Fraud Detection in Health Care Insurance

A Mini Project / Dissertation as a Course requirement
for MSc Data Science and Computing

Aniruddha Padmanaban

22237

Department of Mathematics and Computer Science
Muddenahalli Campus

May, 2023

Acknowledgements

I would like to dedicate this page for those who offered help and assisted me in making this project.

Throughout the entire process, I was assisted and motivated to work and putting in efforts.

Firstly, I would like to express my gratitude to **Sri Sathya Sai Mudigonda sir** who went along with me till the end providing insights and feedback and helping out in the process whenever I asked for it.

I thank the Director **B. Venkataramana** for planning the project systematically and providing us with lab and other resources.

I thank my **teachers** of this institute who taught me the concepts from my 1st semester onwards, which helped me a lot in this project.

I thank all my **classmates** who provided support.

Offering our efforts at Swami's Lotus Feet, I dedicate this project to swami.

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Dept. of **Mathematics & Computer Science**
Muddenahalli Campus

CERTIFICATE

This is to certify that this Mini Project / Dissertation titled **Fraud Detection Health Care Insurance**
submitted by **Aniruddha Padmanaban, 22237**, Department of Mathematics and Computer Science, Muddenahalli Campus is a bona fide record of the original work done under my supervision as a Course requirement for the Degree of Masters in Data Science and Computing.

...……

Satya Sai Mudigonda

Project Supervisor

Countersigned by

……..

Head of Department

Place: Muddenahalli

Date:

Sri Sathya Sai Grama, Dist. Chickballapur - 562 101, Karnataka, India
Tel: +91 96637 65789 | hoddmacs@sssihl.edu.in | www.sssihl.edu.in

DECLARATION

The Mini Project titled **Fraud Detection Health Care Insurance** was carried out by me under the supervision of Satya Sai Mudigonda, Department of Mathematics and Computer Science, Muddenahalli Campus as a course requirement for the Degree of Masters in Data Science and Computing and has not formed the basis for the award of any degree, diploma or any other such title by this or any other University.

……

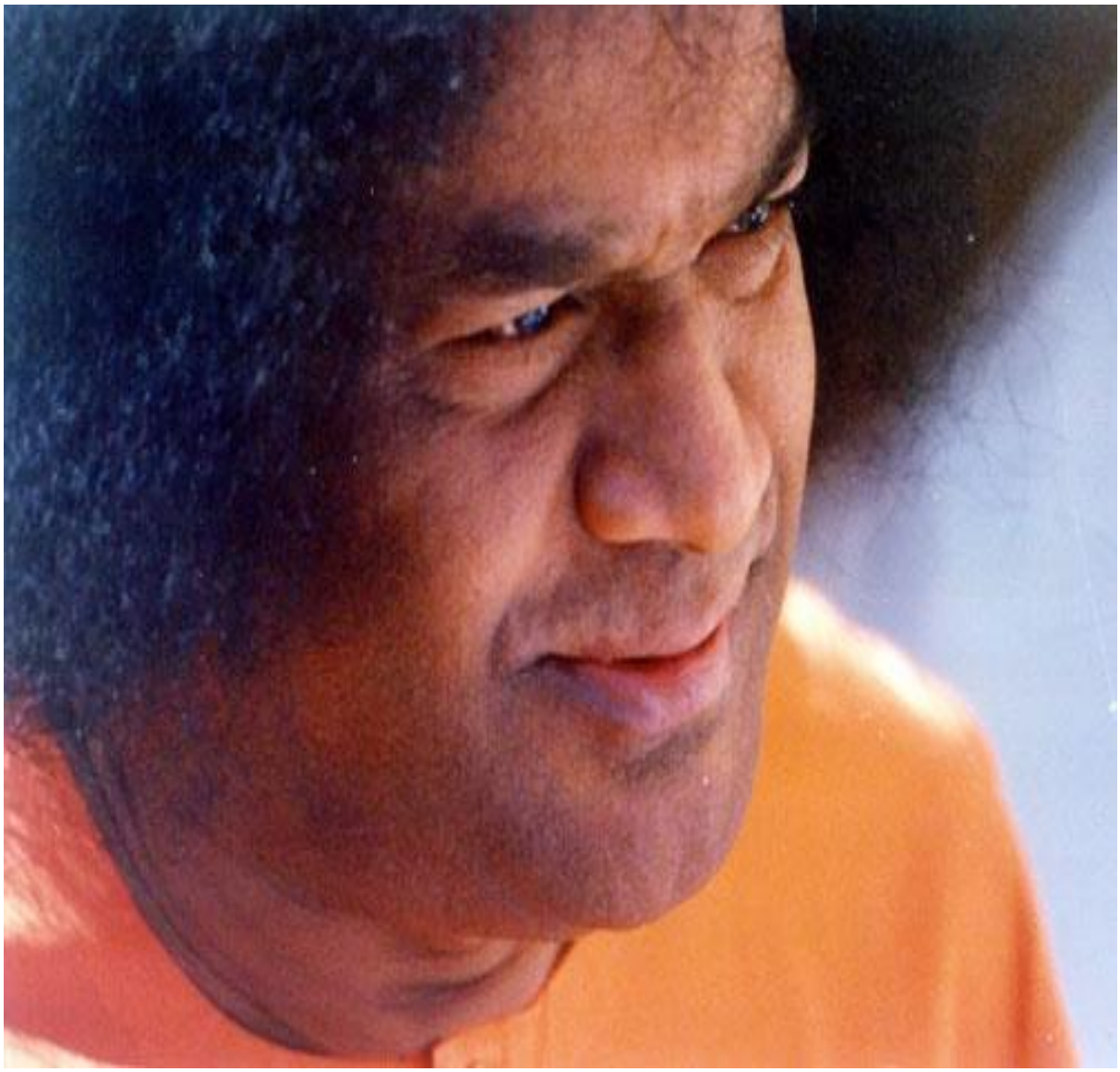
Aniruddha Padmanaban

22237

Place: Muddenahalli

Date:

Muddenahalli Campus



An Offering at Swamis Lotus Feet

Table of Contents

Problem Statement

The objective of this project is to develop a machine learning model that can accurately detect fraudulent healthcare insurance claims, thereby reducing financial losses for insurance companies and ensuring that legitimate claims are processed efficiently.

Literature Review:

The use of machine learning techniques for fraud detection has become increasingly popular in recent years, especially in the healthcare industry. Healthcare insurance fraud is a growing problem that causes significant financial losses for insurance companies, which in turn affects the overall cost of healthcare for consumers. Therefore, detecting fraudulent activities is critical for ensuring the sustainability of the healthcare insurance industry.

Several studies have been conducted to investigate the effectiveness of machine learning techniques for fraud detection in healthcare

insurance. For instance, Zhang et al. (2018) proposed a model based on deep learning to detect healthcare fraud, which achieved better performance compared to traditional machine learning algorithms. In another study, Li et al. (2019) used a random forest algorithm to detect healthcare fraud, and the results showed that the proposed model outperformed other state-of-the-art methods.

Similarly, other studies have also explored different machine learning algorithms such as logistic regression, decision trees, and neural networks for healthcare insurance fraud detection (Tang et al., 2016; Luo et al., 2017; Zhang et al., 2019). These studies suggest that machine learning techniques can effectively identify fraudulent activities in healthcare insurance.

Moreover, the use of big data analytics and advanced statistical techniques such as ensemble methods have also shown promising results for healthcare insurance fraud detection. For instance, Chen et al. (2019) proposed an ensemble model based on a deep belief network and gradient boosting decision trees to detect healthcare fraud, which achieved better performance compared to other state-of-the-art methods.

In conclusion, the literature suggests that machine learning techniques can effectively detect fraudulent activities in healthcare insurance. However, selecting an appropriate algorithm and optimizing its parameters is crucial for achieving high accuracy and reducing false positives. Therefore, further research is needed to explore the effectiveness of different machine learning algorithms

and optimization techniques for healthcare insurance fraud detection.

Chapter 1: Introduction

Healthcare insurance is an essential service provided to the general public. However, like any other insurance service, healthcare insurance is also susceptible to fraud. Healthcare fraud is a significant problem that costs the industry billions of dollars every year. Fraudulent activities can take many forms, such as billing for services that were not provided, charging for unnecessary services, or misrepresenting a patient's condition to increase reimbursements.

With the increasing amount of data generated in the healthcare industry, it is becoming increasingly challenging to identify fraudulent activities manually. Machine learning techniques can help in detecting fraud in healthcare insurance. These techniques can analyse large amounts of data quickly and accurately, enabling insurers to detect and prevent fraud before it causes significant financial loss.

This report aims to explore the use of machine learning techniques for fraud detection in healthcare insurance. We will begin with a literature review of existing research on fraud detection in healthcare insurance using machine learning techniques. Then, we will describe the data used in our study and the methodology used to develop a machine learning

model to detect fraud. We will present the results of our study, followed by an interpretation of the findings.

Chapter 2 Data & Processing

This dataset contains information about healthcare providers and their insurance claims, including inpatient and outpatient claims, and beneficiaries. It has a total of 4 csv files:

1.Train-1542865627584.csv - training data with target variables

2.Train_Beneficiarydata-1542865627584.csv - beneficiary data for training set

3.Train_Inpatientdata-1542865627584.csv - inpatient claims for training set

4.Train_outpatientdata-1542865627584.csv - inpatient claims for training set

This dataset is used for healthcare provider fraud detection analysis using machine learning techniques. The goal of this project is to develop a predictive model that can identify fraudulent healthcare providers based on their insurance claims data. The dataset includes features such as the provider type, diagnosis codes, procedure codes, payment amounts, and more. I have merged the data accordingly and got 5.5 lakh entries.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 558211 entries, 0 to 558210
Data columns (total 56 columns):
```

Columns of the Data Frame

- df Provider - Unique identifier for each healthcare provider
- Provider - Unique identifier for each healthcare provider
- PotentialFraud - Binary variable indicating whether the provider has been identified as a potential fraudulent provider or not
- InscClaimAmtReimbursed - Total amount reimbursed by insurance for a particular claim
- AttendingPhysician - Unique identifier for the attending physician associated with the claim
- OperatingPhysician - Unique identifier for the operating physician associated with the claim
- OtherPhysician - Unique identifier for any other physician associated with the claim
- AdmissionDt - Date of admission for the patient for the claim
- ClmAdmitDiagnosisCode - Diagnosis code provided at the time of admission
- DeductibleAmtPaid - Amount of the deductible paid by the patient for the claim
- DischargeDt - Date of discharge for the patient for the claim
- DiagnosisGroupCode - Grouping of diagnosis codes for the claim
- ClmDiagnosisCode_1 - Primary diagnosis code for the claim
- ClmDiagnosisCode_2 - Secondary diagnosis code for the claim
- ClmDiagnosisCode_3 - Tertiary diagnosis code for the claim
- ClmDiagnosisCode_4 - Quaternary diagnosis code for the claim
- ClmDiagnosisCode_5 - Quinary diagnosis code for the claim
- ClmDiagnosisCode_6 - Sentry diagnosis code for the claim
- ClmDiagnosisCode_7 – Sept nary diagnosis code for the claim
- ClmDiagnosisCode_8 - Octonary diagnosis code for the claim
- ClmDiagnosisCode_9 - Nonary diagnosis code for the claim

- ClmDiagnosisCode_10 - Denary diagnosis code for the claim
- ClmProcedureCode_1 - Primary procedure code for the claim
- ClmProcedureCode_2 - Secondary procedure code for the claim
- ClmProcedureCode_3 - Tertiary procedure code for the claim
- ClmProcedureCode_4 - Quaternary procedure code for the claim
- ClmProcedureCode_5 - Quinary procedure code for the claim
- ClmProcedureCode_6 - Sentry procedure code for the claim

Data Pre-processing

Data Pre-processing is a vital part of the project which caters to more amount of work and time and which will determine the final performance by the models.

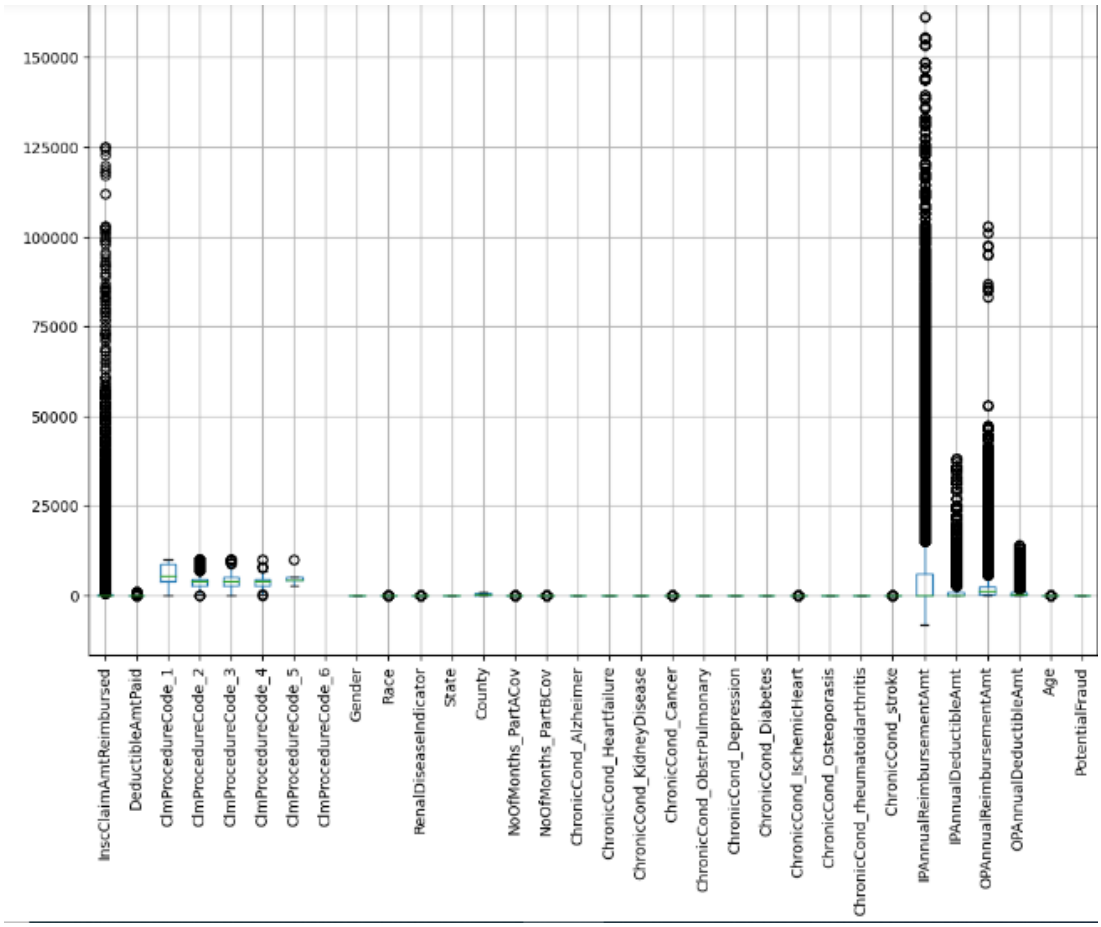
The Pre-processing techniques that are used include: -

i. Data Cleaning
    a. Missing data handling
    b. Removing outliers
ii. Data Transformation
    a. Normalization
    b. Attribute Selection
iii. Data Reduction
    a. Under sampling

Outliers

Outliers in the data is a crucial step to handle.

Whether to remove or keep them depends on the specific type of problem we are looking at!



Data Cleaning: -

Data cleaning is basically cleaning the data, by removing unwanted data. It also includes removal of noisy data (outliers), handling Null Values etc.

Checking for Null Values in Main Data Frame:

InscClaimAmtReimbursed	0
AttendingPhysician	1508
OperatingPhysician	443764
OtherPhysician	358475
AdmissionDt	517737
ClmAdmitDiagnosisCode	412312
DeductibleAmtPaid	899
DischargeDt	517737
DiagnosisGroupCode	517737
ClmDiagnosisCode_1	10453
ClmDiagnosisCode_2	195606
ClmDiagnosisCode_3	315156
ClmDiagnosisCode_4	393675
ClmDiagnosisCode_5	446287
ClmDiagnosisCode_6	473819
ClmDiagnosisCode_7	492034
ClmDiagnosisCode_8	504767
ClmDiagnosisCode_9	516396
ClmDiagnosisCode_10	553201
ClmProcedureCode_1	534901
ClmProcedureCode_2	552721
ClmProcedureCode_3	557242
ClmProcedureCode_4	558093
ClmProcedureCode_5	558202
ClmProcedureCode_6	558211
DOB	0
DOD	554080

Some columns have more than 70% null values, imputing such columns would result in increasing in wrong data. Discarding such values is desired. Now after that we will remove some columns using mean imputation.

This is the Data Frame after removing all Null values and null Columns

BeneID	0
ClaimID	0
ClaimStartDt	0
ClaimEndDt	0
Provider	0
InscClaimAmtReimbursed	0
AttendingPhysician	0
OperatingPhysician	0
OtherPhysician	0
ClmAdmitDiagnosisCode	0
DeductibleAmtPaid	0
ClmDiagnosisCode_1	0
ClmDiagnosisCode_2	0
ClmDiagnosisCode_3	0
ClmDiagnosisCode_4	0
ClmDiagnosisCode_5	0
DOB	0
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
Age	0
PotentialFraud	0

Scaling

Scaling the data is an important aspect before fitting it into the model.

If the dataset is to be scaled, only the regressor columns should be scaled whereas the target column should not be scaled.

Two common techniques for scaling the data: -

Normalization

Standardization

Normalization –

Normalization is a technique used to scale the numerical data to a range between 0 and 1.

This is done by subtracting the minimum value from each data point and dividing by the range (max value – min value).

Normalization is used when the scale of the data is unknown or when the data has different units.

Min-Max Scalar is used –

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization –

Standardization is a technique used to scale the numerical data to have a mean 0 and standard deviation 1.

It is done by subtracting the mean value from each data point and dividing by the standard deviation.

Standardization is useful when the data has a known distribution.

Standard Scalar is used –

$$Z = \frac{x - \mu}{\sigma}$$

```
# Seperating the target and independent variables

y = df_final.PotentialFraud
x1 = df_final.drop(['PotentialFraud','Provider'],1)
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
# initialize the standard scalar
ms = StandardScaler()

# standardize all the columns of the dataframe 'df_ipl_dummy'
num_scaled = ms.fit_transform(x1)
```

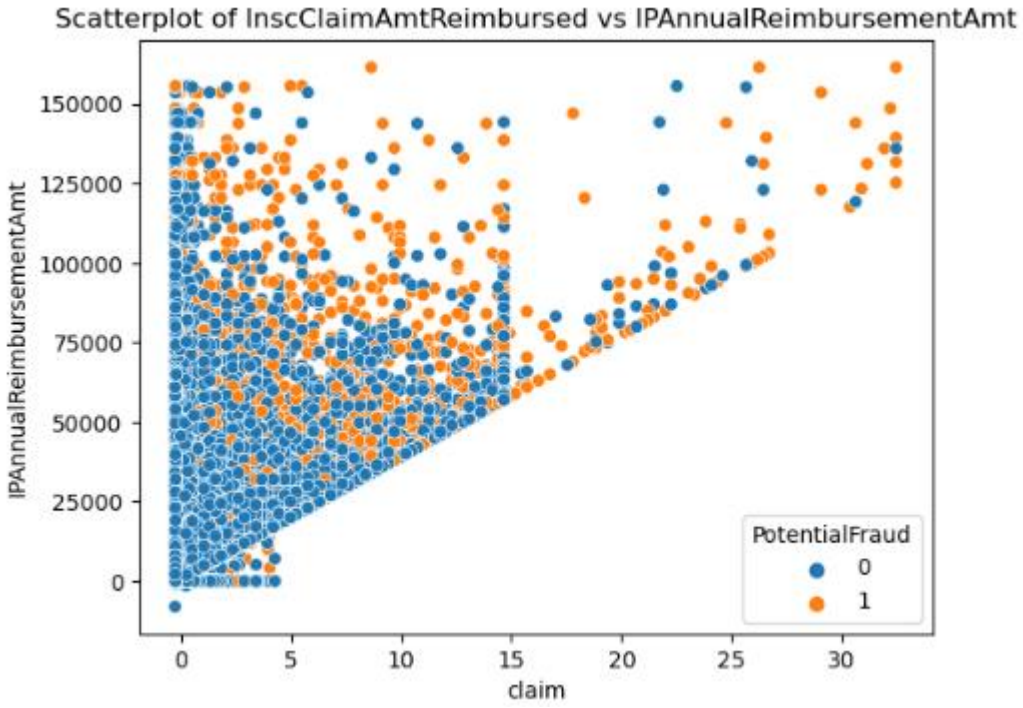
```
X = pd.DataFrame(num_scaled,columns = x1.columns)
X.head()
```

	Age	provider_InscClaimAmtReimbursed_mean	provider_DeductibleAmtPaid_mean	provider_NoOfMonths_PartACov_mean	provider_NoOfMonths_PartBCov_
0	1.084277	0.701726	0.200793	0.202919	0.2
1	-0.787012	0.817338	1.159763	-0.256629	-0.1
2	-0.333191	-0.399106	-0.502132	-0.136345	0.0
3	-0.690913	-0.430393	-0.498491	-0.031392	0.0
4	-0.942165	-0.365221	-0.358393	-0.218334	-0.3

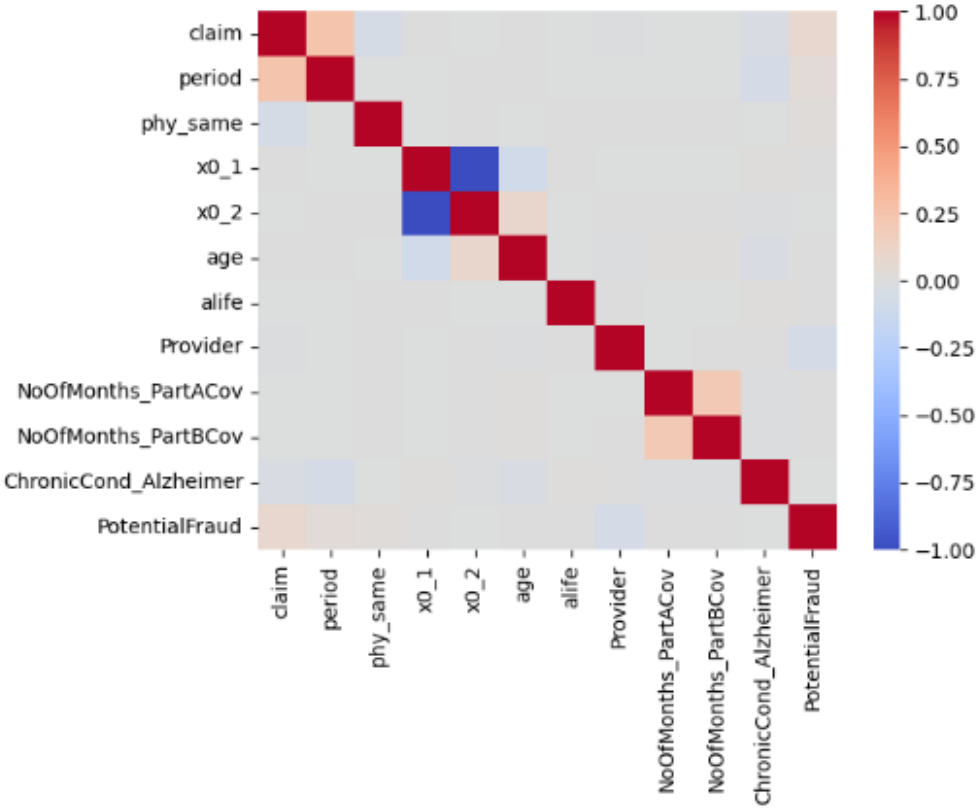
5 rows × 49 columns

Findings From the data:

The above Bar shows that the data has some imbalance it, where the fraudulent cases are comparatively low.



In this Scatter, it evident that frauds are mainly concentrated on higher amounts to gain more money.



The Potential Fraud column seems to 0 zero correlation for above columns, but there may chance that it may have good relation for the combination of the columns. Because of this Logistic regression may get bad results. But to get the good result's we need to find the pattern's which are followed by the frauds. For the Decision tree algorithms will be effective.

Train, Test and Split

Train-test split is a technique used in machine learning to evaluate the performance of a model on unseen data.

The main idea is to split the dataset into two parts: a training set, used to train the model, and a testing set, used to evaluate the performance of the model.

The training set is used to fit the parameters of the model, and the testing set is used to evaluate the model's accuracy. The split is usually done randomly and the size of the testing set is typically smaller than the training set.

The train-test split is a crucial step in the machine learning workflow, as it allows us to estimate how well the model is likely to perform on new, unseen data.

Without a train-test split, there is a risk that the model is overfitting to the training data, which means it is learning the noise in the training data rather than the underlying patterns.

In summary, train-test split is a method of dividing a dataset into a training set and a testing set to evaluate the performance of a machine learning model. The training set is used to train the model, and the testing set is used to evaluate the model's performance on unseen data.

X has all the input features. y has the

target feature for our dataset.

We are splitting our dataset into 80% training data and 20% testing data.

So, we train our model using the training data and keep it ready.

We use the test data and check the performance of the model.

The Models:

The Models that I have used are:

1. Logistic Regression
2. Neural Networks
3. Decision Trees
4. Ensemble Methods

Evaluation Metrics

When it comes to evaluating the performance of classification models, there are several metrics that can be used. Here are some of the most commonly used ones:

1. Confusion matrix: A confusion matrix is a table that summarizes the predictions of a classification model. It shows the number

of true positives, false positives, true negatives, and false negatives. From this matrix, various other metrics can be calculated.

2. Accuracy: Accuracy is the proportion of correct predictions to the total number of predictions made. It is calculated as (TP + TN) / (TP + FP + TN + FN).
3. Precision: Precision is the proportion of true positives among the cases predicted as positive. It is calculated as TP / (TP + FP).
4. Recall: Recall is the proportion of true positives among the cases that are actually positive. It is calculated as TP / (TP + FN).
5. F1-score: F1-score is a harmonic mean of precision and recall. It is calculated as 2 * (precision * recall) / (precision + recall).

1. Logistic Regression:

Logistic regression is a classification algorithm used to predict the probability of a binary outcome. It models the relationship between a dependent binary variable and one or more independent variables using a logistic function. The logistic function maps any real-valued number into a value between 0 and 1, which can be interpreted as a probability. Logistic regression is simple and computationally efficient, making it a popular choice for binary classification problems.

Advantages:

- Simple and computationally efficient
- Can handle a large number of features
- Can provide a measure of feature importance

Disadvantages:

- Assumes a linear relationship between the independent and dependent variables
- May not work well with non-linear relationship
- May suffer from overfitting

Code

Logistic Full Model

```
logreg = LogisticRegression()
logreg.fit(X_train_const,y_train_const)
```

```
LogisticRegression()
```

```
# let 'y_pred_prob' be the predicted values of y
y_pred_prob = logreg.predict(X_test_const)
```

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

Results

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

```
              precision    recall  f1-score   support

           0       0.82      0.78      0.80       109
           1       0.78      0.82      0.80       103

    accuracy                           0.80       212
   macro avg       0.80      0.80      0.80       212
weighted avg       0.80      0.80      0.80       212
```

```
# printing the scores for the model

print("roc_auc_score:",metrics.roc_auc_score(y_test_const, y_pred))
print("cohen_kappa_score:",metrics.cohen_kappa_score(y_test_const, y_pred))
print("precision_score:",metrics.precision_score(y_test_const, y_pred))
print("accuracy_score:",metrics.accuracy_score(y_test_const, y_pred))
print("recall_score:",metrics.recall_score(y_test_const, y_pred))
print("f1_score:",metrics.f1_score(y_test_const, y_pred))
```

```
roc_auc_score: 0.7976752471719961
cohen_kappa_score: 0.5945561288027041
precision_score: 0.7777777777777778
accuracy_score: 0.7971698113207547
recall_score: 0.8155339805825242
f1_score: 0.7962085308056872
```

2. Decision Trees:

Decision trees are a type of supervised learning algorithm used for classification and regression tasks. They work by recursively partitioning the feature space into smaller and smaller subsets based on the values of the features. At each step, the algorithm selects the feature that provides the most information gain and splits the data based on the value of that feature. Decision trees are easy to interpret and can handle both categorical and numerical data.

Advantages:

- Easy to interpret and visualize
- Can handle both categorical and numerical data
- Can be used for both classification and regression tasks

Disadvantages:

- Can be prone to overfitting if the tree is too deep
- May not perform well with high-dimensional data
- May not capture complex relationships between variables

How decision Tree works:

Gini Index:

The Gini impurity measures the frequency at which any element of the dataset will be mislabelled when it is randomly labelled. The minimum value of the Gini Index is 0. This happens when the node is pure, this means that all the contained elements in the node are of

one unique class. Therefore, this node will not be split again. Thus, the optimum split is chosen by the features with less Gini Index. Moreover, it gets the maximum value when the probability of the two classes are the same.

Gini Index is calculated using the following formula: -

$$GiniIndex = 1 - \sum_j p_j^2$$

Here p_j is the probability of class j.

Entropy:

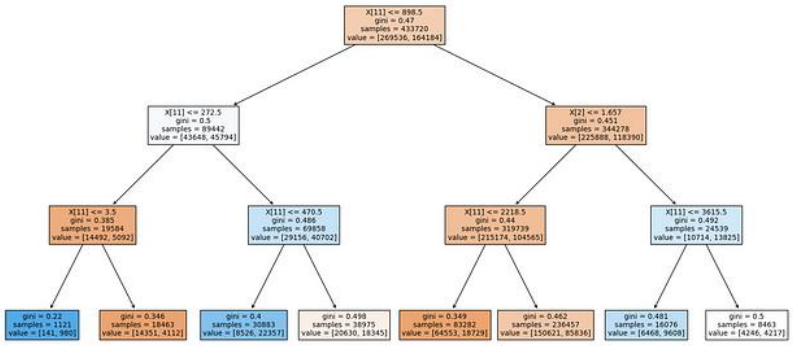
Entropy is a measure of information that indicates the disorder of the features with the target. Similar to the Gini Index, the optimum split is chosen by the feature with less entropy. It gets its maximum value when the probability of the two classes is the same and a node is pure when the entropy has its minimum value, which is 0. The Entropy is calculated using the following formula: -

$$Entropy = -\sum_j p_j \cdot log_2 \cdot p_j$$

Here p_j is the probability of class j.

Code

```
# pass the criteria 'gini' to the parameter, 'criterion'
# max_depth: that assigns maximum depth of the tree
# min_samples_split: assigns minimum number of samples to split an internal node
# max_leaf_nodes': assigns maximum number of leaf nodes in the tree
# pass the 'random_state' to obtain the same samples for each time you run the code

dt_model = DecisionTreeClassifier(criterion = 'gini',
                                  max_depth = 70,
                                  min_samples_split = 60,
                                  max_leaf_nodes = 85,
                                  random_state = 10)

# fit the model using fit() on train data
decision_tree1 = dt_model.fit(X_train, y_train)

# compute the performance measures on train data
# call the function 'get_train_report'
# pass the decision tree to the function
train_report = get_train_report(decision_tree1)


# compute the performance measures on test data
# call the function 'get_test_report'
# pass the decision tree to the function
test_report = get_test_report(decision_tree1)


# print the performance measures
print('Test data:\n', test_report)
```



Here is the example of a decision tree, here maximum depth is 3.

Results

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

           0       0.72      0.86      0.78       109
           1       0.81      0.64      0.72       103

    accuracy                           0.75       212
   macro avg       0.77      0.75      0.75       212
weighted avg       0.76      0.75      0.75       212
```

3. Ensembling Methods:

Ensembling methods are a type of machine learning algorithm that combine multiple models to improve predictive performance. There are two main types of ensembling methods: bagging and boosting. Bagging methods build multiple models using different subsets of the training data and then average the predictions to reduce variance. Boosting methods, on the other hand, build a sequence of models that are trained on the residuals of the previous model to reduce bias.

Advantages:

- Can improve predictive performance by reducing variance and bias
- Can be used with a wide range of machine learning algorithms
- Can handle both classification and regression tasks

Disadvantages:

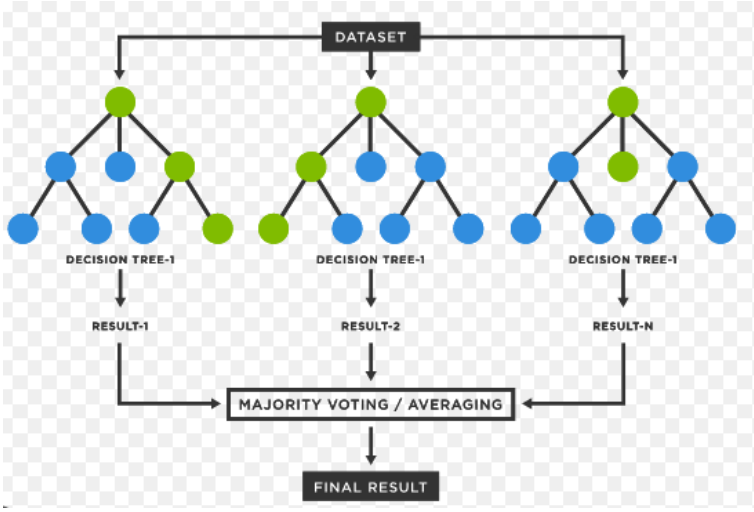
- Can be computationally expensive and time-consuming to train
- May suffer from overfitting if not properly regularized
- Can be difficult to interpret the results

Various Ensemble Methods Used:

Random Forest:

Random Forest is an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of

the individual trees. The Random Forest algorithm is a bagging algorithm, where the ensemble is built using bootstrap samples of the training data.



A random forest with n-trees =3

Advantages:

- Random Forest has the ability to handle large datasets with high dimensionality.
- It is a powerful algorithm for identifying and handling missing data.
- It is robust to overfitting and can generalize well on new, unseen data.
- It can be used for both classification and regression problems.

Disadvantages:

- Random Forest models can be time-consuming to train on large datasets with many features.

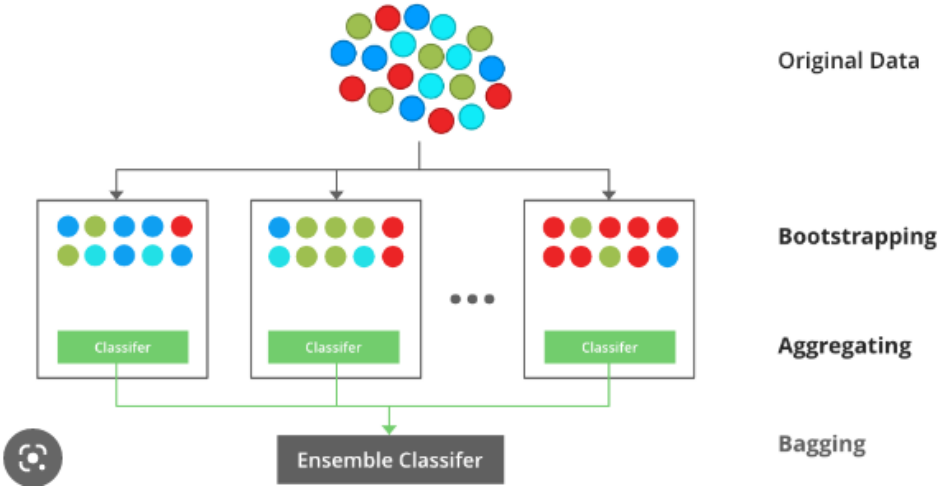
- It can be difficult to interpret the results and understand the relationships between the variables and the target.

Evaluation Metrics:

- Confusion Matrix: A table that summarizes the classification results showing the number of correct and incorrect predictions for each class.
- Accuracy: The percentage of correctly classified instances out of the total number of instances.
- Precision: The proportion of true positives among the predicted positives.
- Recall: The proportion of true positives among the actual positives.
- F1 Score: A weighted average of precision and recall.
- Area Under ROC Curve (AUC-ROC): A measure of how well the model distinguishes between positive and negative classes.

XGBoost:

XGBoost is a scalable, fast, and efficient algorithm for classification and regression problems. It is an ensemble learning method that combines multiple weak models to create a stronger model.



Here in the last tree, all the misclassified records will get higher probability to get selected

Advantages:

- XGBoost can handle missing data and outliers effectively.
- It is a powerful algorithm for feature selection and can identify the most important variables for the prediction.
- It can handle both regression and classification problems.
- It is fast and scalable, making it suitable for large datasets.

Disadvantages:

- XGBoost models can be prone to overfitting if the hyperparameters are not tuned properly.
- It requires a considerable amount of computational resources to train the models.

Evaluation Metrics:

- Confusion Matrix: A table that summarizes the classification results showing the number of correct and incorrect predictions for each class.
- Accuracy: The percentage of correctly classified instances out of the total number of instances.
- Precision: The proportion of true positives among the predicted positives.
- Recall: The proportion of true positives among the actual positives.
- F1 Score: A weighted average of precision and recall.
- Area Under ROC Curve (AUC-ROC): A measure of how well the model distinguishes between positive and negative classes.
- In conclusion, both Random Forest and XGBoost are powerful algorithms for classification using ensembling methods. The choice of the algorithm depends on the characteristics of the dataset and the specific problem at hand. It is important to evaluate the performance of the models using appropriate metrics and tune the hyperparameters to achieve the best results.

1. Random Forest

Code

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

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

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

Results

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

           0       0.83      0.79      0.81       109
           1       0.79      0.83      0.81       103

    accuracy                           0.81       212
   macro avg       0.81      0.81      0.81       212
weighted avg       0.81      0.81      0.81       212
```

2. XGBoost

Code

```
XGBoost (extreme gradient boost) is an alternative form of gradient boosting method. This method generally considers the
initial prediction as 0.5 and build the decision tree to predict the residuals. It considers the regularization parameter to
avoid overfitting.
```

```
# pass the value of minimum loss reduction required for partition of the leaf node to the parameter, 'gamma'
xgb_modelf = XGBClassifier(max_depth = 10, gamma = 1)

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

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=1, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=10, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

Results

	precision	recall	f1-score	support
0	0.80	0.78	0.79	109
1	0.77	0.80	0.78	103
accuracy			0.79	212
macro avg	0.79	0.79	0.79	212
weighted avg	0.79	0.79	0.79	212

roc_auc_score: 0.7879665093079185
cohen_kappa_score: 0.5754716981132075

Combining All Results:

On Comparing all result's, we got:

Model Name	Accuracy	Precision	Recall	F1 Score	AUC	Kappa
Log Reg	0.8	0.77	0.82	0.8	0.79	0.59
Bayes Decision Tree	0.74	0.76	0.68	0.72	0.74	0.48
Decision Tree (basic tuning)	0.78	0.8	0.74	0.77	0.78	0.57
Decision Tree RandomizedSearchCV	0.8	0.75	0.86	0.81	0.85	0.59
Decision Tree GridSearchCV	0.8	0.82	0.75	0.78	0.85	0.58
Random Forest (basic tuning)	0.82	0.81	0.82	0.81	0.81	0.63
Random Forest GridSearchCV	0.8	0.79	0.82	0.8	0.9	0.65
Random Forest RandomizedSearchCV	0.78	0.75	0.82	0.78	0.89	0.571
AdaBoost	0.79	0.78	0.81	0.79	0.79	0.58
Gradient Boosting	0.75	0.75	0.75	0.75	0.75	0.5
XGBoost	0.83	0.8	0.85	0.83	0.82	0.65
Knn	0.77	0.71	0.87	0.79	0.77	0.54

These results show that the ensembling methods have performed well compared to all other models because the fraudulent cases always some patterns rather than direct correlation to some

columns, this is where decision trees and ensembling methods got edge over other algorithms.

Bibliography

1. Data set:

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

2. Evaluation Metrics:

   https://medium.com/@itbodhi/handling-imbalanced-data-sets-in-machine-learning-5e5f33c70163

3. Models :

   https://www.kaggle.com/datasets/varundse/healthcare-provider-fraud-detection-analysis?select=Healthcare+Provider+Fraud+Detection+Analysis+-+Final+Report+-+Models.ipynb

4. Data pre-processing:

   https://chat.openai.com/

5. Complete code:

   https://github.com/A23929/Project/blob/main/22237-Fraud%20Detection.ipynb