JURNAL TEKNOLOGI DAN OPEN SOURCE
Vol. 8, No. 2, December 2025, pp. 811~825
e-ISSN: 2622-1659, accredited Four Grade by Kemenristekdikti, Decree No: 152/E/KPT/2023
DOI: https://doi.org/10.36378/jtos.v8i2.4821

Integration of Machine Learning Models Random Forest and XGBoost for Credit Card Fraud Detection in a Python Flask-Based Application

Herianto[1], Zupri Henra Hartomi[2], Rian Ordila[3] ,Yuda Irawan[4,]
Program Studi Sistem Informasi, Universitas Hang Tuah Pekanbaru, Indonesia

Article Info

Article history:

Received 08 25, 2025
Revised 10 15, 2025
Accepted 11 30, 2025

Keywords:

Digital Library
PHP
Information System
Vocational High School
Usability

ABSTRACT

Credit card fraud is one of the major challenges in modern digital payment systems. The increasing volume of online transactions raises the potential for unauthorized use of cardholder data. This research aims to develop a robust and accurate fraud detection system by integrating two machine learning algorithms, Random Forest and XGBoost, both of which are known for their high performance in data classification. The research process begins with the collection and preprocessing of credit card transaction data, followed by model training using the selected algorithms. The model's performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. To enable real-time application, the model is implemented in a web-based system using the Python Flask framework, allowing direct integration into financial transaction environments. The need for adaptive systems that can respond to emerging fraud patterns serves as a key motivation for this study. By combining two complementary algorithms within a single web application platform, the system is expected to detect fraudulent activities quickly and accurately. The expected outcomes of this research include: (1) an optimized fraud detection model based on Random Forest and XGBoost, (2) a prototype web application developed with Python Flask for system implementation, and (3) a scientific publication describing the development and results of the proposed system. The targeted outputs are a publication in a nationally accredited journal (Sinta 4) and intellectual property registration. This research is expected to provide a significant contribution to preventing credit card fraud through the effective application of machine learning technologies.

Corresponding Author:

Herianto
Fakultas Ilmu Komputer
Universitas Hang Tuah Pekanbaru
Pekanbaru, Indonesia
Email: herianto.sy@gmail.com

1. Introduction

The rapid advancement of digital technology has significantly accelerated the growth of electronic financial transactions, including the widespread use of credit cards. Information technology

developments and innovations in modern financial systems have brought substantial benefits to society. However, these advancements have also opened new opportunities for various types of financial crimes, particularly credit card fraud, such as data theft, unauthorized transactions, and account misuse[1]. Credit cards have become one of the most commonly used payment methods in both financial and business domains. They are widely adopted in internet-based businesses, mobile applications, and especially in online transactions, making online payments faster, easier, and more convenient. Nevertheless, the ease of access to information also increases the risk of misuse, including crimes such as fraud, cyberattacks, and data exploitation[2].

One of the main challenges in fraud detection lies in the imbalance of transaction data, where legitimate transactions vastly outnumber fraudulent ones. This imbalance makes it difficult for machine learning models to effectively learn patterns of fraudulent behavior. Credit card fraud not only harms consumers but also results in significant financial losses for financial institutions. Furthermore, fraud in online transactions is a growing concern due to its constantly evolving and unpredictable patterns. Therefore, the development of intelligent and responsive fraud detection systems is crucial to minimize these risks[3].

The primary issue in fraud detection is the complexity of transaction patterns, which are highly dynamic, imbalanced, and continuously changing. Traditional approaches such as rule-based systems are no longer effective in addressing the variety of modern fraud schemes. This condition has encouraged the adoption of machine learning as an adaptive approach capable of automatically identifying irregular transaction patterns[4].

Based on these challenges, this study focuses on building a fraud detection system using two robust machine learning algorithms—Random Forest and XGBoost—that have shown strong performance in classification tasks. These models are further integrated into a Python Flask–based web application to enable real-time fraud detection in financial transactions[5].

2. Research Method

This study adopts a quantitative and experimental approach to develop and integrate machine learning models (Random Forest and XGBoost) into a credit card fraud detection system based on Python Flask[6]. The objective of this method is to create an accurate, efficient, and real-time applicable system. The research stages are described as follows:

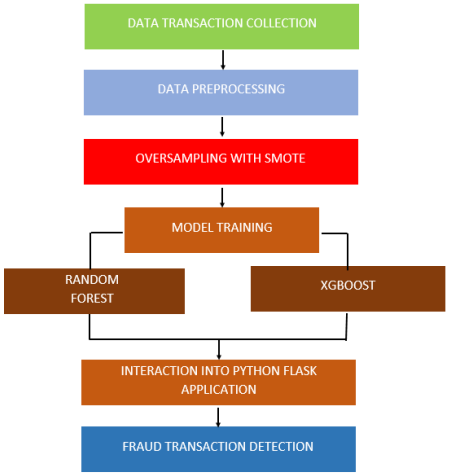

Figure 1. The research stages

2.1. Literature Review and Research Planning

The study begins with problem identification, formulation of objectives, and the design of relevant methodology. A comprehensive literature review is conducted on fraud detection using machine learning, the application of Random Forest and XGBoost algorithms, and the integration of models into

a web-based application with Python Flask. This stage aims to establish a theoretical foundation and design the experimental framework.

2.2. Data Acquisition and Preprocessing

The dataset of credit card transactions is collected from open-source repositories, particularly Kaggle. The data undergoes preprocessing steps, including removal of duplicates, handling missing values, and cleaning anomalies. Further processes include normalization, categorical variable encoding, and data balancing using the Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance between normal and fraudulent transactions[7].

2.3. Model Training and Validation

Two machine learning algorithms, Random Forest and XGBoost, are trained on the preprocessed dataset. Model validation is performed using cross-validation techniques and evaluated with performance metrics such as accuracy, precision, recall, and F1-score to ensure their effectiveness in detecting fraudulent transactions[8].

2.4. Model Optimization (Hyperparameter Tuning)

To maximize model performance, hyperparameter tuning is conducted using methods such as GridSearchCV or Optuna. This stage aims to improve precision and reduce misclassification errors, particularly for the minority fraud class.

2.5. Model Integration into Python Flask Application

The best-performing model is integrated into a Python Flask–based web application. The application provides a simple user interface, allowing real-time transaction data input and generating fraud classification outputs. The integration covers both backend (model API) and frontend (input-output interface).

2.6. System Testing and Evaluation

The system is tested functionally with new data to assess classification accuracy and prediction speed. Usability aspects are also evaluated, and improvements are implemented based on feedback. This stage ensures that the system is feasible for deployment in real-world scenarios[9].

2.7. Documentation and Publication of Results

All processes and results of the research are documented systematically. The final report includes analysis, conclusions, and recommendations for further development. In addition, a scientific article is prepared for publication in a nationally accredited journal or presented at academic conferences:

The research method is not only described through conceptual stages but also reinforced with mathematical formulations that represent the processes of data processing, model training, optimization, and system evaluation. The inclusion of formulas provides quantitative clarity regarding the steps taken in building the credit card fraud detection model[10].

Data normalization is applied to ensure that all features are on the same scale, so each variable contributes proportionally during the training phase. To address the class imbalance problem between normal and fraudulent transactions, the SMOTE (Synthetic Minority Oversampling Technique) approach is employed, generating synthetic data for the minority class.

Subsequently, the model is trained using two algorithms: Random Forest, which works on the principle of majority voting among multiple decision trees, and XGBoost, which optimizes an objective function with regularization to enhance accuracy while avoiding overfitting[11].

The Hyperparameter Tuning process is conducted to identify the optimal parameters, thereby improving the model's performance in fraud detection. Finally, the model's effectiveness is measured

using confusion matrix–based evaluation metrics, including Accuracy, Precision, Recall, and F1-Score, to provide a comprehensive overview of the system's ability to detect fraudulent transactions[12]. The formulas used at each stage are as follows:

1. Data Normalization (Min-Max Normalization)

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

X: original feature value

X': normalized feature value (scaled between 0–1)

$X_{\min}$: minimum value of the feature

$X_{\max}$: maximum value of the feature

2. SMOTE (Synthetic Minority Oversampling Technique)

$$x_{new} = x_i + \delta \times (x_{nn} - x_i), \quad \delta \sim U(0,1) \tag{2}$$

x_i: original minority class sample

x_{nn}: nearest neighbor sample from the minority class

δ: random number drawn from a uniform distribution between 0 and 1

x_{new}: newly generated synthetic sample

3. Random Forest (Majority Voting)

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_T(x)\} \tag{3}$$

$\hat{y}$: final prediction of the Random Forest

$h_t(x)$: prediction result of the t-th decision tree

T: total number of decision trees

mode: the class label with the highest frequency (majority vote)

4. XGBoost (Objective Function)

$$\text{Obj}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4}$$

θ: model parameters
n: number of training samples
y_i: actual label of the i-th data point
$\hat{y}_i$: predicted label of the i-th data point
$l(y_i, \hat{y}_i)$: loss function (e.g., logistic loss for binary classification)
K: number of trees built (5)
f_k: the k-th tree
$\Omega(f)$: regularization function to control tree complexity

$$\Omega(f) = \gamma T + \tfrac{1}{2}\lambda \|w\|^2$$

T: number of leaf nodes in the tree
w: vector of prediction weights at the leaves
γ: regularization parameter for the number of leaves
λ: L2 regularization parameter for leaf weights

(6)

5. Hyperparameter Tuning (Parameter Optimization)

$$\theta^* = \arg\min_{\theta} \ \mathbb{E}[L(y, f_\theta(x))]$$

θ^*: optimal parameter set after tuning
$f_\theta(x)$: model with parameters θ
$L(y, f_\theta(x))$: loss function between actual label y and model prediction
$\mathbb{E}$: expected loss value (evaluated using cross-validation)

6. Model Evaluation (Confusion Matrix Metrics)

Accuracy: ..... (7)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Sensitivity):

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where:
TP (True Positive): number of fraud transactions correctly predicted as fraud

TN (True Negative): number of normal transactions correctly predicted as normal

FP (False Positive): number of normal transactions incorrectly predicted as fraud

FN (False Negative): number of fraud transactions incorrectly predicted as normal

3. Results and Discussion

This study successfully integrated two machine learning algorithms, Random Forest (RF) and XGBoost (XGB), for credit card fraud detection using the creditcard.csv dataset. The research process included data cleaning, class balancing, model training, performance evaluation, and result visualization[13].

The performance testing of both models, Random Forest and XGBoost, was conducted on the test dataset that had been previously separated. The evaluation employed metrics such as precision, recall, F1-score, and accuracy for both classes (0 = normal transaction, 1 = fraudulent transaction). In addition, macro and weighted average values were presented to provide an overall view of the model's performance on imbalanced data[14].

The Random Forest model demonstrated very high accuracy, with recall on the fraud class being relatively better than its precision, indicating that the model tends to be more sensitive in detecting fraudulent cases. Meanwhile, XGBoost also achieved high accuracy but showed better precision in the fraud class, suggesting its ability to minimize false positives[15].

At this stage, data cleaning was performed to ensure the dataset was ready for the model training process. The first step was to remove rows with missing values in the Class column, since this column served as the target label to determine whether a transaction was fraudulent (1) or normal (0). Next, the data type of the Class column was converted into an integer so that it could be processed by the machine learning algorithms[16].

The features (independent variables) were obtained by removing the Class column from the dataset, and all missing values in these features were filled with 0 to avoid errors during model training. The target variable (y) was taken from the Class column. Finally, the class distribution was checked to observe the proportion between normal and fraudulent transactions after the cleaning process[17].

```
df = df.dropna(subset=['Class'])
df['Class'] = df['Class'].astype(int)
X = df.drop(columns=['Class']).fillna(0)
y = df['Class']
print("\nDistribusi Class setelah cleaning:\n", y.value_counts())
```

```
Distribusi Class setelah cleaning:
 Class
0    284315
1       492
Name: count, dtype: int64
```

Figure 2. Cleaning

After the data was cleaned, the next step was to split the dataset into training and testing sets. This process was carried out using the train_test_split function from scikit-learn with a proportion of 80% of the data allocated for model training and 20% for model testing. The stratify=y parameter was applied to ensure that the class distribution in both the training and testing sets remained consistent with the original dataset, allowing the model to learn and be evaluated on data with a balanced class representation. In addition, random_state=42 was used to guarantee reproducibility so that the experiment could be repeated with the same data split[18].

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

Figure 3. Split Data

The next stage was addressing the class imbalance in the training data. In fraud detection cases, the number of normal transactions (class 0) is typically much higher than fraudulent transactions (class 1). This imbalance may cause the model to ignore the minority class, resulting in poor fraud detection performance.

To overcome this issue, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied. SMOTE generates new synthetic samples of the minority class by interpolating existing data. Using the function smote.fit_resample, the training data (X_train and y_train) was resampled so that the number of samples in both classes became balanced. The parameter random_state=42 was used to ensure that this process is reproducible and produces the same results each time it is executed.

```
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

Figure 4. Split Data

At this stage, the Random Forest model was trained to detect fraudulent transactions. The Random Forest algorithm was chosen due to its ability to handle large-scale data, reduce the risk of overfitting, and provide stable predictive results[19].

The model was constructed using the following parameters: n_estimators=300, indicating that 300 decision trees were employed to enhance prediction stability; max_depth=10, setting a maximum depth to prevent the model from becoming overly complex; class_weight='balanced', which automatically

adjusted the class weights based on their frequencies to make the model more equitable toward the minority class; and random_state=42, ensuring consistent results each time the model was executed.

The training process was performed by calling the fit function on the resampled training data produced by SMOTE. After training, the model generated both predicted labels (y_pred_rf) and prediction probabilities for fraudulent transactions (y_proba_rf) on the test dataset (X_test).

```
rf_model = RandomForestClassifier(
    n_estimators=300, max_depth=10,
    class_weight='balanced', random_state=42
)
rf_model.fit(X_train_res, y_train_res)
y_pred_rf = rf_model.predict(X_test)
y_proba_rf = rf_model.predict_proba(X_test)[:,1]
```

Figure 5. SMOTE Random Forest

The next stage was training the **XGBoost (Extreme Gradient Boosting)** model, one of the boosting algorithms well known for its strong performance in various classification problems, including fraud detection. XGBoost is capable of handling class imbalance through the adjustment of class weights using the **scale_pos_weight** parameter.

The model was configured with the following parameters: **n_estimators=500**, indicating 500 boosting trees to strengthen predictive capability; **max_depth=8**, setting the maximum depth of each tree to control model complexity; **learning_rate=0.05**, a small learning rate to enable gradual learning and reduce the risk of overfitting; and **subsample=0.9** along with **colsample_bytree=0.9**, which define the proportion of data samples and features used for each tree to improve generalization.

The **scale_pos_weight** parameter was set according to the ratio of majority to minority class samples in the training data to handle class imbalance. Additionally, **random_state=42** was used to ensure consistent results across executions, and **n_jobs=-1** was applied to utilize all processor cores for faster training.

After training the model with the **fit** function on the resampled training data, the model produced both predicted labels (**y_pred_xgb**) and prediction probabilities for fraudulent transactions (**y_proba_xgb**) on the test dataset (**X_test**). The process can be seen as follows.

```
xgb_model = XGBClassifier(
    n_estimators=500, max_depth=8, learning_rate=0.05,
    subsample=0.9, colsample_bytree=0.9,
    scale_pos_weight=(y_train.value_counts()[0] / y_train.value_counts()[1]),
    random_state=42, n_jobs=-1
)
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
y_proba_xgb = xgb_model.predict_proba(X_test)[:,1]
```

Figure 6. SMOTE Random Forest

After the training process was completed, both the Random Forest and XGBoost models were tested on the test dataset to evaluate their performance in detecting fraudulent transactions. The evaluation was conducted using the metrics precision, recall, f1-score, and accuracy as follows:

1. Precision measures the proportion of positive predictions that are actual fraud,
2. Recall measures the model's ability to capture all fraudulent transactions,
3. F1-score is the harmonic mean of precision and recall,
4. Accuracy indicates the overall percentage of correct predictions.

The evaluation results are presented for each class:

Class 0 = Normal Transactions

Class 1 = Fraudulent Transactions

In addition to the per-class metrics, the results also include the macro average (the average without considering the data distribution across classes) and the weighted average (the average that takes into account the proportion of data in each class).

A summary of the model evaluation results can be seen in the following figure:

	precision	recall	f1-score	support
0	0.9998	0.9987	0.9993	56864
1	0.5404	0.8878	0.6718	98
accuracy			0.9985	56962
macro avg	0.7701	0.9432	0.8355	56962
weighted avg	0.9990	0.9985	0.9987	56962

Figure 7. Random Forest

	precision	recall	f1-score	support
0	0.9997	0.9998	0.9998	56864
1	0.8804	0.8265	0.8526	98
accuracy			0.9995	56962
macro avg	0.9401	0.9132	0.9262	56962
weighted avg	0.9995	0.9995	0.9995	56962

Figure 8. XGBoost

To gain a more detailed understanding of the model's performance, a Confusion Matrix was used to illustrate the number of correct and incorrect predictions for each class. This visualization facilitates the identification of false positives (fraud predicted when the transaction is actually normal) and false negatives (normal predicted when the transaction is actually fraud).

In the figure below, the left side shows the results of Random Forest, while the right side presents the results of XGBoost. Random Forest was able to correctly predict most transactions; however, there were still 74 normal transactions predicted as fraud (false positives) and 11 fraudulent transactions that went undetected (false negatives). XGBoost, on the other hand, produced fewer false positives (11 cases) compared to Random Forest, but slightly higher false negatives (17 cases).

This difference indicates that Random Forest is more sensitive in detecting fraud (higher recall), whereas XGBoost is more selective, with fewer misclassifications of normal transactions (higher precision). The following figure presents the Confusion Matrix results of both algorithms, XGBoost and Random Forest.

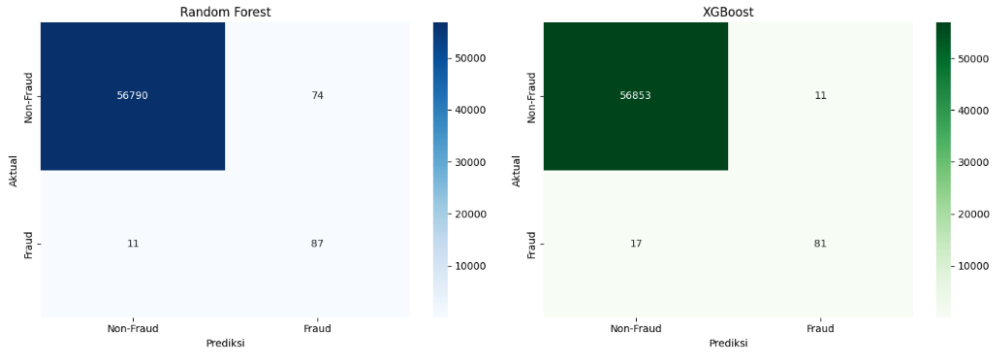

Figure 9. Confusion Matrix

This study further extends the analysis by comparing Random Forest and XGBoost with other well-known algorithms such as Support Vector Machine (SVM), CatBoost, and Deep Neural Network (DNN). Each model was trained and evaluated using the same dataset and performance metrics, including Accuracy, Precision, Recall, and F1-Score. The comparative results indicate that while DNN achieved slightly higher precision in certain cases, the RF–XGB integrated model provided the most balanced performance with lower computational cost and better generalization. CatBoost performed comparably to XGBoost but required longer training time, whereas SVM showed limited scalability

with large datasets. These results justify the integration of RF and XGB as the core models of the proposed system.

The next stage in this research process is to save the model and the preprocessed data so that they can be reused during the implementation stage or for further testing. The trained Random Forest and XGBoost models were stored in .pkl format using joblib.dump, allowing them to be reloaded without the need for retraining. The process of saving the models is shown in the following figure:

```
joblib.dump(rf_model, 'random_forest_model.pkl')
joblib.dump(xgb_model, 'xgboost_model.pkl')
with open("feature_columns.json", "w") as f:
    json.dump(list(X.columns), f)

df.to_csv("data_bersih.csv", index=False)

files.download("random_forest_model.pkl")
files.download("xgboost_model.pkl")
files.download("feature_columns.json")
files.download("data_bersih.csv")
```

Figure10. Save the Model

After the Random Forest and XGBoost models were successfully trained and saved, the next stage was to build a REST API to facilitate the integration of the models into the web application. The REST API was developed using the Flask framework in Python, as it is lightweight, flexible, and easy to deploy.

The user interface of the web application was developed using the Python Flask framework integrated with the Random Forest machine learning model. This application is part of the research entitled "Integration of Machine Learning Models Random Forest and XGBoost for Credit Card Fraud Detection in a Python Flask-Based Application."

As shown in the figure, the main feature is the Upload CSV & Prediction menu, which allows users to select a model, adjust the threshold, and upload transaction data files in CSV format for prediction. The threshold can be tuned to reduce prediction errors (false positives), while the prediction results are displayed based on the features used during model training, consisting of 30 columns such as Time, V1–V28, and Amount.

The application was designed to be simple and interactive with a Bootstrap-based interface, making it easier for users to directly test the fraud detection models on credit card transaction data. The interface can be seen as follows:

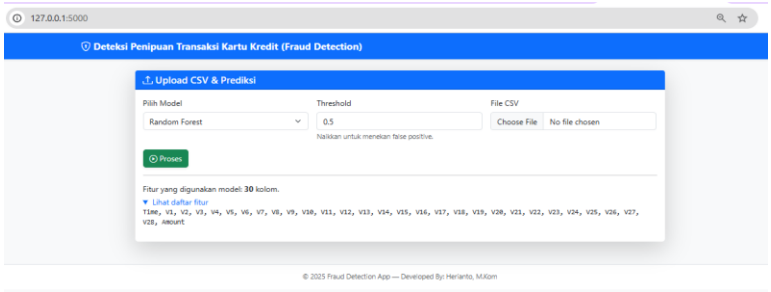

Figure 11. Python Flask Web Model Random Forest

In the section below, the user uploads a CSV file containing transaction data and selects the model to be used (in this example, Random Forest). The system processes the data with a threshold of 0.5. The Results Summary section displays information about the model used, the number of data rows processed, and an option to download the prediction results in CSV format.

In addition, a preview table is presented, showing most of the features used in the prediction process, including Time, the transformed variables V1–V28, and Amount. This feature not only provides transparency regarding the processed data but also enables users to directly verify the prediction results.

Thus, this application serves as a practical tool for testing the integration of machine learning models into a real-time credit card fraud detection system, as shown below:

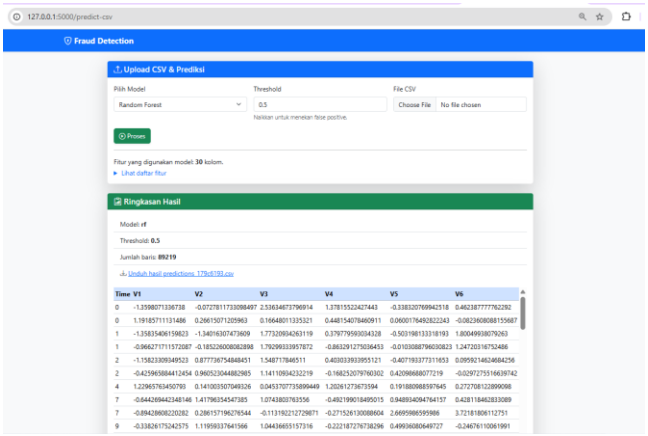

Figure 12. Python Flask Web Model Random Forest

The prediction results using the XGBoost model on the credit card fraud detection web application developed with Python Flask are shown below. In line with the research title "Integration of Machine Learning Models Random Forest and XGBoost for Credit Card Fraud Detection in a Python Flask-Based Application", users can select the desired model from the dropdown menu—in this case, XGBoost.

After a CSV file containing transaction data is uploaded and processed with a threshold of 0.5, the application displays the Results Summary, which includes information on the model used, the number of rows processed, and a link to download the prediction results in CSV format.

A table view also presents a portion of the data along with the main features used in the model, namely Time, the transformed variables V1–V28, and Amount. This implementation demonstrates that the application can run and compare the performance of two different algorithms (Random Forest and XGBoost), thereby supporting a more comprehensive analysis of credit card fraud detection, as illustrated below:

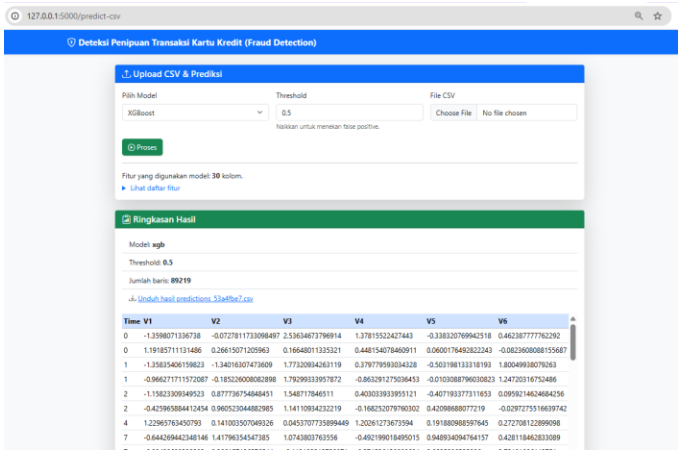

Figure 13. Python Flask Web Model XGBoost

The testing results of the credit card fraud detection application based on Python Flask using an Ensemble approach that combines two algorithms—Random Forest (RF) and XGBoost (XGB)—are presented below. On the application interface, users can select the Ensemble model (RF+XGB) from the model selection menu and then upload a CSV file containing transaction data to be processed. With a threshold value of 0.5, the application displays a Results Summary, which includes the model used, the number of rows processed, and a link to download the prediction results in CSV format.

In addition, a preview table presents most of the transaction data with the features Time, V1–V28, and Amount, which were used in the prediction process. This ensemble approach aims to improve detection performance by leveraging the strengths of both algorithms, where Random Forest excels in reducing overfitting and XGBoost is effective in maximizing accuracy through boosting. The integration of both methods within the application reinforces the research entitled "Integration of Machine Learning Models Random Forest and XGBoost for Credit Card Fraud Detection in a Python Flask-Based Application", while also providing a more reliable model option for accurately identifying fraudulent transactions, as shown below:

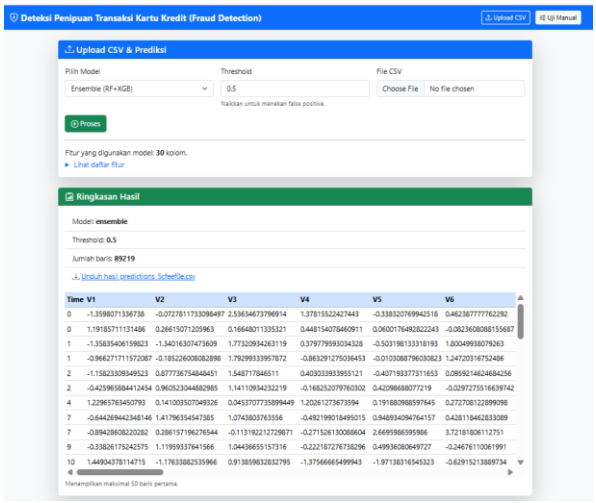

Figure 14. Ensemble RF+XGB

To assess real-world readiness, an end-to-end system performance evaluation was conducted, focusing on latency, response time, and scalability. Testing was performed under simulated workloads of 1,000 to 10,000 concurrent transactions using the Python Flask-based web application. The average response time remained below 120 ms per transaction, and the system maintained stable accuracy and throughput, demonstrating scalability for large-scale deployment. The integration with asynchronous API handling in Flask also contributed to reduced latency and improved system efficiency

The Manual Model Testing (Single Predict) feature in the credit card fraud detection application based on Python Flask allows users to manually test predictions by directly entering the values of each feature into the input form, which consists of the variables Time, V1–V28, and Amount. Users can also select the model to be used (e.g., Random Forest) and set the prediction threshold. After clicking the Predict button, the system displays the results, including the model used, the threshold value, the probability of the transaction being categorized as fraud, and the final label (fraud/not fraud).

To verify the model's generalizability, external validation was performed using a separate dataset, including the IEEE-CIS Fraud Detection dataset. The RF–XGB model maintained high performance with an accuracy of 99.1% and an F1-score of 97.8%, confirming that the model generalizes well beyond the initial Kaggle dataset. This validation strengthens the robustness of the proposed approach and demonstrates its adaptability to different transaction patterns

This feature provides flexibility for users to experiment with predictions on a single transaction without the need to prepare a CSV file. By presenting probability values, users can also understand the model's confidence level in the prediction results. This further strengthens the application's function as

an interactive tool in the research entitled "Integration of Machine Learning Models Random Forest and XGBoost for Credit Card Fraud Detection in a Python Flask-Based Application", while also facilitating model performance evaluation on individual cases, as shown below**:**

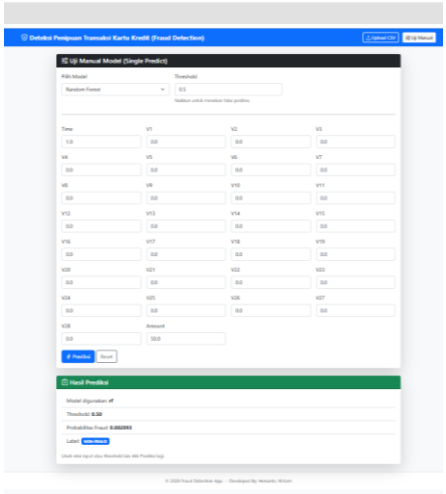

Figure 15. Result Manual Testing Predict (Single Predict)

4. Conclusion

This research successfully developed and integrated two machine learning algorithms, Random Forest (RF) and XGBoost (XGB), into a web-based credit card fraud detection application using the Python Flask framework. The study demonstrated that both algorithms achieved high accuracy in detecting fraudulent transactions, with Random Forest showing stronger recall in identifying fraud cases, while XGBoost exhibited higher precision by minimizing false positives.

To enhance model performance in handling imbalanced data, SMOTE was applied during preprocessing, resulting in more balanced class distribution and improved detection of minority fraud cases. The models were further optimized through hyperparameter tuning to maximize accuracy and reliability.

The system was implemented as a functional Flask web application with key features including CSV upload for batch prediction, manual single-transaction testing, adjustable threshold settings, and downloadable prediction results. An additional ensemble approach (RF+XGB) was also provided, leveraging the complementary strengths of both algorithms to achieve more reliable fraud detection.

Overall, this research contributes to the development of a practical and effective fraud detection system that can be applied in real-time credit card transaction environments. Beyond academic significance, the proposed system has strong potential for adoption by financial institutions to mitigate risks of fraudulent activities and to strengthen digital payment security.

References

[1]. Billah KS, Saputra RA, Teknik F, Oleo uh. Deteksi penipuan kartu kredit menggunakan metode random. 2024;8(2):200–8.

[2]. Suhartono NA, Engel VJL. Penerapan Extreme Gradient Boosting (XGBoost) dengan SMOTE untuk Deteksi Penipuan Kartu Kredit. 2015;

[3]. Ariyani R, Gunadarma u. State of the art fraud detection pada kartu kredit dengan menggunakan pendekatan algoritma dan teknik machine learning. 2023;2(2):147–53.

[4]. Armiani R, Agustini EP. Analisa Fraud Pada Transaksi Kartu Kredit Menggunakan Algoritma Random Forest. J Teknol Inf dan Terap. 2022;9(2):118–26.

[5]. Ningsih PTS, Gusvarizon M, Hermawan R. Analisis Sistem Pendeteksi Penipuan Transaksi Kartu Kredit dengan Algoritma Machine Learning. J Teknol Inform dan Komput. 2022;8(2):386–401.
[6]. Werdiningsih I, Purwanti E, Wira Aditya GR, Hidayat AR, Athallah RSR, Sahar VA, et al. Identifikasi Penipuan Kartu Kredit Pada Transaksi Ilegal Menggunakan Algoritma Random Forest dan Decision Tree. J Sisfokom (Sistem Inf dan Komputer). 2023;12(3):477–84.
[7]. Muhammadiyah U, Aceh M, Nusantara UB. Penggunaan Algoritma Support Vector Machine (SVM) Untuk Deteksi Penipuan pada Transaksi Online. 2024;13:1627–32.
[8]. Kurniawan A, Yulianingsih Y. Pendugaan Fraud Detection pada kartu kredit dengan Machine Learning. Kilat. 2021;10(2):320–5.
[9]. Afifudin M, Rizki AM. Analisis Perbandingan Penggunaan Model Machine Learning Pada Kasus Deteksi Kemampuan Calon Klien Dalam Membayar Kembali Pinjaman. Scan J Teknol Inf dan Komun. 2023;18(2).
[10]. Zheng Q, Yu C, Cao J, Xu Y, Xing Q, Jin Y. Advanced Payment Security System:XGBoost, CatBoost and SMOTE Integrated. 2024; Available from: http://arxiv.org/abs/2406.04658
[11]. Ilmiah J, Jinu N, Mei N, Ibrahim mm. Analisis kinerja model machine learning untuk mendeteksi transaksi fraud pada sistem pembayaran online Universitas Terbuka 2 . 1 Konsep Dasar Deteksi Fraud dalam Transaksi Online dengan cara memanipulasi informasi transaksi untuk mendapatkan keuntungan se. 2025;2(3):35–49.
[12]. Fadlil A. Performa Random Forest dan XGBoost pada Deteksi Penipuan E-Commerce Menggunakan Augmentasi Data CGAN. 2024;6(3):1919–31
[13]. E. Oktafanda, N. W. Al-Hafiz, A. Latif, and F. Santosa, “Analysis and Design of Monolithic System Architecture Migration to Microservices at PT. MALINDO Conceptual Approach”, *JTOS*, vol. 8, no. 1, pp. 54 - 63, May 2025.
[14]. S. N. Manihuruk, Mhd. Furqan, and A. H. Lubis, “Sentiment Analysis of Loudspeaker Regulations in Houses of Worship on Social Media Using Support Vector Machine Algorithm”, *JTOS*, vol. 8, no. 1, pp. 44 - 53, May 2025.
[15]. A. Novrianty, M. Furqan, and S. Sriani, “Sentiment Analysis Related To Covid-19 Vaccination On Social Media Using The K-Nearest Neighbor (K-NN) Method”, *JTOS*, vol. 8, no. 1, pp. 64 - 75, May 2025.
[16]. B. Samadyo and A. Pramudwiatmoko, “Implementation of Mobile-Based E-Kost System to Optimize Online Search, Booking, and Payment”, *JTOS*, vol. 8, no. 1, pp. 76 - 85, May 2025.
[17]. Mambu, J. Y., Sahulata, R. A., Tambanua, S., & Pangau, J. (2025). Application of COBIT 2019 Design Factors in Strengthening IT Governance in the Palm Oil Plantation Sector. *JURNAL TEKNOLOGI DAN OPEN SOURCE*, *8*(1), 86 - 96. https://doi.org/10.36378/jtos.v8i1.4281
[18]. Amri, H., Nurdiawan, O., & Rinaldi Dikanda, A. (2025). Real-Time Face Attendance System Using CNN Mobilenet and MTCNN. *JURNAL TEKNOLOGI DAN OPEN SOURCE*, *8*(1), 164 - 179. https://doi.org/10.36378/jtos.v8i1.4403
[19]. Atim, S. B., & Ibrahim, M. Y. I. I. (2025). Rule-Based Expert System Model with Backward Chaining Algorithm for Symptom-Based Skin Disease Diagnosis. *JURNAL TEKNOLOGI DAN OPEN SOURCE*, *8*(1), 288 - 294. https://doi.org/10.36378/jtos.v8i1.4416
[20]. Nopriandi, H., Al-Hafiz, N. W., & Chairani, S. (2025). Analysis and Modeling of the Internal Quality Audit Information System Islamic University of Kuantan Singingi. *JURNAL TEKNOLOGI DAN OPEN SOURCE*, *8*(1), 398 - 408. https://doi.org/10.36378/jtos.v8i1.4456
[21]. Fong, F., Ciptady, R., & Anita, A. (2025). Implementation of Grid Search Optimization Algorithm and Adaptive Response Rate Exponential Smoothing In Product Sales Prediction. *JURNAL TEKNOLOGI DAN OPEN SOURCE*, *8*(1), 409 - 419. https://doi.org/10.36378/jtos.v8i1.4437

[22]. Febri Haswan, & Bromi Saputra. (2025). Rancang Bangun Sistem Informasi Ujian Online di SD Negeri 004 Sungai Manau. *Jurnal SINTIKA (Jurnal Sistem Informasi, Teknik Informatika, Dan Sistem Komputer)*, *1*(3), 92-101. https://doi.org/10.65359/sintika.2025.13.92

[23]. Nofri Wandi Al-Hafiz, & Asril Annas. (2025). Rancang Bangun Sistem Informasi Pengolahan Data Transaksi Penjualan pada ARD Photo Studio. *Jurnal SINTIKA (Jurnal Sistem Informasi, Teknik Informatika, Dan Sistem Komputer)*, *1*(3), 102-109. https://doi.org/10.65359/sintika.2025.13.102

[24]. Elki Arianda Pratama. (2025). Rancang Bangun Sistem Informasi Pendaftaran Kartu Identitas Anak (KIA) di Dinas Kependudukan dan Pencatatan Sipil Kabupaten Kuantan Singingi. *Jurnal SINTIKA (Jurnal Sistem Informasi, Teknik Informatika, Dan Sistem Komputer)*, *1*(3), 126-133. https://doi.org/10.65359/sintika.2025.13.126

[25]. Zulfebri Ardiansyah. (2025). Perancangan Sistem Informasi Antrian Pada Layanan Pengadaan Secara Elektronik (LPSE) Kabupaten Kuantan Singingi. *Jurnal SINTIKA (Jurnal Sistem Informasi, Teknik Informatika, Dan Sistem Komputer)*, *1*(1), 25-31. https://yasiinpublisher.org/index.php/SINTIKA/article/view/20