

DEPARTMENT OF COMPUTER SCIENCE
Data Privacy and Security - CS 528

Project Report:

Credit card fraud detection

By:

Rohith. H

Iyyapu Lakshmi Apoorva

Buraga Balaji

Contents

1 Team Description	2
2 Application	2
3 Datasets	3
4 Privacy and Security Technique	3
5 Risk Management	4
6 Code Implementation	4
7 Testing	11
8 Results	11
9 Summary	13
10 References	14

1 Team Description

This project is spearheaded by a trio of dedicated master's students, each bringing unique skills and academic focus to the table. Rohith and Iyyapu Lakshmi Apoorva specialize in Cybersecurity, having commenced their studies in the fall of 2023. Their expertise is complemented by Buraga Balaji, who is not only pursuing a Master's in Computer Science but also brings valuable industry experience from two years of working in a software company.

2 Application

Our credit card fraud detection application leverages machine learning algorithms to predict whether a given credit card transaction is fraudulent or not. Users simply input transaction details, such as transaction amount, location, and cardholder information, into the user-friendly interface provided by Gradio. Behind the scenes, our application utilizes a trained machine learning model to analyze these inputs and provide a real-time prediction of fraud likelihood.

With its intuitive design and accurate predictions, our application offers a convenient solution for financial institutions and individuals to quickly assess the risk of fraudulent transactions, thereby enhancing security and mitigating potential financial losses. Powered by Gradio's seamless integration and customizable interface, our application ensures accessibility and ease of use for users of all backgrounds, making it a valuable tool in the fight against credit card fraud.

3 Datasets

Our credit card fraud detection dataset contains various features crucial for identifying fraudulent transactions. These include `Distance_from_home` and `Distance_from_last_transaction`, which flag unusual spatial patterns indicating potential fraud. `Ratio_to_median_purchase_price` highlights abnormal spending behavior, while `Repeat_retailer` identifies irregular transaction patterns. Additionally, `Used_chip` and `Used_pin_number` provide insights into transaction security levels, and `Online_order` indicates vulnerability to online fraud. Using these features and the target variable "Fraud," our learning model is trained to accurately identify fraud, helping to improve our ability to detect and prevent fraudulent activities

4 Privacy and Security Technique

Using differential privacy methods such as the Laplace and exponential mechanisms is crucial in credit card fraud detection. These methods help keep transaction details private while still allowing for effective analysis and model training. The Laplace mechanism, a cornerstone of differential privacy, is employed to add calibrated noise to attributes such as transaction distances, purchase ratios, and transaction histories. For instance, attributes like `Distance_from_home` and `Distance_from_last_transaction`, which reveal spatial patterns and transaction frequencies, respectively, undergo noise addition proportional to their sensitivity. Similarly, features like `Ratio_to_median_purchase_price` and `Repeat_retailer`, indicative of spending behaviors and transaction regularities, receive noise to obscure sensitive information. By calibrating the noise level based on the privacy budget (ϵ), the Laplace mechanism ensures a balance between privacy protection and data utility, preventing adversaries from inferring specific transaction details while preserving the overall analysis accuracy. On the other hand, the exponential mechanism contributes to differential privacy by probabilistically selecting transactions for analysis or model training. By considering the sensitivity of each transaction attribute and its contribution to the overall privacy risk, the exponential mechanism ensures that the selected transactions provide meaningful insights while upholding privacy guarantees. This approach not only strengthens the security posture of credit card fraud detection systems but also fosters trust among stakeholders by demonstrating a commitment to

privacy preservation and regulatory compliance.

5 Risk Management

Risk management is a crucial practice for businesses to identify, assess, and mitigate potential risks that could impact their operations, finances, or reputation. It involves systematically analyzing various risk factors, such as market volatility, regulatory changes, cybersecurity threats, and operational failures. By proactively identifying risks, businesses can develop strategies and controls to mitigate their impact and enhance resilience. This may include implementing risk monitoring systems, diversifying investments, establishing contingency plans, and ensuring compliance with relevant regulations. Effective risk management not only helps minimize potential losses but also enables businesses to seize opportunities and achieve their objectives with greater confidence and stability.

6 Code Implementation

In this project, we aim to enhance data privacy in credit card fraud detection through the implementation of differential privacy techniques. Specifically, we utilize Laplace and Exponential mechanisms to ensure the confidentiality of sensitive information while maintaining the utility of the data for analysis. The following code presents our implementation, demonstrating how these mechanisms can be effectively applied to protect privacy in fraud detection scenarios.

```
[2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings;
warnings.filterwarnings('ignore')
```

Figure.1. Importing all libraries.

After importing all necessary libraries read the data which is in .csv format and further perform EDA on that for further development.

```
[6]: cols = ['repeat_retailer', 'used_chip', 'used_pin_number', 'online_order']
fig = plt.figure(figsize=(7,5))
for index, col in enumerate(cols):
    plt.subplot(2,2, index+1)
    sns.countplot(data = cred_data, x=cred_data[col])
fig.tight_layout(pad=1.0)
```

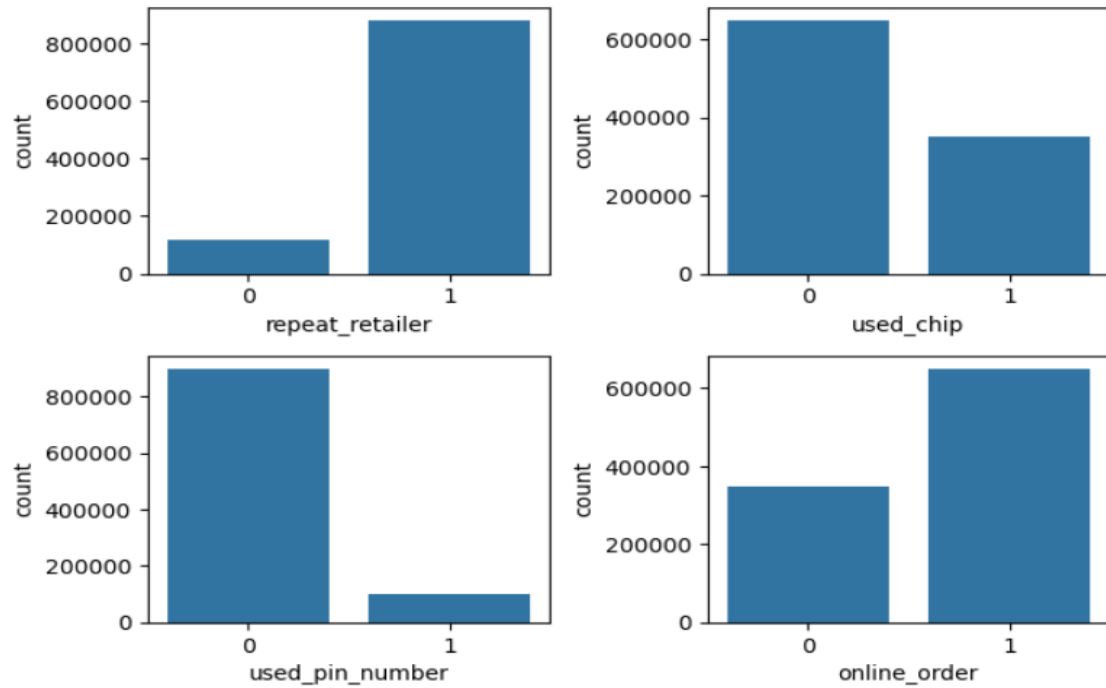


Figure 2: Analyzing the count of different transaction methods.

Now, implementing the Differential privacy to this transaction method by using the Laplace mechanism and Exponential mechanism to analyze and determine the utility score, threshold percentage and mean squared error which mechanism is performing better.

```
[8]: # Defining class Differential Privacy
class DifferentialPrivacy:
    def __init__(self, epsilon):
        self.epsilon = epsilon

    # Initializing function laplace mechanism
    def laplace_mechanism(self, data):
        sensitivity = 1
        beta = sensitivity / self.epsilon
        noise_shape = data.shape
        noise = np.random.laplace(0, beta, noise_shape)
        return data + noise

    # Initializing function exponential mechanism
    def exponential_mechanism(self, data, k=1):
        scores = np.random.exponential(scale=1/self.epsilon, size=len(data))
        selected_indices = np.argsort(scores)[:k]
        return data[selected_indices]

[9]: privacy_list = ['repeat_retailer', 'used_chip', 'used_pin_number', 'online_order'] # containing the specified attributes

# Specifying the privacy (epsilon) value
epsilon = 3.0
dp = DifferentialPrivacy(epsilon) # Initializing the Differential Privacy object with epsilon value

# Applying differential privacy to each attributes specify in the privacy list
for list in privacy_list:
    privacy_data = cred_data[list].values
    noisy_privacy_data = dp.laplace_mechanism(privacy_data) # Applying the Laplace mechanism to the attribute's data
    cred_data[list] = noisy_privacy_data # Updating the data with the noisy attribute data
```

Figure 3: Differential privacy with laplace and exponential mechanism

Now, initializing the class of Differential privacy, which encapsulates the methods for implementing differential privacy mechanisms, initializes an object with a specified privacy parameter called epsilon. The laplace mechanism adds laplace noise to the input data array, ensuring differential privacy with respect to the specified epsilon value. The exponential mechanism method implements selectin the k elements from the data array based on their scores generated using exponential distribution, while maintaining privacy with respect to epsilon. This code provides a foundation for incorporating differential privacy techniques into credit card fraud detection algorithms.

```
[10]: privacy_list = ['repeat_retailer', 'used_chip', 'used_pin_number', 'online_order']
for list1 in privacy_list:
    privacy_data1 = cred_data[list].values
    epsilon = 1.0
    dp = DifferentialPrivacy(epsilon)
    noisy_privacy_data1 = dp.exponential_mechanism(privacy_data1, k=1000000)
    cred_data[list] = noisy_privacy_data1

[11]: print("Original Privacy Data:", privacy_data)
print("Noisy Privacy Data:", noisy_privacy_data)

Original Privacy Data: [0 0 1 ... 1 1 1]
Noisy Privacy Data: [ 0.69485031 -0.27866836  1.04844276 ...  0.94391312  1.55870608
 0.91187657]
```

Figure 5: Original Data and Noisy Privacy Data

The code iterates through a list of privacy-sensitive attributes in a dataset related to the different methods of transactions. For each attribute, it extracts the corresponding data and initializes a differential privacy object where this can be applied for both laplace and exponential mechanism. Finally, it updates the original dataset with the noisy privacy-enhanced data for the current attribute.

```
[12]: def mechanism_score(original_data, noisy_data):
        scores = np.exp(-np.abs(original_data - noisy_data))
        probabilities = scores / np.sum(scores)
        threshold = 0.5
        utility_score = np.sum(probabilities[np.abs(original_data - noisy_data) < threshold])
        return utility_score

        utility_score = mechanism_score(privacy_data, noisy_privacy_data)
        print("Utility Score:", utility_score)

        Utility Score: 0.8645694443030112

[13]: def mechanism_score(original_data, noisy_data):
        scores = np.exp(-np.abs(original_data - noisy_data))
        probabilities = scores / np.sum(scores)
        threshold = 0.5
        utility_score = np.sum(probabilities[np.abs(original_data - noisy_data) < threshold])
        return utility_score

        utility_score = mechanism_score(privacy_data1, noisy_privacy_data1)
        print("Utility Score:", utility_score)

        Utility Score: 0.608261027428557

[14]: from sklearn.metrics import mean_squared_error
        mse = mean_squared_error(privacy_data, noisy_privacy_data)
        print("Mean Squared Error (MSE):", mse)

        Mean Squared Error (MSE): 0.22238548275225078
```

Figure 6: Utility score and mean squared error

The scores represent the utility and accuracy of the differential privacy mechanism applied to the dataset. A utility score of 86% for Laplace mechanism suggest that it preserves the data quality better compared to the exponential mechanism, which achieves a lower utility score of 60%. The mean squared error (MSE) of 0.22 for Laplace indicates the average difference between the original and noisy data, with lower values indicating better preservation of data fidelity. Overall, these metrics provides insights into the trade-offs between privacy preservation and data utility in the context of credit card fraud detection.

```
[53]: threshold = 0.5
        preserved_ratio = np.mean(np.abs(privacy_data - noisy_privacy_data) < threshold)
        preserved_percentage = preserved_ratio * 100
        print("Preserved Percentage:", preserved_percentage)

        Preserved Percentage: 77.65379999999999
```

Figure 7: Preserved percentage

Achieving a preserved percentage of 77% with the laplace mechanism indicates that approximately 77% of the original data's information is retained after applying differential privacy. This suggests a reasonable balance between privacy protection and data utility, demonstrating the effectiveness of the Laplace mechanism in preserving the essential features of the dataset for credit-card fraud detection.

```
[29]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
      epsilon = 0.1
      dp = DifferentialPrivacy(epsilon)
      x_train_noisy = dp.laplace_mechanism(x_train.values)
      x_train_noisy

[29]: array([[ -15.50122231,  -4.3107259 , -13.02157619, ..., -21.28845613,
          10.66270494,  2.95643611],
 [239.88472099, -2.58019928,  8.87468054, ..., -9.92180029,
        -2.32584191, -5.64494043],
 [ 72.06085068, -13.24840112,  2.97251746, ..., -20.58403007,
        6.52623691,  2.40451207],
 ...,
 [ 70.01217592,  3.49441987, 30.64304527, ..., -5.41426858,
        -1.57457714,  5.70558714],
 [ 99.68990551, -13.74246413, -1.07409915, ...,  6.40710086,
        -5.29854445, -0.32022062],
 [ 24.57872364,  1.57140867, 11.69860488, ..., 30.20864693,
        0.50313466, -2.57264295]])

[30]: x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(x, y, test_size=0.2, random_state=42)
      epsilon = 0.1
      dp = DifferentialPrivacy(epsilon)
      x_train_noisy_1 = dp.exponential_mechanism(x_train_1.values, k=1000000)
      x_train_noisy_1

[30]: array([[ 4.99728230e-01,  6.94497333e+00,  4.41041466e-01, ...,
        -1.03983565e-03, -5.66054352e-02,  2.54771959e-01],
 [ 5.04896600e+01,  5.32157647e+01,  3.95783836e+00, ...,
        -6.31315838e-02,  1.67234735e-01,  1.11633708e+00],
 [ 8.86182400e+00,  4.23964491e-01,  2.89781356e+00, ...,
        -2.37423982e-01, -8.87960192e-02,  8.54865568e-01],
```

Figure 8: Splitting the dataset into training and testing

Split of dataset into 80% of training and 20% of testing data and implanting those mechanism for all the attributes of the dataset and check the accuracy by implementing Machine Learning algorithms to check which is performing better and which is more conveyable.

MODEL PREDICTION ON LAPLACE MECHANISM

```
[36]: # RANDOM FOREST
rf = RandomForestClassifier()
clf_rf = rf.fit(x_train_noisy, y_train)
y_pred = clf_rf.predict(x_test)
acc2 = accuracy_score(y_test, y_pred)
print('--RANDOM FOREST CLASSIFIER ACCURACY--: ', acc2)

--RANDOM FOREST CLASSIFIER ACCURACY--: 0.7785378132866367

[37]: # DECISION TREE CLASSIFIER
dt = DecisionTreeClassifier()
clf_dt = dt.fit(x_train_noisy, y_train)
y_pred = clf_dt.predict(x_test)
acc3 = accuracy_score(y_test, y_pred)
print('--DECISION TREE ACCURACY--: ', acc3)

--DECISION TREE ACCURACY--: 0.6153004268359418

[38]: # LOGISTIC REGRESSION
log = LogisticRegression()
clf_log = log.fit(x_train_noisy, y_train)
y_pred = clf_log.predict(x_test)
acc3 = accuracy_score(y_test, y_pred)
print('--LOGISTIC REGRESSION ACCURACY--: ', acc3)

--LOGISTIC REGRESSION ACCURACY--: 0.672485498522491
```

MODEL PREDICTION ON EXPONENTIAL MECHANISM

```
[39]: rf = RandomForestClassifier()
clf_rf = rf.fit(x_train_noisy_1, y_train_1)
y_pred_1 = clf_rf.predict(x_test_1)
acc2 = accuracy_score(y_test_1, y_pred1)
print('--RANDOM FOREST CLASSIFIER ACCURACY--: ', acc2)

--RANDOM FOREST CLASSIFIER ACCURACY--: 0.3245594834190653

[40]: dt = DecisionTreeClassifier()
clf_dt = dt.fit(x_train_noisy_1, y_train_1)
y_pred_1 = clf_dt.predict(x_test_1)
acc3 = accuracy_score(y_test_1, y_pred1)
print('--DECISION TREE ACCURACY--: ', acc3)

--DECISION TREE ACCURACY--: 0.5044872496443034

[42]: log = LogisticRegression()
clf_log = log.fit(x_train_noisy_1, y_train_1)
y_pred_1 = clf_log.predict(x_test_1)
acc3 = accuracy_score(y_test_1, y_pred1)
print('--LOGISTIC REGRESSION ACCURACY--: ', acc3)

--LOGISTIC REGRESSION ACCURACY--: 0.3245594834190653
```

Figure 9: Predictive performance of each algorithm based on laplace and exponential mechanism.

Implementing the Random Forest Classifier, Decision Tree and Logistic Regression for their effective predictive performance on dataset and obtaining the results based on the mechanism we derived in the code and testing them on this dataset which is performing effectively and is convenient to this data and project.

```
[37]: import gradio as gr
fraud_detection_model = load_fraud_detection_model()

input_columns = ['distance_from_home', 'distance_from_last_transaction',
                 'ratio_to_median_purchase_price', 'repeat_retailer',
                 'used_chip', 'used_pin_number', 'online_order']
inputs = [gr.Textbox(label=col.replace('_', ' ').title()) for col in input_columns]
outputs = gr.Textbox(label="Fraud Prediction")

def predict_fraud_wrapper(*input_values):
    input_data = np.array(input_values).reshape(1, -1)
    prediction = predict_fraud(fraud_detection_model, input_data)
    result_text = "Fraud Detected" if prediction == 1 else "No Fraud"
    speak_text(result_text)
    return result_text

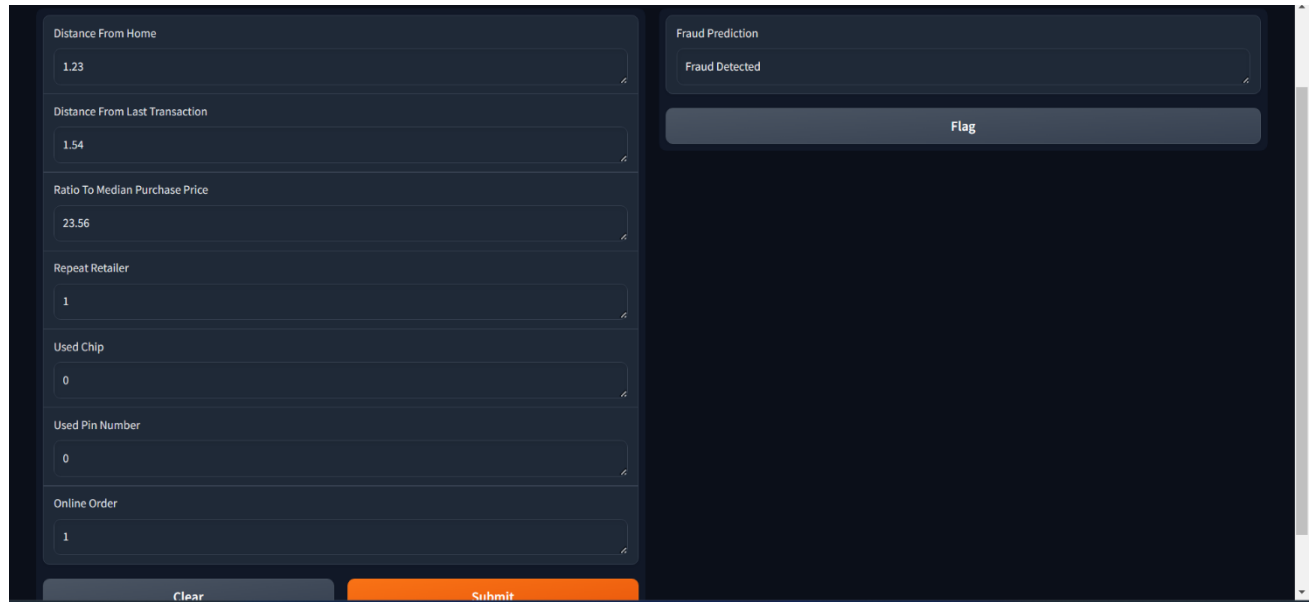
interface = gr.Interface(fn=predict_fraud_wrapper, inputs=inputs, outputs=outputs,
                        title="Credit Card Fraud Detector", description="Predict fraud based on credit card features.")
interface.launch()

Running on local URL: http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

Figure 10: Application launch by using Gradio App

Finally, we develop the application of gradio interface resulting in the detection of whether fraud happened or not based on the attributes. That results how well the model is performing and how the differential privacy plays the vital role in this project.



The screenshot displays a Gradio web interface for a fraud detection application. On the left, a vertical stack of input fields is provided for various attributes: 'Distance From Home' (1.23), 'Distance From Last Transaction' (1.54), 'Ratio To Median Purchase Price' (23.56), 'Repeat Retailer' (1), 'Used Chip' (0), 'Used Pin Number' (0), and 'Online Order' (1). Each input field has a small icon in its top right corner. At the bottom of this stack are two buttons: a grey 'Clear' button and an orange 'Submit' button. On the right side of the interface, there is a 'Fraud Prediction' section containing a text box labeled 'Fraud Detected' and a large, light blue button labeled 'Flag'.

Figure 11: Gradio App

7 Testing

Testing for the credit card fraud detection project with the integration of differential privacy involves several critical steps to ensure the efficacy of privacy-preserving measures and the accuracy of fraud detection algorithms. Firstly, the verification of differential privacy guarantees is essential, confirming that the added noise adheres to specified privacy budgets. Utility assessment follows, evaluating the impact of noise on data quality through metrics like Mean Squared Error or Preserved Percentage. Subsequently, the performance of fraud detection models trained on differentially private data is scrutinized using standard metrics such as precision, recall, and F1-score. Privacy-utility trade-off analysis is conducted to strike a balance between privacy protection and data utility, often involving adjustments to privacy budgets. Robustness testing is imperative, assessing mechanisms' resilience against privacy attacks and data breaches. User feedback is solicited to ensure acceptability and alignment with privacy expectations. Documentation of testing procedures and results facilitates transparency and accountability. Continuous monitoring and improvement mechanisms are established for ongoing refinement based on evolving requirements. Lastly, ethical considerations, including potential biases and unintended consequences, are addressed to uphold ethical standards and societal trust.

8 Results

In this project, the difference in accuracy scores between Laplace and Exponential mechanisms across various machine learning models reflects their impact on the predictive performance of each algorithm. In Laplace mechanism, Random Forest achieves the highest accuracy of 0.77, indicating robust performance in preserving data utility while maintaining privacy. Decision Tree follows with an accuracy of 0.61, demonstrating some reduction in performance compared to Random Forest. Logistic Regression exhibits an accuracy of 0.67, indicating moderate performance. Conversely, the Exponential mechanism yields lower accuracy scores across all models, with each scoring approximately 0.50. This suggests that the Exponential mechanism introduces more noise into the data, compromising the predictive capability of the machine learning models. As a result, the models trained on data perturbed by the Exponential mechanism are less effective in capturing the underlying patterns and making accurate predictions compared to those trained on data perturbed by the Laplace mechanism. These results highlight the trade-off between privacy preservation and predictive

performance. While the Laplace mechanism offers a better balance between privacy and utility, the Exponential mechanism sacrifices more utility for stronger privacy guarantees, resulting in reduced predictive accuracy across machine learning models. The AUC scores summarize the discriminatory power of each model. Random Forest performs the best with an AUC of 0.77, indicating strong classification ability. Logistic Regression follows with an AUC of 0.67, suggesting moderate performance. The Decision Tree model trails behind with the lowest AUC of 0.61, indicating comparatively weaker discriminatory power. In essence, these scores highlight the varying effectiveness of each model in distinguishing between positive and negative classes based on their ROC curves.

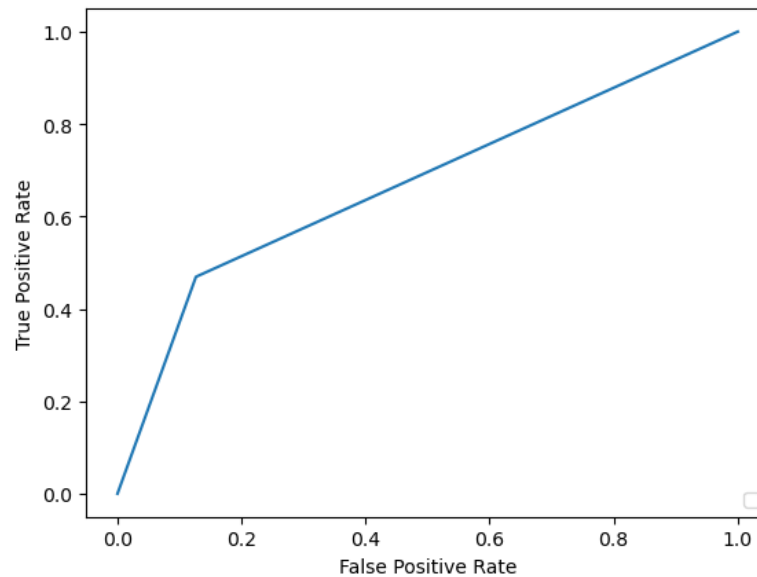


Figure 12: Graph obtained on model performance

As part of application by integrating all the concepts into our project and finally deploying it in gradio app and testing the app with the user input and checking whether fraud happened or not.

9 Summary

In this credit card fraud detection project, the integration of differential privacy introduces a robust layer of privacy protection while maintaining the accuracy of fraud detection algorithms. Leveraging differential privacy mechanisms such as the Laplace and exponential mechanisms, sensitive transaction details are safeguarded through the addition of calibrated noise, preventing unauthorized access and inference of individual transaction information. Testing procedures involve verifying the adherence to privacy guarantees, assessing data utility, and evaluating the performance of fraud detection models trained on differentially private data. Privacy-utility trade-offs are carefully analyzed to strike a balance between privacy protection and data utility, with adjustments made to privacy budgets as necessary. Throughout the project, user feedback, documentation, and continuous monitoring facilitate transparency, accountability, and ongoing refinement of privacy protection measures. Ethical considerations, including biases and unintended consequences, are addressed to uphold ethical standards and societal trust in the project's outcomes. Overall, the integration of differential privacy enhances the security and privacy of credit card transaction data, ensuring compliance with regulations and fostering trust among stakeholders.

References

- [1] Raj, S. Benson Edwin, and A. Annie Portia. "Analysis on credit card fraud detection methods." In 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET), pp. 152- 156. IEEE, 2011.
- [2] Ghosh, Sushmito, and Douglas L. Reilly. "Credit card fraud detection with a neural network." In System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on, vol. 3, pp. 621- 630. IEEE, 1994.
- [3] Chaudhary, Khyati, Jyoti Yadav, and Bhawna Mallick. "A review of fraud detection techniques: Credit card." International Journal of Computer Applications 45, no. 1 (2012): 39-44.
- [4] Srivastava, Abhinav, Amlan Kundu, Shamik Sural, and Arun Majumdar. "Credit card fraud detection using hidden Markov model." IEEE Transactions on Dependable and secure computing 5, no. 1 (2008): 37- 48.
- [5] Awoyemi, John O., Adebayo O. Adetunmbi, and Samuel A. Oluwadare. "Credit card fraud detection using machine learning techniques: A comparative analysis." In 2017 international conference on computing networking and Informatics (ICCNi), pp. 1-9. IEEE, 2017.
- [6] Sahin, Yusuf, and Ekrem Duman. "Detecting credit card fraud by ANN and logistic regression." In 2011 international symposium on Innovations in intelligent systems and Applications, pp. 315-319. IEEE, 2011.
- [7] Kiran, Sai, Jyoti Guru, Rishabh Kumar, Naveen Kumar, Deepak Katariya, and Maheshwar Sharma. "Credit card fraud detection using Naïve Bayes model based and KNN classifier." International Journal of Advance Research, Ideas, and Innovations in Technology 4, no. 3 (2018): 44.
- [8] Husejinovic, Admel. "Credit card fraud detection using naive Bayesian and c4. 5 decision tree classifiers." Husejinovic, A.(2020). Credit card fraud detection using naive Bayesian and C 4 (2020): 1-5.
- [9] Saheed, Yakub K., Moshood A. Hambali, Micheal O. Arowolo, and Yinusa A. Olasupo. "Application of GA feature selection on Naive Bayes, random forest and SVM for credit card

fraud detection." In 2020 international conference on decision aid sciences and Application (DASA), pp. 1091-1097. IEEE, 2020.