AUTOMOBILE INSURANCE FRAUD PREDICTION SYSTEM (AIFPS)

Prepared by:

NURSYAZA NISA BINTI ARFARIZAL

Who's at fault?

Ferrari owner wins RM1.5m compensation from insurance company after four-year battle

By New Straits Times - May 4, 2023 @ 3:57pm

Background Project

- Auto insurance fraud refers to the deliberate act of deceiving an auto insurance provider or manipulating insurance claims for personal gain.

- Auto insurance fraud impacts both insurance companies and policyholders. Insurance companies experience financial losses due to fraudulent claims, leading to higher premiums for honest policyholders.

- Hence, this project aims to perform auto insurance analysis and develop predictive models that can identify patterns, anomalies, and indicators of potential fraudulent activities known as Auto Insurance Fraud Prediction System (AIFPS).

Objectives

1. To investigate the features that affect fraud prediction modelling
2. To find the best machine learning algorithms for fraud prediction
3. To construct fraud profiler analysis by using Fuzzy Inference System

Problem Statement

- The insurance industry faces significant difficulties in identifying and stopping fraudulent activities, resulting in monetary setbacks, increased premiums, and reduced confidence.

- The problem statement highlights the challenges faced by the insurance industry in detecting fraud, with a particular emphasis on the identification of different fraudulent activities and their financial implications. Additionally, the statement acknowledges the issue of trust erosion.

- To tackle this issue, it is necessary to create novel fraud detection mechanisms, employ data-centric methodologies, and cooperate with industry partners to enhance the fight against insurance fraud.

Method

STANDARD SCALAR

Why Standard Scalar is chosen?

- Feature Ranking: RFE ranks the features based on their importance or contribution to the model.
- Wrapper Method: RFE is a wrapper method, meaning it selects features by training and evaluating a model.
- Improved Model Performance: By selecting the most relevant features and eliminating irrelevant or redundant ones, RFE can improve model performance.

```python
# Perform data pre-processing by importing standardscaler
# To standardize and normalize numerical input variables for classification
from sklearn.preprocessing import StandardScaler

def scale_and_encode(df):
    # Split columns into numerical and categorical
    num_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
    cat_cols = df.select_dtypes(include=['object']).columns.tolist()

    # Standardize numerical columns
    scaler = StandardScaler()
    df[num_cols] = scaler.fit_transform(df[num_cols])

    # One-hot encode categorical columns
    df = pd.get_dummies(df, columns=cat_cols,drop_first = True)

    return df
```

Method

SYNTHETHIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)

Why SMOTE is chosen?

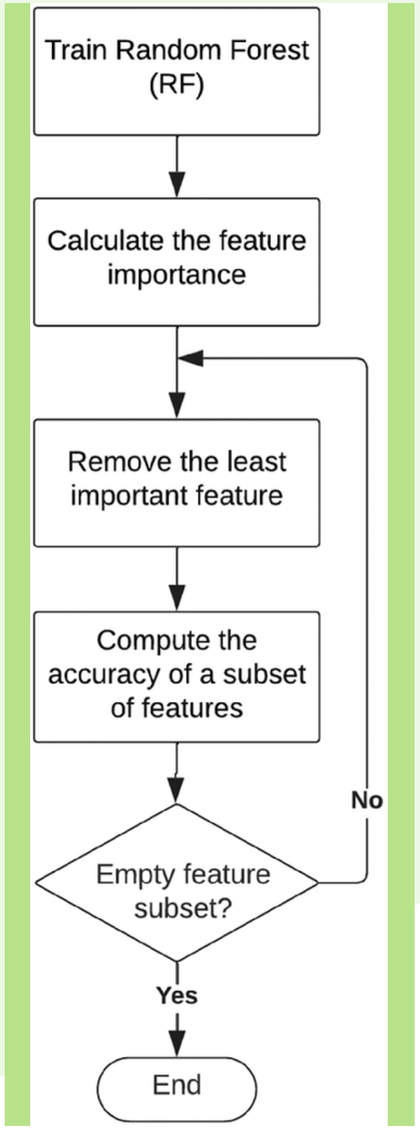
- Improved Model Performance: SMOTE can help improve the performance of machine learning models by addressing the class imbalance issue.
- Avoiding Overfitting: SMOTE generates synthetic instances that are not exact duplicates of existing minority class instances, introducing additional variability into the dataset.
- Retaining Information: Unlike simple oversampling techniques that duplicate existing instances, SMOTE creates synthetic instances that are based on existing instances.

```python
# Perform SMOTE to address class imbalance
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

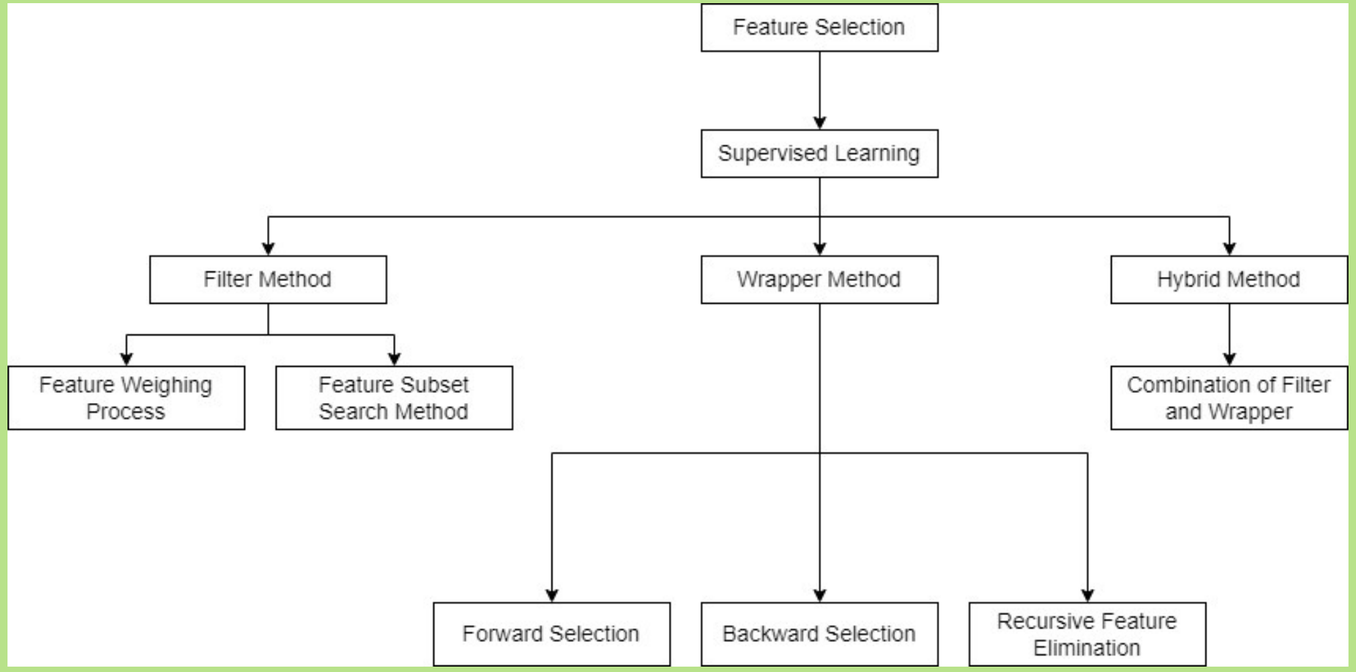
Method

Objective 1:

Recursive Feature Elimination (RFE)



Recursive Feature Elimination (RFE)



- RFE stands for Recursive Feature Elimination. It is a feature selection technique used to select the most relevant features from a given dataset.

Why RFE is chosen?

- Feature Ranking: RFE ranks the features based on their importance or contribution to the model.
- Wrapper Method: RFE is a wrapper method, meaning it selects features by training and evaluating a model.
- Improved Model Performance: By selecting the most relevant features and eliminating irrelevant or redundant ones, RFE can improve model performance.

```python
# Create the RFE object and specify the number of features to select
num_features = 10  # Number of top features to select
from sklearn.feature_selection import RFE
rfe = RFE(estimator=dt, n_features_to_select=num_features)

# Fit the RFE object to the training data
rfe.fit(X_train, y_train)

# Apply feature selection to both training and testing sets
X_train_selected = rfe.transform(X_train)
X_test_selected = rfe.transform(X_test)

# Get the top selected feature indices
selected_feature_indices = rfe.get_support(indices=True)

# Get the feature names corresponding to the selected indices
selected_feature_names = X.columns[selected_feature_indices]

# Print the top selected feature names
print("Top 10 selected features:")
for feature_name in selected_feature_names:
    print(feature_name)
```

Output

```
Top 10 selected features:
months_as_customer
policy_annual_premium
capital-loss
incident_hour_of_the_day
injury_claim
property_claim
vehicle_claim
incident_severity_Minor Damage
incident_severity_Total Loss
incident_severity_Trivial Damage
```

```python
# Create the RFE object and specify the number of features to select
num_features = 10  # Number of top features to select
from sklearn.feature_selection import RFE
rfe = RFE(estimator=xgb, n_features_to_select=num_features)

# Fit the RFE object to the training data
rfe.fit(X_train, y_train)

# Apply feature selection to both training and testing sets
X_train_selected = rfe.transform(X_train)
X_test_selected = rfe.transform(X_test)

# Get the top selected feature indices
selected_feature_indices = rfe.get_support(indices=True)

# Get the feature names corresponding to the selected indices
selected_feature_names = X.columns[selected_feature_indices]

# Print the top selected feature names
print("Top 10 selected features:")
for feature_name in selected_feature_names:
    print(feature_name)
```

Output

```
Top 10 selected features:
insured_education_level_Masters
insured_occupation_handlers-cleaners
insured_occupation_other-service
insured_occupation_priv-house-serv
insured_occupation_prof-specialty
insured_relationship_own-child
incident_severity_Minor Damage
incident_severity_Total Loss
incident_severity_Trivial Damage
incident_state_VA
```

```python
# Create the RFE object and specify the number of features to select
num_features = 10  # Number of top features to select
from sklearn.feature_selection import RFE
rfe = RFE(estimator=rf, n_features_to_select=num_features)

# Fit the RFE object to the training data
rfe.fit(X_train, y_train)

# Apply feature selection to both training and testing sets
X_train_selected = rfe.transform(X_train)
X_test_selected = rfe.transform(X_test)

# Get the top selected feature indices
selected_feature_indices = rfe.get_support(indices=True)

# Get the feature names corresponding to the selected indices
selected_feature_names = X.columns[selected_feature_indices]

# Print the top selected feature names
print("Top 10 selected features:")
for feature_name in selected_feature_names:
    print(feature_name)
```

Output

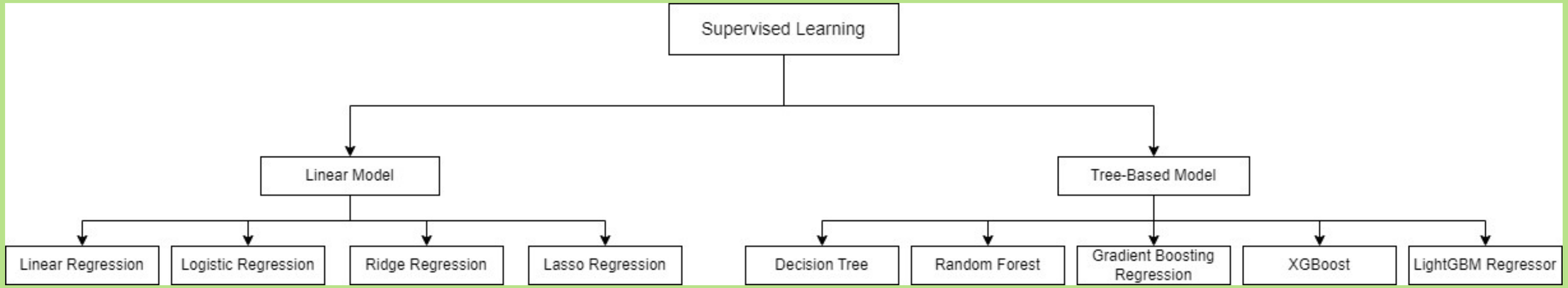
```
Top 10 selected features:
months_as_customer
policy_annual_premium
capital-gains
capital-loss
incident_hour_of_the_day
injury_claim
property_claim
vehicle_claim
incident_severity_Minor Damage
incident_severity_Total Loss
```

Method

Objective 2:

Tree-Based Models

Tree-Based Models



- In the proposed system AIFPS, this study will only focus on supervised learning tree-based models which are Decision Tree, XGBoost and Random Forest.

Why Tree-Based Models are chosen?

- Interpretability: Tree-based models provide interpretability by representing decisions and rules in a hierarchical structure.
- Non-linearity: Tree-based models can capture non-linear relationships between features and the target variable.
- Feature Importance: Tree-based models provide a measure of feature importance, indicating which features have the most influence on the model's predictions.

Decision Tree

- Decision trees are a popular and intuitive supervised learning algorithm used for both classification and regression tasks.
- Interpretability: Decision trees provide a transparent and understandable decision-making process. The tree structure can be visualized, allowing users to interpret and explain how decisions are made.
- Overall, decision trees are a versatile algorithm with a range of applications. They offer interpretability, handle nonlinear relationships, and are suitable for both small and large datasets.

Decision Tree

```
# Define the hyperparameters to tune and their possible values
parameters = {'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'min_impurity_decrease': [0.0, 0.1, 0.2]}

# Create the grid search object
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=dt, param_grid=parameters, cv=5)

# Fit the grid search object to the training data with selected features
grid_search.fit(X_train_selected, y_train)

# Get the best model with tuned hyperparameters
dt_best_model = grid_search.best_estimator_
```

Decision Tree

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Calculate the training accuracy of the best model
DecisionTree_train_acc = accuracy_score(y_train, dt_best_model.predict(X_train_selected))

# Calculate the test accuracy of the best model
DecisionTree_test_acc = accuracy_score(y_test, dt_predictions)

# Print the training and test accuracy of the Decision Tree
print(f"Training accuracy of Decision Tree is: {DecisionTree_train_acc}")
print(f"Test accuracy of Decision Tree is: {DecisionTree_test_acc}")

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, dt_predictions))

# Print the classification report
print("Classification Report:")
print(classification_report(y_test, dt_predictions))
```

XGBoost

- XGBoost (eXtreme Gradient Boosting) is a popular supervised learning algorithm that belongs to the family of gradient boosting methods.

- High Performance: XGBoost is known for its high predictive performance and is often considered one of the top-performing algorithms in many machine learning tasks. It is designed to handle large and complex datasets efficiently.

- Overall, XGBoost is a powerful and versatile algorithm that excels in predictive performance, handles missing values, and provides feature importance information.

XGBoost

```
# Define the hyperparameters to tune and their possible values
parameters={
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 500, 1000],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# Create the grid search object
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=xgb, param_grid=parameters, cv=5)

# Fit the grid search object to the training data with selected features
grid_search.fit(X_train_selected, y_train)

# Get the best model with tuned hyperparameters
xgb_best_model = grid_search.best_estimator_
```

XGBoost

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Calculate the training accuracy of the best model
XGBoost_train_acc = accuracy_score(y_train, best_model.predict(X_train_selected))

# Calculate the test accuracy of the best model
XGBoost_test_acc = accuracy_score(y_test, predictions)

# Print the training and test accuracy of the Decision Tree
print(f"Training accuracy of XGBoost is: {XGBoost_train_acc}")
print(f"Test accuracy of XGBoost is: {XGBoost_test_acc}")

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, xgb_predictions))

# Print the classification report
print("Classification Report:")
print(classification_report(y_test, xgb_predictions))
```

Random Forest

- XGBoost (eXtreme Gradient Boosting) is a popular supervised learning algorithm that belongs to the family of gradient boosting methods.

- High Performance: XGBoost is known for its high predictive performance and is often considered one of the top-performing algorithms in many machine learning tasks. It is designed to handle large and complex datasets efficiently.

- Overall, XGBoost is a powerful and versatile algorithm that excels in predictive performance, handles missing values, and provides feature importance information.

Random Forest

```
# Define the hyperparameters to tune and their possible values
parameters = {
    'n_estimators': [100, 200, 300],
                'max_depth': [None, 5, 10],
                'min_samples_split': [2, 5, 10]}

# Create the grid search object
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=parameters, cv=5)

# Fit the grid search object to the training data with selected features
grid_search.fit(X_train_selected, y_train)

# Get the best model with tuned hyperparameters
rf_best_model = grid_search.best_estimator_
```

Random Forest

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Calculate the training accuracy of the best model
XGBoost_train_acc = accuracy_score(y_train, best_model.predict(X_train_selected))

# Calculate the test accuracy of the best model
XGBoost_test_acc = accuracy_score(y_test, predictions)

# Print the training and test accuracy of the Decision Tree
print(f"Training accuracy of XGBoost is: {XGBoost_train_acc}")
print(f"Test accuracy of XGBoost is: {XGBoost_test_acc}")

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, xgb_predictions))

# Print the classification report
print("Classification Report:")
print(classification_report(y_test, xgb_predictions))
```

Method

Objective 3:

CONTINUE FYP 2

RESULT

Result Decision Tree

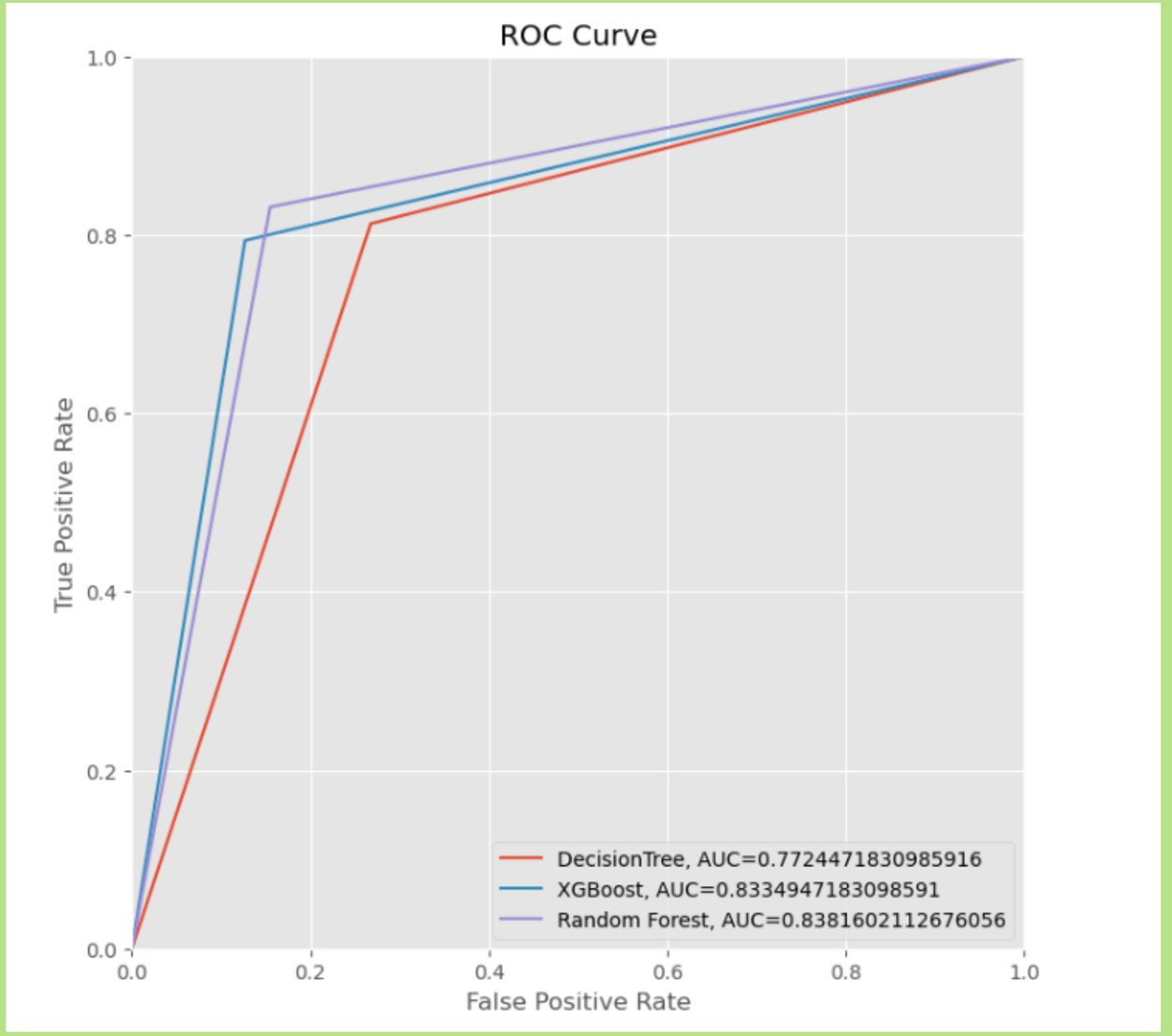
```
Training accuracy of Decision Tree is: 0.782392026578073
Test accuracy of Decision Tree is: 0.7748344370860927
Confusion Matrix:
[[104  38]
 [ 30 130]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       142
           1       0.77      0.81      0.79       160

    accuracy                           0.77       302
   macro avg       0.77      0.77      0.77       302
weighted avg       0.77      0.77      0.77       302
```

Result XGBoost

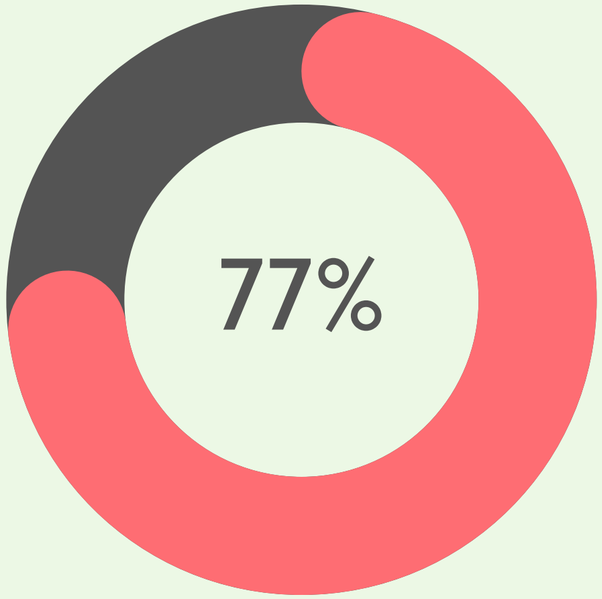
```
Training accuracy of XGBoost is: 0.5681063122923588
Test accuracy of XGBoost is: 0.847682119205298
Confusion Matrix:
[[124  18]
 [ 33 127]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.87      0.83       142
           1       0.88      0.79      0.83       160

    accuracy                           0.83       302
   macro avg       0.83      0.83      0.83       302
weighted avg       0.84      0.83      0.83       302
```

Result Random Forest

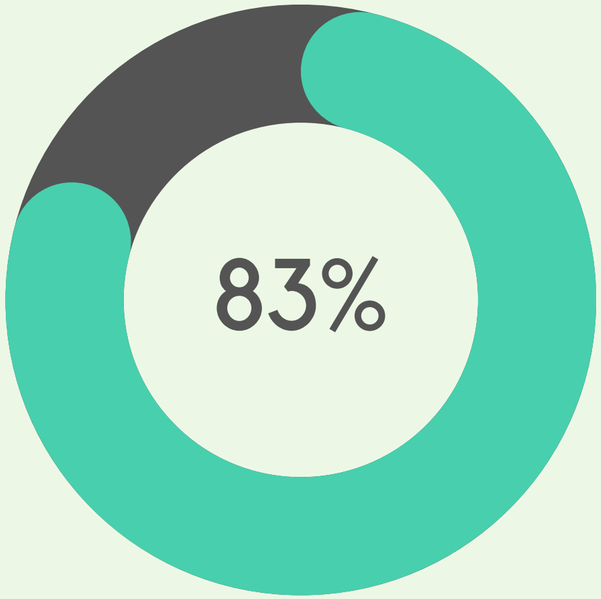
```
Training accuracy of Random Forest is: 0.9568106312292359
Test accuracy of Random Forest is: 0.8377483443708609
Confusion Matrix:
[[120  22]
 [ 27 133]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.85      0.83       142
           1       0.86      0.83      0.84       160

    accuracy                           0.84       302
   macro avg       0.84      0.84      0.84       302
weighted avg       0.84      0.84      0.84       302
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
DecisionTree, AUC=0.7724471830985916
XGBoost, AUC=0.8334947183098591
Random Forest, AUC=0.8381602112676056

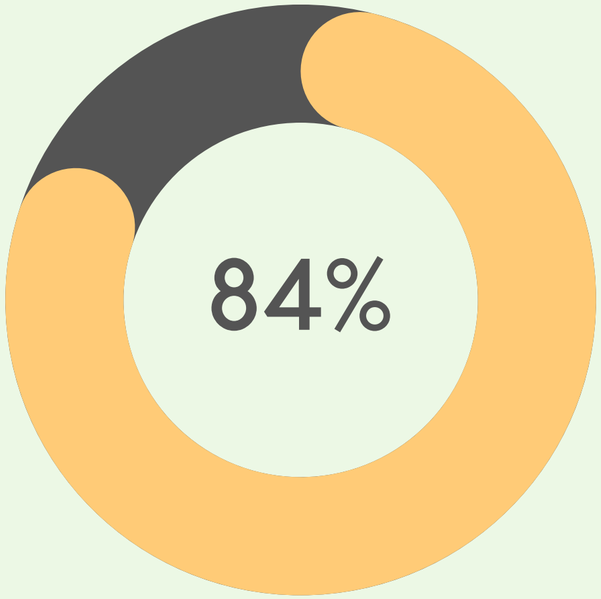
Accuracy



DECISION TREE



XGBOOST

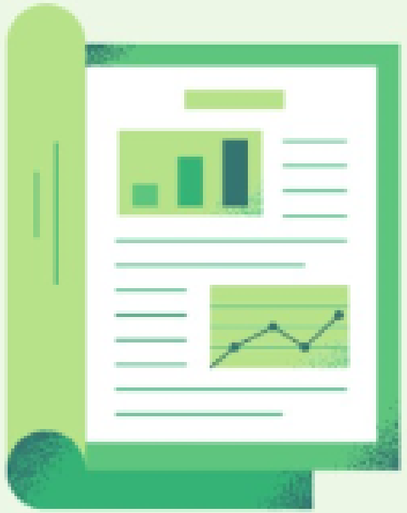


RANDOM FOREST

DEMO

AUTOMOBILE INSURANCE FRAUD PREDICTION SYSTEM (AIFPS)

LET'S GET STARTED



PREDICTION

PLEASE ENTER THE FOLLOWING INFORMATION

months_as_customer:	328
age:	48
property_claim:	15020
injury_claim:	6510
vehicle_claim:	52080

Submit

Result: FRAUD