WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Machine Learning-Based Fraud Detection in Financial Transactions: A Random Forest Approach

Dr.Farheen Sultana, Mir Mohammed Ali, Gulam Rabbani Adnan, Mohammed Obaid UR Rahman

Department of IT, Nawab Shah Alam Khan College of Engineering and Technology, Hyderabad, India
Email: mirmohammedali106@gmail.com

A B S T R A C T:

In recent years, the widespread adoption of digital payments has led to a significant rise in fraudulent financial activities. Traditional fraud detection systems based on static rules often fail to recognize evolving and complex fraud patterns. This paper presents a machine learning-based web application that utilizes a Random Forest Classifier to detect fraudulent transactions with high accuracy. The system is trained on a real-world-inspired dataset and employs feature engineering, class balancing through under-sampling, and normalization techniques. To ensure model reliability, stratified cross-validation is applied during training. The final model achieves an Area Under Curve (AUC) score of 92%, indicating strong performance. A user-friendly web interface is developed using Flask to allow real-time predictions. The integration of machine learning with web deployment demonstrates the practical viability of fraud detection at scale. This system can serve as a prototype for secure transaction monitoring in financial institutions. It also highlights the importance of intelligent automation in combating financial fraud.

KeywordsFraud Detection, Machine Learning, Random Forest Classifier, Real-Time Prediction, Flask Framework, Web Application, Imbalanced Dataset, Feature Engineering, Cybersecurity, Financial Transactions.

1. Introduction:

The increasing reliance on digital platforms for financial transactions has introduced unprecedented convenience and speed, but it has also amplified the risk of fraud. Fraudulent activities in banking, e-commerce, and digital payment systems are becoming more sophisticated, making traditional rule-based detection methods increasingly ineffective. These legacy systems are often rigid, manually updated, and incapable of adapting to rapidly evolving fraud patterns, resulting in high false negatives and missed threats. To address this critical challenge, machine learning (ML) techniques offer dynamic and adaptive solutions that can learn from historical data and generalize to new fraud behaviors. In this study, we propose a fraud detection system that uses the Random Forest Classifier, a robust ensemble learning method known for its high accuracy and resistance to overfitting. The dataset is subjected to essential preprocessing steps including categorical encoding, class balancing using RandomUnderSampler, and feature scaling to ensure optimal model performance. The trained model is integrated into a user-friendly web application built with Flask, enabling real-time fraud predictions based on live transaction inputs. This approach not only enhances detection accuracy but also provides a scalable framework for practical deployment. The goal of this research is to bridge the gap between ML capabilities and real-world fraud prevention in digital financial ecosystems.

2. Literature Review:

The Fraud detection has been a prominent research topic in financial technology due to the increasing prevalence of online transaction fraud. Early systems relied heavily on manually crafted rules, which proved ineffective against evolving fraud tactics. As a result, machine learning techniques emerged as adaptive and data-driven alternatives. Researchers have applied a range of algorithms, from traditional classifiers to deep learning models, to improve fraud detection accuracy. This section highlights key contributions and advancements in the field based on previous studies:

- **Bhattacharyya et al. (2011)**
  The authors performed a comparative analysis of multiple machine learning techniques for credit card fraud detection using real-world banking

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: author@institute.xxx

datasets. Random Forest and decision tree models outperformed logistic regression in accuracy and recall. They concluded that ensemble methods offer better adaptability to fraud classification tasks under data imbalance conditions.

- **Chen, Liaw &Breiman (2004)**

  In this foundational work, the authors introduced the Random Forest algorithm as an ensemble classifier robust against overfitting and capable of handling high-dimensional, imbalanced datasets. Their experiments confirmed its efficiency in improving both classification accuracy and generalization on noisy datasets.

- **Chawla et al. (2002)**

  The researchers proposed the Synthetic Minority Over-sampling Technique (SMOTE) to tackle class imbalance in binary classification problems. They showed that SMOTE significantly boosts the sensitivity of classifiers by creating synthetic minority class examples rather than simply duplicating existing ones.

- **Dal Pozzolo et al. (2015)**

  This study evaluated under-sampling strategies for credit card fraud detection and emphasized the importance of calibration for classifier reliability. The authors demonstrated that well-calibrated under-sampling, combined with ensemble models, reduces false negatives without compromising overall accuracy.

- **Jurgovsky et al. (2018)**

  The authors explored the use of sequence modeling, particularly LSTM networks, for transaction-based fraud detection. They demonstrated that deep learning models outperform traditional classifiers when temporal behavior patterns are incorporated, especially in large-scale, real-time financial systems.

3. Methodology:

This section outlines the systematic steps followed to develop the fraud detection model and its integration into a web-based application. It covers data preprocessing, feature engineering, class balancing, model training, evaluation, and deployment. The methodology followed these key steps:

3.1 Dataset Description

The dataset used in this study is a publicly available synthetic financial transaction dataset frequently employed in fraud detection research. It contains multiple attributes such as transaction type, amount, origin and destination accounts, and the account balances before and after each transaction. The target variable is a binary label indicating whether a transaction is fraudulent or legitimate. The dataset is highly imbalanced, with fraudulent transactions comprising less than 1% of the total samples.

3.2 Data Preprocessing

Initial preprocessing involved checking for missing values, duplicates, and irrelevant attributes. Features such as nameOrig, nameDest, newbalanceOrig, and newbalanceDest were removed as they served no predictive purpose and introduced noise. The categorical feature type, representing transaction type (e.g., PAYMENT, TRANSFER), was numerically encoded to make it suitable for machine learning algorithms. The data was then shuffled to ensure randomness in training and evaluation splits.

3.3 Feature Scaling

To standardize the range of feature values and improve model convergence, feature scaling was applied using the StandardScaler from scikit-learn. This technique transforms all numeric input features to a standard normal distribution with a mean of zero and a standard deviation of one. Feature scaling helps prevent dominance of larger-valued features and ensures uniform weight distribution during model training.

3.4 Handling Class Imbalance

Due to the rarity of fraudulent transactions, the dataset exhibited significant class imbalance, which could bias the model toward predicting the majority class. To address this, the RandomUnderSampler technique from the imblearn library was used. This method randomly under-samples the majority class (non-fraudulent transactions) to balance the distribution of classes. Although this reduces the overall training size, it significantly improves the model's ability to detect minority class patterns.

3.5 Model Training and Validation

A Random Forest Classifier was selected due to its high accuracy, interpretability, and robustness to overfitting. The model was trained using stratified 5-fold cross-validation to ensure that each fold preserved the same class ratio as the original dataset. This technique minimized sampling bias and improved

model generalization. Hyperparameters such as the number of estimators and tree depth were tuned for optimal performance. Evaluation metrics included precision, recall, F1-score, and AUC (Area Under Curve).

3.6 ModelDeployment

After training, the final model was serialized using Python's pickle module for reuse in a web environment. A lightweight web application was developed using the Flask framework to provide real-time fraud predictions. The interface accepts transaction inputs from the user, processes the data in the same way as during training, and returns an instant prediction. The frontend was designed using HTML, Bootstrap, and JavaScript, ensuring an intuitive user experience and seamless model integration.

4. Illustrations:

```python
# Load dataset and drop irrelevant features
import pandas as pd

df = pd.read_csv('fraud.csv')
df.drop(['nameOrig', 'nameDest', 'newbalanceOrig', 'newbalanceDest'], axis=1, inplace=True)

# Encode categorical feature
df['type'] = df['type'].map({
    'PAYMENT': 0, 'CASH_IN': 1, 'DEBIT': 2,
    'CASH_OUT': 3, 'TRANSFER': 4
})
```

Fig. 1 – Data Preprocessing

```python
# Normalize features and handle class imbalance
from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df.drop('isFraud', axis=1))
y = df['isFraud']

rus = RandomUnderSampler()
X_resampled, y_resampled = rus.fit_resample(X_scaled, y)
```

Fig. 2 – Feature Selectionand class balancing

```python
# Train the model and evaluate its performance
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score

model = RandomForestClassifier(class_weight='balanced', random_state=42)
model.fit(X_resampled, y_resampled)

# Make predictions and print evaluation metrics
y_pred = model.predict(X_resampled)
print("Accuracy:", accuracy_score(y_resampled, y_pred))
print("AUC Score:", roc_auc_score(y_resampled, model.predict_proba(X_resampled)[:, 1]))
```

Fig. 3 –Model Training and Evaluation

5. Result:

The developed fraud detection system successfully identified fraudulent transactions with high accuracy and consistency. After training on a balanced and scaled dataset, the Random Forest Classifier achieved an AUC score of 92%, demonstrating excellent discriminatory power. The confusion matrix indicated a low false negative rate, which is critical in fraud detection applications. Precision and recall values were also notably high, confirming the model's effectiveness in detecting true fraud cases without excessive false alerts. The system was deployed through a Flask-based web interface that allowed users to input transaction data and receive instant predictions. Usability tests confirmed smooth integration between the frontend and the backend model. The application provided real-time feedback with minimal latency, making it practical for operational use. Model performance remained stable across different data splits using 5-fold cross-validation. Overall, the system demonstrated robustness, accuracy, and efficiency, highlighting its potential as a deployable tool for financial institutions.

6. Requirements:

6.1. Hardware Requirements

- **Processor:** Intel Core i5 or equivalent (minimum dual-core architecture recommended)
- **RAM:** Minimum 4 GB (8 GB or higher preferred for smooth data processing and model training)
- **Storage:** At least 250 GB of hard disk space for dataset storage, Python libraries, and project files
- **Graphics (optional):** Integrated graphics sufficient; no GPU required for this classification model
- **Display:** Standard 14" monitor or higher, supporting modern IDEs and browser-based testing

6.2. Software Requirements

- **Operating System:** Windows 10 / Ubuntu 20.04 or any compatible Linux distribution
- **Programming Language:** Python 3.8 or later
- **IDE/Editor:** PyCharm, Visual Studio Code, or Jupyter Notebook (for model training and debugging)
- **Web Framework:** Flask 2.x (for backend deployment)
- **Libraries/Dependencies:** pandas, numpy, scikit-learn, imbalanced-learn, matplotlib, pickle, flask
- **Browser:** Google Chrome or Mozilla Firefox (for UI testing)

7. Conclusion:

The Fraud Detection project is a machine learning-based solution designed to identify and prevent financial fraud in digital transactions. Developed as a web-based application using Flask, the system enables users to input transaction details and receive real-time fraud predictions. By leveraging a Random Forest Classifier, the system achieves high accuracy in distinguishing fraudulent from legitimate transactions. Through careful data preprocessing, feature scaling, and handling of class imbalance, the model was trained effectively and evaluated with a 92% AUC score. The modular structure of the project

ensures easy scalability, maintainability, and integration with live financial systems. Overall, the project emphasizes intelligent automation, user-centric design, and practical deployment, offering a scalable solution for secure digital financial services.

Step 1: Import Required Libraries

- Import libraries like pandas, numpy, and scikit-learn for data processing and model training
- Use imbalanced-learn to handle class imbalance in the dataset
- Employ matplotlib and seaborn for performance visualization
- Use pickle for saving the trained model
- Use Flask for deploying the model as a web application

Step 2: Dataset Collection and Setup

- Use a structured transaction dataset (fraud.csv) simulating real-world banking transactions
- Label each transaction as **fraudulent** or **legitimate**
- Organize features such as transaction type, amount, and balance history into a clean CSV file

Step 3: Feature Engineering

- Encode the type column to convert categorical data into numerical format
- Drop noisy or irrelevant features such as nameOrig, nameDest, and raw balance columns
- Retain essential numerical features for better prediction accuracy

Step 4: Data Preprocessing

- Scale numerical features using StandardScaler to normalize input ranges
- Apply RandomUnderSampler to balance the number of fraud vs. non-fraud samples
- Prepare the data using train_test_split and verify using cross-validation

Step 5: Model Training and Evaluation

- Train a RandomForestClassifier with class_weight='balanced' to avoid bias
- Evaluate using metrics such as Accuracy, AUC, Precision, and Recall
- Achieved an AUC of **92%**, showing excellent detection performance with minimal false positives

Step 6: Web Application Deployment

- Develop a frontend using HTML and Bootstrap for user inputs
- Build a Flask backend to load the trained model and process user requests
- Host locally or deploy on a server for real-time fraud prediction and visualization

Appendix B. Survey Questionnaire

The following questionnaire was used to gather user feedback on the Fraud Detection Web Application:

- How easy was it to input and submit transaction details through the system for fraud analysis?
- Was the web interface intuitive and responsive while using the fraud detection tool?
- Were the prediction results (fraudulent or legitimate) clearly displayed and easy to interpret?
- Did the system provide helpful feedback in case of incorrect or missing input values?
- Do you feel confident in the model's ability to accurately detect and classify fraudulent transactions?

References

1. Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). *Data mining for credit card fraud: A comparative study*. Decision Support Systems, 50(3), 602–613.
2. Chen, C., Liaw, A., &Breiman, L. (2004). *Using Random Forest to Learn Imbalanced Data*. University of California, Berkeley.
3. Chawla, N. V., Bowyer, K. W., Hall, L. O., &Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research, 16, 321–357.

4. Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). *Calibrating Probability with Undersampling for Unbalanced Classification*. IEEE Symposium Series on Computational Intelligence.
5. Jurgovsky, J., Granitzer, G., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). *Sequence classification for credit-card fraud detection*. Expert Systems with Applications, 100, 234–245.
6. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.
7. Flask Documentation. (2023). *Flask Web Framework*. Available at: https://flask.palletsprojects.com/
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
9. Sun, Y., Wong, A. K., & Kamel, M. S. (2009). *Classification of imbalanced data: A review*. International Journal of Pattern Recognition and Artificial Intelligence, 23(04), 687–719.
10. Wang, Y., Ma, R., & Huang, L. (2021). *A secure real-time fraud detection system using ensemble learning*. Journal of Information Security and Applications, 60, 102–119.