CAPSTONE PROJECT: HEALTHCARE FRAUD DETECTION

Presenter: QAISARA MARDHIAH BINTI ROSLAN

INTRODUCTION

PROBLEM STATEMENT:

Detecting healthcare fraud is a complex task due to the evolving tactics used by fraudsters. The problem is to develop an effective healthcare fraud detection system using a Supervised Learning model that can identify fraudulent activities within healthcare claims and transactions.

OBJECTIVES:

- Develop a system capable of identifying a wide range of fraudulent activities within healthcare claims.
- Utilizing supervised learning models and algorithms to enhance the accuracy of fraud detection.

DATA GATHER & PREPARATIONS

Data Cleaning

3. Identify Missing Data

```
Beneficiary.isnull().sum()
```

4. Understanding missing data

```
# create a new DataFrame to store the information about nulls
null_df = pd.DataFrame(Beneficiary.isnull().sum(), columns=['Count of Nulls'])


#null_df.index.name = "Column"
null_df.sort_values(['Count of Nulls'], ascending=False)
```

Data Cleaning

```
# 'DiagnosisGroupCode' is not null, it assigns 1; otherwise, it assigns 0.
# Drops the 'DiagnosisGroupCode' column to 'Hospt'


PtData['Hospt'] = (PtData['DiagnosisGroupCode'].notnull()).astype(int)
PtData.drop(['DiagnosisGroupCode'], axis=1, inplace=True)
```

DATA PREPARATION

```python
# Calculate the number of unique claims per beneficiary
PtData['NumUniqueClaims'] = PtData.groupby('BeneID')['ClaimID'].transform('nunique')

# Calculate ExtraClm (extra claims beyond unique claims)
PtData['ExtraClm'] = PtData['NumClaims'] - PtData['NumUniqueClaims']

# Convert date columns to datetime format
date_columns = ['AdmissionDt', 'DischargeDt', 'ClaimStartDt', 'ClaimEndDt', 'DOB', 'DOD']
PtData[date_columns] = PtData[date_columns].apply(pd.to_datetime, format='%Y-%m-%d')

# Calculate AdmissionDays (number of days in hospital)
PtData['AdmissionDays'] = (PtData['DischargeDt'] - PtData['AdmissionDt']).dt.days + 1

# Calculate ClaimDays (number of days a claim spans)
PtData['ClaimDays'] = (PtData['ClaimEndDt'] - PtData['ClaimStartDt']).dt.days + 1

# Calculate Age
PtData['Age'] = ((PtData['ClaimStartDt'] - PtData['DOB']).dt.days + 1) / 365
```

Number of Unique Claims per Beneficiary

- ❑ Identify unusual claim pattern
- ❑ Create a feature for the model

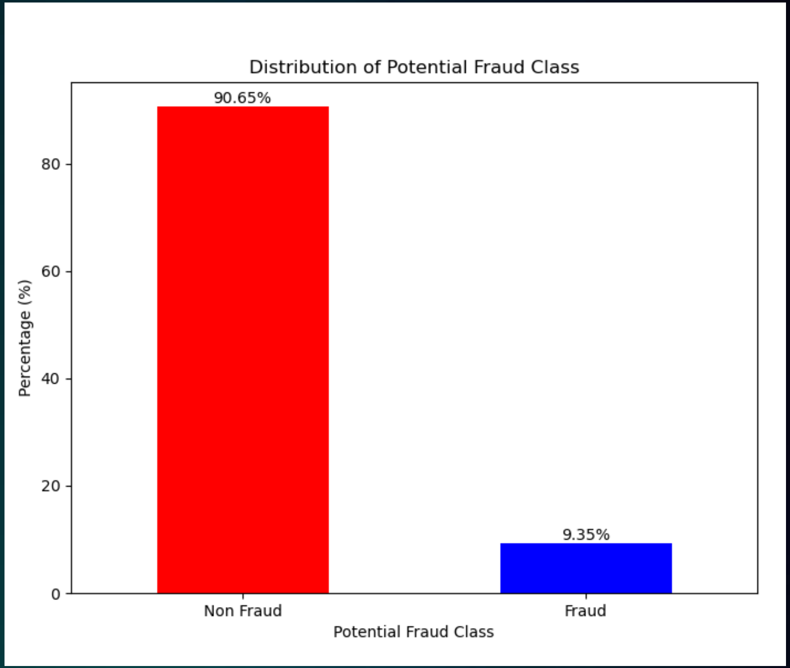
Claim Days

- ❑ Unusual long or short claim durations can be indicative of potential fraud

Age

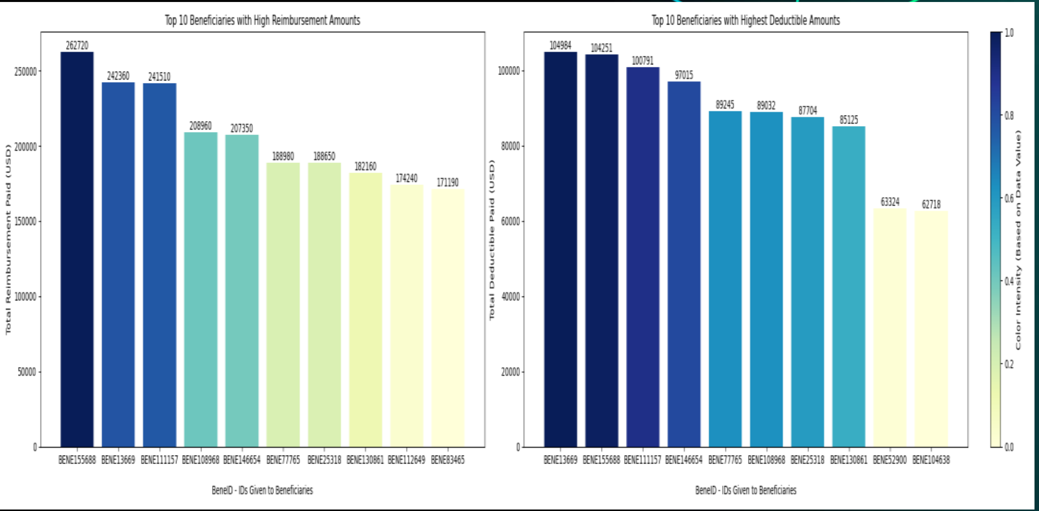
- ❑ Detecting age inconsistencies between the reported age and the DOB

EDA



- Bar plot highlights class distribution in the dataset.
- Demonstrates substantial class imbalance in fraud detection data.
- About 90.65% of data labeled as ‘Non-Fraud.’
- The 'Fraud' class constitutes a smaller portion, around 9.35% of the dataset.

EDA



- ❑ Mismatch between deductible and reimbursement amounts may suggest potential fraud.
- ❑ BENE155688, with a reimbursement of $262,720 and a much lower deductible of $104,251.
- ❑ This substantial imbalance raises concerns and merits further investigation as a possible red flag for fraud activity.

SUPERVISED LEARNING

```python
# Split the data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100)

# Initialize models
models = {
            'Logistic Regression': LogisticRegression(),
            'Decision Tree': DecisionTreeClassifier(),
            'Random Forest': RandomForestClassifier(),
            'KNN': KNeighborsClassifier()
         }

# Scale the features
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Train and evaluate each model with scaled features
accuracies = {}

for name, model in models.items():
    model.fit(x_train_scaled, y_train)
    y_pred = model.predict(x_test_scaled)
    accuracy = accuracy_score(y_test, y_pred) * 100
    accuracies[name] = accuracy
    print(f'{name} Test Accuracy: {accuracy:.2f}%')
```

```
Logistic Regression Test Accuracy: 93.07%
Decision Tree Test Accuracy: 91.13%
Random Forest Test Accuracy: 92.88%
KNN Test Accuracy: 92.33%
```

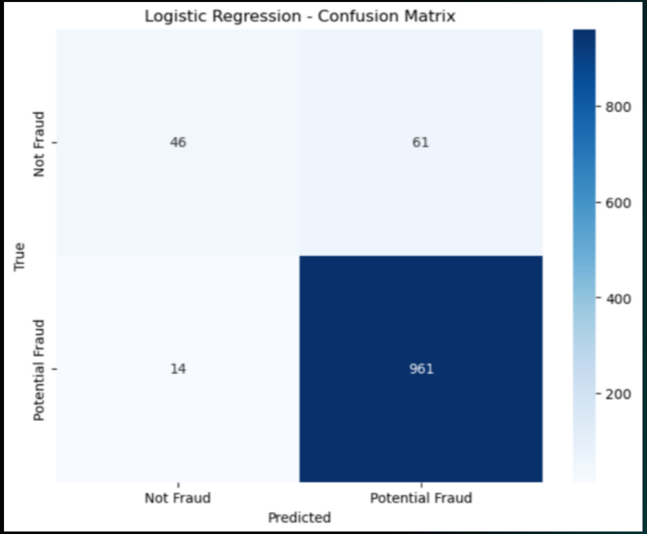
Logistic Regression

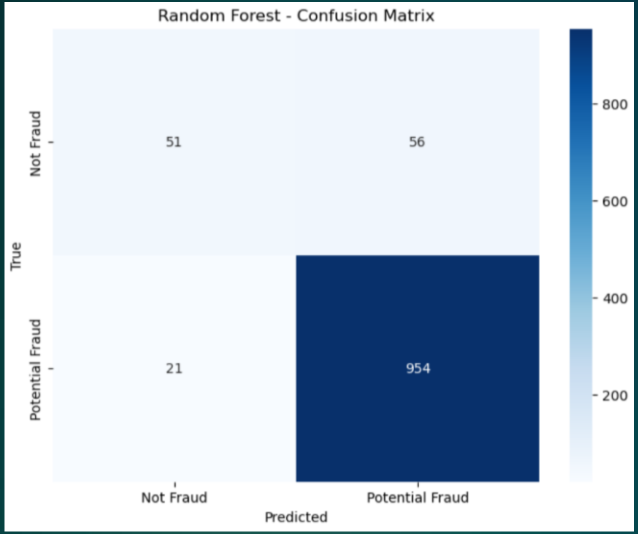
used for binary classification, modeling the probability that a given input belongs to one of two classes

Random Forest

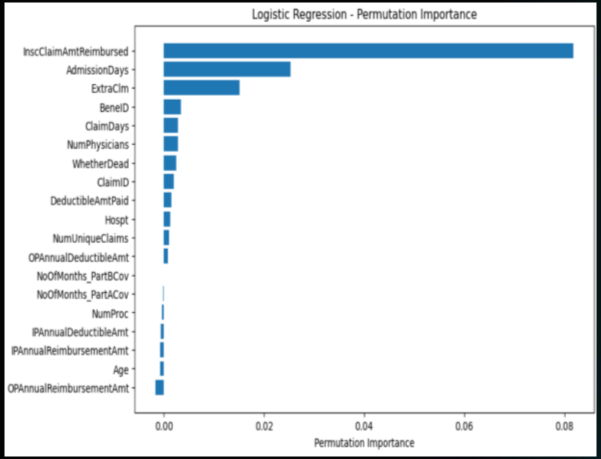
suitable for handling a large number of features to assess the importance of each feature.

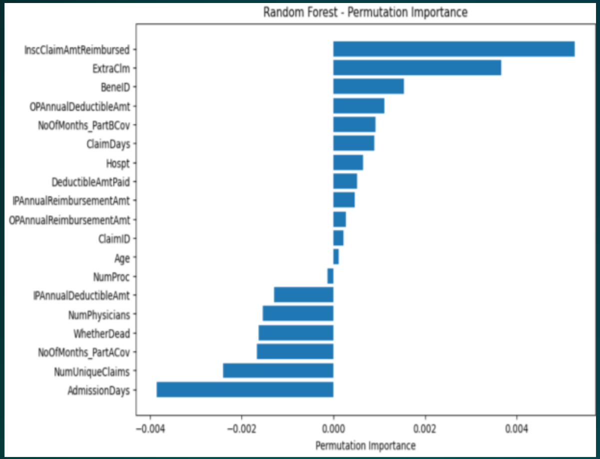
EVALUATION

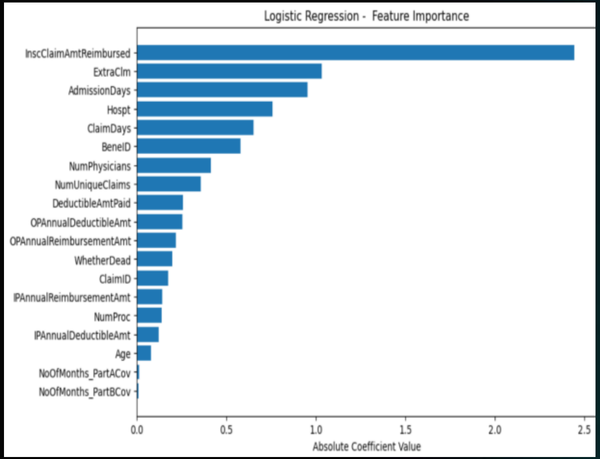

Logistic Regression - Confusion Matrix
True
Not Fraud
Potential Fraud
46
61
14
961
800
600
400
200
Not Fraud
Potential Fraud
Predicted

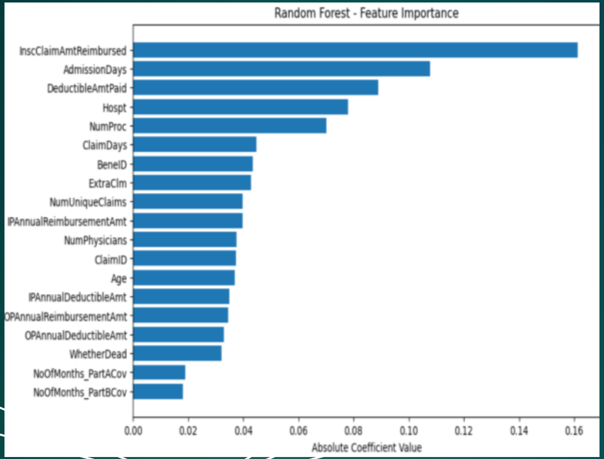

Random Forest - Confusion Matrix
True
Not Fraud
Potential Fraud
51
56
21
954
800
600
400
200
Not Fraud
Potential Fraud
Predicted

EVALUATION


Logistic Regression - Permutation Importance
InscClaimAmtReimbursed
AdmissionDays
ExtraClm
BeneID
ClaimDays
NumPhysicians
WhetherDead
ClaimID
DeductibleAmtPaid
Hospt
NumUniqueClaims
OPAnnualDeductibleAmt
NoOfMonths_PartBCov
NoOfMonths_PartACov
NumProc
IPAnnualDeductibleAmt
IPAnnualReimbursementAmt
Age
OPAnnualReimbursementAmt
0.00
0.02
0.04
0.06
0.08
Permutation Importance


Random Forest - Permutation Importance
InscClaimAmtReimbursed
ExtraClm
BeneID
OPAnnualDeductibleAmt
NoOfMonths_PartBCov
ClaimDays
Hospt
DeductibleAmtPaid
IPAnnualReimbursementAmt
OPAnnualReimbursementAmt
ClaimID
Age
NumProc
IPAnnualDeductibleAmt
NumPhysicians
WhetherDead
NoOfMonths_PartACov
NumUniqueClaims
AdmissionDays
−0.004
−0.002
0.000
0.002
0.004
Permutation Importance


Logistic Regression - Feature Importance
InscClaimAmtReimbursed
ExtraClm
AdmissionDays
Hospt
ClaimDays
BeneID
NumPhysicians
NumUniqueClaims
DeductibleAmtPaid
OPAnnualDeductibleAmt
OPAnnualReimbursementAmt
WhetherDead
ClaimID
IPAnnualReimbursementAmt
NumProc
IPAnnualDeductibleAmt
Age
NoOfMonths_PartACov
NoOfMonths_PartBCov
0.0
0.5
1.0
1.5
2.0
2.5
Absolute Coefficient Value


Random Forest - Feature Importance
InscClaimAmtReimbursed
AdmissionDays
DeductibleAmtPaid
Hospt
NumProc
ClaimDays
BeneID
ExtraClm
NumUniqueClaims
IPAnnualReimbursementAmt
NumPhysicians
ClaimID
Age
IPAnnualDeductibleAmt
OPAnnualReimbursementAmt
OPAnnualDeductibleAmt
WhetherDead
NoOfMonths_PartACov
NoOfMonths_PartBCov
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Absolute Coefficient Value

```python
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=40)
```

```python
# Create and fit a Logistic Regression model
lr = LogisticRegression(random_state=100)
lr.fit(x_train, y_train)

# Make predictions on the test data
y_pred = lr.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred) * 100
print(f'Accuracy of Logistic Regression Classifier: {accuracy:.2f}%')
```

```
Accuracy of Logistic Regression Classifier: 93.44%
```

```python
# Create and fit a Random Forest classifier model
random_forest_model = RandomForestClassifier(random_state=100)
random_forest_model.fit(x_train, y_train)

# Make predictions on the test data
y_pred = random_forest_model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred) * 100
print(f'Accuracy of Random Forest Classifier: {accuracy:.2f}%')
```

```
Accuracy of Random Forest Classifier: 93.07%
```

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Logistic Regression	0.9344	0.7808	0.5089	0.6162	0.7462
Random Forest	0.9307	0.7342	0.5179	0.6073	0.7481

The top three features were selected, and accuracy was recalculated. As a result, the accuracy for both models increased to 93.44% and 93.07%, respectively

RECOMMENDATION

Increasing the amount of fraud data in the training dataset is able to improve machine learning model performance. Balancing the dataset with a sufficient representation of both fraud and non-fraud data enhances the model's ability to accurately learn and distinguish fraud patterns