Insurance Claims Fraud Detection

Submitted by: Subhashri Ravichandran

Submitted to: Prof. Dr. Vinay Singh

Problem Statement

Fraudulent insurance claims are a challenge for insurers, costing billions annually. Detecting such claims is critical to reduce financial losses and maintain trust with legitimate claimants. However, fraudulent activities are often sophisticated, involving false documents, exaggerated claims, or coordinated efforts by organized groups. Traditional rule-based systems, which rely on predefined conditions to flag claims, are inadequate in this dynamic environment. They struggle to keep up with evolving fraud patterns and frequently produce a high volume of false positives. This leads to inefficiencies in investigation processes, resulting in wasted resources on genuine claims while missing potential fraud. Furthermore, manual processes are time-consuming and lack the scalability required to handle the growing volume of insurance claims. The industry needs an intelligent, adaptive, and scalable solution to detect and mitigate fraud effectively.

Solution

Machine learning offers a robust and scalable approach to detect fraudulent claims, complement traditional methods and address their limitations. By leveraging advanced algorithms and large datasets, insurers can identify subtle and complex fraud patterns that manual or rule-based systems might overlook.

Supervised learning – Models such as Random Forests, Gradient Boosting can be trained using labeled historical claims data, enabling them to classify claims as fraudulent or genuine with high accuracy. For example, a Gradient Boosting model can be trained with labeled data containing features like claim amount, frequency, and time since policy inception to predict fraudulent claims. By identifying patterns such as unusually high claim amounts submitted shortly after policy initiation, the model can flag potential fraud with high precision.

Unsupervised Learning – Models like clustering and anomaly detection using algorithms such as k-means or Isolation Forests are highly effective in identifying previously unknown fraud patterns. For instance, an Isolation Forest algorithm can analyze claim data to detect anomalies, such as claims with extremely high repair costs in regions where average repair costs are significantly lower. These anomalies can then be flagged for further investigation, uncovering fraud cases that were not previously recognized.

Combining Supervised and Unsupervised Learning – Using supervised and unsupervised approaches through ensemble methods can further improve detection performance. For example, anomaly detection can pre-filter suspicious claims, which are then further analyzed by supervised models for classification.

Advanced Techniques

Natural language processing (NLP) can analyze textual claim descriptions, emails, and supporting documents for inconsistencies.

Graph-based techniques can reveal connections between seemingly unrelated claims, uncovering organized fraud rings.

Explainable AI (XAI) techniques can enable investigators understand why a claim was flagged as fraudulent. This builds trust in the system and ensures regulatory compliance.

Regular retraining of models with new data ensures adaptability to emerging fraud tactics, enabling insurers to stay ahead of evolving threats. By incorporating these technologies into claims management workflows, insurers can enhance efficiency, reduce investigation costs, minimize false positives, and uphold operational integrity while safeguarding financial resources.

Implementation – Supervised Learning

Data Source- https://www.kaggle.com/datasets/buntyshah/auto-insurance-claims-data/data

Analysis Setup

The dataset undergoes extensive preprocessing, including handling missing values, encoding categorical variables, and addressing class imbalance with SMOTE. Irrelevant columns are removed to enhance model efficiency. A train-test split (75:25) is performed to separate the data into training and testing sets. A Random Forest classifier is employed, with hyperparameter tuning conducted via GridSearchCV to optimize performance. The data is standardized, and the model is trained on a balanced dataset. Key evaluation metrics such as classification reports, confusion matrices, ROC-AUC, and precision-recall curves are analyzed. Feature importance is visualized to understand the impact of different variables on fraud detection.

Code – Random Forest Model

```
# -*- coding: utf-8 -*-
"""
Updated on Wed Jan 24 2025
@author: Subhashri Ravichandran
"""

# Importing required libraries
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc,
precision_recall_curve
from imblearn.over_sampling import SMOTE

# Set plot style
plt.style.use('ggplot')

# Load Dataset
df = pd.read_csv("C:\\Users\\subha\\OneDrive\\Dokumente\\BCM\\insurance_claims.csv")

# Data Cleaning
df.replace('?', np.nan, inplace=True)
df['collision_type'] = df['collision_type'].fillna(df['collision_type'].mode()[0])
df['property_damage'] = df['property_damage'].fillna(df['property_damage'].mode()[0])
df['police_report_available'] = df['police_report_available'].fillna(df['police_report_available'].mode()[0])

# Drop unnecessary columns
drop_columns = ['policy_number', 'policy_bind_date', 'policy_state', 'insured_zip', 'incident_location',
'incident_date','incident_state', 'incident_city', 'insured_hobbies', 'auto_make', 'auto_model', 'auto_year',
            'age', 'total_claim_amount', '_c39']
df.drop(drop_columns, inplace=True, axis=1)

# Define features and target variable
X = df.drop('fraud_reported', axis=1)
y = df['fraud_reported']

# Encode target variable
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)  # 'N' -> 0, 'Y' -> 1

# Encode categorical features in X
cat_cols = X.select_dtypes(include=['object']).columns
X[cat_cols] = X[cat_cols].apply(lambda col: label_encoder.fit_transform(col))

# Handle class imbalance using SMOTE
smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y_encoded)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_balanced, y_balanced, test_size=0.25, random_state=42)


# Standardize numeric columns
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
```

```
X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)
X_train.columns

# Hyperparameter tuning for Random Forest
param_grid = {
    'n_estimators': [100, 140, 200],
    'max_depth': [5, 10, 20],
    'min_samples_split': [2, 3, 5],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42), param_grid=param_grid,
                           scoring='roc_auc', cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_

# Train best model
best_model.fit(X_train, y_train)

# Predictions
y_pred = best_model.predict(X_test)
y_pred_proba = best_model.predict_proba(X_test)[:, 1]

# Feature Importance
importances = best_model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance':
importances}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:\n", feature_importance_df)

# Evaluate Model Performance
print("\nClassification Report (Default Threshold):\n")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Not Fraud', 'Fraud'])
disp.plot(cmap='Blues')
plt.title('Confusion Matrix - Random Forest')
plt.show()

# ROC Curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Random Forest (AUC = {roc_auc:.2f})", color='darkorange')
plt.plot([0, 1], [0, 1], 'k--', color='blue', lw=1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
```

```
plt.show()

# Precision-Recall Curve
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label="Random Forest")
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
plt.grid(True)
plt.show()

# Visualize Feature Importance
plt.figure(figsize=(10, 6))
sns.barplot(
    x=feature_importance_df['Importance'],
    y=feature_importance_df['Feature'],
    palette='viridis'
)
plt.title('Feature Importance - Random Forest', fontsize=16)
plt.xlabel('Importance', fontsize=14)
plt.ylabel('Feature', fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```

Results

The Random Forest model demonstrated strong performance in detecting fraudulent insurance claims. Out of 200 fraud cases, 176 were correctly identified as fraudulent, while 24 were misclassified as non-fraud. The model's ability to capture fraudulent claims effectively highlights its reliability. The use of SMOTE to balance class distribution contributed to improved fraud detection, reducing bias toward the majority class. The evaluation metrics, including classification reports and ROC-AUC, confirm the model's effectiveness in distinguishing fraudulent from non-fraudulent claims.

Conclusion

Despite the model’s strong performance, the presence of false negatives indicates room for further optimization. Incorporating additional fraud indicators such as claim filing delay, altered documents, multiple claims from the same contact address and so on could enhance detection accuracy. Future improvements could also explore advanced models to further minimize errors.

References

https://www.researchgate.net/publication/367795142_Prediction_of_Insurance_Fraud_Detection_using_Machine_Learning_Algorithms

https://ieeexplore.ieee.org/abstract/document/8074258

https://www.mdpi.com/2227-9091/10/12/230

https://ieeexplore.ieee.org/document/10430658