National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Aniket Jambukar
Student ID: x20185014

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aniket Jambukar
Student ID:	x20185014
Programme:	MSc Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	16/12/2021
Project Title:	Fraudulent Healthcare Providers detection using Machine Learning Algorithms
Word Count:	XXX
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Aniket Jambukar
x20185014

1 Introduction

This manual is a detailed specification of hardware and softwares that were used to develop the research project Fraudulent Healthcare Providers detection using Machine Learning Algorithms.

2 PC Requirements

This section has deatisl of harware and software configurations on which the research project was developed.

2.1 Hardware Specifications

The hardware configuration is as follows figure 1

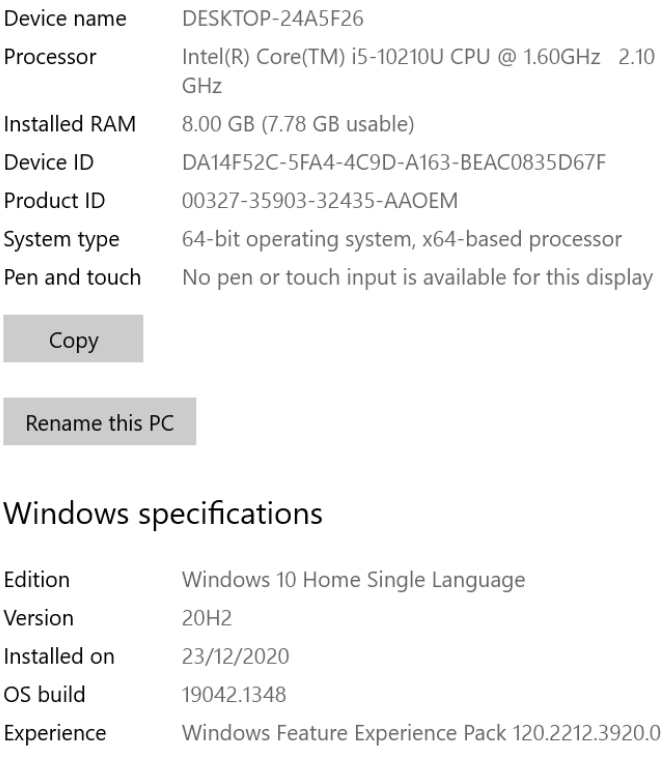

Figure 1: Hardware specifications

3 Software specifications

Anaconda navigator must be installed, the version used during the project is as shown in figure 2 1.10.0.

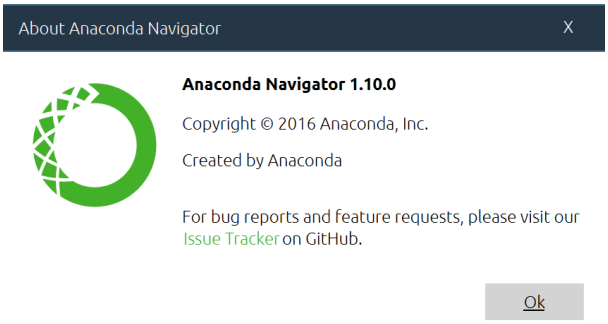

Figure 2: Anaconda Navigator specification

Anaconda navigator should have Jupyter notebook on it and the version used is 6.1.14 and the Python version is 3.8.5as shown in below figure 3.

Figure 3: Jupyter notebook specification

4 Jupyter Notebook

From Anaconda Navigator open Jupyter notebook that will launch on a browser. Click on new on left side of the screen and create a new notebook.

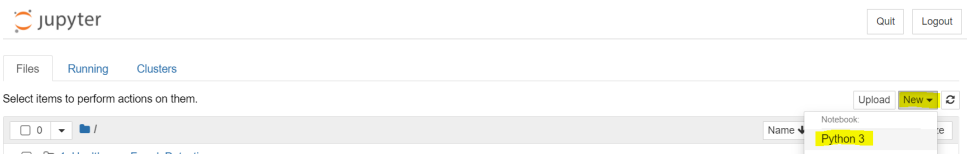

Figure 4: Create notebook

5 Python Libraries

Install all python libraries using command **pip install library name** as below in figure 5. The version during project developed is as shown in below figure 5. The underlying libraries should also be available as present in the codes below.

```
panads version      : 1.1.3
numpy version       : 1.19.2
seaborn version     : 0.11.0
matplotlib version  : 3.3.2
scipy version       : 1.5.2
sklearn version     : 1.0.1
statsmodels version : 0.12.0
imblearn version    : 0.8.1
xgb version         : 1.5.1
pygam version       : 0.8.0
```

Figure 5: Python Libraries

6 Project Code and execution

6.1 Introduction

The code of the project is been divided in three parts. First part is the data pre-processing and feature engineering. Second part is EDA and feature selection. Third part are the models and there evaluations. The datasets were referred form Kaggle[1]. Four training files need to be downloaded and stored in same directory of the Juypter notebook.

6.2 Notebook 1 Data pre-processing and feature engineering

```
# libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')


# souurce files from kaggle from above link
#read all train files from local directory
Train_Data=pd.read_csv('Train-1542865627584.csv')
TRN_Beneficiary_Data=pd.read_csv('Train_Beneficiaryd
↪  ata-1542865627584.csv')
```

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

```
TRN_Inpatient_Data=pd.read_csv('Train_Inpatientdata-1542865627584.csv')
TRN_Outpatient_Data=pd.read_csv('Train_Outpatientdata-1542865627584.csv')
```

6.2.1 Pre-processing of Beneficiary Dataset

```
TRN_Beneficiary_Data.isnull().sum()
```

The output of above code to check null is shown in figure 6

```
BeneID                              0
DOB                                 0
DOD                            137135
Gender                              0
Race                                0
RenalDiseaseIndicator               0
State                               0
County                              0
NoOfMonths_PartACov                 0
NoOfMonths_PartBCov                 0
ChronicCond_Alzheimer               0
ChronicCond_Heartfailure            0
ChronicCond_KidneyDisease           0
ChronicCond_Cancer                  0
ChronicCond_ObstrPulmonary          0
ChronicCond_Depression              0
ChronicCond_Diabetes                0
ChronicCond_IschemicHeart           0
ChronicCond_Osteoporasis            0
ChronicCond_rheumatoidarthritis     0
ChronicCond_stroke                  0
IPAnnualReimbursementAmt            0
IPAnnualDeductibleAmt               0
OPAnnualReimbursementAmt            0
OPAnnualDeductibleAmt               0
dtype: int64
```

Figure 6: Null values in Beneficiary Dataset

```
# Added column IsDead in Train Beneficiary Dataset
TRN_Beneficiary_Data.loc[TRN_Beneficiary_Data.DOD.isna(),'IsDead']=0
TRN_Beneficiary_Data.loc[TRN_Beneficiary_Data.DOD.notna(),'IsDead']=1

# Converted DOB and DOD to data format in Train Beneficiary Dataset
TRN_Beneficiary_Data['DOB'] = pd.to_datetime(TRN_Beneficiary_Data['DOB']
↪  , format = '%Y-%m-%d')
TRN_Beneficiary_Data['DOD'] = pd.to_datetime(TRN_Beneficiary_Data['DOD']
↪  , format = '%Y-%m-%d')

# Calculate Age in Train Beneficiary Dataset by subtracting Date of
↪  date and Date of Birth
TRN_Beneficiary_Data['Age'] = round(((TRN_Beneficiary_Data['DOD'] -
↪  TRN_Beneficiary_Data['DOB']).dt.days)/365)

# Records with DOD null were considered with last DOD from unique
↪  values to calculate age in Train Beneficiary Dataset
```

```
TRN_Beneficiary_Data.Age.fillna(round(((pd.to_datetime('2009-12-01' ,
↪  format = '%Y-%m-%d') -
↪  TRN_Beneficiary_Data['DOB']).dt.days)/365),inplace=True)

# From above unique values replacing RenalDiseaseIndicator from Y to 1
TRN_Beneficiary_Data['RenalDiseaseIndicator'] =
↪  TRN_Beneficiary_Data['RenalDiseaseIndicator'].replace(['Y'],'1')

# Label columns have 2 as No while 1 as Yes indicators
lst_ChronicCond = ['ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
↪  'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
↪  'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
↪  'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
↪  'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
↪  'ChronicCond_stroke']

# Replaced 2 and NO to 0 so that we have standerd label in all columns
↪  as 0 to NO and 1 to YES
for cols in lst_ChronicCond:
    TRN_Beneficiary_Data[cols] =
    ↪  TRN_Beneficiary_Data[cols].replace([2],0)
```

6.2.2 Pre-processing Inpatient dataset

```
# Check null values in inpatient dataset
TRN_Inpatient_Data.isnull().sum()
```

The output of above code to check null is shown in figure 7

```
BeneID                     0
ClaimID                    0
ClaimStartDt               0
ClaimEndDt                 0
Provider                   0
InscClaimAmtReimbursed     0
AttendingPhysician       112
OperatingPhysician     16644
OtherPhysician         35784
AdmissionDt                0
ClmAdmitDiagnosisCode      0
DeductibleAmtPaid        899
DischargeDt                0
DiagnosisGroupCode         0
ClmDiagnosisCode_1         0
ClmDiagnosisCode_2       226
ClmDiagnosisCode_3       676
ClmDiagnosisCode_4      1534
ClmDiagnosisCode_5      2894
ClmDiagnosisCode_6      4838
ClmDiagnosisCode_7      7258
ClmDiagnosisCode_8      9942
ClmDiagnosisCode_9     13497
ClmDiagnosisCode_10    36547
ClmProcedureCode_1     17326
ClmProcedureCode_2     35020
ClmProcedureCode_3     39509
ClmProcedureCode_4     40358
ClmProcedureCode_5     40465
ClmProcedureCode_6     40474
dtype: int64
```

Figure 7: Null values in Inpatient Dataset

```
# Replaced null values of physicians as Missing
TRN_Inpatient_Data.loc[TRN_Inpatient_Data.Attend\\
ingPhysician.isna(),'AttendingPhysician']='Missing'

TRN_Inpatient_Data.loc[TRN_Inpatient_Data.Operat\\
ingPhys\\
ician.isna(),'OperatingPhysician']='Missing'

TRN_Inpatient_Data.loc[TRN_Inpatient_Data.OtherPhysician.isna(),'OtherPhysician']='Mi

# Calculate total physician count for every claim
TRN_Inpatient_Data['NumberOfPhysicians'] =
↪  ((TRN_Inpatient_Data["AttendingPhysician"] != "Missing")*1) +
↪  ((TRN_Inpatient_Data["OperatingPhysician"] != "Missing")*1) +
↪  ((TRN_Inpatient_Data["OtherPhysician"] != "Missing")*1) +
↪  ((TRN_Inpatient_Data["AttendingPhysician"] == "Missing")*0) +
↪  ((TRN_Inpatient_Data["OperatingPhysician"] == "Missing")*0) +
↪  ((TRN_Inpatient_Data["OtherPhysician"] == "Missing")*0)

# Replaced null values of DeductibleAmtPaid with 0 as DeductibleAmtPaid
↪  for others is 1068
TRN_Inpatient_Data.loc[TRN_Inpatient_Data.DeductibleAmtPaid.isna(),'Ded
↪  uctibleAmtPaid']=0

# Converted AdmissionDt and DischargeDt to data format in Test
↪  Inpatient Data
TRN_Inpatient_Data['AdmissionDt'] =
↪  pd.to_datetime(TRN_Inpatient_Data['AdmissionDt'] , format =
↪  '%Y-%m-%d')
TRN_Inpatient_Data['DischargeDt'] =
↪  pd.to_datetime(TRN_Inpatient_Data['DischargeDt'] , format =
↪  '%Y-%m-%d')

# Calculating Days admitted for Inpatients by adding 1 to dates
↪  difference to get right output
TRN_Inpatient_Data['DaysAdmitted'] = ((TRN_Inpatient_Data['DischargeDt']
↪  - TRN_Inpatient_Data['AdmissionDt']).dt.days)+1
```

6.2.3 Pre-processing Outpatient Dataset

```
# Check null values in outpatient dataset
TRN_Outpatient_Data.isnull().sum()
```

The output of above code to check null is shown in figure 8

```
BeneID                       0
ClaimID                      0
ClaimStartDt                 0
ClaimEndDt                   0
Provider                     0
InscClaimAmtReimbursed       0
AttendingPhysician        1396
OperatingPhysician      427120
OtherPhysician          322691
ClmDiagnosisCode_1       10453
ClmDiagnosisCode_2      195380
ClmDiagnosisCode_3      314480
ClmDiagnosisCode_4      392141
ClmDiagnosisCode_5      443393
ClmDiagnosisCode_6      468981
ClmDiagnosisCode_7      484776
ClmDiagnosisCode_8      494825
ClmDiagnosisCode_9      502899
ClmDiagnosisCode_10     516654
ClmProcedureCode_1      517575
ClmProcedureCode_2      517701
ClmProcedureCode_3      517733
ClmProcedureCode_4      517735
ClmProcedureCode_5      517737
ClmProcedureCode_6      517737
DeductibleAmtPaid            0
ClmAdmitDiagnosisCode   412312
dtype: int64
```

Figure 8: Null values in Outpatient Dataset

```
# Created column DaysAdmitted as 0 for outpatient dataset as they were
↪  not admitted
TRN_Outpatient_Data['DaysAdmitted'] = 0

# Replaced null value of physicial as missing
TRN_Outpatient_Data.loc[TRN_Outpatient_Data.Atten\\
dingPhysician.isna(),'AttendingPhysician']='Missing'
TRN_Outpatient_Data.loc[TRN_Outpatient_Data.Operating\\
Physician.isna(),'OperatingPhysician']='Missing'
TRN_Outpatient_Data.loc[TRN_Outpatient_Data.OtherPhys\\
ician.isna(),'OtherPhysician']='Missing'


# Calculate total number of physicians attended
TRN_Outpatient_Data['NumberOfPhysicians'] =
↪  ((TRN_Outpatient_Data["AttendingPhysician"] != "Missing")*1) +
↪  ((TRN_Outpatient_Data["OperatingPhysician"] != "Missing")*1) +
↪  ((TRN_Outpatient_Data["OtherPhysician"] != "Missing")*1) +
↪  ((TRN_Outpatient_Data["AttendingPhysician"] == "Missing")*0) +
↪  ((TRN_Outpatient_Data["OperatingPhysician"] == "Missing")*0) +
↪  ((TRN_Outpatient_Data["OtherPhysician"] == "Missing")*0)
```

6.2.4 Pre-processing for Train Data

```
# converted label of Yes of potentail fraud to 1
Train_Data['PotentialFraud'] =
↪  Train_Data['PotentialFraud'].replace(['Yes'],'1')

# converted label of No of potentail fraud to 0
Train_Data['PotentialFraud'] =
↪  Train_Data['PotentialFraud'].replace(['No'],'0')
```

6.2.5 Merge all data to create a final dataset

```
# taking common columns in inpatient and outpatient datasets to merge
↪  datasets
columnsToCombine = [ idx for idx in TRN_Inpatient_Data.columns if idx in
↪  TRN_Outpatient_Data.columns]

# Merge Inpatient and Outpatient datasets
Fin_TRN_Dataset_v1 = pd.merge(TRN_Inpatient_Data, TRN_Outpatient_Data,
↪  left_on = columnsToCombine, right_on = columnsToCombine, how =
↪  'outer')

# Merge Benificiary data with Inpatient and Outpatient data
Fin_TRN_Dataset = pd.merge(Fin_TRN_Dataset_v1, TRN_Beneficiary_Data,
↪  left_on = 'BeneID', right_on = 'BeneID', how = 'inner')

# there are no values in ClmProcedureCode_6 column so dropped the
↪  column
del Fin_TRN_Dataset['ClmProcedureCode_6']

# List of Diagnosis Codes columns
Diagnosis_Codes = ['ClmAdmitDiagnosisCode', 'ClmDiagnosisCode_1',
↪  'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4',
↪  'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7',
↪  'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10' ]


# List of Procedure Code columns
Proc_Codes = ['ClmProcedureCode_1', 'ClmProcedureCode_2',
↪  'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5']


# A function that will take unique values from all columns of Procedure
↪  and Diagnosis codes
def unqvals(dataframe):
    return np.array([len(set([a for a in y[~pd.isnull(y)]])) for y in
    ↪  dataframe.values])
```

```
# Calculating number of Diagnosis the patient received
Fin_TRN_Dataset['UnqDiagCodes'] =
↪  unqvals(Fin_TRN_Dataset[Diagnosis_Codes])
# Calculating number of Procedures the patient received
Fin_TRN_Dataset['UnqProcCodes'] = unqvals(Fin_TRN_Dataset[Proc_Codes])


# Added column DiagCodeLabel
Fin_TRN_Dataset.loc[Fin_TRN_Dataset.DiagnosisGroupCode.isna(),'DiagCod
↪  eLabel']=0
Fin_TRN_Dataset.loc[Fin_TRN_Dataset.DiagnosisGroupCode.notna(),'DiagCod
↪  eLabel']=1

# Added column DeductAmtLabel
Fin_TRN_Dataset.loc[Fin_TRN_Dataset.DeductibleAmtPaid.isna(),'Ded
↪  uctAmtLabel']=0
Fin_TRN_Dataset.loc[Fin_TRN_Dataset.DeductibleAmtPaid.notna(),'Ded
↪  uctAmtLabel']=1

# Adding columns if the patient visited patients
Fin_TRN_Dataset['AttendingPhysicianCnt'] =
↪  ((Fin_TRN_Dataset["AttendingPhysician"] != "Missing")*1) +
↪  ((Fin_TRN_Dataset["AttendingPhysician"] == "Missing")*0)
Fin_TRN_Dataset['OperatingPhysicianCnt'] =
↪  ((Fin_TRN_Dataset["OperatingPhysician"] != "Missing")*1) +
↪  ((Fin_TRN_Dataset["OperatingPhysician"] == "Missing")*0)
Fin_TRN_Dataset['OtherPhysicianCnt'] =
↪  ((Fin_TRN_Dataset["OtherPhysician"] != "Missing")*1) +
↪  ((Fin_TRN_Dataset["OtherPhysician"] == "Missing")*0)
```

6.2.6 Create Aggregated Datasets

```
# Calculating total unique number of benificary the provider treated
↪  and total number of claims the provider submitted
FIN_COUNTS = Fin_TRN_Dataset[['BeneID',
↪  'ClaimID']].groupby(Fin_TRN_Dataset['Provider']).nunique().reset_ind
↪  ex()


# calculating mean of Age of patients, Number of months of Part A and B
↪  coverage of insurance
FIN_MEANS = round(Fin_TRN_Dataset.groupby(['Provider'], as_index =
↪  False)[['NoOfMonths_PartBCov', 'NoOfMonths_PartACov', 'Age']].mean())


# created a dataset with provider ID with all sum of features
FIN_DATASET = Fin_TRN_Dataset.groupby(['Provider'], as_index =
↪  False)[cols_to_sum].sum()
```

6.2.7 Merge datasets

```
# merged Beneficary and claims counts to sum dataset
FIN_DATASET = pd.merge(FIN_DATASET, FIN_COUNTS, left_on = 'Provider',
↪  right_on = 'Provider', how = 'inner')

# merged mean data to final dataset
FIN_DATASET = pd.merge(FIN_DATASET, FIN_MEANS, left_on = 'Provider',
↪  right_on = 'Provider', how = 'inner')

# merged provider label of potential fraud to final dataset
FIN_DATASET = pd.merge(FIN_DATASET, Train_Data, left_on = 'Provider',
↪  right_on = 'Provider', how = 'inner')

# droping provider ID columns that has unique ID's of all providers
del FIN_DATASET['Provider']

# save dataframe to local directory
FIN_DATASET.to_pickle('FIN_DATASET.pkl')
```

A .pkl file would be saved that need to be used in next input of notebook.

6.3 Notebook 2 EDA and feature selection

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot
import matplotlib.pyplot as plt
from scipy import stats
from scipy.special import boxcox1p
from sklearn.preprocessing import PowerTransformer
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.multivariate.manova import MANOVA
from statsmodels.stats.multicomp import pairwise_tukeyhsd

import warnings
warnings.filterwarnings('ignore')

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.model_selection import train_test_split


# read dataframe from local directory
FIN_DATASET = pd.read_pickle('FIN_DATASET.pkl')
```

```
\\
# pie plot of potential fraud label
# this a highly imbalanced dataset
plt.pie(FIN_DATASET['PotentialFraud'].value_counts(), autopct="%1.2f%%",
↪  labels = ['Not-Fraud', 'Fraud'], explode = [0, 0.1])
plt.suptitle('Potential Fraud Pie Plot')
```

The output of above code of class imbalance is in figure 9

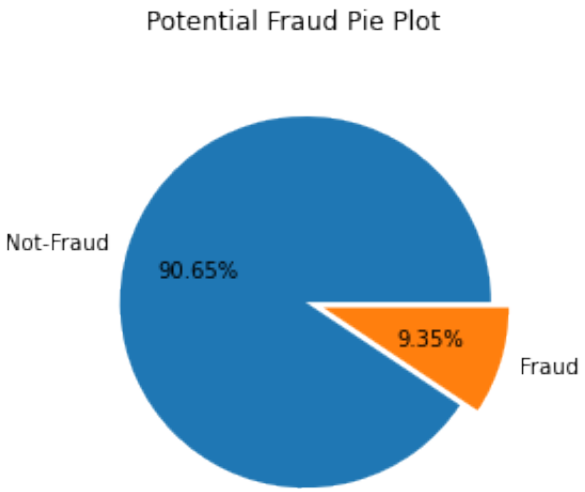

Figure 9: Pie chart of Class imbalance in dataset

```
sns.barplot(FIN_DATASET["PotentialFraud"],FIN_DATASET["BeneID"],
↪  hue=FIN_DATASET["PotentialFraud"])
plt.xlabel('Potential Fraud')
plt.ylabel('BeneID Counts')
plt.suptitle('Potential Fraud vs BeneID Counts')

# This barplot shows mean of Beneficiary ID for potential fraud
# we can say there is potental fraud when Benificary count is more as
↪  its mean is more
```

The output of above code in figure 10

```
sns.barplot(FIN_DATASET["PotentialFraud"],FIN_DATASET["ClaimID"],
↪  hue=FIN_DATASET["PotentialFraud"])
plt.xlabel('PotentialFraud')
plt.ylabel('ClaimID Counts')
plt.suptitle('PotentialFraud vs ClaimID Counts')

# This barplot shows mean of Claims ID for potential fraud
# we can say there is potental fraud when Claims count is more as its
↪  mean is more
```

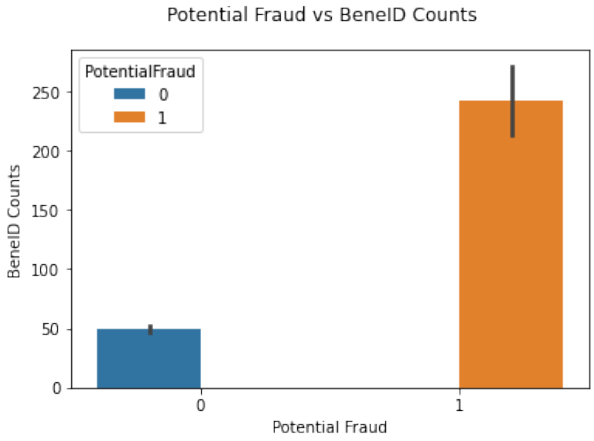

Figure 10: BeneID mean plot

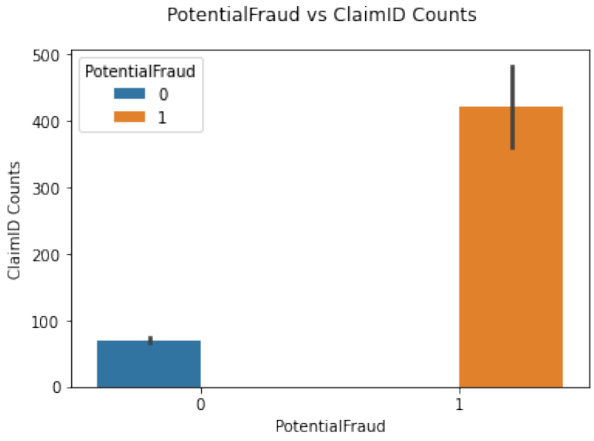

Figure 11: ClaimID Mean Plot

The output of above code is shown in figure 11

```
sns.barplot(FIN_DATASET["PotentialFraud
↪  "],FIN_DATASET["InscClaimAmtReimbursed"],
↪  hue=FIN_DATASET["PotentialFraud"])
plt.xlabel('PotentialFraud')
plt.ylabel('InscClaim Amount Reimbursed')
plt.suptitle('PotentialFraud vs InscClaim Amount Reimbursed')

# This barplot shows mean of Insurance claims amount for potential
↪  fraud
# we can say there is potental fraud when Insurance claims amount is
↪  more as its mean is more
```

The output of above code is shown in figure 13

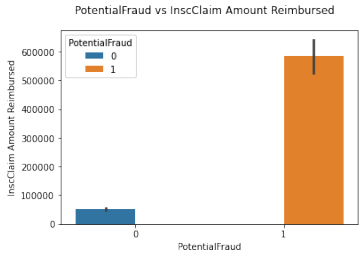

Figure 12: InscClaimAmtReimbursed Mean plot

```
# As values contains 0 to transform used boxcox1 method that will add a
↪  constant to all values and then log transform them
transform_pt = PowerTransformer(method='yeo-johnson')
FIN_DATASET_PT = FIN_DATASET
FIN_DATASET_v1 = FIN_DATASET
transform_pt.fit(FIN_DATASET_PT)
col_num = 0
for cols in distcols:
    FIN_DATASET_v1[cols] = boxcox1p(FIN_DATASET[cols],
    ↪  transform_pt.lambdas_[col_num])
    col_num = col_num + 1
```

6.3.1 MANOVA Hypothesis Testing

```
maov_tst = MANOVA.from_formula('InscClaimAmtReimbursed+Attend
↪  ingPhysicianCnt+OperatingPhysicianCnt+OtherPhysicianCnt+NumberOfPhysicians\\
+RenalDiseaseIndicator+ChronicCond_Alzheimer+ChronicCond
↪  _Heartfailure+ChronicCond_KidneyDisease+ChronicCond
↪  _Cancer+ChronicCond_ObstrPulmonary+ChronicCond
↪  _Depression+ChronicCond_Diabetes+ChronicCond
↪  _IschemicHeart+ChronicCond_Osteoporasis+ChronicCond_rheumatoid
↪  arthritis+ChronicCond_stroke+IPAnnualReimbursementAmt+IPAnnualDed
↪  uctibleAmt+OPAnnualReimbursementAmt+OPAnnualDeductibleAmt+IsDead
↪  +UnqDiagCodes+UnqProcCodes+DiagCodeLabel+DeductibleAmtPaid+DaysAd
↪  mitted+BeneID+ClaimID+Age~PotentialFraud',
↪  data=FIN_DATASET_v1)

print(maov_tst.mv_test())

##Performing Univariate Analysis on all features was done.

reg = ols('InscClaimAmtReimbursed~PotentialFraud',d
↪  ata=FIN_DATASET_v1).fit()
aov = sm.stats.anova_lm(reg,type=2)
print(aov)


## Post hoc Analysis was done on all features
mc = pairwise_tukeyhsd(FIN_DATASET_v1['InscClaimAmtReimbursed
↪  '],FIN_DATASET_v1['PotentialFraud'],alpha=0.001)
print(mc)

mc = pairwise_tukeyhsd
↪  (FIN_DATASET_v1['Age'],FIN_DATASET_v1['PotentialFraud'],alpha=0.001)
print(mc)
```

Age feature was removed from following result in figure 13

```
 Multiple Comparison of Means - Tukey HSD, FWER=0.00
====================================================
group1 group2 meandiff p-adj   lower    upper  reject
----------------------------------------------------
     0      1   -4.039 0.7465 -43.5768 35.4989  False
----------------------------------------------------
```

Here with the rejection status of False means that there is an Alternate Hypothesis as there is no significance difference in mean Age existing in between PotentialFraud

Figure 13: Post hoc analysis of Age feature

```
# checking correlations of all features in dataset
corl=FIN_DATASET_v1.corr()
corl
```

```
# print corealtion matrix
pyplot.figure(figsize=(15,10))
sns.heatmap(corl, annot= True)
```

The output of above code is shown in figure 14

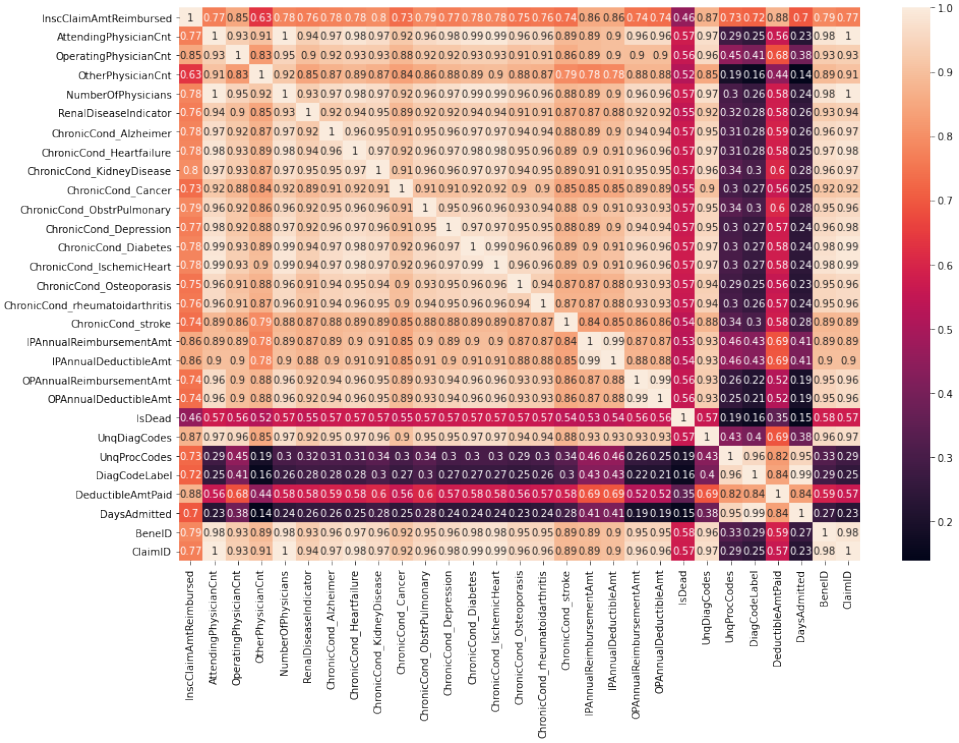

Figure 14: Co-relation Matrix

6.3.2 Feature selection ANOVA f-test

```
# create dataframe
X_val=FIN_DATASET_v1.drop("PotentialFraud",axis=1)
y_val=FIN_DATASET_v1.PotentialFraud

# split to train and test
X_train, X_test, y_train, y_test = train_test_split(X_val, y_val,
↪  test_size=0.30, random_state=1)
print('Training data shape', X_train.shape, y_train.shape)
print('Testing data shape', X_test.shape, y_test.shape)

# feature selection
sel_features = SelectKBest(score_func=f_classif, k='all')
sel_features.fit(X_train, y_train)
Xfstrain = sel_features.transform(X_train)
Xfstest = sel_features.transform(X_test)
```

```
# f-scores of all features
f_scores = []
for a in range(len(sel_features.scores_)):
    print('Score of feature %d: %f' % (a, sel_features.scores_[a]),
    ↪  X_val.columns[a])
    f_scores.append(sel_features.scores_[a])


# selecting best features
cols_selected = []
for i in range(len(sel_features.scores_)):
    if sel_features.scores_[i] > 700:
        print('Feature Score %d: %f' % (i, sel_features.scores_[i]),
        ↪  X_val.columns[i])
        cols_selected.append(X_val.columns[i])

cols_selected.append('PotentialFraud')

# new dataframe with selected features
FIN_DATASET_v2 = FIN_DATASET_v1[cols_selected]

# save dataframe to local directory
FIN_DATASET_v2.to_pickle('FIN_DATASET_MODEL.pkl')
```

Dataframe is saved in .pkl file that is used for next notebook.

7 Section 5

Your fifth section. Change the header and label to something appropriate.

8 Section 6

Your sixth section. Change the header and label to something appropriate.

8.1 Notebbok 3 Models and Evaluation

```
import pandas as pd
import numpy as np
from imblearn.combine import SMOTETomek
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import
↪  confusion_matrix,accuracy_score,classification_report,recall_score
import seaborn as sns
import matplotlib.pyplot as plt
from imblearn.combine import SMOTETomek
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.svm import SVC
from xgboost import XGBClassifier
from pygam import LogisticGAM
import warnings
warnings.filterwarnings('ignore')


# read dataframe from local directory
FIN_DATASET_MODEL = pd.read_pickle('FIN_DATASET_MODEL.pkl')

# create dataframes for models
X=FIN_DATASET_MODEL.drop("PotentialFraud",axis=1)
y=FIN_DATASET_MODEL.PotentialFraud

# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
↪  random_state=1)

print('Training Dataframe:', X_train.shape, y_train.shape)
print('Testing Dataframe:', X_test.shape, y_test.shape)

# As dataset only has 9.35% of fraud instances using SMOTETomek
↪  sampling technique to balance dataset
SMT_sampling = SMOTETomek(random_state = 42)
X_train_st,y_train_st=SMT_sampling.fit_resample(X_train,y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_st)))


# plotting sampled data
fig, axes = plt.subplots(1, 2, figsize=(10,5))
ax = sns.countplot(y_train, ax=axes[0], palette=["#fc9272", "#fee0d2"])
ax = sns.countplot(y_train_st, ax=axes[1], palette=["#fc9272",
↪  "#fee0d2"])
```

The output of above code is shown in figure 15

8.1.1 Random Forest Model

```
RFC_MODEL=RandomForestClassifier(random_state=42)
rfc_MODEL=RFC_MODEL.fit(X_train_st,y_train_st)
y_pred_rf=rfc_MODEL.predict(X_test)

print('Random Forest Model: Classification Report
↪  \n',classification_report(y_test,y_pred_rf))
sns.heatmap(confusion_matrix(y_test,y_pred_rf),annot=True, cmap="Blues",
↪  fmt="d",
            xticklabels = ['Predicted Non-fraud', 'Predicted Fraud'],
            yticklabels = ['Actual Non-fraud', 'Actual Fraud'])
```

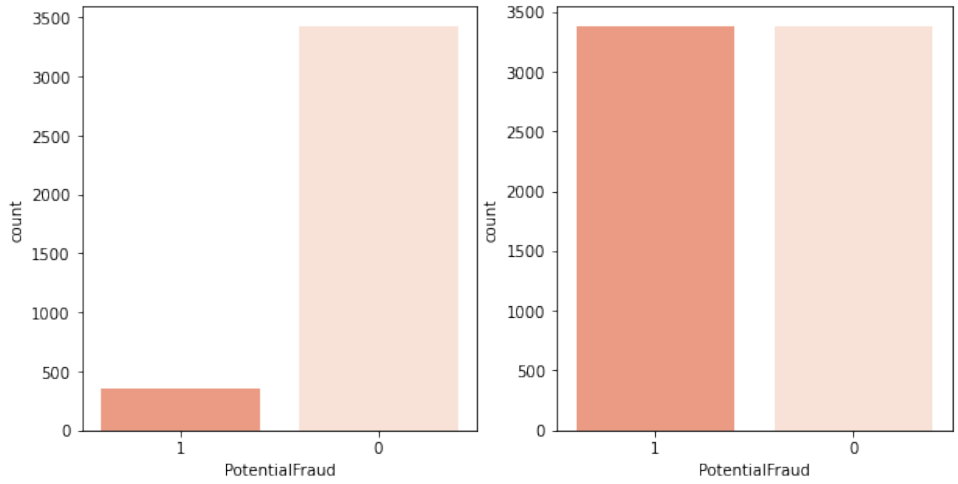

Figure 15: SMOTETomek Sampling

The output of above code is shown in figure 16

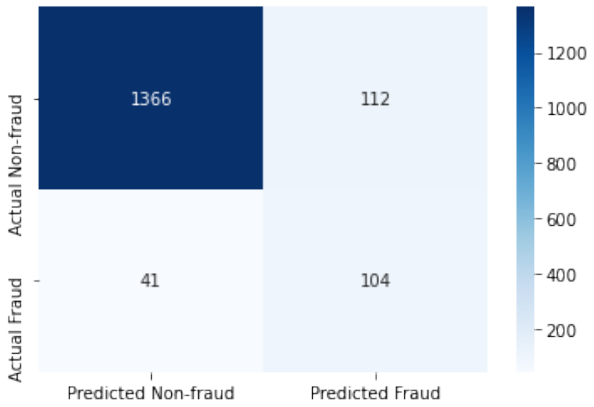

Figure 16: Random forest classification matrix

8.1.2 SVM Model

```
SVM_MODEL=SVC(C=1.0, kernel='rbf', degree=3, gamma='auto')
svm_MODEL=SVM_MODEL.fit(X_train_st,y_train_st)
y_pred_svm=svm_MODEL.predict(X_test)
```

```
print('SVM Model: Classification Report
↪  \n',classification_report(y_test,y_pred_svm))
sns.heatmap(confusion_matrix(y_test,y_pred_svm),annot=True, cmap="Blues",
↪  fmt="d",
            xticklabels = ['Predicted Non-fraud', 'Predicted Fraud'],
            yticklabels = ['Actual Non-fraud', 'Actual Fraud'])
```

The output of above code is shown in figure 17

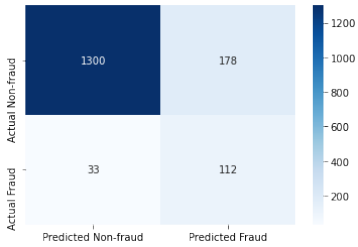

Figure 17: SVM classification matrix

8.1.3 XGBoost Model

```
xg_model=XGBClassifier(max_depth=2, min_child_weight=5,
↪  learning_rate=0.1, \
        n_estimators=100, seed=0, subsample=0.8, colsample_bytree=0.8, \
        objective='binary:logistic')
xgb_MODEL=xg_model.fit(X_train_st,y_train_st)
y_pred_xgb=xgb_MODEL.predict(X_test)

print('XGBoost Model: Classification Report
↪  \n',classification_report(y_test,y_pred_xgb))
sns.heatmap(confusion_matrix(y_test,y_pred_xgb),annot=True, cmap="Blues",
↪  fmt="d",
            xticklabels = ['Predicted Non-fraud', 'Predicted Fraud'],
            yticklabels = ['Actual Non-fraud', 'Actual Fraud'])
```

The output of above code is shown in figure 18

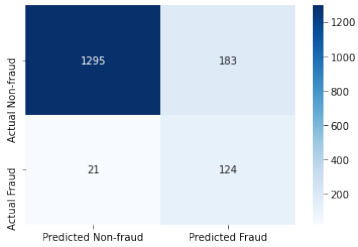

Figure 18: XGBoost classification matrix

8.1.4 Logistic GAM Model

```
lambda_ = 0.5
n_splines = 10
constraints = None

gm = LogisticGAM(constraints=constraints, lam=lambda_,
↪  n_splines=n_splines)
gm_MODEL=gm.fit(X_train_st,y_train_st)
y_pred_gam=gm_MODEL.predict(X_test)
```

```
y_pred_gam

new_gam = y_pred_gam.astype(str)

mmap = {"False": "0", "True": "1"}
n_new_gam = np.vectorize(mmap.get)(new_gam)

new_pred_gam = n_new_gam.astype(np.object)

new_pred_gam

print('LogisticGAM model: Classification
↪  Report\n',classification_report(y_test,new_pred_gam))

sns.heatmap(confusion_matrix(y_test,new_pred_gam),annot=True,
↪  cmap="Blues", fmt="d",
            xticklabels = ['Predicted Non-fraud', 'Predicted Fraud'],
            yticklabels = ['Actual Non-fraud', 'Actual Fraud'])
```

The output of above code is shown in figure 19

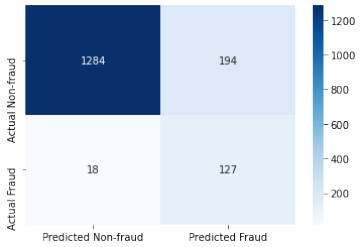

Figure 19: LogisticGAM classification matrix

The comparative visualization of four models is as below figure 20

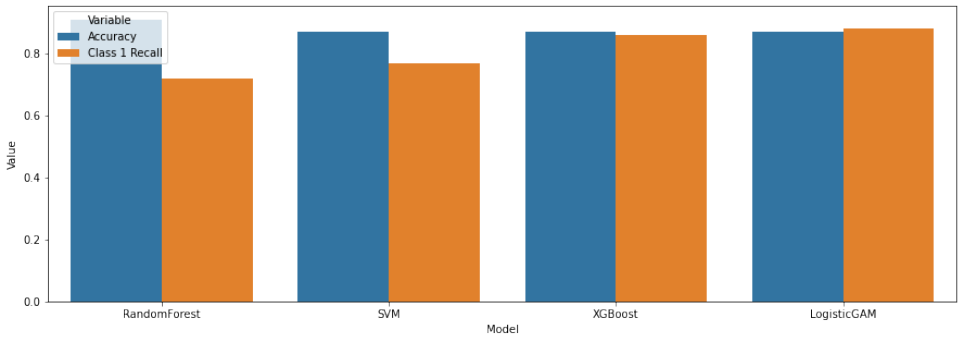

Figure 20: Comparative evaluation