**Credit Card Fraud Detection Using Machine Learning**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**(E23CSEU0253) Kalinnd Sharma**

Submitted to

**DR. Amit Soni**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**Abstract**

In this project, we address the issue of detecting fraudulent credit card transactions using machine learning. We use a publicly available dataset containing anonymized transaction data and build a Random Forest classifier to predict fraud. The dataset is highly imbalanced, and we employ SMOTE to address this issue. Features are scaled using robust scaling, and the transaction amount is log-transformed for better feature distribution. The trained model achieves an accuracy of 99%, precision of (89%,100%), and recall of (86%,100%). The model is deployed using Streamlit Cloud, making it accessible as a web-based fraud detection tool. This project demonstrates an end-to-end solution, from model building to deployment.

1. **Introduction**

The increasing number of online transactions has resulted in a growing risk of credit card fraud. Fraudulent transactions can lead to financial losses, damage to consumer trust, and security breaches. This project seeks to address the challenge of detecting fraud using machine learning techniques. A key obstacle in fraud detection is the imbalance in the dataset—legitimate transactions far outnumber fraudulent ones, making it difficult for standard classification models to effectively identify fraud. This project uses a Random Forest classifier, along with oversampling techniques, to address this challenge. The model is deployed on Streamlit Cloud to provide a user-friendly, web-based fraud detection tool.

1. **Methodology:**

The project follows a structured approach to detect fraudulent credit card transactions. First, the publicly available dataset is preprocessed by scaling features with robust scaling and applying a log transformation to the transaction amount for improved distribution. Due to the dataset's high imbalance, Synthetic Minority Oversampling Technique (SMOTE) is used to generate synthetic samples of the minority class. A Random Forest classifier is selected for its robustness and ability to handle imbalanced data effectively. The model is trained and validated using a train-test split, and performance metrics like accuracy, precision, and recall are evaluated. Finally, the trained model is deployed as a web-based application using Streamlit Cloud, offering users an accessible and interactive fraud detection tool.

1. **Hardware/Software Required**

This section lists all the sources, tools, and datasets used during the project, giving proper credit to the creators and contributors. It includes references to research papers, datasets, software tools, machine learning libraries, and frameworks that were essential to building, training, and deploying the credit card fraud detection model. Accurate citation of these resources is important for transparency and reproducibility, as well as for acknowledging the foundational work that made the project possible.

**1. Datasets:**

- Credit Card Fraud Detection Dataset: The dataset used in this project was sourced from Kaggle. It contains transactions made by European cardholders in September 2013. The dataset includes time, amount, and 28 principal component analysis (PCA) features. -)

**2. Research Papers:**

* Original Research on Credit Card Fraud Detection:
* The dataset's creators, Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi, provided an insightful analysis of credit card fraud detection. Their paper formed the foundation of the dataset and the analytical methods used in this project.
* Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating Probability with Undersampling for Unbalanced Classification. IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 159–166.
* Available at: [https://ieeexplore.ieee.org/document/7217582](<https://ieeexplore.ieee.org/document/7217582>)
* - Handling Imbalanced Datasets:
* - Research on methods for dealing with imbalanced datasets helped in the application of techniques like SMOTE (Synthetic Minority Over-sampling Technique). The SMOTE method has been widely cited for its effectiveness in balancing datasets in classification tasks.
* - 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, pp. 321
* - Available at: [https://www.jair.org/index.php/jair/article/view/10302](https://www.jair.org/index.php/jair/article/view/10302)

**3. Python Libraries and Tools:**

* + scikit-learn: This project extensively used the `scikit-learn` library for machine learning tasks, including building the Random Forest Classifier, data splitting, model evaluation, and implementing SMOTE.
  + Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, pp. 2825–2830.
  + Available at: [https://scikit-learn.org/stable/](<https://scikit-learn.org/stable/>)
  + imbalanced-learn: This Python package was used to apply the SMOTE technique for handling imbalanced classes in the dataset.
  + Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, 18(17), pp. 1–5.
  + Available at: [https://imbalanced-learn.org/stable/](<https://imbalanced-learn.org/stable/>)
  + joblib: This package was used for saving the trained Random Forest model to disk and loading it back during deployment.
  + Joblib Documentation: [https://joblib.readthedocs.io/en/latest/](https://joblib.readthedocs.io/en/latest/)
  + pandas: The `pandas` library was used for data manipulation and preprocessing tasks, including loading the dataset and performing transformations.
  + McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, pp. 51–56.
  + Available at: [https://pandas.pydata.org/](<https://pandas.pydata.org/>)
  + numpy: The `numpy` library was crucial for handling arrays and performing numerical operations throughout the project.
  + Oliphant, T. E. (2006). A Guide to NumPy. USA: Trelgol Publishing.
  + Available at: [https://numpy.org/](<https://numpy.org/>)
  + matplotlib & seaborn: These libraries were used for generating visualizations to explore data distributions and plot performance metrics.
  + matplotlib: Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), pp. 90–95.
  + Available at: [https://matplotlib.org/](https://matplotlib.org/)
  + seaborn: Waskom, M., et al. (2021). seaborn: Statistical Data Visualization. Journal of Open Source Software, 6(60), 3021.
  + Available at: [https://seaborn.pydata.org/](<https://seaborn.pydata.org/>)
  + Streamlit: Streamlit was used to build the web interface for the machine learning model, providing an easy-to-use framework for deploying the application on the web.
  + Streamlit Documentation: [https://docs.streamlit.io/](https://docs.streamlit.io/)

**4. Deployment:**

* 1. Streamlit Cloud: This platform was used to deploy the web app and make it publicly accessible for users to input transaction details and get fraud predictions.
  2. Streamlit Cloud Documentation: [https://streamlit.io/cloud](<https://streamlit.io/cloud>)
  3. GitHub: Version control and collaboration were handled using GitHub, with the codebase stored and linked to Streamlit Cloud for continuous deployment.
  4. GitHub Documentation: [https://docs.github.com/en](https://docs.github.com/en)

**5. Additional Resources:**

- Cross-validation Techniques: Research on the application of cross-validation and grid search methods for hyperparameter tuning in Random Forest models.

- Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, pp. 1137–1143.

- Available at: [https://dl.acm.org/doi/10.5555/1643031.1643047](https://dl.acm.org/doi/10.5555/1643031.1643047)

1. **Experimental Results :**

In this section, we present the final outcomes of the credit card fraud detection model and the deployed Streamlit web application. The results focus on the performance of the model based on key evaluation metrics and the application demonstration, showcasing how the app functions in real-time when users input transaction data.

**Model Performance: Final Performance Metrics of the Model on Test Data**

After training the Random Forest classifier, the model was evaluated on a separate test dataset to measure its effectiveness in predicting fraudulent transactions. The following metrics were used to assess the model's performance:

1. Accuracy:

* + Accuracy measures the overall correctness of the model, i.e., the percentage of correctly predicted transactions (both fraud and non-fraud) out of the total number of transactions.
  + While accuracy can be useful, it may not fully reflect the model's performance on imbalanced datasets, where the majority class (non-fraud) dominates.
  + Formula :Accuracy = (True Positives + True Negatives)/(Total Transactions)

2. Precision:

* Precision focuses on the proportion of predicted fraud cases that were actually fraud. High precision indicates that the model rarely misclassifies non-fraudulent transactions as fraud.
* Formula:Precision = True Positives / (True Positives + False Positives)

3. Recall (Sensitivity or True Positive Rate):

- Recall measures the ability of the model to identify fraudulent transactions from all actual fraud cases. Higher recall means the model correctly identifies more fraudulent transactions, which is crucial in fraud detection.

Formula:Recall = True Positives/(True Positives + False Negatives)

4. F1-Score:

* The F1-score balances precision and recall, providing a single metric that reflects the trade-off between identifying as many fraudulent transactions as possible (high recall) and minimizing false positives (high precision).
* Formula:F1-Score = 2 x Precision Recall/(Precision + Recall)

5. AUC-ROC (Area Under the Receiver Operating Characteristic Curve):

- The AUC-ROC score evaluates the model’s ability to distinguish between fraud and non-fraud transactions. A higher AUC means the model has a better chance of correctly ranking a randomly chosen fraud case higher than a randomly chosen non-fraud case.

Summary of Performance:

- Accuracy: 99.92%

- Precision: 0.90 (90%)

- Recall: 0.82 (82%)

- F1-Score: 0.86 (86%)

- AUC-ROC: 0.98

The model achieved an excellent AUC-ROC score of 0.98, indicating that it is highly effective in distinguishing between fraudulent and non-fraudulent transactions. The recall value of 82% shows that the model can correctly identify a significant number of actual frauds, which is critical in reducing financial loss due to undetected fraud. The precision of 90% indicates that most of the predicted fraud cases were accurate, minimizing false positives.

1. **Conclusions:**

In conclusion, the project achieved its goal of developing an effective fraud detection model . Through careful data preprocessing, and model selection, the model was able to provide accurate predictions for credit card fraud. Future improvements could focus on exploring alternative models such as XGBoost or neural networks, enhancing interpretability, enriching the dataset with new features, and deploying the solution on more scalable platforms like AWS or GCP. These advancements would not only increase the model's accuracy but also make it more practical for real-time, large-scale fraud detection applications.

1. **Future Scope**

The project opens avenues for further enhancements and broader applications in fraud detection:

1. Exploration of Advanced Models Incorporate machine learning models like XGBoost or deep learning architectures such as neural networks to improve predictive accuracy.

2. Feature Engineering: Introduce additional features, such as temporal patterns or user behavior metrics, to enhance model robustness and generalizability.

3. Real-Time Detection: Transition from batch processing to real-time fraud detection using streaming data pipelines for faster response times.

4. Deployment on Scalable Platforms: Migrate the model to cloud platforms like AWS or GCP for better scalability, reliability, and integration with enterprise systems.

5. Explainability and Transparency: Implement tools like SHAP or LIME to enhance the interpretability of model predictions, boosting trust among stakeholders.

6. Dataset Enrichment:Utilize domain-specific datasets or simulated data to improve the model's adaptability to different fraud scenarios.

These advancements would strengthen the solution's practicality and efficiency in real-world applications.

1. **GitHub Link of Your Complete Project**

Link is https://github.com/kalinndsharma/Credit-card-fraud-detection/tree/main