Big Data-Medicare Fraud Detection

ARIZONA STATE UNIVERSITY

Advanced Analytics of Big data

Dr. Robert Rucker

12/3/2022

Shalini Pilli-1223275235

Shreya Naresh- 1222887120

Harsha Bapatla- 1223535820

TABLE OF CONTENTS

1) Why are you choosing this task? **3**

2) What are the organizational benefits of doing so, that is, list at least 5 questions your proposal will address. **4**

3) What dataset are you proposing to use? **5**

4) What technologies do you propose to us? **7**

5) Now comment on the CRISP steps and how your proposal matches them. **8**

TECHNOLOGIES EMPLOYED: **11**

RESULTS: **12**

1. Identify if a particular case is a potential fraud or not? 12
2. Identifying the region with most frauds in the medicare industry? 12
3. Which is the best model used that gives higher accuracy value based on the comparison of different models tested on a given dataset? 15
4. Which age group has committed the most number of fraudulent cases? 23
5. Identify the top providers with fraudulent cases. 26

SUMMARY **27**

CONCLUSION **28**

REFERENCES **29**

1. Why are you choosing this task?

In the United States, there is a sizable healthcare business with both private and public organizations. The cost of medical services keeps rising, in part as a result of the aging population growth. Between 2012 and 2014, U.S. health insurance spending increased by 6.7% to reach $3 trillion, while Medicare spending, at nearly $600 billion, accounts for 20% of all human services spending in the country. Due to the growing older population and the rising expenditures of Medicare, cost-cutting measures are required. Reducing fraud is one way to assist cover costs and lower overall payments. The goal of this project is to develop big data solutions that have significant applicability at the intersection of the health care and protection sectors.

2) What are the organizational benefits of doing so, that is, list at least 5 questions your proposal will address.

1. Identify if a particular case is a potential fraud or not?
2. Identifying the region with most frauds in the medicare industry?
3. Which is the best model used that gives higher accuracy value based on the comparison of different models tested on a given dataset?
4. Which age group has committed the most number of fraudulent cases?
5. Identify the top providers with fraudulent cases.

3) What dataset are you proposing to use?

We have selected the database from three different resources that sums up our problem. This dataset was created using data from administrative claims submitted by Medicare beneficiaries who are Part D program participants and are available from the CMS Chronic Condition Data Warehouse.They are taken from:

1) CMS Prescriber Data 2017
2) Payment Data 2017
3) Excluded (LEIE) dataset

In the first dataset, we have divided the provider details as with only one unique column that is the ProviderID. BeneID (Int), ClaimID (Int), ClaimStartDt, ClaimEndDt are unique values from the column. BeneID refers to a String value known as the Beneficiary ID, ClaimID is the person ID who claims the insurance of the patient. ClaimStart Date is the start of the Health Insurance claim and the end date denoted as ClaimEndDt.

Further, we have The Provider details. This includes:

1) ProviderID(Int)
2) InsuranceProvider(String)
3) Operating PhysicianID(If Applicable) (Int)

Next, the beneficiary data includes;

1) BeneficiaryID(Int)
2) DOB (Date)

3) Gender (1-Male, 2-Female)
4) Race(Int)- Represented with the specific race IDs assigned.
5) County(Int)- Represented with the specific county IDs.
6) Number of Months(Int)

Potential Fraud rate is termed Yes/ No with the help of logistic regression, It finds the column matching and links to the data considering the ProviderID as the unique field.

4) What technologies do you propose to us?

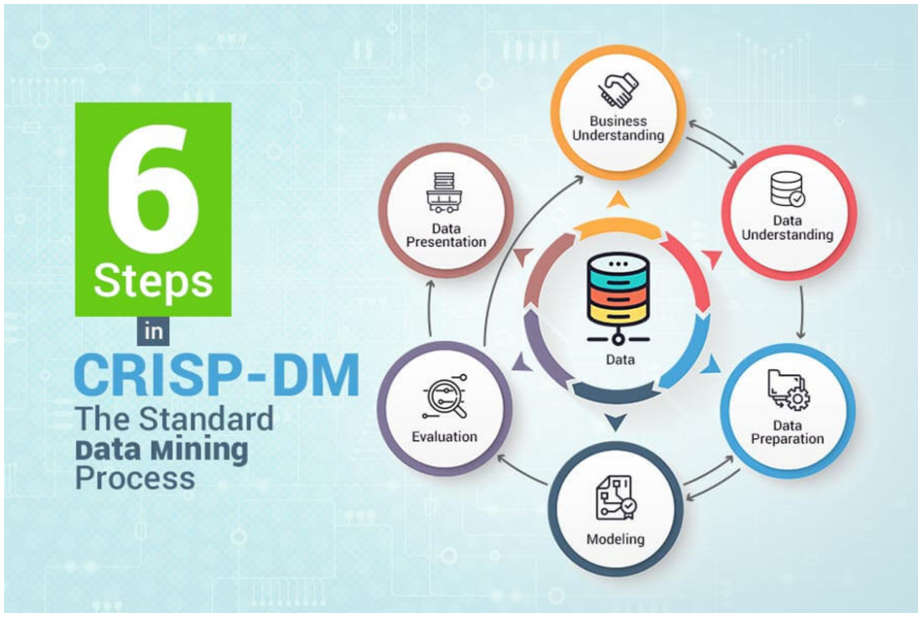
Here we are going to use Python, Tableau and Spark. Python has some advantages as:

1) Easy to learn, read and write
2) Improved Productivity
3) Vast Library Support

We further plan to use Jupyter notebook as our IDE to work with the project. Jupyter is ideal for data scientists who need a way to create quick data visualizations. However, source code is stored as HTML and readable by Jupyter rather than Python IDEs.. With Jupyter we can have In-line code code execution using the blocks which supports a single line graphing. Jupyter does have unique coding features as well, but mostly aimed at visualization. This includes the ability to graph or visualize individual lines of code or data.

When using tableau we can make better visualizations for the data and so is the case while using the spark insight tool. We also have pre-processing tools readily available on tableau instead of performing the data cleaning on jupyter everytime and storing the data. At the end, the Spark insight tools uses in-memory caching and efficient query execution to provide rapid queries against any size of data. Spark, in a nutshell, is a quick and general engine for large-scale data processing.

5) Now comment on the CRISP steps and how your proposal matches them.



The CRISP-DM cycle involves six steps:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Business Understanding

Under this section, we are trying to understand the different models that we are using here. The first step is to visualize the data in various forms. These include Data exploration, cleansing and preparation. Then we are building a simple data model to join all datasets. Here, we plan to use feature engineering to choose the effective feature sets for the different fraud patterns. Later we are building a machine learning model to detect the different fraud patterns.

Data Understanding

This step involves understanding the raw data that will be used to solve the problem. This means understanding the data attributes and problems as well as the relationship between data attributes and the problem. That is the one main reason we do data pre-processing to eliminate unwanted fields and null values from the dataset. The pre-processing steps include cleaning the data by removing duplicates, removal of outliers, factoring the categorical data, removing the data based on general information only needed for analysis and performing data sampling. The dataset looks more imbalanced and so in terms of fraud detection context the dataset is very skewed with 99% no fraudulent cases and less than 1% fraudulent cases.

Data Preparation

After understanding the business and data, the next step is data preparation. In this step, some classification codes are run on the data set using machine learning or data mining library. For data mining activities, each library provides functions that take data from files and require the data set to be saved in a specific format (CSV). In order to use machine learning or a data mining library, we have identified the data for our project and will store it in CSV format.

Modeling

This step involves applying data mining methods, running actual code, building models to solve the data mining task. In this step, we have performed logistic regression for machine learning modeling and Gaussian naive bayes classifier. We will ensure that these patterns represent the true regularities in the data and not some anomalies that do not generalize beyond the data used for mining. Standardscaler is used for the main task when performing feature engineering. Random forest classifier and Gradient boosting classifier are two main concepts we use for analyzing the data after feature engineering is performed.

Evaluation

After running the code and performing the required tasks, the results are obtained based on the modeling we performed with the data. We evaluate the dataset based on the results we obtained. We are making sure that these patterns reflect the data's genuine regularities rather than some anomalies that are specific to the data utilized for mining. We check if all types of fraud patterns are detected and we figure out the best resulting model considering all the factors used for ML and analysis.

Deployment

This is the concluding stage for the data analysis and the machine learning process we have performed to be brought into execution. In order to get some return on investment, it includes applying the data mining results in decision-making. To automate the decision-making process, more data is gathered and we use the data mining techniques to make better visualizations and perform better analysis. At the end, we see that a random forest classifier gives an accuracy rate of 72%.

TECHNOLOGIES EMPLOYED:

Python: We have used pandas to read and check the column number using a shape function, a data frame that drops the column that has the most frequently repeating or common values.

Exploratory Data Analysis: EDA is used in our 1st question where we check the beneficiary data and deduce that most of the beneficiaries are from the YEAR - 1919 to 1943.

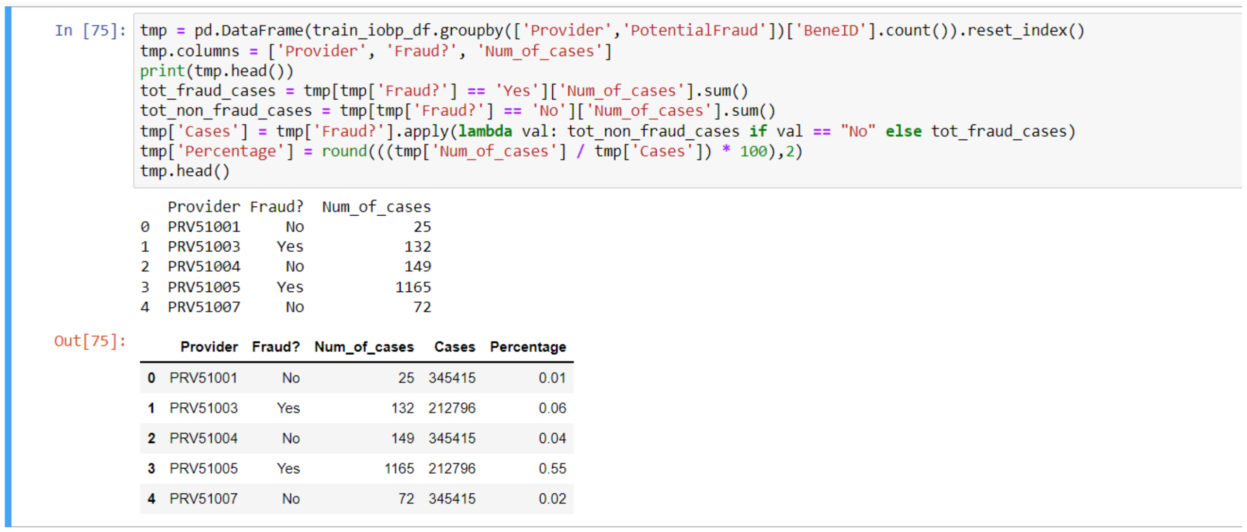
Feature engineering: An example of using feature engineering is where we see that there is no difference in the distribution of Claim Duration for Potentially Fraud and Non-Fraud Providers. Here, we reach the conclusion that Claim Duration alone might not be useful in segregating the Fraud cases.

RESULTS:

1. Identify if a particular case is a potential fraud or not?

This can be a potential sign of fraud prevention because many hospitals in rural areas don't have much facilities or equipment to perform surgeries and provide medical care. The insurance companies are targeting these types of hospitals on a large scale and filing fraudulent claims with them. The identifying if the particular provider is fraudulent or not is done below and we found that the provider with the providerID PRV51005 has the most number of cases.

CODE:



```
In [75]: tmp = pd.DataFrame(train_iobp_df.groupby(['Provider','PotentialFraud'])['BeneID'].count()).reset_index()
         tmp.columns = ['Provider', 'Fraud?', 'Num_of_cases']
         print(tmp.head())
         tot_fraud_cases = tmp[tmp['Fraud?'] == 'Yes']['Num_of_cases'].sum()
         tot_non_fraud_cases = tmp[tmp['Fraud?'] == 'No']['Num_of_cases'].sum()
         tmp['Cases'] = tmp['Fraud?'].apply(lambda val: tot_non_fraud_cases if val == "No" else tot_fraud_cases)
         tmp['Percentage'] = round(((tmp['Num_of_cases'] / tmp['Cases']) * 100),2)
         tmp.head()

            Provider Fraud?  Num_of_cases
         0  PRV51001     No            25
         1  PRV51003    Yes           132
         2  PRV51004     No           149
         3  PRV51005    Yes          1165
         4  PRV51007     No            72
```

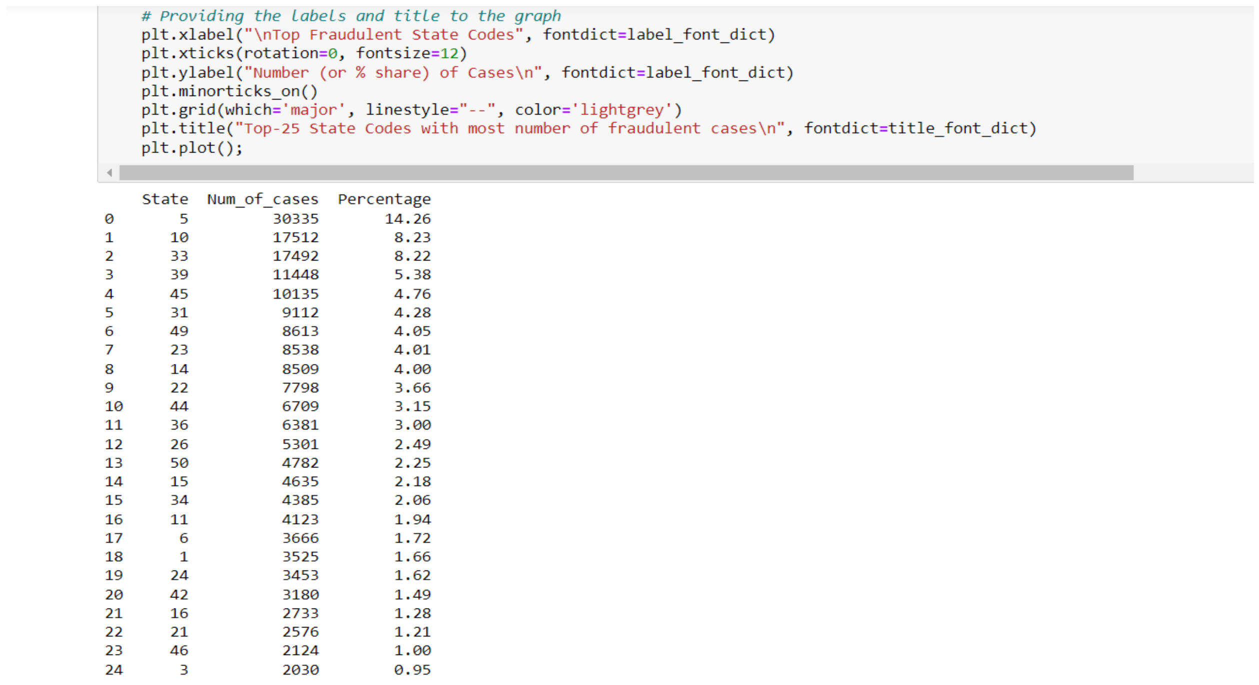
Out[75]:

	Provider	Fraud?	Num_of_cases	Cases	Percentage
0	PRV51001	No	25	345415	0.01
1	PRV51003	Yes	132	212796	0.06
2	PRV51004	No	149	345415	0.04
3	PRV51005	Yes	1165	212796	0.55
4	PRV51007	No	72	345415	0.02

2. Identifying the region with most frauds in the medicare industry?

This can prove to help certain different places around the region to identify the most frauds and later the medical person can be more cautious towards beneficiaries proving to have any medical illness and can be cautious with their medical claims too. Hence, we identified that the region with state code 5 (i.e Arkansas) has the most number of frauds in the medicare industry. [1]

CODE:

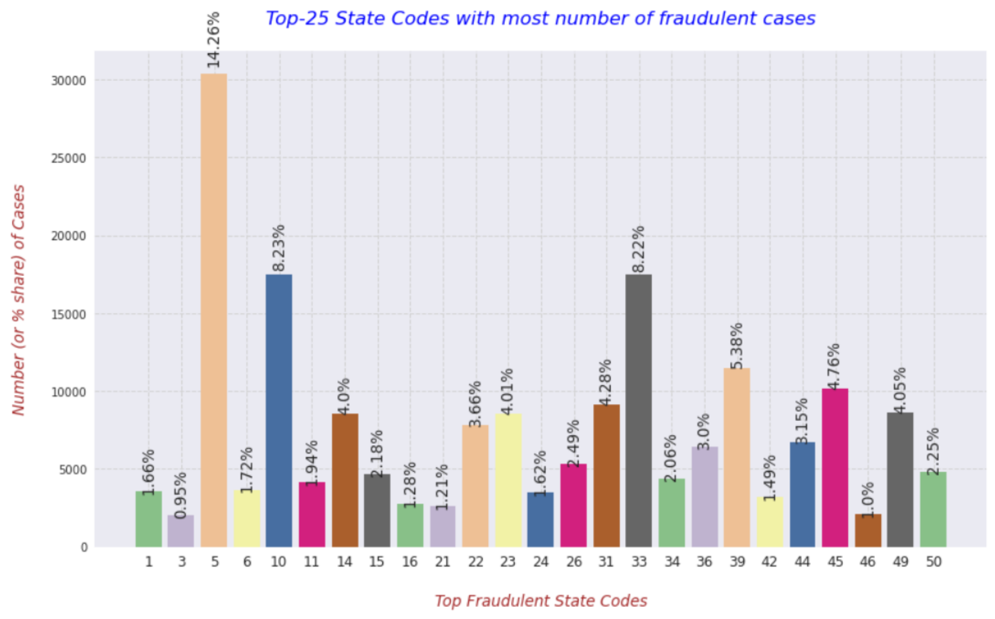


```
# Providing the labels and title to the graph
plt.xlabel("\nTop Fraudulent State Codes", fontdict=label_font_dict)
plt.xticks(rotation=0, fontsize=12)
plt.ylabel("Number (or % share) of Cases\n", fontdict=label_font_dict)
plt.minorticks_on()
plt.grid(which='major', linestyle="--", color='lightgrey')
plt.title("Top-25 State Codes with most number of fraudulent cases\n", fontdict=title_font_dict)
plt.plot();
```

```
    State  Num_of_cases  Percentage
0       5         30335       14.26
1      10         17512        8.23
2      33         17492        8.22
3      39         11448        5.38
4      45         10135        4.76
5      31          9112        4.28
6      49          8613        4.05
7      23          8538        4.01
8      14          8509        4.00
9      22          7798        3.66
10     44          6709        3.15
11     36          6381        3.00
12     26          5301        2.49
13     50          4782        2.25
14     15          4635        2.18
15     34          4385        2.06
16     11          4123        1.94
17      6          3666        1.72
18      1          3525        1.66
19     24          3453        1.62
20     42          3180        1.49
21     16          2733        1.28
22     21          2576        1.21
23     46          2124        1.00
24      3          2030        0.95
```

RESULTS:

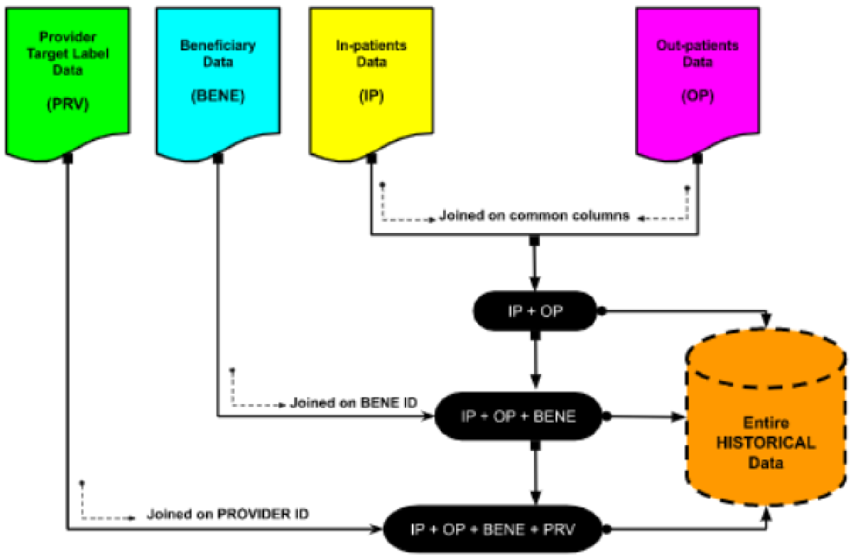
```
22      21          2576        1.21
23      46          2124        1.00
24       3          2030        0.95
```

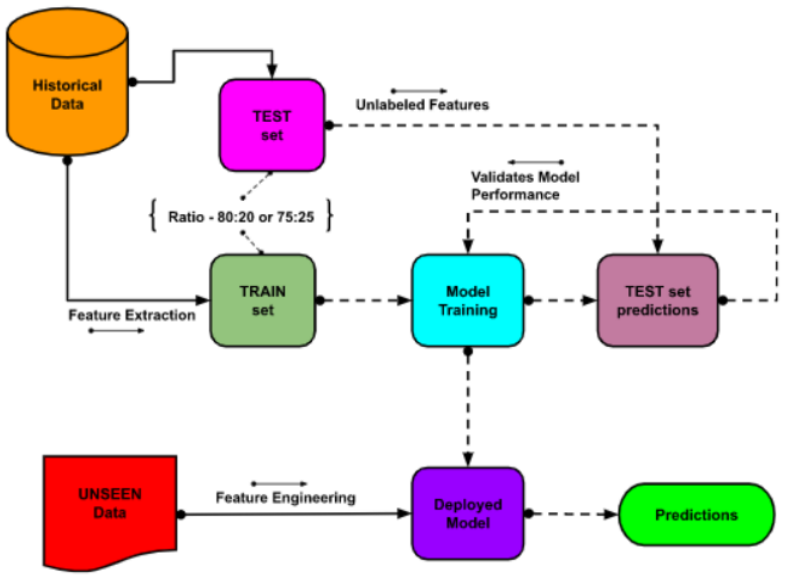


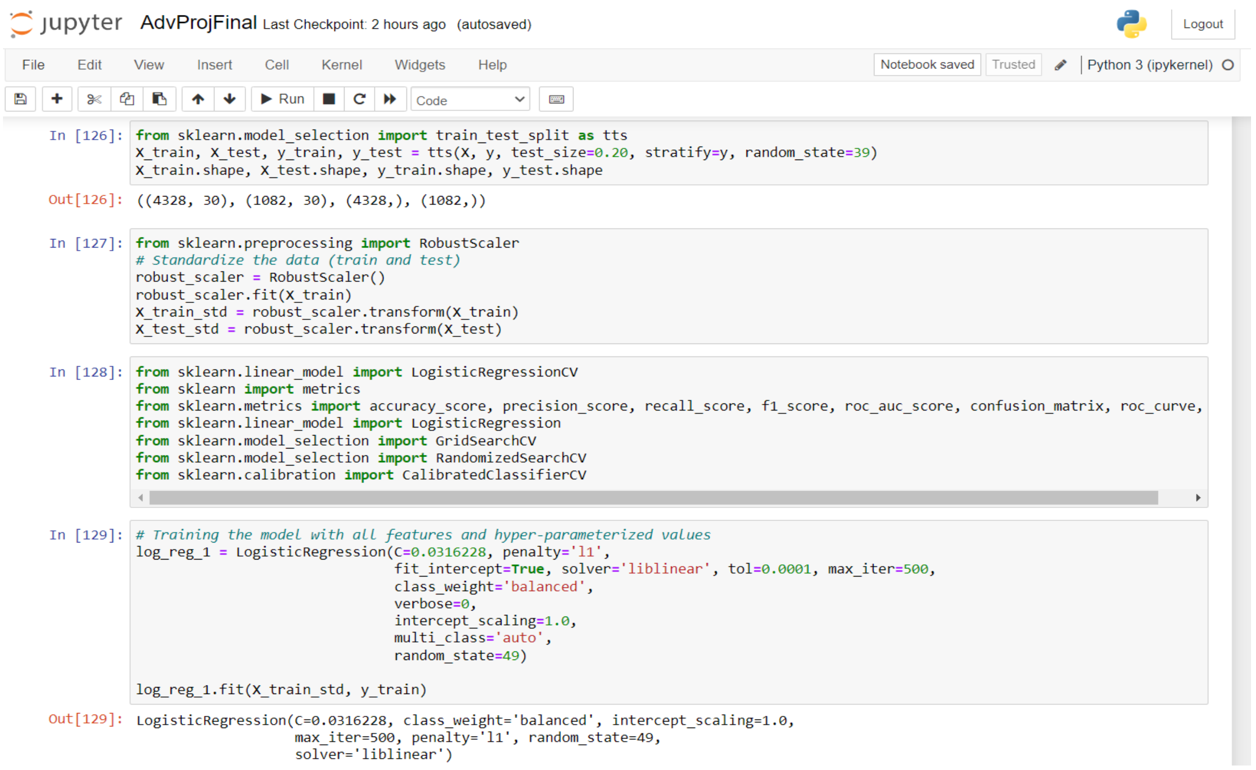
3. **Which is the best model used that gives higher accuracy value based on the comparison of different models tested on a given dataset?**

We could identify the best model using the machine learning Models. First, we filtered the unwanted columns and added the required columns to the dataset. We merged the final EDA dataset, so we could not start our data segregation process. The main objective is to predict the Medicare Provider Fraud. Thus, here we are grouping the entire dataset at the level of PROVIDER and taking SUM of all the columns to create n-dimensional representation of each provider. Train and Validation sets were created for measuring the performance of the ML models. The first model performed was the logistic regression with an accuracy of 0.9592. The second model performed was the Decision Tree 0.9507. Then, we performed a random forest classifier that had an accuracy of 0.9511. After performing these ML models we further performed minority synthetic oversampling so we can create the train and validation sets and get every feature into the same scale. Out of the models we performed and the visualizations we got, We concluded that the random forest classifier is the best performing model for our dataset. Below is the process that we followed:

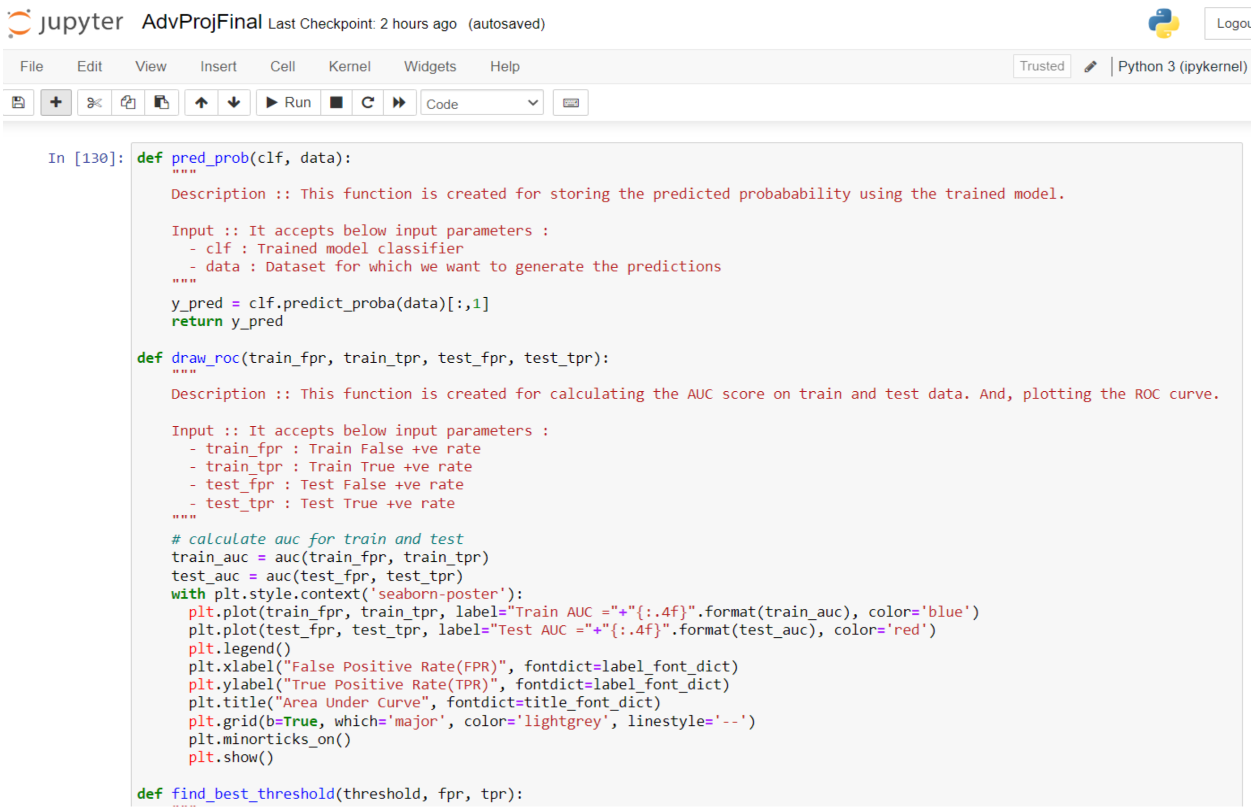
RESULTS:



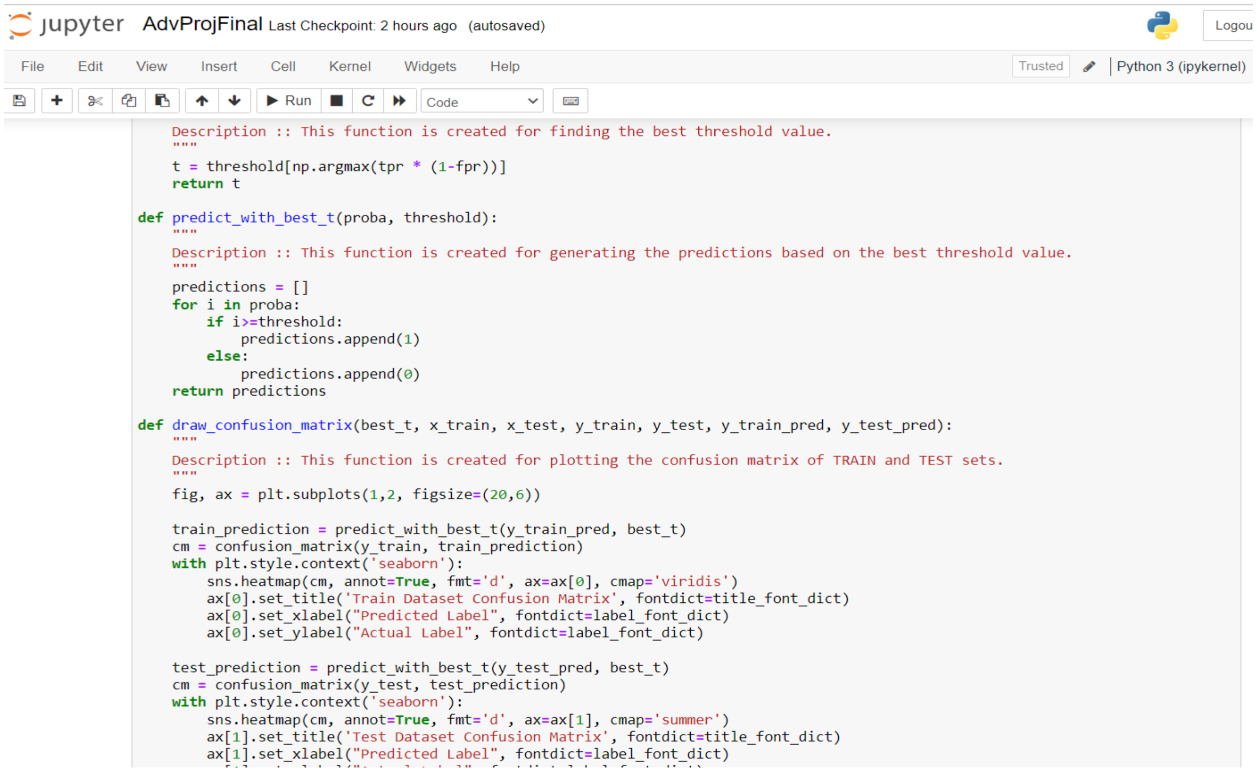



jupyter AdvProjFinal Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Notebook saved Trusted Python 3 (ipykernel)
Run Code

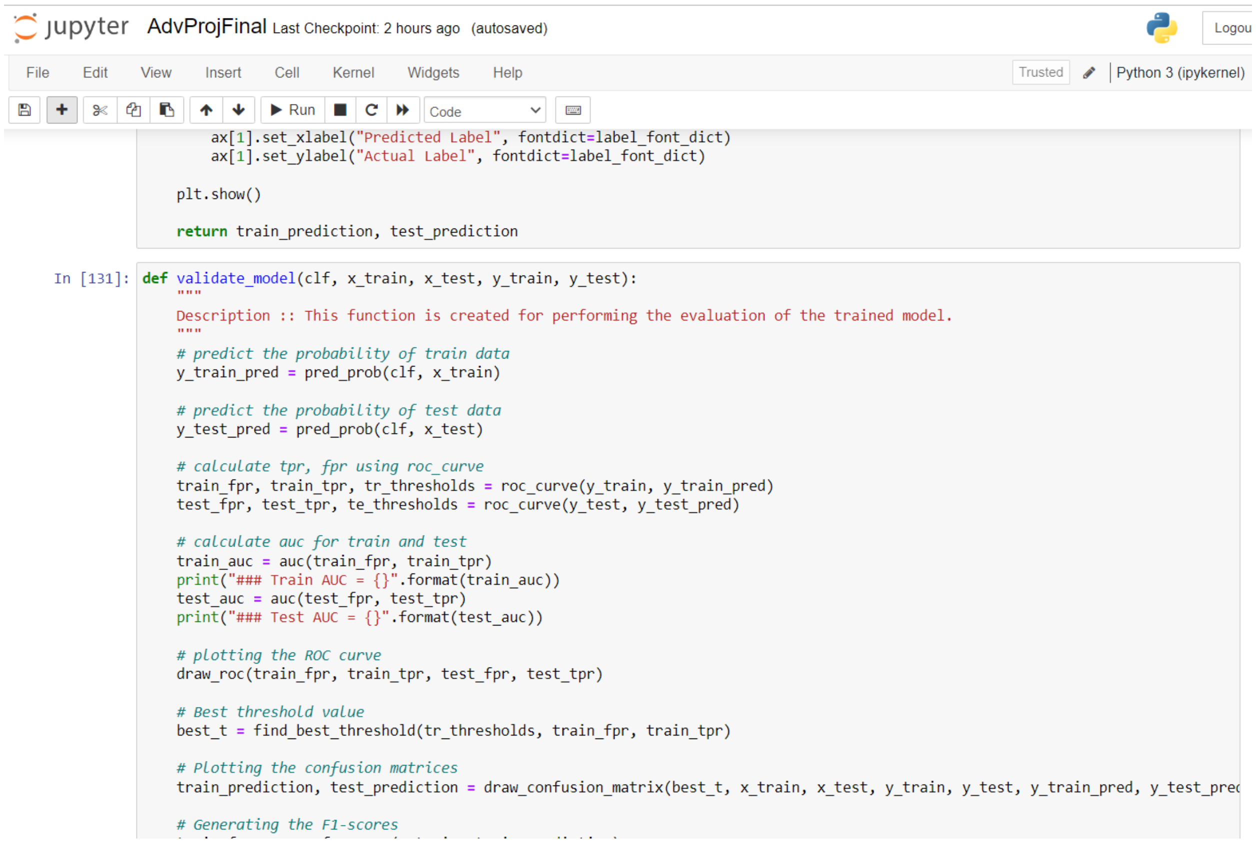
```
In [126]: from sklearn.model_selection import train_test_split as tts
          X_train, X_test, y_train, y_test = tts(X, y, test_size=0.20, stratify=y, random_state=39)
          X_train.shape, X_test.shape, y_train.shape, y_test.shape

Out[126]: ((4328, 30), (1082, 30), (4328,), (1082,))

In [127]: from sklearn.preprocessing import RobustScaler
          # Standardize the data (train and test)
          robust_scaler = RobustScaler()
          robust_scaler.fit(X_train)
          X_train_std = robust_scaler.transform(X_train)
          X_test_std = robust_scaler.transform(X_test)

In [128]: from sklearn.linear_model import LogisticRegressionCV
          from sklearn import metrics
          from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, roc_curve,
          from sklearn.linear_model import LogisticRegression
          from sklearn.model_selection import GridSearchCV
          from sklearn.model_selection import RandomizedSearchCV
          from sklearn.calibration import CalibratedClassifierCV

In [129]: # Training the model with all features and hyper-parameterized values
          log_reg_1 = LogisticRegression(C=0.0316228, penalty='l1',
                                         fit_intercept=True, solver='liblinear', tol=0.0001, max_iter=500,
                                         class_weight='balanced',
                                         verbose=0,
                                         intercept_scaling=1.0,
                                         multi_class='auto',
                                         random_state=49)

          log_reg_1.fit(X_train_std, y_train)

Out[129]: LogisticRegression(C=0.0316228, class_weight='balanced', intercept_scaling=1.0,
                             max_iter=500, penalty='l1', random_state=49,
                             solver='liblinear')
```


jupyter AdvProjFinal Last Checkpoint: 2 hours ago (autosaved)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code

```
In [130]: def pred_prob(clf, data):
              """
              Description :: This function is created for storing the predicted probabability using the trained model.

              Input :: It accepts below input parameters :
                - clf : Trained model classifier
                - data : Dataset for which we want to generate the predictions
              """
              y_pred = clf.predict_proba(data)[:,1]
              return y_pred

          def draw_roc(train_fpr, train_tpr, test_fpr, test_tpr):
              """
              Description :: This function is created for calculating the AUC score on train and test data. And, plotting the ROC curve.

              Input :: It accepts below input parameters :
                - train_fpr : Train False +ve rate
                - train_tpr : Train True +ve rate
                - test_fpr : Test False +ve rate
                - test_tpr : Test True +ve rate
              """
              # calculate auc for train and test
              train_auc = auc(train_fpr, train_tpr)
              test_auc = auc(test_fpr, test_tpr)
              with plt.style.context('seaborn-poster'):
                plt.plot(train_fpr, train_tpr, label="Train AUC ="+"{:.4f}".format(train_auc), color='blue')
                plt.plot(test_fpr, test_tpr, label="Test AUC ="+"{:.4f}".format(test_auc), color='red')
                plt.legend()
                plt.xlabel("False Positive Rate(FPR)", fontdict=label_font_dict)
                plt.ylabel("True Positive Rate(TPR)", fontdict=label_font_dict)
                plt.title("Area Under Curve", fontdict=title_font_dict)
                plt.grid(b=True, which='major', color='lightgrey', linestyle='--')
                plt.minorticks_on()
                plt.show()

          def find_best_threshold(threshold, fpr, tpr):
```

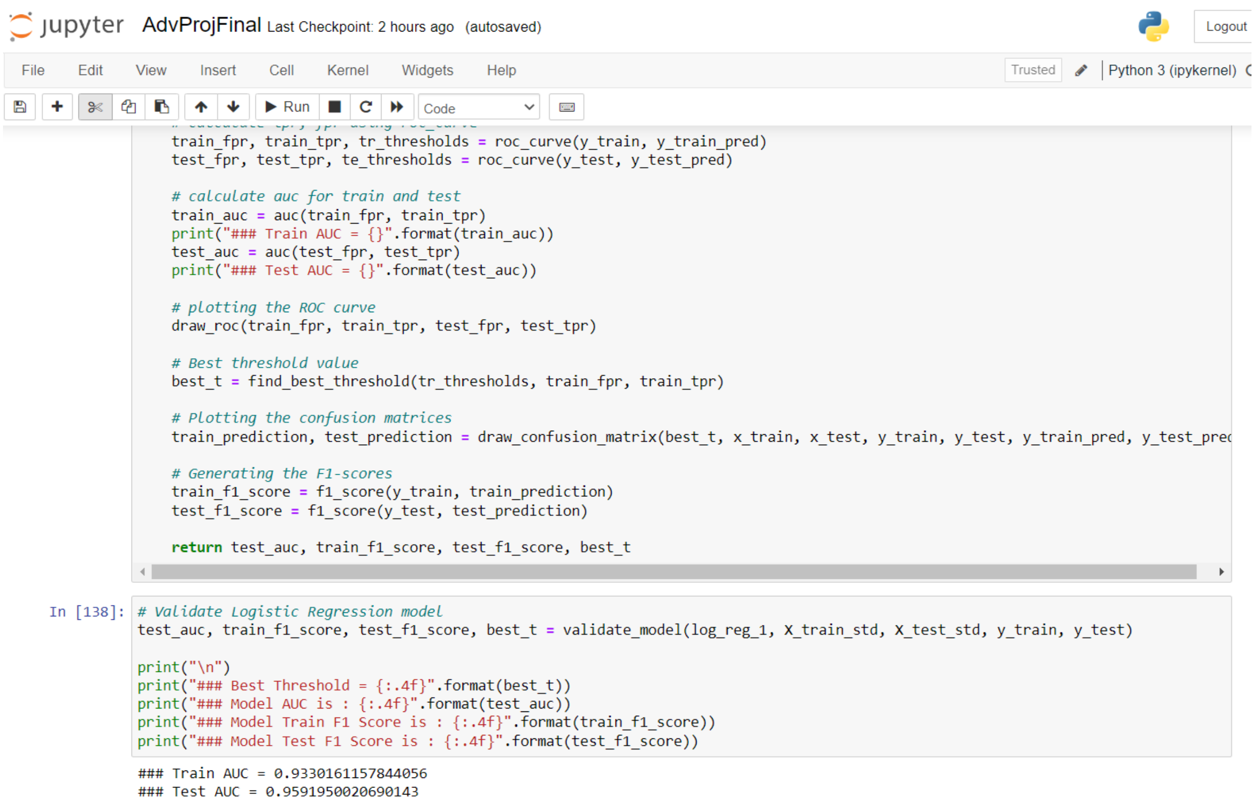

jupyter AdvProjFinal Last Checkpoint: 2 hours ago (autosaved)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
    Description :: This function is created for finding the best threshold value.
    """
    t = threshold[np.argmax(tpr * (1-fpr))]
    return t

def predict_with_best_t(proba, threshold):
    """
    Description :: This function is created for generating the predictions based on the best threshold value.
    """
    predictions = []
    for i in proba:
        if i>=threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions

def draw_confusion_matrix(best_t, x_train, x_test, y_train, y_test, y_train_pred, y_test_pred):
    """
    Description :: This function is created for plotting the confusion matrix of TRAIN and TEST sets.
    """
    fig, ax = plt.subplots(1,2, figsize=(20,6))

    train_prediction = predict_with_best_t(y_train_pred, best_t)
    cm = confusion_matrix(y_train, train_prediction)
    with plt.style.context('seaborn'):
        sns.heatmap(cm, annot=True, fmt='d', ax=ax[0], cmap='viridis')
        ax[0].set_title('Train Dataset Confusion Matrix', fontdict=title_font_dict)
        ax[0].set_xlabel("Predicted Label", fontdict=label_font_dict)
        ax[0].set_ylabel("Actual Label", fontdict=label_font_dict)

    test_prediction = predict_with_best_t(y_test_pred, best_t)
    cm = confusion_matrix(y_test, test_prediction)
    with plt.style.context('seaborn'):
        sns.heatmap(cm, annot=True, fmt='d', ax=ax[1], cmap='summer')
        ax[1].set_title('Test Dataset Confusion Matrix', fontdict=title_font_dict)
        ax[1].set_xlabel("Predicted Label", fontdict=label_font_dict)
```


jupyter AdvProjFinal Last Checkpoint: 2 hours ago (autosaved)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
        ax[1].set_xlabel("Predicted Label", fontdict=label_font_dict)
        ax[1].set_ylabel("Actual Label", fontdict=label_font_dict)

    plt.show()

    return train_prediction, test_prediction
```

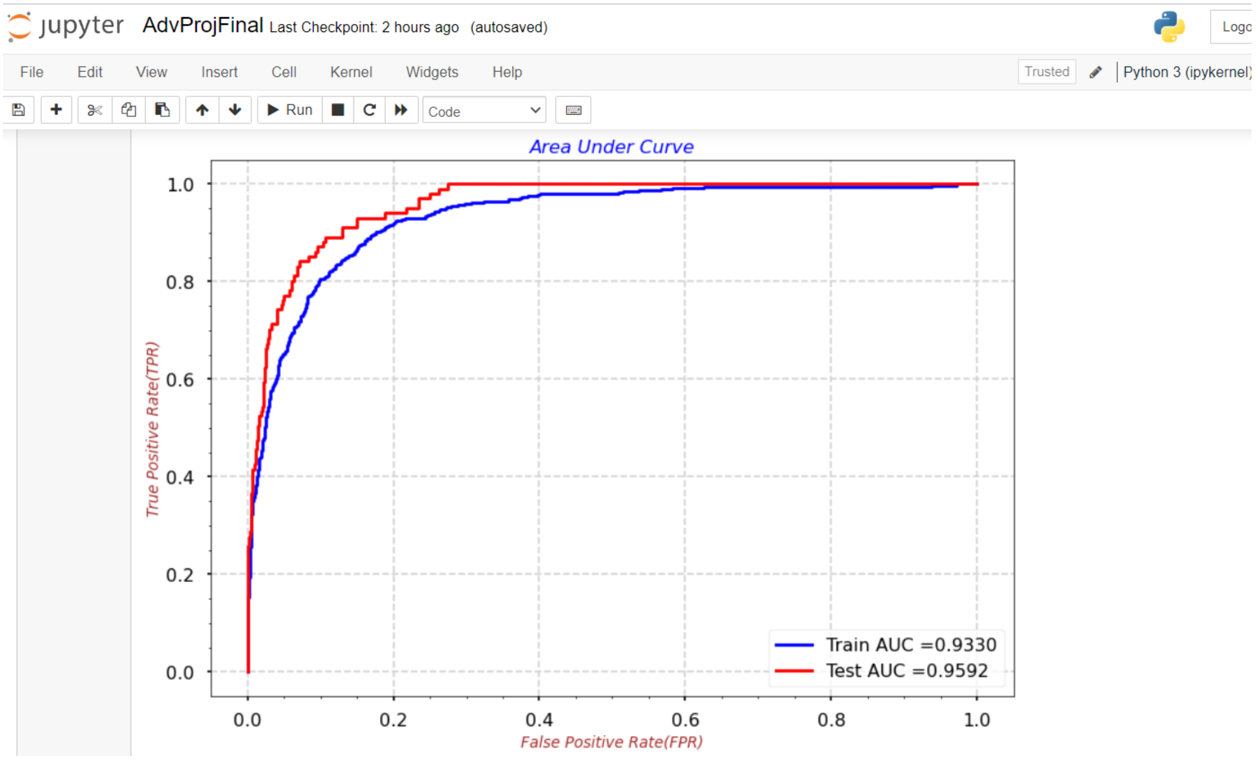
```
In [131]: def validate_model(clf, x_train, x_test, y_train, y_test):
              """
              Description :: This function is created for performing the evaluation of the trained model.
              """
              # predict the probability of train data
              y_train_pred = pred_prob(clf, x_train)

              # predict the probability of test data
              y_test_pred = pred_prob(clf, x_test)

              # calculate tpr, fpr using roc_curve
              train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
              test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

              # calculate auc for train and test
              train_auc = auc(train_fpr, train_tpr)
              print("### Train AUC = {}".format(train_auc))
              test_auc = auc(test_fpr, test_tpr)
              print("### Test AUC = {}".format(test_auc))

              # plotting the ROC curve
              draw_roc(train_fpr, train_tpr, test_fpr, test_tpr)

              # Best threshold value
              best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)

              # Plotting the confusion matrices
              train_prediction, test_prediction = draw_confusion_matrix(best_t, x_train, x_test, y_train, y_test, y_train_pred, y_test_pred

              # Generating the F1-scores
```

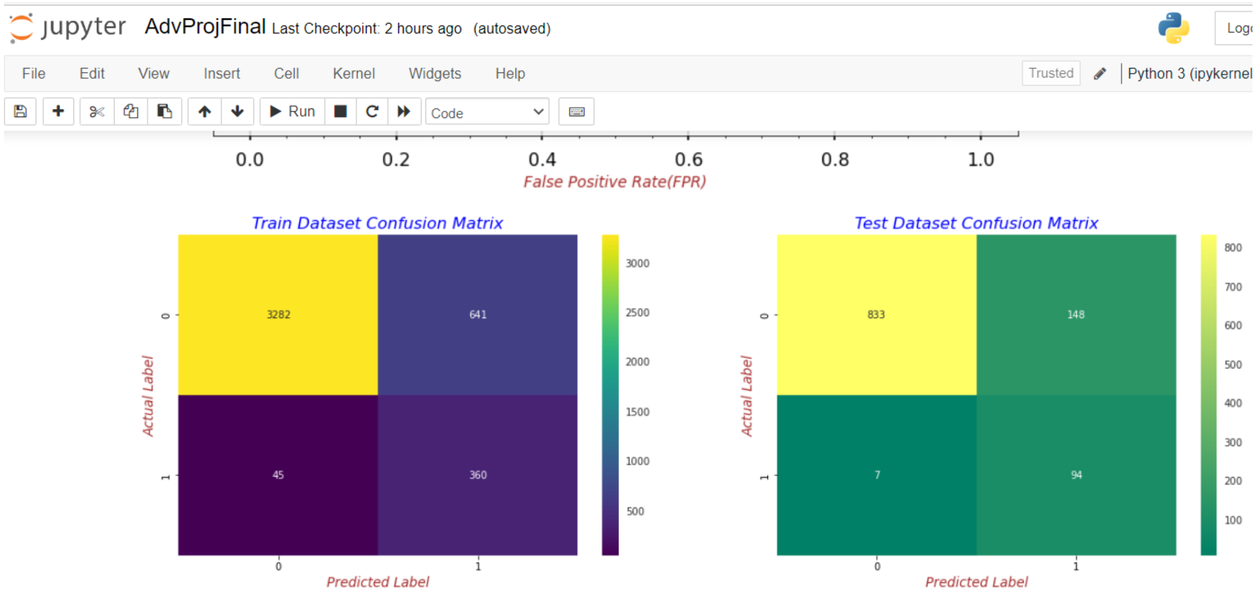

jupyter AdvProjFinal Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
    # calculate tpr, fpr using roc_curve
    train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
    test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

    # calculate auc for train and test
    train_auc = auc(train_fpr, train_tpr)
    print("### Train AUC = {}".format(train_auc))
    test_auc = auc(test_fpr, test_tpr)
    print("### Test AUC = {}".format(test_auc))

    # plotting the ROC curve
    draw_roc(train_fpr, train_tpr, test_fpr, test_tpr)

    # Best threshold value
    best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)

    # Plotting the confusion matrices
    train_prediction, test_prediction = draw_confusion_matrix(best_t, x_train, x_test, y_train, y_test, y_train_pred, y_test_pre

    # Generating the F1-scores
    train_f1_score = f1_score(y_train, train_prediction)
    test_f1_score = f1_score(y_test, test_prediction)

    return test_auc, train_f1_score, test_f1_score, best_t
```

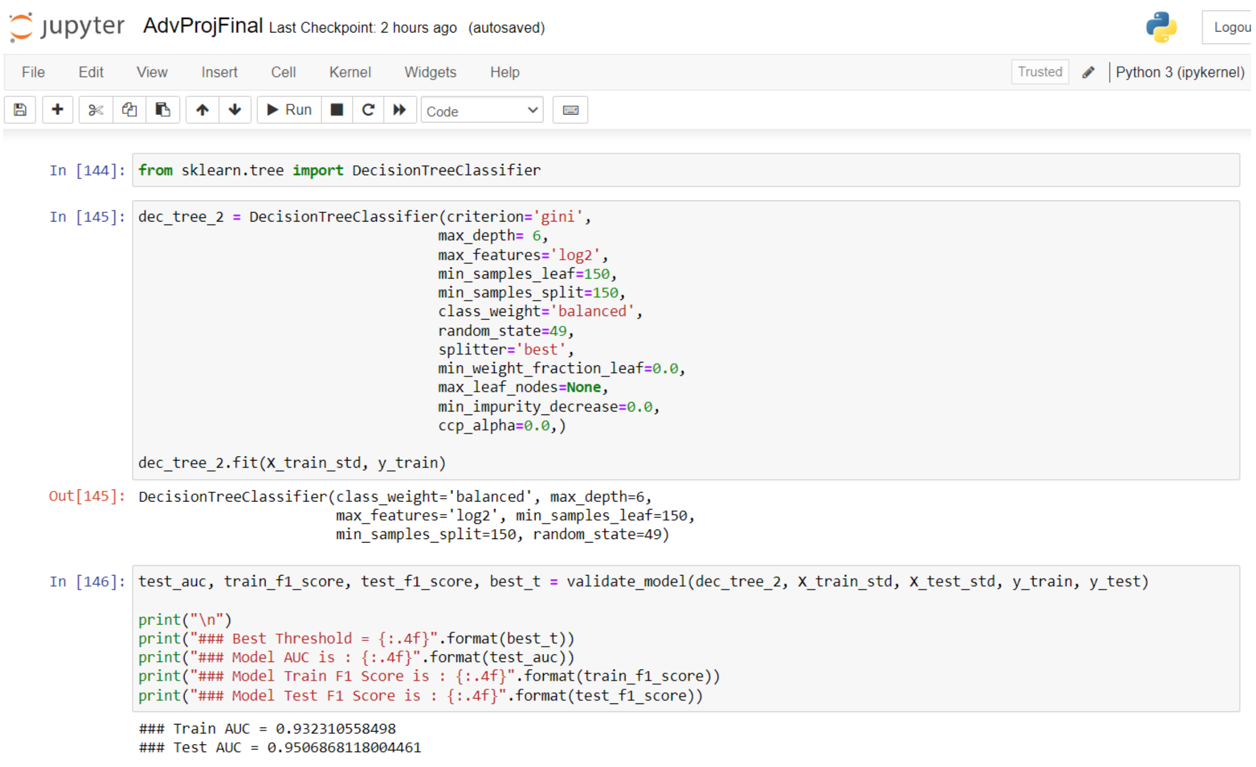
```
In [138]: # Validate Logistic Regression model
          test_auc, train_f1_score, test_f1_score, best_t = validate_model(log_reg_1, X_train_std, X_test_std, y_train, y_test)

          print("\n")
          print("### Best Threshold = {:.4f}".format(best_t))
          print("### Model AUC is : {:.4f}".format(test_auc))
          print("### Model Train F1 Score is : {:.4f}".format(train_f1_score))
          print("### Model Test F1 Score is : {:.4f}".format(test_f1_score))
```

```
### Train AUC = 0.9330161157844056
### Test AUC = 0.9591950020690143
```

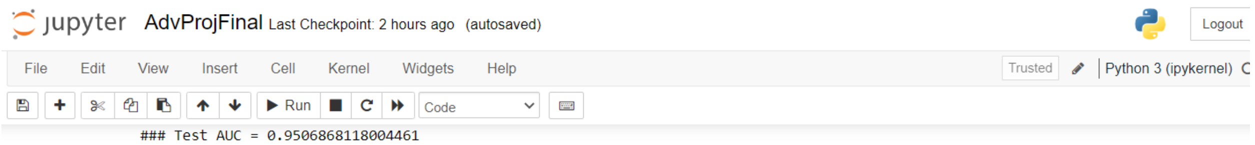




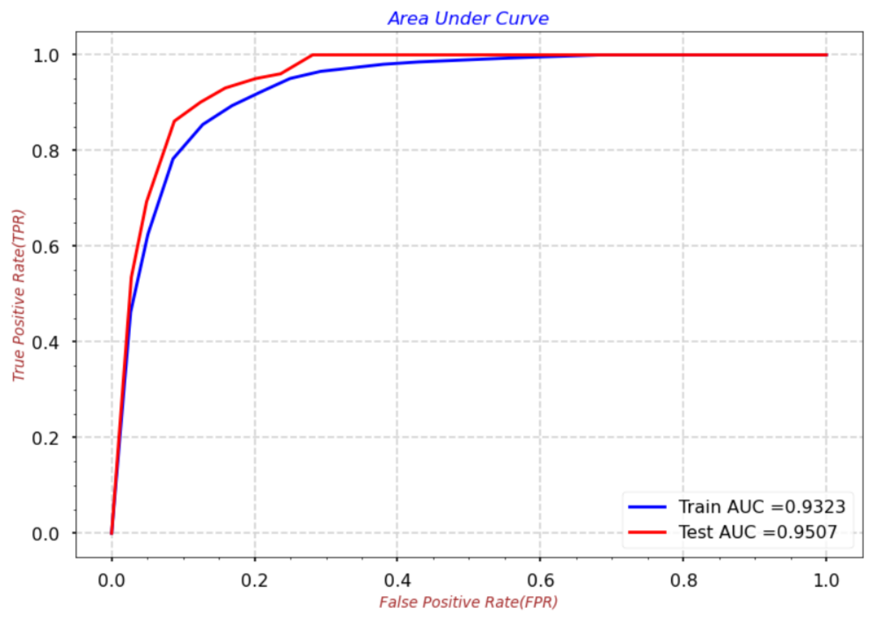
```
### Best Threshold = 0.3468
### Model AUC is : 0.9592
### Model Train F1 Score is : 0.5121
### Model Test F1 Score is : 0.5481
```

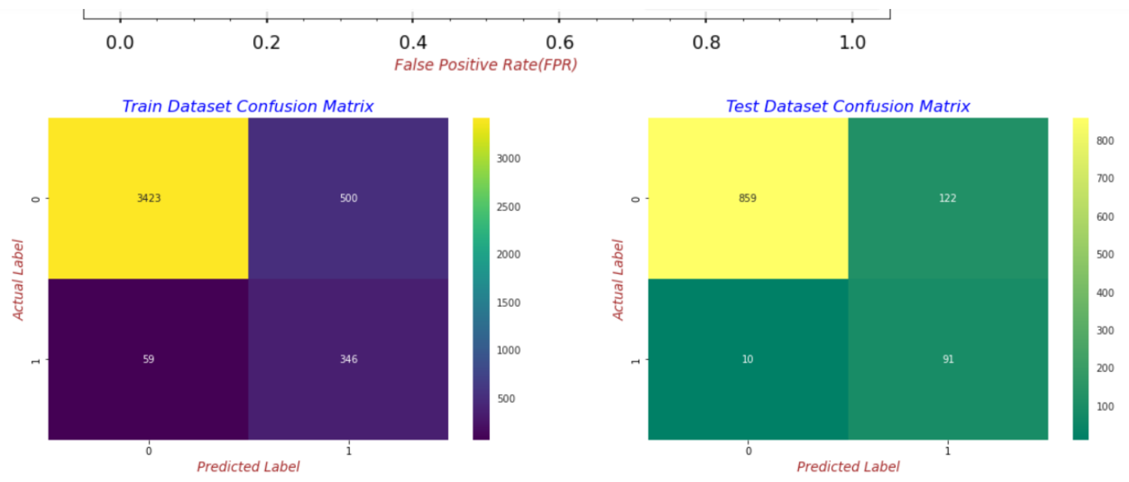

jupyter AdvProjFinal Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [144]: from sklearn.tree import DecisionTreeClassifier

In [145]: dec_tree_2 = DecisionTreeClassifier(criterion='gini',
                                    max_depth= 6,
                                    max_features='log2',
                                    min_samples_leaf=150,
                                    min_samples_split=150,
                                    class_weight='balanced',
                                    random_state=49,
                                    splitter='best',
                                    min_weight_fraction_leaf=0.0,
                                    max_leaf_nodes=None,
                                    min_impurity_decrease=0.0,
                                    ccp_alpha=0.0,)

          dec_tree_2.fit(X_train_std, y_train)

Out[145]: DecisionTreeClassifier(class_weight='balanced', max_depth=6,
                       max_features='log2', min_samples_leaf=150,
                       min_samples_split=150, random_state=49)

In [146]: test_auc, train_f1_score, test_f1_score, best_t = validate_model(dec_tree_2, X_train_std, X_test_std, y_train, y_test)

          print("\n")
          print("### Best Threshold = {:.4f}".format(best_t))
          print("### Model AUC is : {:.4f}".format(test_auc))
          print("### Model Train F1 Score is : {:.4f}".format(train_f1_score))
          print("### Model Test F1 Score is : {:.4f}".format(test_f1_score))

          ### Train AUC = 0.932310558498
          ### Test AUC = 0.9506868118004461
```

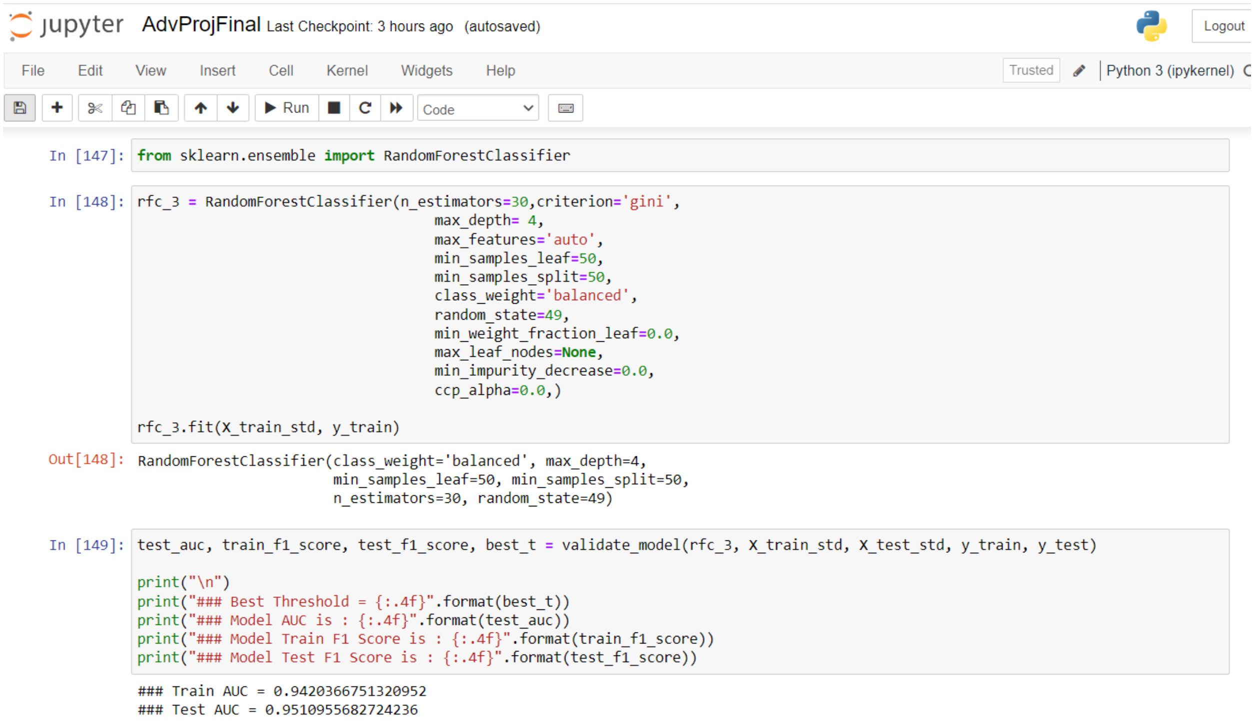


```
### Test AUC = 0.9506868118004461
```

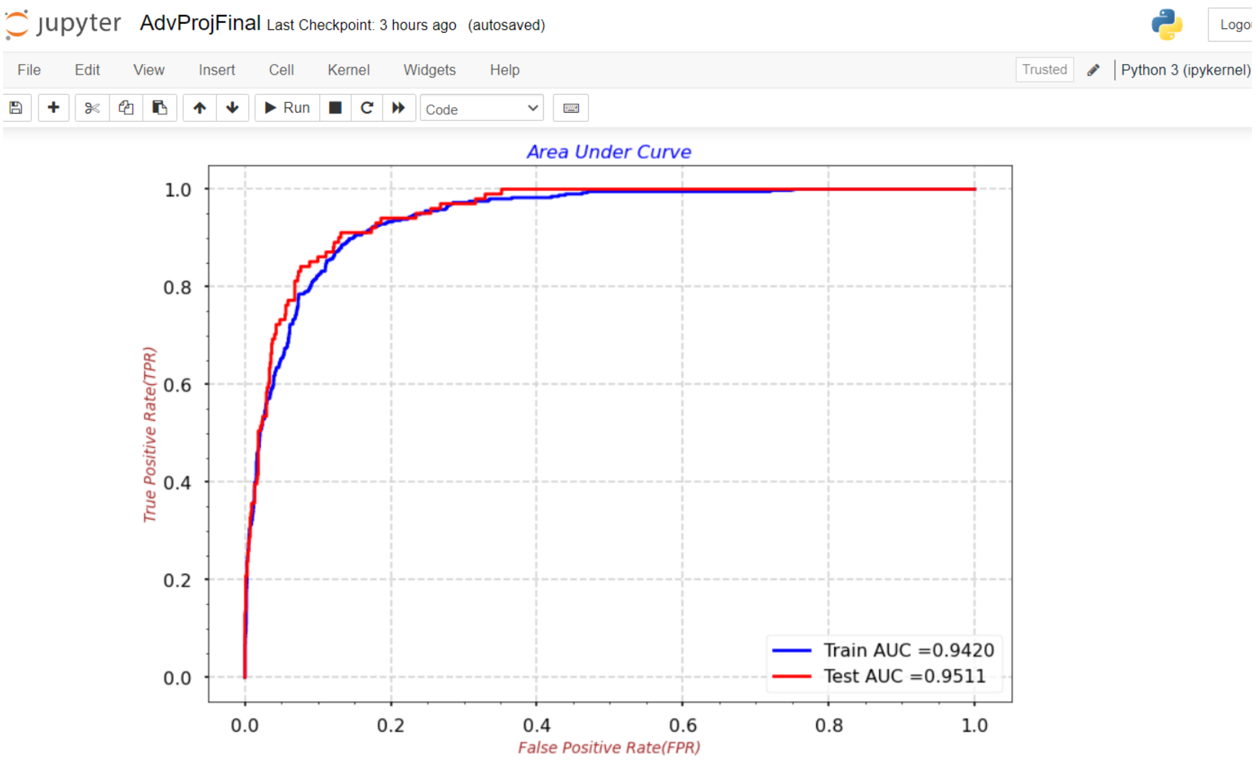


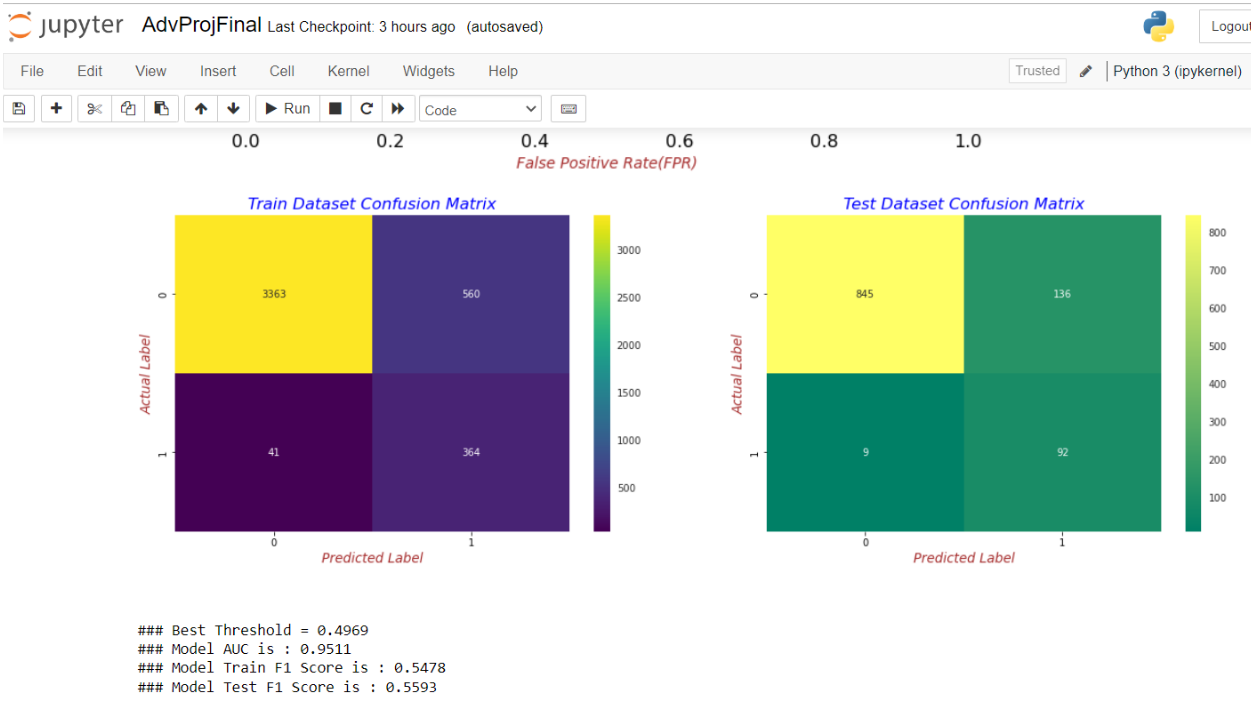


```
### Best Threshold = 0.6328
### Model AUC is : 0.9507
### Model Train F1 Score is : 0.5532
### Model Test F1 Score is : 0.5796
```


jupyter AdvProjFinal Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [147]: from sklearn.ensemble import RandomForestClassifier

In [148]: rfc_3 = RandomForestClassifier(n_estimators=30,criterion='gini',
                                         max_depth= 4,
                                         max_features='auto',
                                         min_samples_leaf=50,
                                         min_samples_split=50,
                                         class_weight='balanced',
                                         random_state=49,
                                         min_weight_fraction_leaf=0.0,
                                         max_leaf_nodes=None,
                                         min_impurity_decrease=0.0,
                                         ccp_alpha=0.0,)

          rfc_3.fit(X_train_std, y_train)

Out[148]: RandomForestClassifier(class_weight='balanced', max_depth=4,
                                 min_samples_leaf=50, min_samples_split=50,
                                 n_estimators=30, random_state=49)

In [149]: test_auc, train_f1_score, test_f1_score, best_t = validate_model(rfc_3, X_train_std, X_test_std, y_train, y_test)

          print("\n")
          print("### Best Threshold = {:.4f}".format(best_t))
          print("### Model AUC is : {:.4f}".format(test_auc))
          print("### Model Train F1 Score is : {:.4f}".format(train_f1_score))
          print("### Model Test F1 Score is : {:.4f}".format(test_f1_score))

          ### Train AUC = 0.9420366751320952
          ### Test AUC = 0.9510955682724236
```



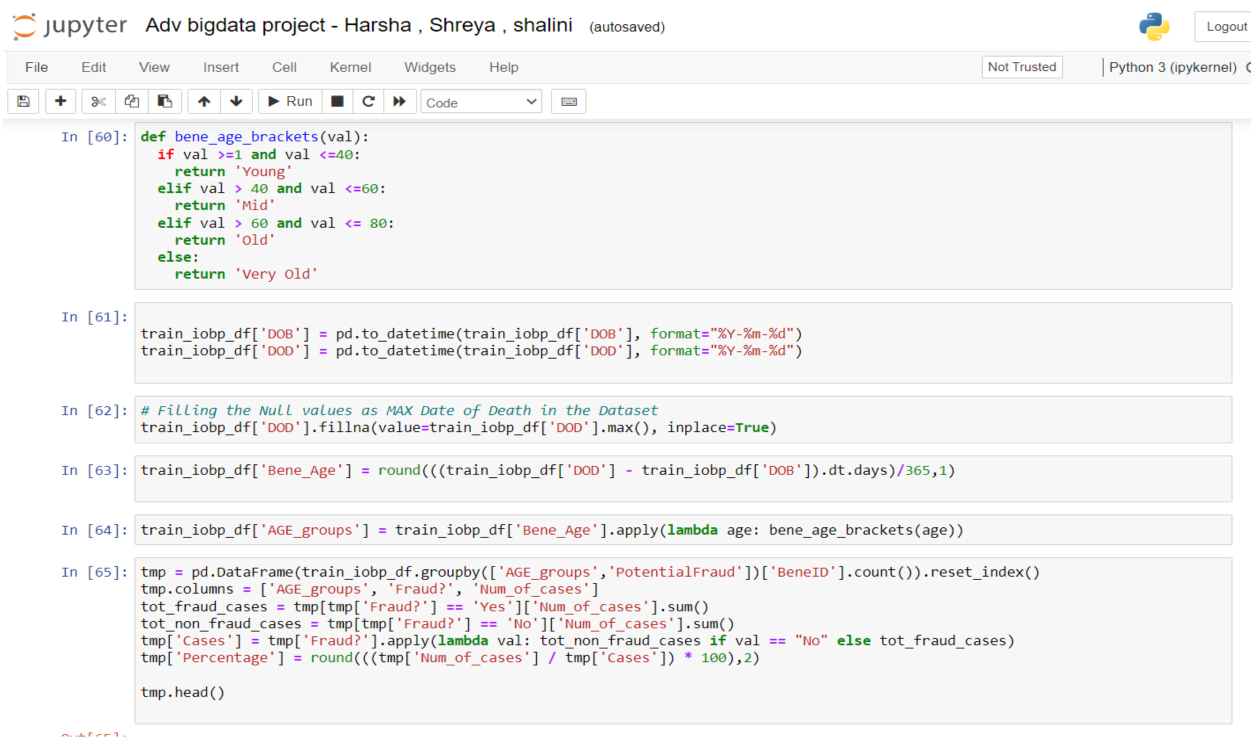


```
### Best Threshold = 0.4969
### Model AUC is : 0.9511
### Model Train F1 Score is : 0.5478
### Model Test F1 Score is : 0.5593
```

4. Which age group has committed the most number of fraudulent cases?

Here, we identified the most number of fraudulent cases as per the age group. Below is the process of how the work was done using our dataset in Python. We have grouped ages with 1-40 years as ‘Young’ , 40-60 years age as ‘Mid’, 60- 80 years age as ‘Old’ and above 80 years as ‘Very Old’ .

RESULTS:


jupyter
Adv bigdata project - Harsha , Shreya , shalini (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

In [60]:
```
def bene_age_brackets(val):
    if val >=1 and val <=40:
        return 'Young'
    elif val > 40 and val <=60:
        return 'Mid'
    elif val > 60 and val <= 80:
        return 'Old'
    else:
        return 'Very Old'
```

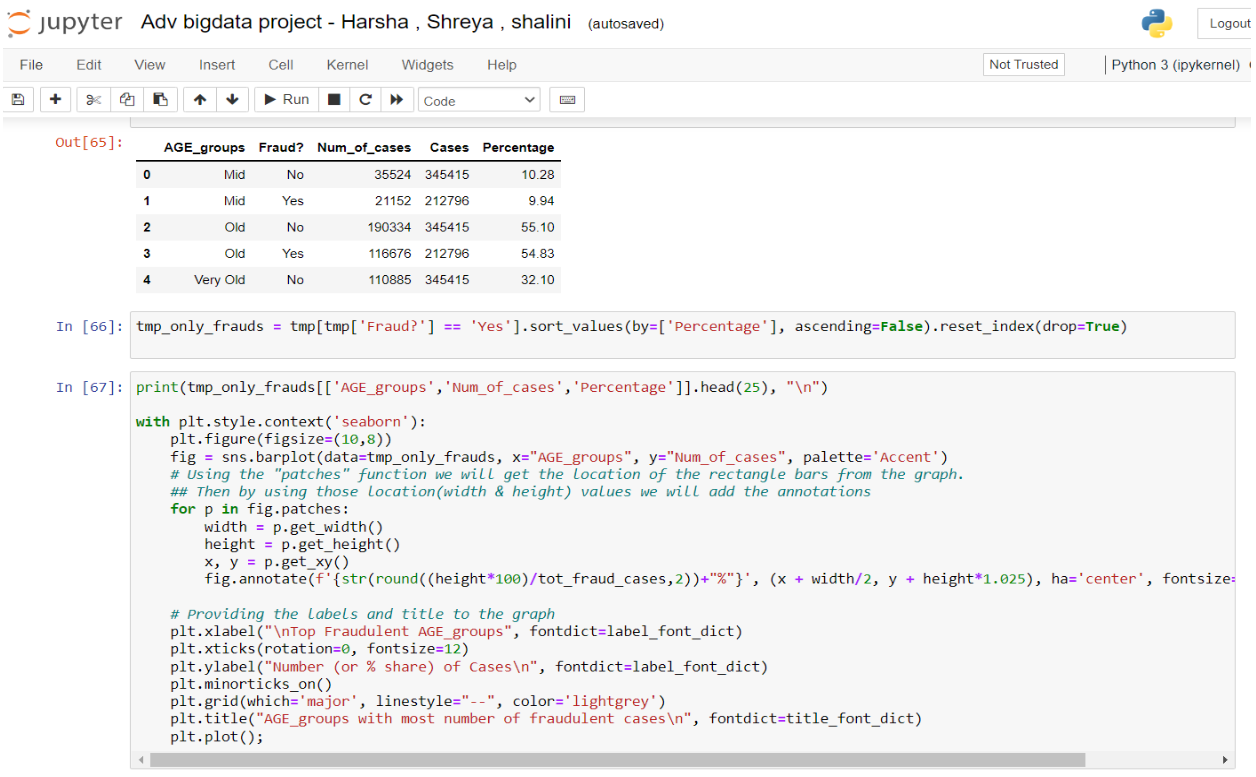
In [61]:
```
train_iobp_df['DOB'] = pd.to_datetime(train_iobp_df['DOB'], format="%Y-%m-%d")
train_iobp_df['DOD'] = pd.to_datetime(train_iobp_df['DOD'], format="%Y-%m-%d")
```

In [62]:
```
# Filling the Null values as MAX Date of Death in the Dataset
train_iobp_df['DOD'].fillna(value=train_iobp_df['DOD'].max(), inplace=True)
```

In [63]:
```
train_iobp_df['Bene_Age'] = round(((train_iobp_df['DOD'] - train_iobp_df['DOB']).dt.days)/365,1)
```

In [64]:
```
train_iobp_df['AGE_groups'] = train_iobp_df['Bene_Age'].apply(lambda age: bene_age_brackets(age))
```

In [65]:
```
tmp = pd.DataFrame(train_iobp_df.groupby(['AGE_groups','PotentialFraud'])['BeneID'].count()).reset_index()
tmp.columns = ['AGE_groups', 'Fraud?', 'Num_of_cases']
tot_fraud_cases = tmp[tmp['Fraud?'] == 'Yes']['Num_of_cases'].sum()
tot_non_fraud_cases = tmp[tmp['Fraud?'] == 'No']['Num_of_cases'].sum()
tmp['Cases'] = tmp['Fraud?'].apply(lambda val: tot_non_fraud_cases if val == "No" else tot_fraud_cases)
tmp['Percentage'] = round(((tmp['Num_of_cases'] / tmp['Cases']) * 100),2)

tmp.head()
```

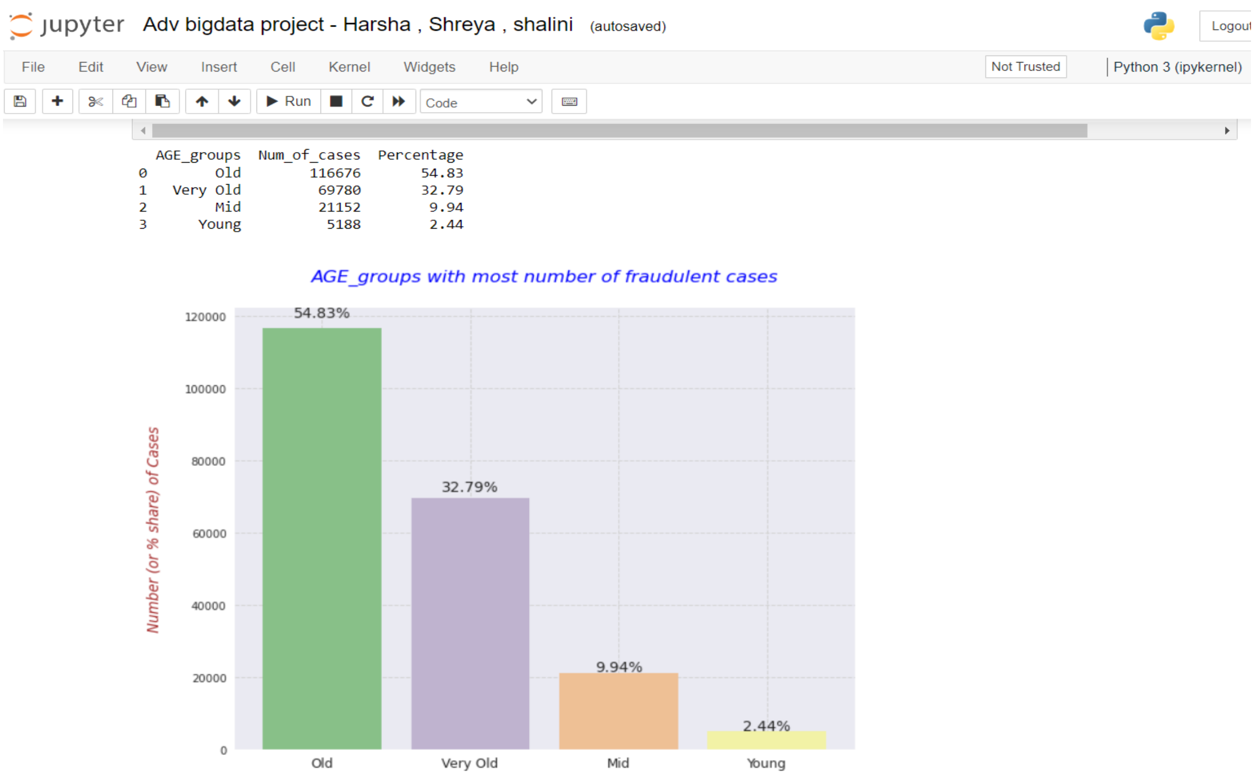

jupyter
Adv bigdata project - Harsha , Shreya , shalini (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

Out[65]:

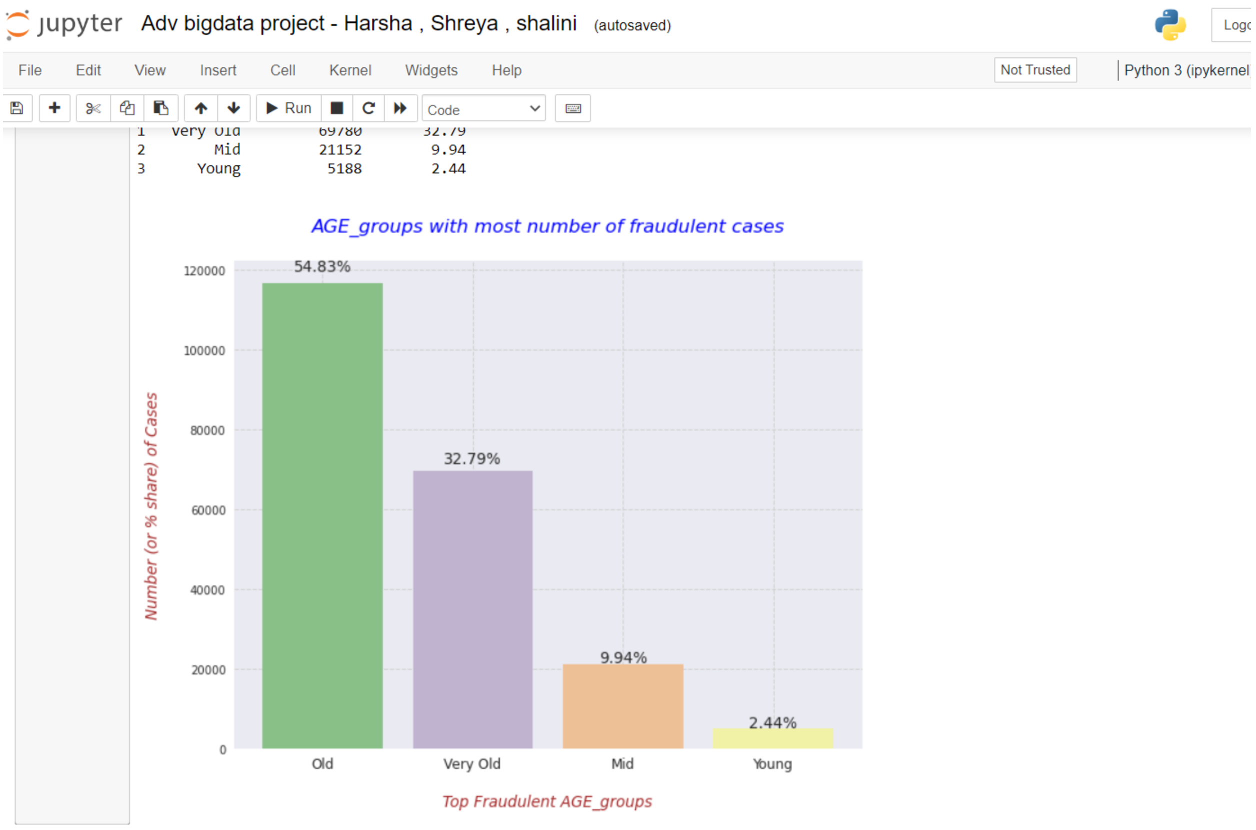
	AGE_groups	Fraud?	Num_of_cases	Cases	Percentage
0	Mid	No	35524	345415	10.28
1	Mid	Yes	21152	212796	9.94
2	Old	No	190334	345415	55.10
3	Old	Yes	116676	212796	54.83
4	Very Old	No	110885	345415	32.10

In [66]:
```
tmp_only_frauds = tmp[tmp['Fraud?'] == 'Yes'].sort_values(by=['Percentage'], ascending=False).reset_index(drop=True)
```

In [67]:
```
print(tmp_only_frauds[['AGE_groups','Num_of_cases','Percentage']].head(25), "\n")

with plt.style.context('seaborn'):
    plt.figure(figsize=(10,8))
    fig = sns.barplot(data=tmp_only_frauds, x="AGE_groups", y="Num_of_cases", palette='Accent')
    # Using the "patches" function we will get the location of the rectangle bars from the graph.
    ## Then by using those location(width & height) values we will add the annotations
    for p in fig.patches:
        width = p.get_width()
        height = p.get_height()
        x, y = p.get_xy()
        fig.annotate(f'{str(round((height*100)/tot_fraud_cases,2))+"%"}', (x + width/2, y + height*1.025), ha='center', fontsize=

    # Providing the labels and title to the graph
    plt.xlabel("\nTop Fraudulent AGE_groups", fontdict=label_font_dict)
    plt.xticks(rotation=0, fontsize=12)
    plt.ylabel("Number (or % share) of Cases\n", fontdict=label_font_dict)
    plt.minorticks_on()
    plt.grid(which='major', linestyle="--", color='lightgrey')
    plt.title("AGE_groups with most number of fraudulent cases\n", fontdict=title_font_dict)
    plt.plot();
```



```
  AGE_groups  Num_of_cases  Percentage
0        Old        116676       54.83
1   Very Old         69780       32.79
2        Mid         21152        9.94
3      Young          5188        2.44
```

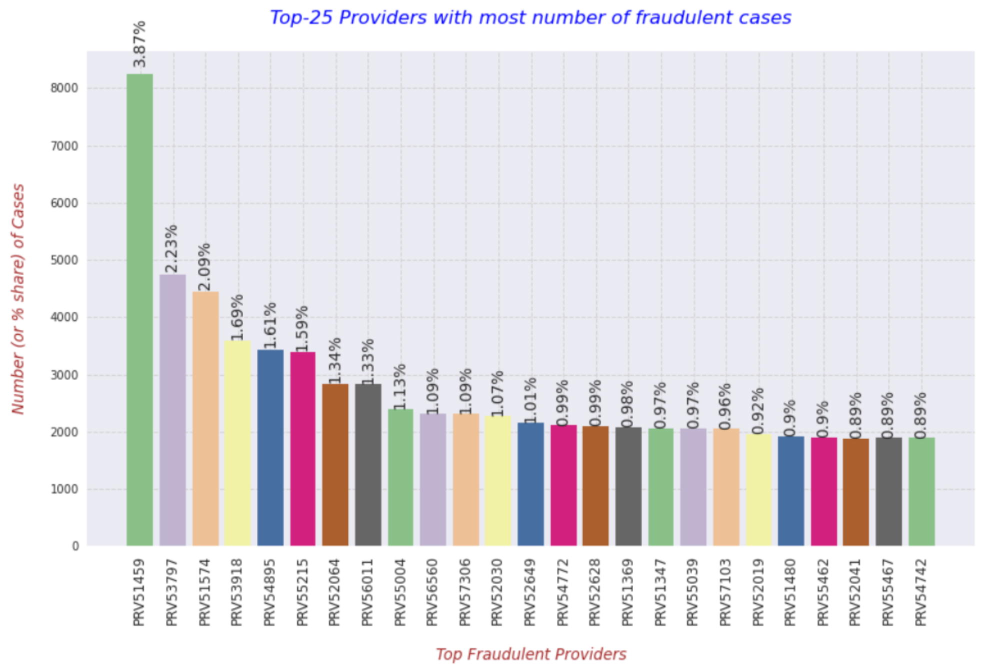

jupyter
Adv bigdata project - Harsha , Shreya , shalini (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel
Run
Code
AGE_groups with most number of fraudulent cases
Number (or % share) of Cases
120000
100000
80000
60000
40000
20000
0
54.83%
32.79%
9.94%
2.44%
Old
Very Old
Mid
Young
Top Fraudulent AGE_groups

```
1   Very Old         69780       32.79
2        Mid         21152        9.94
3      Young          5188        2.44
```

In reference to the above visualization, we can infer that Old people between the age group 60 to 80 tend to commit the most fraud.

5. Identify the top providers with fraudulent cases.

We found the top provider with the most fraudulent cases so we can be aware of the provider who has the most number of cases filed. We created columns that stated the number of columns, provider details and the percentage. Here, we concluded that the most number of fraud cases is committed by PRV51459.



SUMMARY

Building a cutting-edge data science model that helps in fraud detection in the medicare industry using real-time analysis and classification algorithms. The government can employ this technology to assist patients, pharmacies, and doctors, which will eventually aid in establishing credibility for the business, battle rising healthcare expenses, and manage the effects of fraud.A major problem that leads to significant financial suffering in the Medicare/Medicaid and protection companies is the misrepresenting of medical services. So the main objectives of our project are to create a simple data model that highlights the relationships between the various datasets and identifies the essential traits for extortion recognitions and build a sophisticated AI model that can identify misrepresentation-based designs based on different highlights and make the transactions transparent.

CONCLUSION

1. Adding Aggregated features at Provider, Beneficiary, Attending physician, Operating physician all these levels certainly helped in achieving the good performance scores.
2. Specifically for the age distribution over the beneficiaries data having to check with the span, overall the percentage of number of beneficiaries with YEAR of birth from 1939 to 1943 is the highest. Whereas, it is lowest from 1978 to 1983.
3. The reasoning behind checking to see whether FRAUD claims are higher for specific AGE Groups is that there may be a potential pattern that providers file a higher number of FRAUD claims for either a younger age group or very old age group.
4. Next point to understand is that there can be ONE to MANY relationship b/w a BENEFICIARY and CLAIM filed, thus I'm making an assumption here is that beneficiaries with DOD as NA are alive, however this cannot be always true once we join it with CLAIMS data.
5. Out of the ML Models Performed, the best-resulting model is Random Forest with an AUC of 72 %.

REFERENCES

1. https://www23.statcan.gc.ca/imdb/p3VD.pl?Function=getVD&TVD=53971
2. Part D Prescriber Data CY 2017. (n.d.). Retrieved June 23, 2020, from https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/PartD2017
3. LEIE Downloadable Databases: Office of Inspector General: U.S. Department of Health and Human Services. (2020, June 10). Retrieved June 23, 2020, from https://oig.hhs.gov/exclusions/exclusions_list.asp
4. Dataset Downloads. (n.d.). Retrieved June 23, 2020, from https://www.cms.gov/OpenPayments/Explore-the-Data/Dataset-Downloads
5. *Scikit Learn - Boosting Methods*. (n.d.). Retrieved October 15, 2022, from https://www.tutorialspoint.com/scikit_learn/scikit_learn_boosting_methods.htm
6. Navlani, A. (n.d.). *Understanding Random Forests Classifiers in Python Tutorial*. www.datacamp.com. Retrieved October 15, 2022, from https://www.datacamp.com/tutorial/random-forests-classifier-python