CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

Abstract—In this paper, we are building a classifier to classify the inpatient Medicare Claims as Fraud, based on the Inpatient and Beneficiaries Data, containing details of various health conditions, insurance specifics, diagnosis Codes and procedures that patient were subjected to while admitted to a healthcare institution.

Index Terms—Data Cleaning, Data Transformation, Insights, F1 Score, Scaling, Logistic Regression, Random Forest, Naïve Bayes Classifier, Extra Regressor Trees, HTML, UI.

INTRODUCTION

In developed nations, every medical procedure (from a regular checkup to surgery) is underwritten by Medicare covers, from different healthcare providers. In this scenario, frauds committed during the claims process have become a major concern. Especially in the case of patients requiring immediate hospitalization. According to government data, the total Medicare spends have grown exponentially because of the frauds in Medicare Claims where all stakeholders (providers, physicians and beneficiaries) profit by making false claims and getting them approved. Procedural loopholes at various stages of inpatient care are not visible with the narrow perspective that we get at a particular stage of approval. To identify these loopholes a rigorous analysis of Medicare data of beneficiaries and various aspects of claim, collected at every stage of the patient care and claim process was carried out.

Generally ambiguous diagnosis codes, Insurance reimbursement costs, Health Care conditions, procedures are used to commit frauds. Medicare aids, especially those provided to inpatients are the most vulnerable welfare initiatives impacted due to fraudulent practices. Due to this reason, there has been a large increase in premiums and healthcare spending.

The goal of this project is to "*predict potentially fraudulent claims for the inpatient data*" based on the claims filed by providers. Along with this, I will also highlight important variables helpful in detecting the behavior of potentially fraudulent providers. I will also analyze fraudulent patterns in the provider's claims to understand the future behavior of providers.

RESOURCES USED:

- **Microsoft Excel (for data visualization)**
- **Google COLAB PRO (25.51 GB RAM and 64GB ROM Storage)**
- **Notepad++.**
- **Anaconda Spyder.**

DATA SOURCE FOR ANALKYSIS

The following data sources have been used for this project:

- **Train_Beneficiarydata-1542865627584**: Comprising of the beneficiary data with details such as state, county, chronic health conditions etc., which was used to train the model.
- **Train_Inpatientdata-1542865627584**: Comprising of in-patient data with details such as insurance cover, physicians, providers, procedures carried out, conditions of hospitalization and claim.
- **Train-1542865627584**: Comprising of training tags of potential Frauds and Providers.
- **Test_Beneficiarydata-1542969243754**: Comprising of the corresponding Test Beneficiary Data.
- **Test_Inpatientdata-1542969243754**: Comprising of the corresponding Test In-patient Data.
- **Test-1542969243754**: Comprising of the ~~F~~final test tags.

The single dataset with all these sources of the data can be found at:

https://www.kaggle.com/rohitrox/healthcare-provider-fraud-detection-analysis

DATA CLEANING AND DATA TRANSFORMATION

It was very challenging to work on these datasets, as:

a. there were a lot of missing values
b. I had to decode the meaning of the values used
c. Also most columns contained data which was a combination of categorical and numerical values.

The approach I have taken to address these problems for across datasets, has been delineated below:

NOTE: The columns of "**Provider**" and "**BeneID**" are kept constant in all the Datasets as they are unique keys constant across all the datasets, later used for merging datasets.

<u>DATA OF **Train_Beneficiarydata-1542865627584:**</u>

County, Race and State: These 3 columns didn't have any missing values. I used them to deduce that the data belongs to African subcontinent (as only it had the following specifics: 54 countries and 5 races). This had to be done as the values in the later columns regarding the Chronic Conditions, was not explicit.

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

DOB and DOD: Date of Birth (DOB) and Date of Death (DOD) were 2 datetime64 columns that were good for time series analysis of the frauds but proved to be of no help in the modelling for Fraud classification. Hence I deleted them.

Age: Although the 2 columns DOD and DOB were deleted, their data was not wasted. By feature engineering I added a new column called *age* that was the difference between the DOB, DOD. The cases where the data was missing in the DOD column, I assumed the patient was not dead, and populated the current date; thus giving me the required age. This could give an insight into the age groups targeted by the Fraudsters.

RenalDiseaseIndicator: This column had no missing data. I replaced the "Y" tags with numerical 1 and 0 defined the "NO"; I then converted the values into the int64 datatype.

Nosofmonths_PartACov and Nosofmonths_PartBCov: These 2 columns defined the number of months covered for the beneficiary in Part A (Inpatient) cover and Part B (Out Patient) Cover. As there were no value changes in the whole dataset, in these 2 columns, for all the beneficiaries covered for 12 months, I deleted the 2 columns.

ChronicCond_Alzheimer and ChronicCond_HeartFailure: In these 2 columns it was seen that the columns values were 1 and 2 respectively. By doing some research on the 2 above it was seen that 1 out of every 10 individuals in the African Subcontinent generally has *Alzheimer* and a similar proportion have *Heart Disease*. Owing to this, the frequency of 1 tag more closely resembled the above ratio of 1/10.Hence I replaced the value 2 with 0 or NO.

ChronicCond_KidneyDisease, ChronicCond_ObstrPulmonary and ChronicCond_Cancer: On facing a similar problem in the 3 columns mentioned above, it was seen that 13.7% of people have Kidney diseases, 34.7% have cancer. Based on this the 2 tag was changed to 0 in all the above columns.

ChronicCond_Diabates, ChronicCond_IschemicHeart Disease and ChronicCond_Osteoporaisis: According to WHO 60% population, 81% population and 20% population respectively have the above-mentioned diseases, which is seen in the same proportion in the tag 1-YES and 2- NO, Hence the changes.

ChronicCond_ Stroke, ChronicCond_ rheumatoid arthritis and ChronicCond_Depression: Here same changes were done as in case of the *other* diseases.

IPAnnualReimbursementAmt, IPAnnualDeductibleAmt, OPAnnualReimbursementAmt, OPAnnualDeductibleAmt: For these columns there were no missing values. All values were floating point numbers and completely discrete so the models would be able to decipher it very easily. Hence, no changes were made.

DATA OF **Train_Inpatientdata-1542865627584:**

As it is seen that there is higher risk of loss in inpatient cases, I have focused my project on analyzing frauds for the Inpatients segment. The changes made to columns of this dataset as per the focus of the analysis are:

ClaimID: As IDs of the claims are unique to each transaction or claim, they won't be repeat in future cases. So it cannot be an index for fraudulent claims. I have deleted the same.

ClaimStartDt and ClaimEndDt: These 2 being datetime64 columns are not continuous variables and its co variance with respect to other columns cannot be found. Hence I deleted these columns.

AttendingPhysician, OperatingPhysician and OtherPhysician: In these 3 columns there were lot of samples/observations missing. Hence I used the following methods to address the missing values:

a. Where attending physicians were there but either the Operating Physician or the other physician were absent, I copied the attending physician to either of the missing or both.
b. Where the operating physician was there but the Attending Physicians were missing, in these cases I just copied the operating physician to the Attending Physician, as the diagnosis codes and Procedures that the inpatient would have been subjected would be as prescribed by the Operating Physician.
c. Where the Other Physician was missing. Here the precedence was followed with attending Physician having a higher precedence, followed by the Operating Physician.
d. Where all the 3 were absent or only the other Physician not directly responsible for the patient was present. In these conditions I used the "**Hot-deck method of imputation"** (where the missing values are filled using the values seen in similar cases.). As per this method, I grouped the data into 3 groups by Dataset.Groupby() command. The 3 groups were

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

```
attendingPhy_trainset = tr1.groupby(['St
ate','County','Provider','DiagnosisGroup
Code'])
attendingPhy_tr1 = tr2.groupby(['State',
'County','Provider'])
attendingPhy_tr2 =tr3.groupby(['State'])
```

Using these the above 3 groups, I also attached the 3 groups with their maximum occurring AttendingPhysician and collected them in 3 dictionaries (with the group names as keys and max occurring AttendingPhysician as values), then for the missing cases I collected the corresponding BeneID and then tallied the best match across all the keys of the 3 dictionaries. On finding a match the corresponding AttendingPhysician was filled. Then the corresponding OperatingPhysician and OtherPhysician were filled required.

Lastly as all the values in these 3 columns were categorical or string they could cause problems, later in the analysis. To avoid this, I **Label Encoded (assigned a label to all the string values in the columns and then replaced the actual values by their respective numerical labels)** the values, and stored the corresponding Value-label combinations as Dictionary (Physician_dict_df: - found in the Dictionaries_for_processing folder.)

ClmAdmitDiagnosisCode: As column had categorical values, I **label encoded** these values and replaced the labels for the corresponding Values in the table (Corresponding Dictionary: ClmAdmitDiagnosisCodeDict_df: - found in the Dictionaries_for_processing folder.)

DiagnosisCodeGroup: These columns already had numerical value except for the "OTH" value which I replaced with 0.

AdmissionDt and DischargeDt: The dates could not be discrete for modelling so I deleted these columns.

Duration_of_Stay: I derived this column through feature engineering on the 2 columns AdmissionDt and DischargeDt. Values in this column were the difference of the values in AdmissionDt and DischargeDt columns.

DeductibleAmtPaid: This column signifies the amount already paid by the medicare for the inpatient. I just replaced the missing values as NO amount paid i.e. with the value 0.0

ClmDiagnosisCode_1-10 Columns: These columns capture the diagnosed conditions for which the claims were made. These columns allowed up to 10 such codes to be claimed, but models or algorithms don't understand list of codes. So to assign the codes to respective inpatients replaced the 10 columns with 25 such columns (with given space computational resources of 25.51GB Google Colab PRO, this is the max number of columns that the system and models were able to handle.), after short-listing 25 of the most frequent ClmDiagnosisCodes; I grouped the ClmDiagnosisCodes for all the inpatients and then stored them in a dictionary with the respective BeneID as keys. Next for all the BeneID values I enumerated their respective values in the dictionary. If I found any of the 25 codes in them I replaced the value of that column as 1 in the main dataset where I had replaced the 10 DiagnosisCodes with the 25 frequently occurring diseases. Having completed the procedure for all the inpatients it became a more useful set of columns for analysis. The 25 columns used have been preserved in the transfer_list1.csv (Stored in the Dictionaries_for_processing folder.)

ClmProcedureCode_1-6: These columns have information about the procedures that have been claims against them. Though this follows the same procedure as the former ClmDiagnosisCodes_1-10 columns, but due to limited computational resources I did not convert each procedure into a column as that would have resulted in 1000 columns, and the system was unable to handle the load. As the columns were already numerical, I just populated all the missing values with 0, signifying no procedures.

InscClaimAmtReimbursed: This column captures the total Claim amount that was reimbursed. It had all the values populated filled, discrete, and floating so no changes were made. And due to the computational limitation, I had to delete all rows below the 5000 value of this column to reduce the size of dataset below 900MB

DATA OF **Test-1542969243754:**

This was the last dataset that contained all the respective target tags. The changes made to the columns were:

PotentialFraud: This column had only 2 values namely "NO" and "YES", which were replaced by 0 and 1.

Having cleaned, discretized and transformed data across all datasets, I then merged all the data sets into a single large dataset by:

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

1. Merging the **Train_Inpatientdata-1542865627584** and **Train_Beneficiarydata-1542865627584** using the "BeneID" column of patients.
2. Merging the dataset formed after **1.;** (**work_dat_notDead_large** in codes) with the **Test-1542969243754,** using the Provider column.

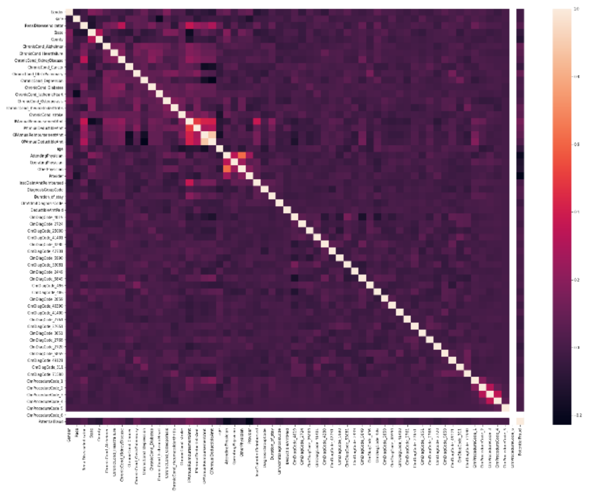
Having combined the datasets, I realized that the *BeneID* column was specific to each patient and hence would not give any insight into fraud analysis. So I deleted it. Regarding the *Provider* column, I **label Encoded** it and replaced the values with their respective labels.

DATA INSIGHTS:

Having transformed the data, I moved towards finding various interesting insights that could establish relationships among the columns. This would help in better analysis of fraudulent practices and ways to counter them

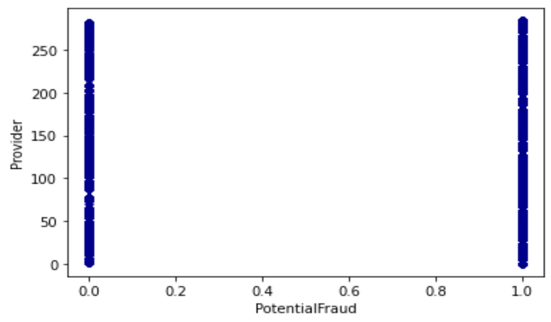
.

To identify the best place to start looking for relationships, I made a seabborn.Heatmap():



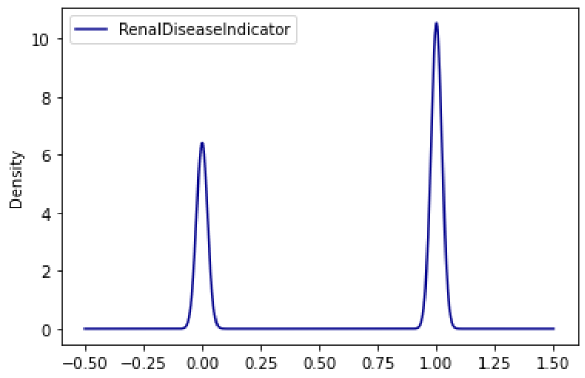
From the above I was able to gain the following valuable insights:

- **Provider vs Potential Frauds:**



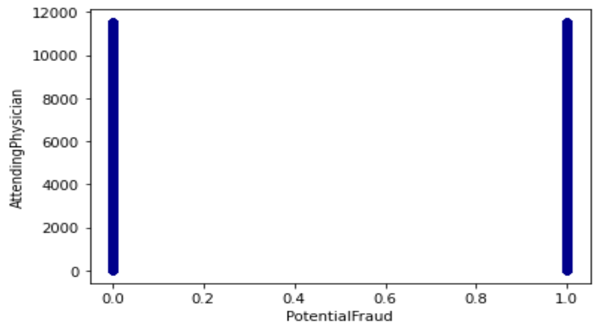
The Providers have strong –ve correlation with the potential frauds. Which implies that the providers are not the source of fraudulent practices

- **RenalDisease vs Potential Frauds:**



Here I saw a high density of Potential Frauds with RenalDiseaseIndicator being 1 (ChronicCondKidneyDisease), implying that the renal conditions are being used for fraudulent cases prominently. Hence more investigations should be done in renal condition cases.

- **AttendingPhysician vs Potential Frauds:**



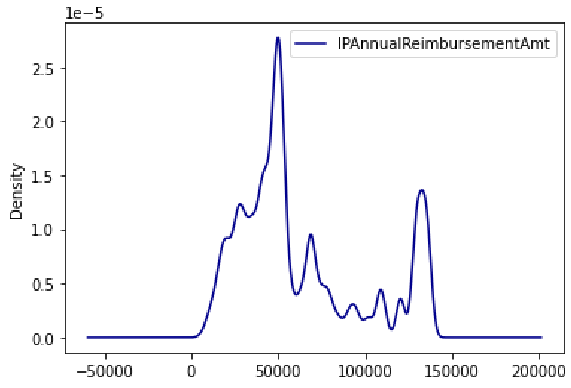
None of the Physicians are even slightly related

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

To Potential Frauds, implying they are not responsible for the any Fraudulent Behaviors in Inpatient Data.

- **IPAnnualReimbursement vs Potential Frauds:**

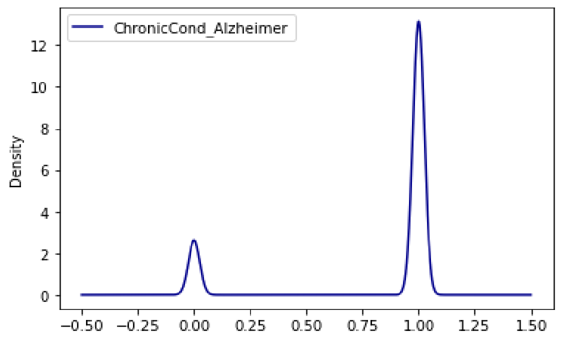


Here I observed a positive covariance in the IPAnnualReimbursement and Potential Frauds i.e. there are more fraudulent cases as IPAnnualReimbursement increases. At IPAnnualReimbursement value of $50000 the density peaks. Hence, I would keep an eye out for cases with high IPAnnualReimbursement.

A similar trend is observed in High InscClaimAmtReimbursed cases and High IPAnnualDeductibleAmt cases.

Moreover, the InscClaimAmtReimbursed was also seen to have positive covariance with the ChronicCond_KidneyDisease, which implies that the Renal Diseases have high Insurance Claim Amount reimbursed and are also a case for high Potential Frauds. Hence, all Renal related cases need to be minutely scrutinized to avoid fraudulent practices.

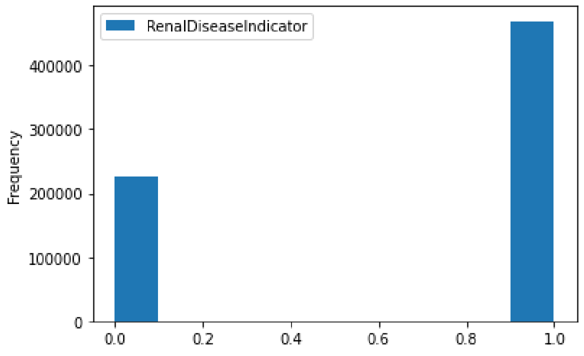
- **ChronicCond_Alzheimer Frequency:**



The ChronicCond_Alzheimer and Potential Frauds have a higher positive covariance compared to other diseases, as evidenced by the Heatmap. This implies that Alzheimer inpatients are being used for making Fraudulent claims. Especially because of high concentration of inpatients having Alzheimer cases.

A similar trend is seen in case of ChronicCondObstrPulmonary, condition which also displays a positive covariance with Potential Frauds i.e. both of them co-occurring at high rates.

- **RenalDiseaseIndicator Frequency:**

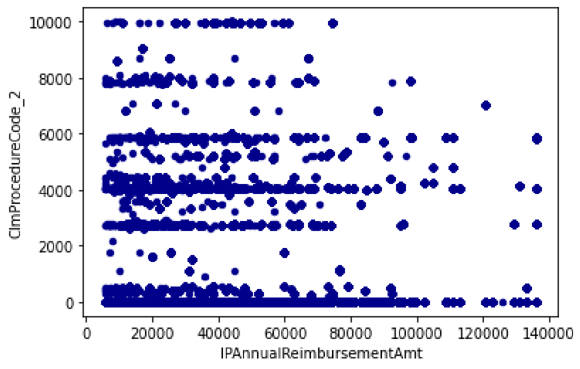


From the heatmap I observed:

a. IPAnnualReimbursement and RenalDiseaseIndicator having +ve covariance,
b. RenalDiseaseIndicator and Potential Fraud having +ve Covariance with the.

The RenalDiseaseIndicator also having a high frequency, points to the fact that the Fraudulent claims are being made in case of ChronicCond_KidneyDisease (RenalDiseaseIndicator = 1) more frequently.

- **IPAnnualReimbursement vsClmProcedureCode_1**



One of the more peculiar covariances was seen in the case

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

of the IPAnnualReimbursement, Potential fraud and ClmProcedureCode_1, implying that procedure 1 is being used for fraudulent claims more than the other procedures.

Other such correlations observed in the Heat map were Positive covariance between high Insurance Claim Amount Reimbursed (InscClmAmtReimbursed) and:

a. ChronicCond_Stroke,
b. ChronicCond_HeartFailure,
c. ChronicCond_KidneyDIsease,

which implies that Fraudulent cases were seen to have good correlation with the InscClmAmtReimbursed. These 3 Chronic Conditions also need to be looked at more closely to avoid high value Fraudulent Claims.

DATA BALANCING:

On considering the number of cases of Fraud and not Fraud, number of instances of Potential Frauds cases (1) were as below:

```
1    3839343
0    1913967
Name: PotentialFraud, dtype: int64
```

i.e. the number of potential fraud cases are estimated to be 2 times as compared to the non-Fraud cases.

In this scenario I can use various Data Balancing techniques like the OverSampling (Adding the same non Fraud columns again), UnderSampling (Deleting the Fraud cases), Bootstrapping and SMOTE. But they would all result in deleting data, which is not desirable. So I will calculate my model success by using the F1-**score (Precision*Recall/Precision + Recall**), which is used specifically for imbalanced data.

DATA STANDARDIZATION AND MODELLING:

Before I train any model with the transformed data, it is a necessity to first divide the data into Target and feature datasets and then standardize them so as to bring all the features to an equal scale, for ensuring higher accuracy.

To split the data into target and feature datasets, I first separated the data as:

```
ClassFraud_y = work_dat_notDead_large['
PotentialFraud'].copy()

ClassFraud_x = work_dat_notDead_large[wo
rk_dat_notDead_large.columns.difference(
['PotentialFraud'])]
```

Now depending on the models used, I standardized the data using different scalers (stored along with the models as. pkl files in Models Folder.).

The next section of my report, will deal with the details of the various models I used for my analysis of the dataset. The detailed explanations for each model used are listed below:

Logistic Regression:

As the transformed data in my hands did not have sparse data (so no Minimax()or MinAbs()), and I wasn't sure of any outliers; I used the RobustScaler here followed by splitting using the testrain_split of preprocessing package().

```
Robust_scaler_X = RobustScaler()
Robust_scaler_X.fit(ClassFraud_x)
```

Thus limiting my data to mean and quantiles, I proceeded to model the data using the **GridSearchCV()**, with the following parameters:

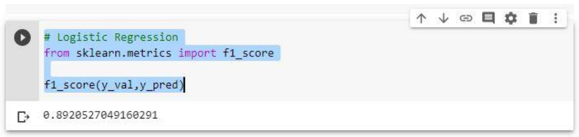
```
## 2>> Logistic regression
model_lr=linear_model.LogisticRegression
()
param1 = {}
param1['penalty'] = ['l1','l2']
param1['C'] = np.logspace(-5, 5, 20)
param1['solver'] = ['sag','newton-
cg','slbfgs']
param1['verbose'] = [1.0,0.5,2.0]
param1['max_iter'] = [800,1000]

clf = GridSearchCV(model_lr, param_grid=
param1, cv = 3, verbose=True,scoring='ro
c_auc',n_jobs=-1)

best_model = clf.fit(X_train, y_train)
```

with the GridSearchCV () having an Cross-Validation =3 folds

The result I got for F1-scoring on the validation set was a



```
# Logistic Regression
from sklearn.metrics import f1_score

f1_score(y_val,y_pred)
```

```
0.8920527049160291
```

F1-score of `0.892` which is good.

Fisher-Discriminant Classifier:

In Fischer Discriminant Classifier I used a StandardScaler() as it doesn't get affected by

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)

EAS 574 MACHINE LEARNING Project

SPRING 2020

outliers.

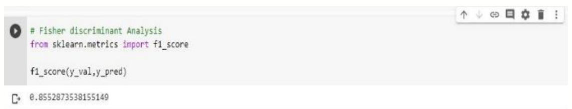
```
scaler = StandardScaler()
ClassFraud_y = work_dat_notDead_large['P
otentialFraud'].copy()
ClassFraud_x = work_dat_notDead_large[wo
rk_dat_notDead_large.columns.difference(
['PotentialFraud'])]
scaler.fit(ClassFraud_x)
```

Followed by splitting, ~~and then the~~I fed the training data ~~was fed~~ into the GridSearchCV() with the following parameters~~ as follows~~:

```
X_train, X_val, y_train, y_val = train_t
est_split(ClassFraud_x_Scaled, ClassFrau
d_y, test_size=0.33)
fisher = LinearDiscriminantAnalysis()

param_grid = {
    'solver' : ['svd','Isqr']
}

CV_rfc = GridSearchCV(estimator=fisher,
param_grid=param_grid, cv= 3, scoring='r
oc_auc', n_jobs=-1)
CV_rfc.fit(X_train, y_train)
```



```
# Fisher discriminant Analysis
from sklearn.metrics import f1_score

f1_score(y_val,y_pred)
```

```
0.8552873538155149
```

The F1 score I observed for the Fisher-Discriminant classifier was 0.8552, lesser as compared to logistic regression.

Naïve Bayes Classifier:

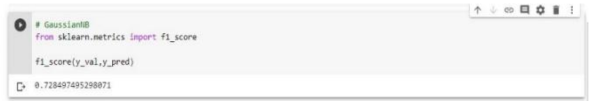
The Naïve Bayes Classifier works on the principle of Bayes theorem i.e.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

For ease of calculation of probability, it relies on calculating the mean and variance and arranging the data features itself, just the way the Scalers do it. Hence, I didn't require any scaler in this case. Moving directly to the model:

```
#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)
```



```
# GaussianNB
from sklearn.metrics import f1_score

f1_score(y_val,y_pred)
```

```
0.728497495298071
```

And the F1 score so given was 0.7284974952

Random Forest:

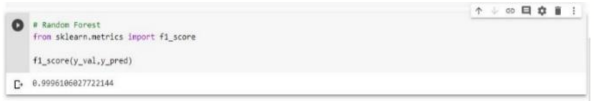
The Random Forest partly works on the information gain got from entropy (summation of all $P(X_i)\log2P(X_i)$) for all "I" values. And splits based on the calculation of these entropies.

So for calculating Probability it does the scaling itself. Moving to the model :

```
rfc = RandomForestClassifier(n_jobs=-
1,n_estimators=50, oob_score = True)

param_grid = {
    'max_features': ['auto', 'log2']
}

CV_rfc = GridSearchCV(estimator=rfc, par
am_grid=param_grid, cv= 3, scoring='roc_
auc', n_jobs=-1)
CV_rfc.fit(X_train, y_train)
#print CV_rfc.best_params_
```



```
# Random Forest
from sklearn.metrics import f1_score

f1_score(y_val,y_pred)
```

```
0.9996106027722144
```

The F1-score so obtained 0.9996106027722144

ExtraRegressorTrees:

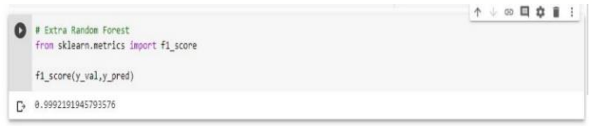
The extraregressor trees are just an extension of the random forest where it repeats the Random Forest with different sets of features used for splitting in each case and then it finally takes an average of all the trees. So generally it should give a better F1 score.

Here I have used the same standard Scaler().

So the code that we tried was:

```
ext_rfc = ExtraTreesClassifier(max_featu
res='auto',n_jobs=-1,n_estimators=50)

ext_rfc.fit(X_train, y_train)
```



```
# Extra Random Forest
from sklearn.metrics import f1_score

f1_score(y_val,y_pred)
```

```
0.9992191945793576
```

And then I calculated the F1 score of the as 0.9992191945793576

CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS

Siddharth Satyakam (UB Number 50293092)
EAS 574 MACHINE LEARNING Project
SPRING 2020

MODEL CONCLUSION:

For the purpose of **Fraud Analysis of Medicare Aid in Inpatient cases,** I have tested 5 models with their F1 Scores mentioned below:

- Logistic Regression: 0.892
- Fischer Discriminant Classifier: 0.8552
- Naïve Bayes: 0.72
- **Random Forest: 0.99961**
- ExtraRegressorTrees: 0.99926

Which shows that **Random Forest Model** run with **GridSearchCV()** with the following specifications

```
RandomForestClassifier(bootstrap=Tru
e, ccp alpha=0.0, class weight=None,
criterion='gini', max_depth=None,
max_features='auto',
max leaf nodes=None,
max samples=None,
min_impurity_decrease=0.0,
min impurity split=None,
min samples leaf=1,
min_samples_split=2,
min_weight_fraction_leaf=0.0,
n_estimators=50, n_jobs=-1,
oob_score=True, random_state=None,
verbose=0, warm_start=False
```

with **GridSearchCV() having cross validation of 3 folds and n_jobs =1, gives the best F1 score of 0.99962**

A peculiar observation in the modelling was seen that generally the Extra Regressor trees being an averaging of multiple random forest should always improve the Classification score however here it rather marginally degraded to 0.9992.

UI DEVELOPMENT:

The UI comprises of 2 groups of HTML pages whose details are provided below:

- **Relation UI:** This is the combination of Visualization_Initiator Page and DemoPage. The Visualization_Initiator page collects the relation graph to be shown using the "SelectList" element of the HTML and then passes the graph to the Iframe Page DemoPage which shows the output graph. On clicking on the graph, it shows the **Inference** derived from the graph.
- **Model UI:** These are 2nd group of pages that also contain Model_Visualization_Initiator and DemoPage1. The Model_Visulization_Initiator page gets the model from the user and sends it to the following Iframe page DemoPage1. The DemoPage1 shows the score and on clicking on the score it gives the inferences of the model.

CONCLUSION:

From my analysis I have concluded that the following health conditions have a higher incidence of Inpatient Medicare claim frauds:

1. Stroke,
2. Heart Failure,
3. Kidney Disease,
4. Pulmonary Obstruction
5. Alzheimer
6. Insurance Claim Amount Being Reimbursed
7. Cases where IPAnnualReimbursed Amt is large

Also, **Claim procedure 1** is being used for fraudulent claims more than the other procedures.

Also, At IPAnnualReimbursement where the value of loss peaks at $50000 for medicare.

Under With the given available computational power we I were was able to design the model with GridSearch() Random Forest model (random_Forest_model.pkl) giving the best results with standardScaler(), for the given dataset.(IN Models Folder)

The model along with the required Scaler and an python script (NewDataModNB : In UI folder) that can accept any new dataset, and convert it to the required transformed data it that and when fed to scaler and model in same order will give us the required results.

However, with the HTML page will provide the score of the model selected for analysis. For independent verification of model execution given resources I was not able to merge the model to HTML page, but it can be done with the resources as mentioned above fulfilled using the Scaler and the python code, I have provided.

CONTRIBUTION:

Siddharth Satyakam: Complete Project.

<u>CLASSIFICATION OF INPATIENT DATA FOR HEALTHCARE FRAUD ANALYSIS</u>

Siddharth Satyakam (UB Number 50293092)

EAS 574 MACHINE LEARNING Project

SPRING 2020