)

signi_feat_rfe = feat_index[feat_index==1].index
print(signi_feat_rfe)
```

xecuted in 1h 26m 41s, finished 18:04:52 2022-03-22

['InscClaimAmtReimbursed', 'State', 'County', 'IPAnnualReimbursementAmt', 'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age', 'Total_PhysiciansChecked']

5. Statistical Tests

5.1.1. Numerical Features Vs the Categorical Target Feature

Target Feature: *'PotentialFraud'*

Numerical Features:
'InscClaimAmtReimbursed','DeductibleAmtPaid','NoOfMonths_PartACov', 'NoOfMonths_PartBCov','IPAnnualReimbursementAmt','IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age', 'Days_stayed_inhospital', 'Days_to_settle_claim','Total_PhysiciansChecked','TotalClmDiagnosisCount', 'TotalClmProcedureCount', 'TotalDiseaseCount'

- We will first Check for Normality of the Data, so we will go for Shapiro-Wilk Test

Null Hypothesis: Data is normal.
Alt. Hypothesis: Data is not not normal.

```
[11] p_val= []
     for i in final_df_num.columns:
       pvalue = stats.shapiro(final_df_num[i])[1]
       p_val.append(pvalue)

     print(p_val)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
## As for all the numerical featutes we have found that the p-value <0.05
## so we reject the H0 , so we can conclude that data is not normal.
## Same thing we are able to conclude from the distribution plots done in EDA.
```

- ❖ We will now Check for equality of Variance, so we will go for Levene Test

Null Hypothesis: Variances are equal
Alt. Hypothesis: Variances are not equal

```
[13] p_val= []
     for i in final_df_num.columns:
       pvalue = stats.levene(final_df_num[i],final_df['PotentialFraud'])[1]
       p_val.append(pvalue)

     print(p_val)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.3198887789700883e-24, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
## As for all Numerical features the p-value <0.05
## so we reject the H0 , so we can conclude that variences are not equal.
```

As we can see that the Data distribution is not normal and also, they don't have equal variances so we have to go for Non-Parametric Test instead of Two sample independent T-Test (Parametric-Test). So here we will go for Mann-Whitney-U test.

Null Hypothesis: The means are equal, so they are not statistically significant.

Alt. Hypothesis: The means are not equal, so they are far away, so, statistically significant.

```
[17] pval ={}

     for i in final_df_num.columns:
       if i == 'PotentialFraud':
         continue
       else:
         p_value = stats.mannwhitneyu(final_df_num[final_df_num['PotentialFraud']== 0][i],  final_df_num[final_df_num['PotentialFraud']== 1][i])[1]
         pval.update({i:p_value})


     {key:value for (key,value) in pval.items()}
```

```
{'Age': 1.1683727437179946e-10,
 'Days_stayed_inhospital': 0.0,
 'Days_to_settle_claim': 0.0,
 'DeductibleAmtPaid': 0.0,
 'IPAnnualDeductibleAmt': 1.1647762070853345e-257,
 'IPAnnualReimbursementAmt': 3.3208283387829644e-245,
 'InscClaimAmtReimbursed': 7.575700740775726e-290,
 'NoOfMonths_PartACov': 8.567584705443157e-06,
 'NoOfMonths_PartBCov': 0.19089833755215163,
 'OPAnnualDeductibleAmt': 3.7664505031202693e-05,
 'OPAnnualReimbursementAmt': 3.5579705823098955e-05,
 'TotalClmDiagnosisCount': 1.9669585548006024e-13,
 'TotalClmProcedureCount': 9.04748623460157e-45,
 'TotalDiseaseCount': 1.3808911391214714e-27,
 'Total_PhysiciansChecked': 1.8221120837812875e-21}
```

```
## In above whereever we find p-value >0.05, there we fail to reject the H0, so those features are insignificant.

## So the Numerical significant features are:  InscClaimAmtReimbursed, DeductibleAmtPaid,NoOfMonths_PartACov,IPAnnualReimbursementAmt,
##                                              IPAnnualDeductibleAmt, OPAnnualReimbursementAmt,OPAnnualDeductibleAmt, Age,
##                                              Days_stayed_inhospital,Days_to_settle_claim,Total_PhysiciansChecked
##                                              ,TotalClmDiagnosisCount, 'TotalClmProcedureCount','TotalDiseaseCount'
```

5.1.2. Categorical Features Vs the Categorical Target Feature

Categorical Features: *'Gender', 'Race', 'State', 'County', 'Patient_visitType', 'DeadorNot', 'Repeted_Physician'*

Target: *'PotentialFraud'*

Here we check the P value to be < 0.05(with 95% confidence interval) to consider that variable to be a significant variable.

- **County vs Target Feature:** As County has more than two subcategories and deg. Of freedom is more than 2, so we will go for chi Square Test.

```
[21] stats.chi2_contingency(pd.crosstab(final_df_cat['PotentialFraud'],final_df_cat['County']))[1]

     0.0

[22] ## as p-value<0.5, County is significant
```

- **State vs Target Feature:** As State has more than two subcategories. Deg. of freedom is more than 2, so we will go for chi Square Test.

```
[23] stats.chi2_contingency(pd.crosstab(final_df_cat['PotentialFraud'],final_df_cat['State']))[1]

     0.0

[24] ## as p-value<0.5, State is significant
```

- **Gender vs Target Feature:** As Gender and the Target both has only two subcategories. So, Deg. of freedom is less than 2, so we will go for 2 sample proportion Test.

```
[27] n_1 = len(df_nl)
     n_2 = len(df_sg)


     l_1 = len(df_nl[df_nl['Gender'] == 1])
     l_2 = len(df_sg[df_sg['Gender'] == 1])

[28] import statsmodels.api as sm
     sm.stats.proportions_ztest(count = np.array([l_1, l_2]),nobs = np.array([n_1, n_2]),alternative='two-sided')[1]

     0.7313608396662625

[29] ## So Gender is not significant
```

- **Race vs Target Feature:** As Race and the Target both has only two subcategories. So, Deg. of freedom is less than 2, so we will go for 2 sample proportion Test.

```
[30] df_nl = final_df_cat[final_df_cat['PotentialFraud'] == 0]
     df_sg = final_df_cat[final_df_cat['PotentialFraud'] == 1]

     n_1 = len(df_nl)
     n_2 = len(df_sg)


     l_1 = len(df_nl[df_nl['Race'] == 1])
     l_2 = len(df_sg[df_sg['Race'] == 1])

     sm.stats.proportions_ztest(count = np.array([l_1, l_2]),nobs = np.array([n_1, n_2]),alternative='two-sided')[1]

     2.1704265129162852e-20

[31] ## So Race is significant
```

- **Patient_visitType vs Target Feature:** As Patient_visitType and the Target both has only two subcategories. So, Deg. of freedom is less than 2, so we will go for 2 sample proportion Test.

```
[32] df_nl = final_df_cat[final_df_cat['PotentialFraud'] == 0]
     df_sg = final_df_cat[final_df_cat['PotentialFraud'] == 1]

     n_1 = len(df_nl)
     n_2 = len(df_sg)


     l_1 = len(df_nl[df_nl['Patient_visitType'] == 1])
     l_2 = len(df_sg[df_sg['Patient_visitType'] == 1])

     sm.stats.proportions_ztest(count = np.array([l_1, l_2]),nobs = np.array([n_1, n_2]),alternative='two-sided')[1]

     0.0

[33] ## so Patient_visitType is significant
```

- **DeadorNot vs Target Feature:** As DeadorNot and the Target both has only two subcategories. So, Deg. of freedom is less than 2, so we will go for 2 sample proportion Test.

```
[34] df_nl = final_df_cat[final_df_cat['PotentialFraud'] == 0]
     df_sg = final_df_cat[final_df_cat['PotentialFraud'] == 1]

     n_1 = len(df_nl)
     n_2 = len(df_sg)


     l_1 = len(df_nl[df_nl['DeadorNot'] == 1])
     l_2 = len(df_sg[df_sg['DeadorNot'] == 1])

     sm.stats.proportions_ztest(count = np.array([l_1, l_2]),nobs = np.array([n_1, n_2]),alternative='two-sided')[1]

     0.32230337793643193

[35] ## So DeadorNot is not significant
```

- **Repeted_Physician vs Target Feature:** As Repeted_Physician and the Target both has only two subcategories. So, Deg. of freedom is less than 2, so we will go for 2 sample proportion Test.

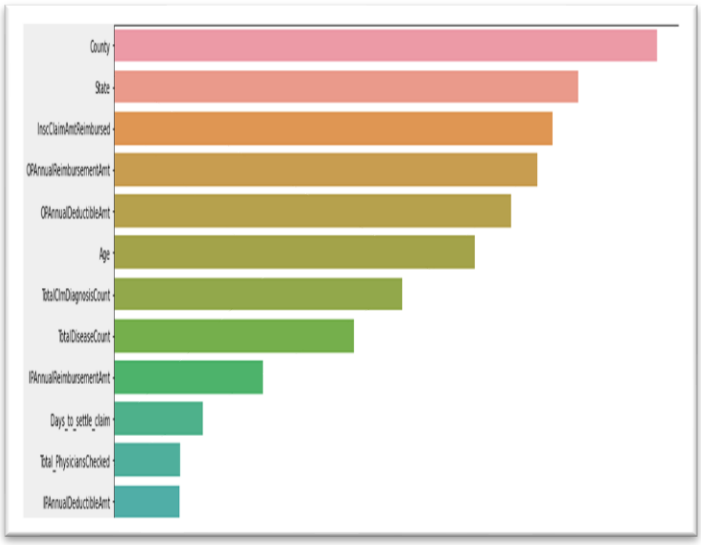
```
[36] df_nl = final_df_cat[final_df_cat['PotentialFraud'] == 0]
     df_sg = final_df_cat[final_df_cat['PotentialFraud'] == 1]

     n_1 = len(df_nl)
     n_2 = len(df_sg)


     l_1 = len(df_nl[df_nl['Repeted_Physician'] == 1])
     l_2 = len(df_sg[df_sg['Repeted_Physician'] == 1])

     sm.stats.proportions_ztest(count = np.array([l_1, l_2]),nobs = np.array([n_1, n_2]),alternative='two-sided')[1]

     0.0

[37] ## so Repeted_Physician is significant
```

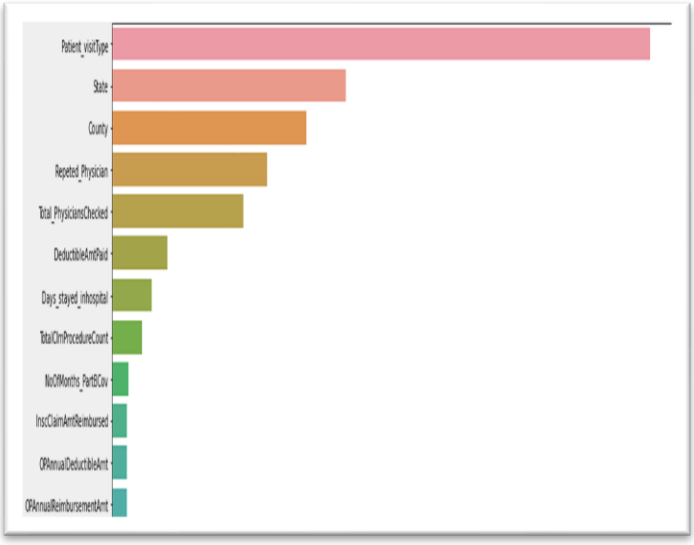
So, we find all the Numerical and Categorical Significant Features are as below:

'County','State','Race','Patient_visitType','Repeted_Physician','InscClaimAmtReimbursed', 'DeductibleAmtPaid','NoOfMonths_PartACov', 'IPAnnualReimbursementAmt','IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt','OPAnnualDeductibleAmt', 'Age', 'Days_stayed_inhospital', 'Days_to_settle_claim','Total_PhysiciansChecked','TotalClmDiagnosisCount', 'TotalClmProcedureCount','TotalDiseaseCount'

5.2.Using Feature importance from Best Performing Baseline Model

Random Forest Model **XG Boost Model**





From both the above Base-line models we have created two models with top 10 and 8 features respectively.

6. Model Building

6.1.1. Train-Test Split

Before applying various classification techniques to predict the fraudulent claims of the Healthcare Providers, let us split the dataset in train and test set.

'test_size' returns the proportion of data to be included in the test set. We have set the test size of 0.2

Shape after splitting the dataset :

Xtrain (446568, 22)

ytrain (446568,)

Xtest (111643, 22)

ytest (111643,)

6.1.2. Machine learning Base Model

- **Algorithm**
  - Logistic Regression (Linear Model) and Random Forest (Non-Linear Model)
- **Properties**
  - Quick and Easy implementation for a Baseline Model building.
  - Not to be overly sensitive to missing features or inherent noise present in the data

Ranking on Important Features from Statistical tests and Feature Importance

- **Evaluation Matrix:**

- Accuracy
- Recall score for Positive Class
- ROC-AUC Score
- Cohen kappa score
- Confusion Matrix
- F1 Score

Summary Of Initial Findings:

	Logistic Regression	Random Forest	Decision Tree	10-Fold CV For RF
Accuracy Score	**62.94**	**73.75**	**70**	**73.32**
AUC_ROC	**52.96**	**72.42**	**68.45**	**72.27**
Cohen Kappa	**7.02**	**42.13**	**36.70**	**NA**
Recall for Class:1	**11**	**55**	**61**	**55.55**
FN	**37853**	**19007**	**16400**	**N. A**
F1 Score	**48.68**	**61.79**	**58.06**	**61.04**

Apart from the above model also Cat Boost, Light GBM, ADA Boost, Gradient Boost and XG Boost has been performed.

From our Bassline Model (Logistic Regression), we can see that the train and test accuracy scores are 63.00 and 62.94 respectively. So, we can say that our model accuracy score is low and also it is slightly underfitted.

From the EDA and the baseline models we can also see that the data is not fit for Linear model. So, we take the Random Forest and the XG Boost model for the best Baseline Model, on which we will farther evaluate and will try to improve the performance and robustness of our future models.

6.1.3. Building Model with the Features from Select KBest:

Here we work with the all the 14 significant features we have got from the Select KBest

Train Test Split for K-Best Features

In [36]:
```
x=df[['NoOfMonths_PartACov', 'NoOfMonths_PartBCov','IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
      'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age', 'Days_to_settle_claim','TotalClmDiagnosisCount',
     'TotalClmProcedureCount','TotalDiseaseCount','Total_PhysiciansChecked','Race','DeadorNot']]
y=df['PotentialFraud']
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=10)
print('xtrain',xtrain.shape)
print('ytrain',ytrain.shape)
print('xtest',xtest.shape)
print('ytest',ytest.shape)
```

```
xtrain (446568, 14)
ytrain (446568,)
xtest (111643, 14)
ytest (111643,)
```

Decision Tree Model

In [8]:
```
dt = DecisionTreeClassifier(random_state=10)
dt.fit(xtrain,ytrain)

y_pred_dt = dt.predict(xtest)
y_train_dt = dt.predict(xtrain)
ypred_proba_dt = dt.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_dt))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_dt), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_dt))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_dt),''

print(confusion_matrix(ytest,y_pred_dt),'\n')

print(classification_report(ytest,y_pred_dt))
```

executed in 4.76s, finished 16:38:14 2022-03-23

```
Train Accuracy:  0.9713705415524624
Test Accuracy:  0.6223677256970881

ROC-AUC Score:  0.6019552158301816
Cohen Cappa Score:  0.1968920336232599

[[48368 20856]
 [21304 21115]]

              precision    recall  f1-score   support

           0       0.69      0.70      0.70     69224
           1       0.50      0.50      0.50     42419

    accuracy                           0.62    111643
   macro avg       0.60      0.60      0.60    111643
weighted avg       0.62      0.62      0.62    111643
```

XG Boost Model

[9]:
```
from xgboost import XGBClassifier
xgb = XGBClassifier(random_state=10)
xgb.fit(xtrain,ytrain)

y_pred_xgb = xgb.predict(xtest)
y_train_xgb = xgb.predict(xtrain)

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_xgb))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_xgb), '\n\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,y_pred_xgb))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_xgb),'\n\n')

print(classification_report(ytest,y_pred_xgb))
```

executed in 12.3s, finished 16:38:27 2022-03-23

```
[16:38:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.
3.0, the default evaluation metric used with the objective 'binary:logistic'
ly set eval_metric if you'd like to restore the old behavior.
Train Accuracy:  0.6440094229770158
Test Accuracy:  0.6343613123975529


ROC-AUC Score:  0.5363671539857882
Cohen Cappa Score:  0.08569624442047508


              precision    recall  f1-score   support

           0       0.64      0.94      0.76     69224
           1       0.59      0.13      0.21     42419

    accuracy                           0.63    111643
   macro avg       0.61      0.54      0.49    111643
weighted avg       0.62      0.63      0.55    111643
```

From all the models we build using the above important features we can see that the above two models are working best among them but from the business perspective the performance of these two models are not well at all. The Recall score for Positive class is found to be very low also the TN is found to be very high which we can't afford in this business problem. So we will try building models from other feature selection techniques.

6.1.4. Building Model with the best Features from RFE:

Best Model from the best 8 features selected by RFE

```
[48]:
X1 = final_df[['InscClaimAmtReimbursed', 'State', 'County', 'IPAnnualReimbursementAmt',
       'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age',
       'Total_PhysiciansChecked']]
y1 = final_df['PotentialFraud']

xtrain,xtest,ytrain,ytest = train_test_split(X1,y1, train_size=0.8,random_state=10 )
```

executed in 485ms, finished 18:05:23 2022-03-22

['InscClaimAmtReimbursed', 'State', 'County', 'IPAnnualReimbursementAmt', 'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age', 'Total_PhysiciansChecked']

Random Forest with Top 8 Feature

```
[49]:
rf10 = RandomForestClassifier(random_state=10)
rf10.fit(xtrain,ytrain)

y_pred_rf = rf10.predict(xtest)
y_train_rf = rf10.predict(xtrain)
ypred_proba_rf = rf10.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_rf))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_rf), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_rf))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_rf),'\n')

print(confusion_matrix(ytest,y_pred_rf),'\n')

print(classification_report(ytest,y_pred_rf))
```

executed in 4m 5s, finished 18:09:33 2022-03-22

```
Train Accuracy:  0.9846115261281596
Test Accuracy:  0.736884533737001

ROC-AUC Score:  0.7879445944640695
Cohen Cappa Score:  0.4315768395901963

[[56464 12760]
 [16615 25804]]

              precision    recall  f1-score   support

           0       0.77      0.82      0.79     69224
           1       0.67      0.61      0.64     42419

    accuracy                           0.74    111643
   macro avg       0.72      0.71      0.72    111643
weighted avg       0.73      0.74      0.73    111643
```

Random Forest with Top 5 Features

```
rf10 = RandomForestClassifier(random_state=10)
rf10.fit(xtrain,ytrain)

y_pred_rf = rf10.predict(xtest)
y_train_rf = rf10.predict(xtrain)
ypred_proba_rf = rf10.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_rf))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_rf), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_rf))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_rf),'\n')

print(confusion_matrix(ytest,y_pred_rf),'\n')

print(classification_report(ytest,y_pred_rf))
```

executed in 3m 3s, finished 17:36:06 2022-03-23

```
Train Accuracy:  0.9683922717256946
Test Accuracy:  0.7396343702695198

ROC-AUC Score:  0.7983471815653977
Cohen Cappa Score:  0.44126332778270116

[[55896 13328]
 [15740 26679]]

              precision    recall  f1-score   support

           0       0.78      0.81      0.79     69224
           1       0.67      0.63      0.65     42419

    accuracy                           0.74    111643
   macro avg       0.72      0.72      0.72    111643
weighted avg       0.74      0.74      0.74    111643
```

We run the model both with top 5, 8 and top 10 features. And find the best result with top 5 features in RF. From all the models we build using the top 8 and top 5 important features we can see that the above two models are working best among them but from the business perspective the performance of these two models are not much improved from the baseline model performance.

The Recall score for Positive class is found to be 61 also the FN is found to be around 15700, which is still very high which we can't afford in this business problem. So, we will try building models from other feature selection techniques.

6.1.4. Building Model with the Statistically Significant Features:

Decision Tree

```
[ ] dt = DecisionTreeClassifier(random_state=10)
    dt.fit(xtrain,ytrain)

    y_pred_dt = dt.predict(xtest)
    y_train_dt = dt.predict(xtrain)
    ypred_proba_dt = dt.predict_proba(xtest)[:,1]

    print ('Train Accuracy: ',accuracy_score(ytrain,y_train_dt))
    print ('Test Accuracy: ',accuracy_score(ytest,y_pred_dt), '\n')

    print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_dt))
    print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_dt),'\n')

    print(confusion_matrix(ytest,y_pred_dt),'\n')

    print(classification_report(ytest,y_pred_dt))
```

```
Train Accuracy:  0.9980786800666416
Test Accuracy:  0.7019159284505074

ROC-AUC Score:  0.6861375832704092
Cohen Cappa Score:  0.36973043177954457

[[52173 17051]
 [16228 26191]]

              precision    recall  f1-score   support

           0       0.76      0.75      0.76     69224
           1       0.61      0.62      0.61     42419

    accuracy                           0.70    111643
   macro avg       0.68      0.69      0.68    111643
weighted avg       0.70      0.70      0.70    111643
```

Cat Boost

```
from catboost import CatBoostClassifier
cat = CatBoostClassifier(random_state=10)
cat.fit(xtrain,ytrain)

y_pred_cat = cat.predict(xtest)
y_train_cat = cat.predict(xtrain)
ypred_proba_cat = cat.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_cat))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_cat), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_cat))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_cat),'\n')

print(confusion_matrix(ytest,y_pred_cat),'\n')

print(classification_report(ytest,y_pred_cat))
```

```
Learning rate set to 0.139456
Train Accuracy:  0.7759445370022034
Test Accuracy:  0.7693272305473697

ROC-AUC Score:  0.825721487373845
Cohen Cappa Score:  0.493747647943528

[[59958  9266]
 [16487 25932]]

              precision    recall  f1-score   support

           0       0.78      0.87      0.82     69224
           1       0.74      0.61      0.67     42419

    accuracy                           0.77    111643
   macro avg       0.76      0.74      0.75    111643
weighted avg       0.77      0.77      0.76    111643
```

From all the model we built we can find only DT and Cat Boost is working best. Here we can find a slight improvement in the Recall score. So, to farther investigate we will run Randomized search CV to check how much the model is improving and we will also check for the FN score if it's decreasing significantly.

Here we run a Randomized Search CV to on the Decision Tree to tune the model and check if the performance of the Model is improving.

```
[ ]  from sklearn.model_selection import RandomizedSearchCV

     params = {'n_estimators': [5,25,50,100,150],
               'max_depth': [3,6,8,10,12,16],
               'criterion': ['gini','entropy'],
               'min_samples_split':[5,10,15],
               'min_samples_leaf':[5,10,15]
              }

     rf = RandomForestClassifier(random_state = 10)

     randmcv = RandomizedSearchCV(estimator=rf, param_distributions = params,
                                  cv = 5, n_iter = 10, n_jobs=-1)


     randmcv.fit(xtrain, ytrain)

     randmcv.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 16,
 'min_samples_leaf': 10,
 'min_samples_split': 10,
 'n_estimators': 25}
```

```
dt = DecisionTreeClassifier(criterion= 'gini',max_depth=16,min_samples_leaf=15,min_samples_split=18,random_state=10)
dt.fit(xtrain,ytrain)

y_pred_dt = dt.predict(xtest)
y_train_dt = dt.predict(xtrain)
ypred_proba_dt = dt.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_dt))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_dt), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_dt))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_dt),'\n')

print(confusion_matrix(ytest,y_pred_dt),'\n')

print(classification_report(ytest,y_pred_dt))
```

```
Train Accuracy:  0.787671754357679
Test Accuracy:  0.7668819361715468

ROC-AUC Score:  0.8284939660405846
Cohen Cappa Score:  0.49989733011521986

[[57382 11842]
 [14184 28235]]

              precision    recall  f1-score   support

           0       0.80      0.83      0.82     69224
           1       0.70      0.67      0.68     42419

    accuracy                           0.77    111643
   macro avg       0.75      0.75      0.75    111643
weighted avg       0.76      0.77      0.77    111643
```

This is the best Recall score we have got so far which is 67. Here we can find the FN is 14184, which is still very costly from the business perspective. So, for farther evaluation we will consider the important feature from the Feature importance of the best Baseline Models i.e RF and XGB.

6.1.5. Building Model with the Feature selection from Best Performing Baseline Models (RF & XGB):

With top 10 Features from the RF model

```
from xgboost import XGBClassifier
xgb = XGBClassifier(random_state=10)
xgb.fit(xtrain,ytrain)

y_pred_xgb = xgb.predict(xtest)
y_train_xgb = xgb.predict(xtrain)
ypred_proba_xgb = xgb.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_xgb))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_xgb), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_xgb))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_xgb),'\n')

print(confusion_matrix(ytest,y_pred_xgb),'\n')

print(classification_report(ytest,y_pred_xgb))
```

executed in 2m 54s, finished 17:20:43 2022-03-22

```
licitly set eval_metric if you'd like to restore the old behavior.
Train Accuracy:  0.7746815714516042
Test Accuracy:  0.7686644035004434

ROC-AUC Score:  0.8252836180880692
Cohen Cappa Score:  0.49182765281647856

[[60018  9206]
 [16621 25798]]

              precision    recall  f1-score   support

           0       0.78      0.87      0.82     69224
           1       0.74      0.61      0.67     42419

    accuracy                           0.77    111643
   macro avg       0.76      0.74      0.74    111643
weighted avg       0.77      0.77      0.76    111643
```

With top 6 Features from the RF model

```
rf = RandomForestClassifier(random_state=10)
rf.fit(xtrain,ytrain)

y_pred_rf = rf.predict(xtest)
y_train_rf = rf.predict(xtrain)
ypred_proba_rf = rf.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_rf))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_rf), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_rf))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_rf),'\n')

print(confusion_matrix(ytest,y_pred_rf),'\n')

print(classification_report(ytest,y_pred_rf))
```

executed in 7m 25s, finished 17:39:24 2022-03-22

```
Train Accuracy:  0.9709652281399473
Test Accuracy:  0.7332389849789059

ROC-AUC Score:  0.7923083759309082
Cohen Cappa Score:  0.4278769948042419

[[55475 13749]
 [16033 26386]]

              precision    recall  f1-score   support

           0       0.78      0.80      0.79     69224
           1       0.66      0.62      0.64     42419

    accuracy                           0.73    111643
   macro avg       0.72      0.71      0.71    111643
weighted avg       0.73      0.73      0.73    111643
```

Using the baseline RF full model Feature importance, we select top 10, 6 and top 5 features to rebuild the model and we find that the above two models are giving the best performance where the scope of improvement of the Recall score and the FN is huge compare to the previous model, we built considering the Statistical Tests. So now we will work with the best features we got from the Baseline XGB full model.

With top 8 Features from the XGB Model

```
rf10 = RandomForestClassifier(random_state=10)
rf10.fit(xtrain,ytrain)

y_pred_rf = rf10.predict(xtest)
y_train_rf = rf10.predict(xtrain)
ypred_proba_rf = rf10.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_rf))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_rf), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_rf))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_rf),'\n')

print(confusion_matrix(ytest,y_pred_rf),'\n')

print(classification_report(ytest,y_pred_rf))
```

executed in 45.1s, finished 21:29:32 2022-03-23

```
Train Accuracy:  0.8163997420325684
Test Accuracy:  0.7800578630097722

ROC-AUC Score:  0.8450121461734943
Cohen Cappa Score:  0.5258467359219108

[[58647 10577]
 [13978 28441]]

              precision    recall  f1-score   support

           0       0.81      0.85      0.83     69224
           1       0.73      0.67      0.70     42419

    accuracy                           0.78    111643
   macro avg       0.77      0.76      0.76    111643
weighted avg       0.78      0.78      0.78    111643
```

With top 8 Scaled Features from the XGB

```
rf_sacled = RandomForestClassifier(random_state=100)
rf_sacled.fit(scaled_xtrain,ytrain)

y_pred_rf_scaled = rf_sacled.predict(scaled_xtest)
y_pred_train_scaled = rf_sacled.predict(scaled_xtrain)
ypred_proba_rf = rf_sacled.predict_proba(scaled_xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_pred_train_scaled))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_rf_scaled), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,y_pred_rf_scaled))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_rf_scaled),'\n')

print(confusion_matrix(ytest,y_pred_rf_scaled),'\n')

print(classification_report(ytest,y_pred_rf_scaled))
```

executed in 46.8s, finished 21:31:06 2022-03-23

```
Train Accuracy:  0.8164176564375414
Test Accuracy:  0.7800757772542838

ROC-AUC Score:  0.7600705459178116
Cohen Cappa Score:  0.5270474904595046

[[58383 10841]
 [13712 28707]]

              precision    recall  f1-score   support

           0       0.81      0.84      0.83     69224
           1       0.73      0.68      0.70     42419

    accuracy                           0.78    111643
   macro avg       0.77      0.76      0.76    111643
weighted avg       0.78      0.78      0.78    111643
```

```
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()

scaled_xtrain = mm.fit_transform(xtrain)
scaled_xtrain = pd.DataFrame(scaled_xtrain, columns= xtrain.columns)
```

ded in 65ms. finished 21:31:13 2022-03-23

```
scaled_xtest= mm.transform(xtest)
scaled_xtest = pd.DataFrame(scaled_xtest, columns= xtest.columns)
```

ded in 19ms. finished 21:31:16 2022-03-23

Min Max-Scaling

With top 8 Features from the XGB Model

```
from xgboost import XGBClassifier
xgb10 = XGBClassifier(random_state=10)
xgb10.fit(xtrain,ytrain)

y_pred_xgb = xgb10.predict(xtest)
y_train_xgb = xgb10.predict(xtrain)
ypred_proba_xgb = xgb10.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_xgb))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_xgb), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_xgb))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_xgb),'\n')

print(confusion_matrix(ytest,y_pred_xgb),'\n')

print(classification_report(ytest,y_pred_xgb))
```

executed in 11.5s, finished 21:29:58 2022-03-23

```
licitly set eval_metric if you'd like to restore the old behavior.
Train Accuracy:  0.7773418605901005
Test Accuracy:  0.7747015039008267

ROC-AUC Score:  0.8344306591606886
Cohen Cappa Score:  0.5069990629186791

[[59945  9279]
 [15874 26545]]

              precision    recall  f1-score   support

           0       0.79      0.87      0.83     69224
           1       0.74      0.63      0.68     42419

    accuracy                           0.77    111643
   macro avg       0.77      0.75      0.75    111643
weighted avg       0.77      0.77      0.77    111643
```

With top 8 Scaled Features from the XGB

```
from xgboost import XGBClassifier
xgb_scaled = XGBClassifier(random_state=10)
xgb_scaled.fit(scaled_xtrain,ytrain)

y_pred_xgb_scaled = xgb_scaled.predict(scaled_xtest)
y_train_xgb_scaled = xgb_scaled.predict(scaled_xtrain)
ypred_proba_xgb = xgb_scaled.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_xgb_scaled))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_xgb_scaled), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_xgb))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_xgb_scaled),'\n')

print(confusion_matrix(ytest,y_pred_xgb_scaled),'\n')

print(classification_report(ytest,y_pred_xgb_scaled))
```

executed in 15.9s, finished 21:31:22 2022-03-23

```
licitly set eval_metric if you'd like to restore the old behavior.
Train Accuracy:  0.7773418605901005
Test Accuracy:  0.7747015039008267

ROC-AUC Score:  0.4978129723867412
Cohen Cappa Score:  0.5069990629186791

[[59945  9279]
 [15874 26545]]

              precision    recall  f1-score   support

           0       0.79      0.87      0.83     69224
           1       0.74      0.63      0.68     42419

    accuracy                           0.77    111643
   macro avg       0.77      0.75      0.75    111643
weighted avg       0.77      0.77      0.77    111643
```

After building the Model from the best 8 features selected from the XG Boost Model we can find that the RF and the XGB model is performing the best. When we do the scaling with the help of MinMaxScaler on that 8 features we can find that the model performance is increasing a bit and on the scaled data, we find the lowest FN score so far i.e., 13712. The Recall for positive class is also better compared to the un-scaled model and best so far i.e., 68.

Here the RF model we found to be little overfitted.

Next, we have built Bagging and Stacking Classifier to evaluate the model performance farther.

Bagging Classifier with Base Estimator as RF

```
from sklearn.ensemble import BaggingClassifier
rf =RandomForestClassifier()

bc = BaggingClassifier(base_estimator =rf, random_state=10)

bc.fit(xtrain,ytrain)
ypred_bc = bc.predict(xtest)
y_train_bc = bc.predict(xtrain)
ypred_proba_bc = bc.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_bc))
print ('Test Accuracy: ',accuracy_score(ytest,ypred_bc), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_bc))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,ypred_bc),'\n')

print(confusion_matrix(ytest,ypred_bc),'\n')
print(classification_report(ytest,ypred_bc))
```

executed in 6m 6s, finished 23:00:08 2022-03-23

```
Train Accuracy:  0.8127519213199333
Test Accuracy:  0.7811595890472309

ROC-AUC Score:  0.8470870144903084
Cohen Cappa Score:  0.528228408654425

[[58707 10517]
 [13915 28504]]

              precision    recall  f1-score   support

           0       0.81      0.85      0.83     69224
           1       0.73      0.67      0.70     42419

    accuracy                           0.78    111643
   macro avg       0.77      0.76      0.76    111643
weighted avg       0.78      0.78      0.78    111643
```

Stacking Classifier with Final Estimator as XGB

```
from sklearn.ensemble import StackingClassifier
from catboost import CatBoostClassifier

dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
xgb = XGBClassifier()
cat = CatBoostClassifier()

base_learners = [('DT Model',dt),
                 ('RF Model',rf),
                 ('CatB Model',cat),
                 ('XGB Model', xgb)]

stack_model = StackingClassifier(estimators=base_learners, final_estimator=xgb )

stack_model.fit(scaled_xtrain, ytrain)
ypred_stack = stack_model.predict(scaled_xtest)
y_train_stack = bc.predict(scaled_xtrain)
ypred_proba_stack = bc.predict_proba(scaled_xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_stack))
print ('Test Accuracy: ',accuracy_score(ytest,ypred_stack), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_stack))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,ypred_stack),'\n')

print(confusion_matrix(ytest,ypred_stack),'\n')

print(classification_report(ytest,ypred_stack))
```

executed in 9m 45s, finished 01:15:15 2022-03-24

```
Train Accuracy:  0.7787526199817273
Test Accuracy:  0.7876087170713794

ROC-AUC Score:  0.836471837051513
Cohen Cappa Score:  0.5421482856118252

[[59063 10161]
 [13551 28868]]

              precision    recall  f1-score   support

           0       0.81      0.85      0.83     69224
           1       0.74      0.68      0.71     42419

    accuracy                           0.79    111643
   macro avg       0.78      0.77      0.77    111643
weighted avg       0.79      0.79      0.79    111643
```

So, finally from the above processes we found our best model is Stacking Classifier with the base learner as DT, RF, XGB and Cat Boost and final Estimator as XG Boost, we find the Recall Score to be the highest so far which is 69 and we find the FN is 13551 also the ROC-AUC score in final model found to be 83.64, which is by far the best and it's reduced compared to our baseline model. Though in the business perspective the high FN is still very costly. We are unable to farther tune our final model due to the limited computational resources.

6.1.6. Revesting the Feature Engineering Process:

As the model performance is not up to the mark so we again revisit the feature engineering techniques. Earlier we dropped Claim Diagnosis and Procedure Codes. As the model score is not improving, we want to add Claim Diagnosis Code 1 to 10, Claim Procedure Code 1 to 5 and Claim Admit Diagnosis Code these 16 features to the existing model and with the total 40 features we again ran the model.

Baseline Best Performing DT Model

```
dt = DecisionTreeClassifier(random_state=100)
dt.fit(xtrain,ytrain)

y_pred_dt = dt.predict(xtest)
y_train_dt = dt.predict(xtrain)
ypred_proba_dt = dt.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_dt))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_dt),'\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_dt))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_dt),'\n')

print(confusion_matrix(ytest,y_pred_dt),'\n')

print(classification_report(ytest,y_pred_dt))
```

executed in 8.95s, finished 13:26:12 2022-03-22

```
Train Accuracy:  0.9999977606993784
Test Accuracy:  0.6924572073484231

ROC-AUC Score:  0.6755265817978182
Cohen Cappa Score:  0.350178893107868

[[51563 17446]
 [16889 25745]]

              precision    recall  f1-score   support

           0       0.75      0.75      0.75     69009
           1       0.60      0.60      0.60     42634

    accuracy                           0.69    111643
   macro avg       0.67      0.68      0.68    111643
weighted avg       0.69      0.69      0.69    111643
```

XGB Model with Top 10 Features

```
xgb10 = XGBClassifier(random_state=100)
xgb10.fit(xtrain,ytrain)

y_pred_xgb = xgb10.predict(xtest)
y_train_xgb = xgb10.predict(xtrain)
ypred_proba_xgb = xgb10.predict_proba(xtest)[:,1]

print ('Train Accuracy: ',accuracy_score(ytrain,y_train_xgb))
print ('Test Accuracy: ',accuracy_score(ytest,y_pred_xgb), '\n')

print('ROC-AUC Score: ',roc_auc_score(ytest,ypred_proba_xgb))
print('Cohen Cappa Score: ',cohen_kappa_score(ytest,y_pred_xgb),'\n')

print(confusion_matrix(ytest,y_pred_xgb),'\n')

print(classification_report(ytest,y_pred_xgb))
```

executed in 25.0s, finished 13:54:49 2022-03-22

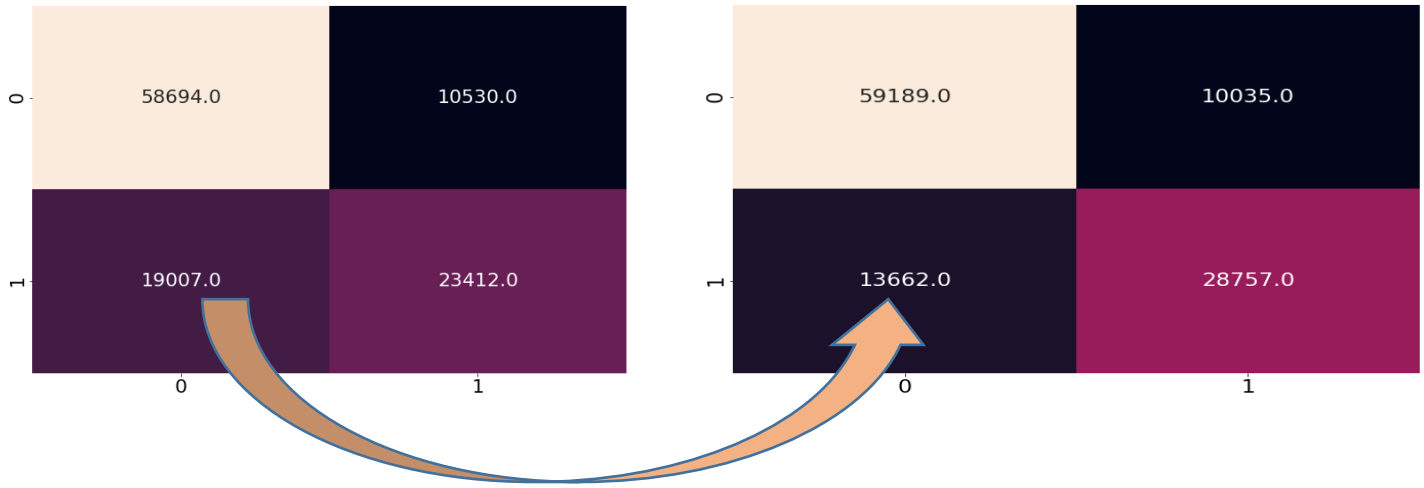
```
[13:54:24] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
3.0, the default evaluation metric used with the objective 'binary:logisti
ly set eval_metric if you'd like to restore the old behavior.
Train Accuracy:  0.7717503269378908
Test Accuracy:  0.7631647304354057

ROC-AUC Score:  0.8220501754452562
Cohen Cappa Score:  0.47776668361875874

[[60181  8828]
 [17613 25021]]

              precision    recall  f1-score   support

           0       0.77      0.87      0.82     69009
           1       0.74      0.59      0.65     42634

    accuracy                           0.76    111643
   macro avg       0.76      0.73      0.74    111643
weighted avg       0.76      0.76      0.76    111643
```

From the above models we can see that the model performance doesn't increases even when we revisit and add those 16 features back. The model performance in fact decreases by a margin.

So, we can conclude that there is no information loss while we only work with the 23 features.

6.1.7. Final Model Evaluation:

CM for Baseline RF Model **CM for Final Stacking Classifier Model**



CV : We perform a 10-Fold CV to check the robustness of the model and we find the scores: (0.67371016,0.6734 3585, 0.66815354, 0.67443362, 0.672262, 0.66744923, 0.66457331,0.66885785, 0.67142102, 0.6753536)

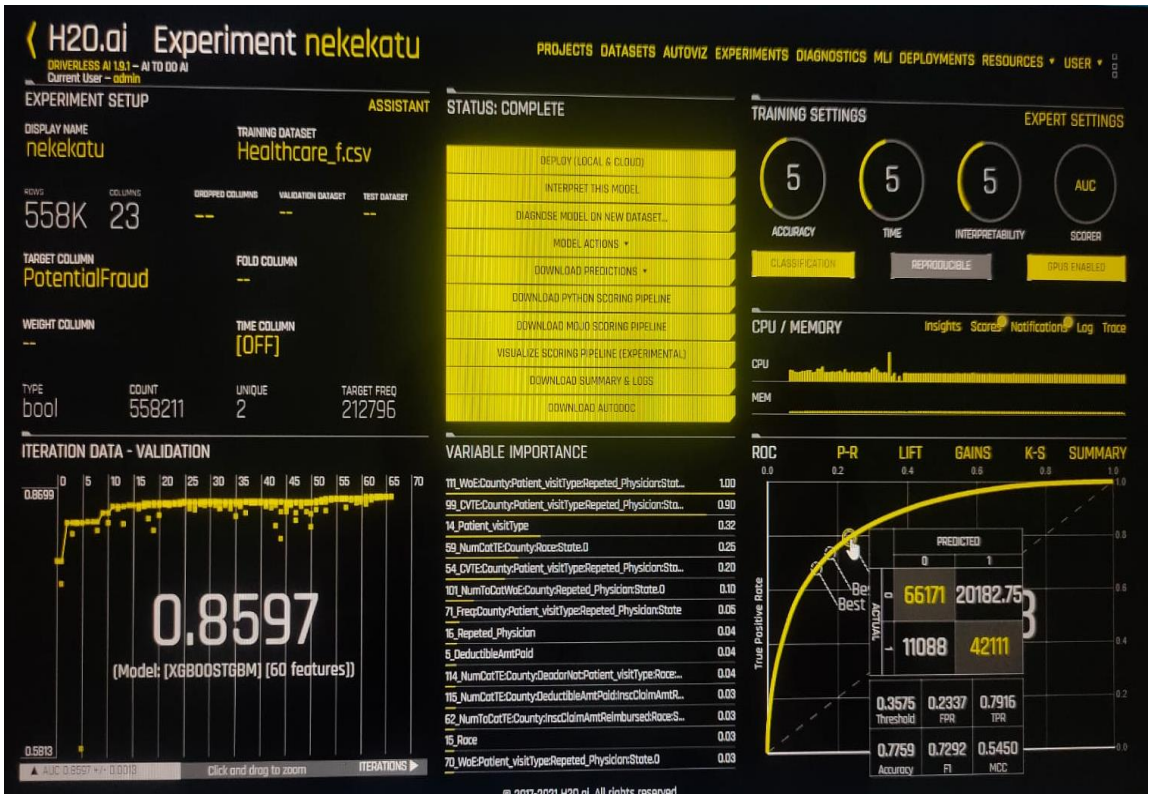
Average Recall Score: 0.6709650222140342

So, we can conclude from the cross validation that we generalize the model well. Though there is still scope of improvement.

7. Comparison and Implications:

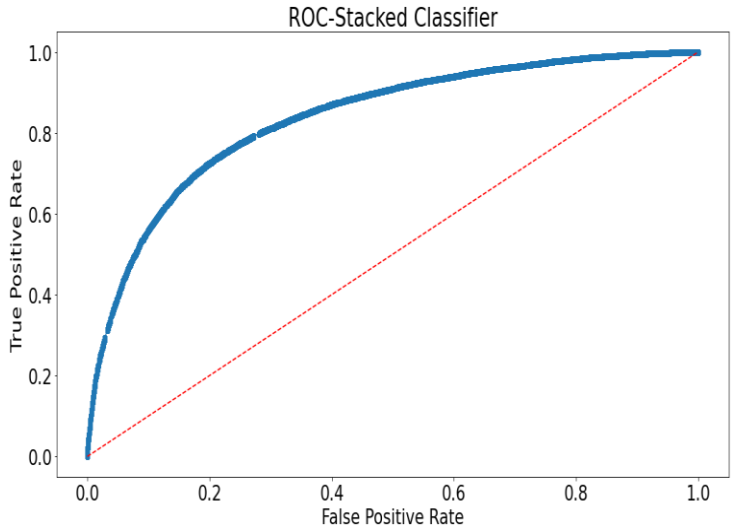
7.1. Comparison of the Benchmark:

We use the H20.ai Auto-ML platform for the benchmarking.

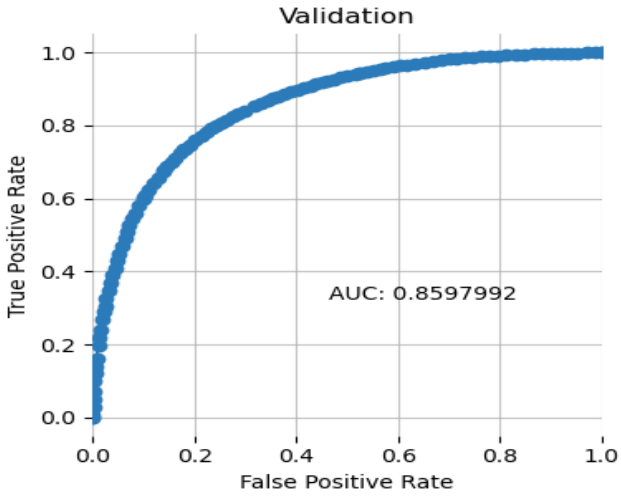


when we evaluate our model against it, we can find that the AUC in our best model found to be 84 where in the H20 it's found to be 85.97. We found our Recall score to be 69 where in H20 it's found to be 79. We have found Lowest FN to be 13551 where as in the Auto-ML tool it is 11008. Though the accuracy in both the models are similar, around 78.5.

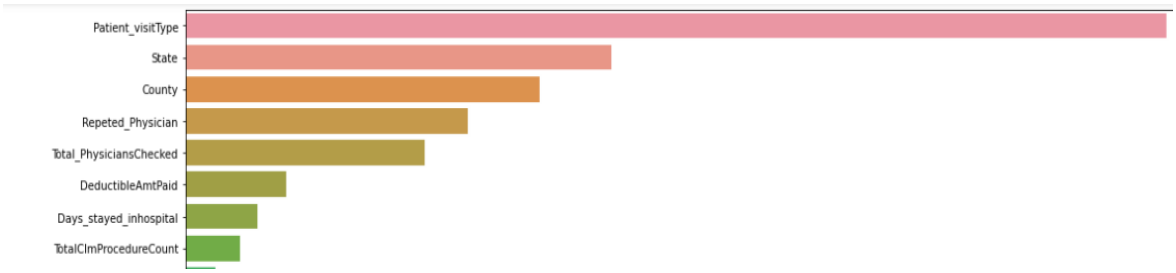
On our Best Model



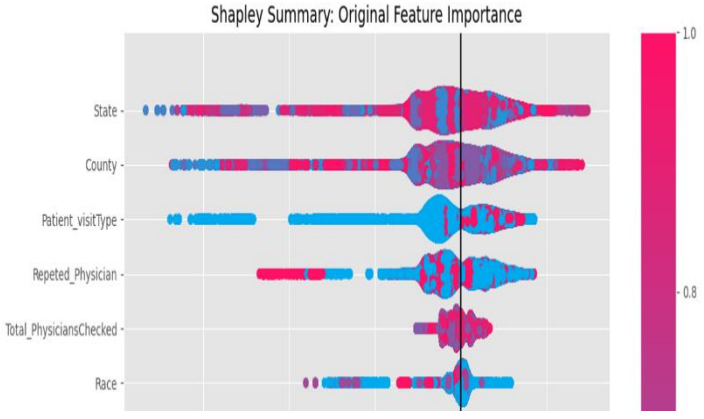
On the H2O Auto-ML Model



We are also able to find the Top 5 features from our model and compared it against the benchmark Auto-ML Model.



In the Auto-ML Model we find the Top 5 features.



We can evaluate that we get the similar features compared to the Auto-ML tool that are having highest (Top 5) significant are: **Patient Visit Type, State, County, Repeated Physician and Total Physician Checked.**

7.2.Implications:

Our implication is the project is that if we use these features Patient Visit Type, State, County, Repeated Physician and Total Physician Checked for the future data, then there is probability that we can differentiate the Fraud to Non-Fraud Claims by 84%. Patient Visit Type alone can serve as the clear separator between Fraud and Non-Fraud Claims, as it is the most important predictor. We can conclude that if in a particular claim a greater number of physicians are involved and if the Same Physician is repeated multiple times, then, that claim is more likely to be Fraud.

Fraudulent Claim Detection from the Healthcare Providers is one of vital aspects for an Insurance Business as it can have a direct impact on reducing the monetary loss.

- So, in the future, any potential fraud can be predicted beforehand in real-time, so intern the loss can be minimized.
- As we have discovered the most important features, with that we can detect the behavior of potential Fraud Healthcare provider's claims and these features also serve as a flag for future business prospect.
- Reduction in the monetary loss of the business can indirectly help the business to cater the same service at a reasonable rate to a greater number of customers.

7.2.1. Recommendations:

- We will recommend using tuned Stacked Classifier model to make prediction of Fraudulent Claims because it is giving maximum Recall score for positive class, 69%.
- To increase the Robustness of the model a greater number of data points are needed. As we had only 2 years (2008-2009) of data we find it challenging and insufficient to build a robust model.
- Some of the anticipated features are found missing, Premium Amount, Max Out of Pocket Amount, Co-Pay Amount, Claimed Amount, which could have been a great addition to the business problem to predict the fraud claims more accurately.

8. Limitations and Scope:

8.1. Limitations:

- Lack of in-depth Domain Understanding.
- With the Limited understanding of the business, we are unable to extract more number features. For the same reason we revisited the Feature Engineering process and brought back 16 features which were dropped earlier. Though some more time and resources spend on understanding the business would have been more helpful to extract more meaningful features, that makes business sense.
- Limited Computational Resources impacted our feature engineering and model tuning phases, which we have overcome to a certain extent using Google Collab.
- The Recall score that we have achieved using manual process is moderate (69). But it's not on par with the benchmark (H20.ai Auto-ML Tool) which is found to be 79. This might be due to the limitation in computational resources.
- Absence of few anticipated features like Premium Amount, Max Out of Pocket Amount, Co-Pay Amount, Claimed Amount have clearly impacted our model performance and presence of them could have been very fruitful in predicting the Fraud Claims with more prediction score.

8.2. Future Scope:

- Instead of Tree based Feature Importance, the Permutation Feature Importance would be a better choice as there could be inherit bias in the dataset and the Permutation Feature Importance can combat that and can make the model more robust.
- More time to be spend in understanding the Business problem could help with more stringent feature engineering and feature selection process.
- With high end Computational resources, the process of feature engineering and fine tuning the Model can be leveraged.

References:

- *https://www.kaggle.com/rohitrox/healthcare-provider-fraud-detection-analysis?select=Test_Inpatientdata-1542969243754.csv*
- *https://journalofbigdata.springeropen.com/articles/10.1186/s40537-018-0138-3*
- *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7579458/*
- *http://www.differencebetween.net/science/health/difference-between-cpt-and-icd-codes/*
- *https://explained.ai/rf-importance/*
- *https://scikit-learn.org/stable/modules/permutation_importance.html*
- *https://www.sciencedirect.com/science/article/pii/S0213911121002661*
- *https://www.jaad.org/article/S0190-9622(20)30481-3/pdf*
- *https://h2o.ai/solutions/use-case/claims-fraud-detection/?_ga=2.216876310.1666353940.1648117677-662288051.1647874332*
- *https://jespublication.com/upload/2020-110466.pdf*
- ***Working procedure for Coverages in Health care Insurance -*** *https://www.medicare.gov/what-medicare-covers/what-part-a-covers*
- ***Advanced fraud Detection Techniques*** *http://www.actuariesindia.org/downloads/25062021_IAI_Fraud%20Detection%20in%20Health%20Insurance%20Claims.pdf*

- ***Feature Importance and Feature Selection with XGBoost*** *-* *https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/*
- *Morley, N. J., Ball, L. J., & Ormerod, T. C. (2006). How the detection of insurance fraud succeeds and fails. Psychology, Crime & Law, 12(2), 163-180.*
- *https://nycdatascience.com/blog/student-works/healthcare-fraud-detecting-inconsistencies-in-provider-data/*
- *https://morioh.com/p/5fd7f32049ce*
- *https://www.tigergraph.com/blogs/machine-learning/machine-learning-applied-to-detecting-fraud-in-healthcare/*
- *https://www.roselladb.com/healthcare-fraud-detection.htm*
- *https://repository.kca.ac.ke/bitstream/handle/123456789/578/Oundo-Model%20For%20Assessing%20Fraudulent%20Medical%20Claims%20%20An%20Application%20Of%20Machine%20Learning%20Algorithms%20In%20Health%20Care.pdf?sequence=1&isAllowed=y*
- *https://rohansoni-jssaten2019.medium.com/healthcare-provider-fraud-detection-and-analysis-machine-learning-6af6366caff2*

- *https://kundusoumya98.medium.com/healthcare-provider-fraud-detection-and-analysis-using-machine-learning-632f7a380c79*
- *https://www.uplevel.work/blog/pinpointing-medicare-healthcare-fraud-with-machine-learning*