Credit Card Fraud Detection

A MACHINE LEARNING APPROACH TO COMBAT FINANCIAL CRIME

Presented by
Yara Lihawa
14 May 2025

Background & Problem Statement

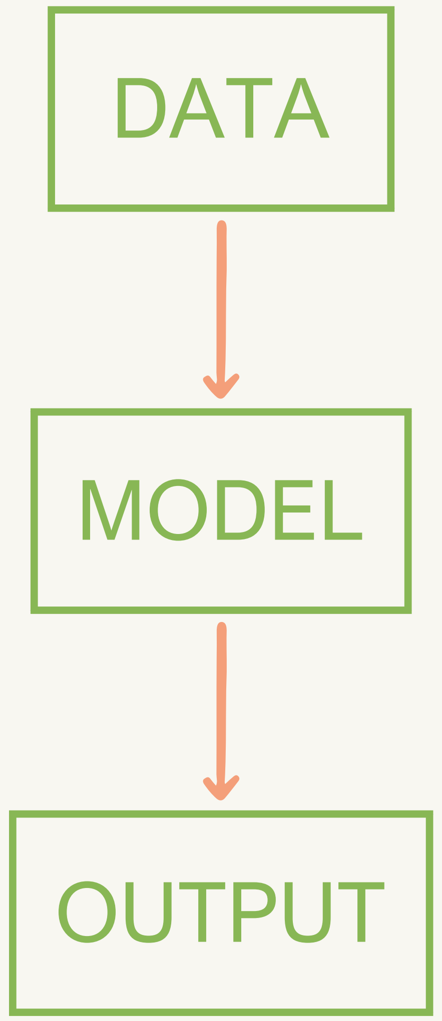
Background

Credit card fraud is a growing concern in the digital era. With millions of transactions happening every day, detecting fraudulent activity quickly and accurately is more important than ever. Traditional fraud detection systems rely heavily on static rules, which often fail to adapt to new and evolving fraud techniques. This creates a demand for smarter, data-driven solutions.

Problem Statement

The rise in digital payments has increased the risk of credit card fraud.
Manual or rule-based systems struggle to adapt to evolving fraud tactics.
Fraudulent transactions represent only 0.17% of all cases, making detection extremely difficult.

Objectives & Scope



Why This Project?

Credit card fraud is a growing concern in the digital era. With millions of transactions happening every day, detecting fraudulent activity quickly and accurately is more important than ever. Traditional fraud detection systems rely heavily on static rules, which often fail to adapt to new and evolving fraud techniques. This creates a demand for smarter, data-driven solutions.

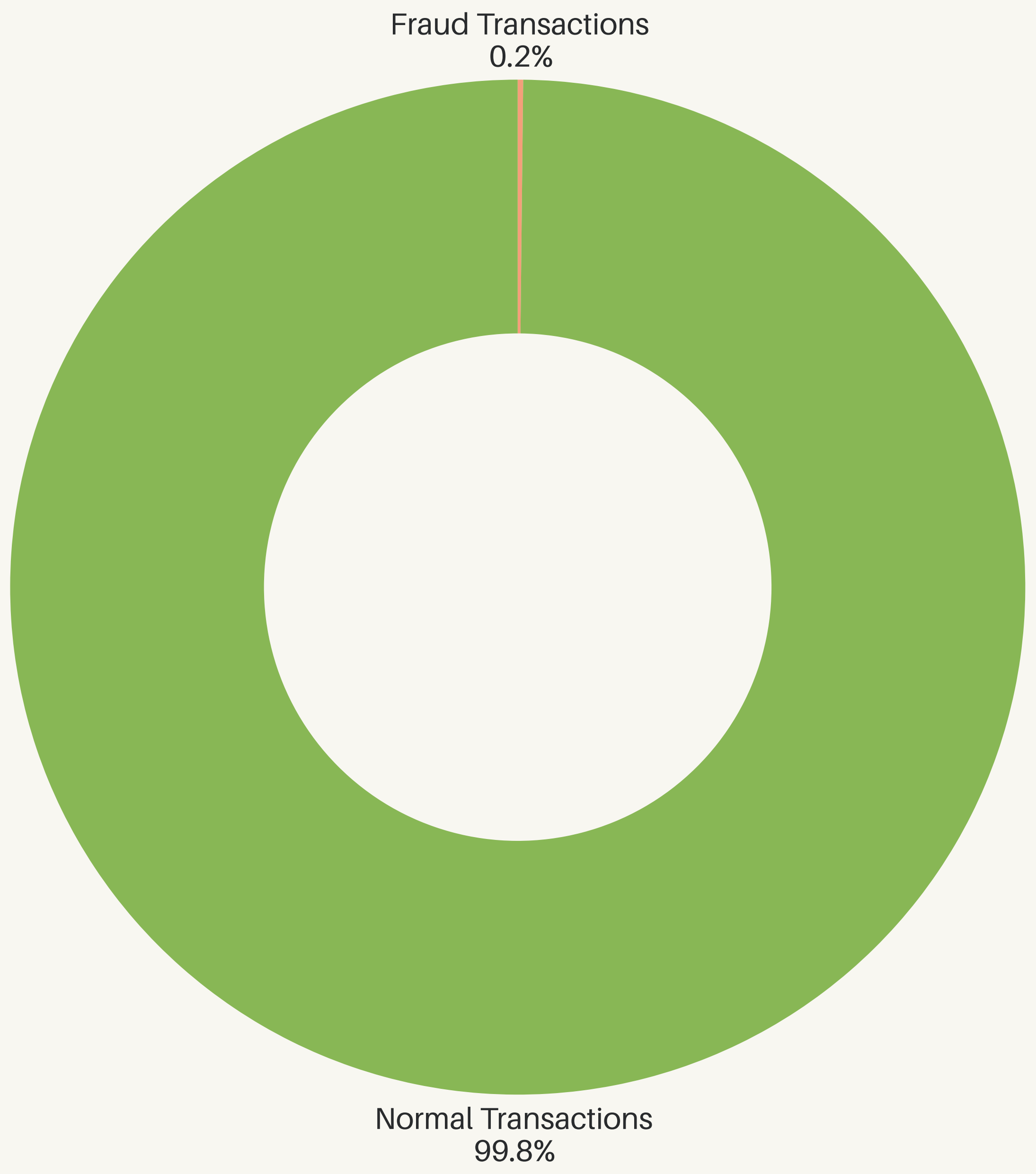
Project Objectives

- Use machine learning to detect credit card fraud.
- Handle class imbalance in the dataset.
- Focus on recall to catch the maximum number of fraudulent transactions.
- Evaluate performance and select the best approach.

Data Collection & Preperation

Data Info

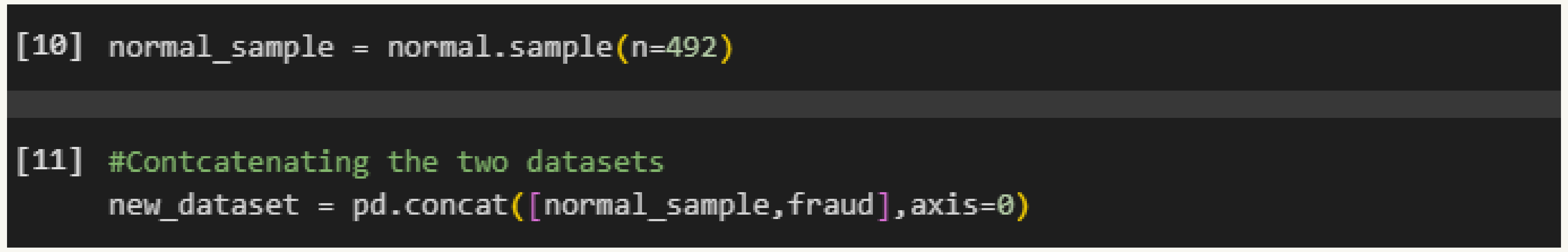
- **Rows:** 284,807
- **Fraud Cases:** 492 (~0.17%)
- **Features:**
  - V1–V28: PCA-transformed features
  - Time and Amount (raw)
- **Class:** Target (0 = Normal, 1 = Fraud)



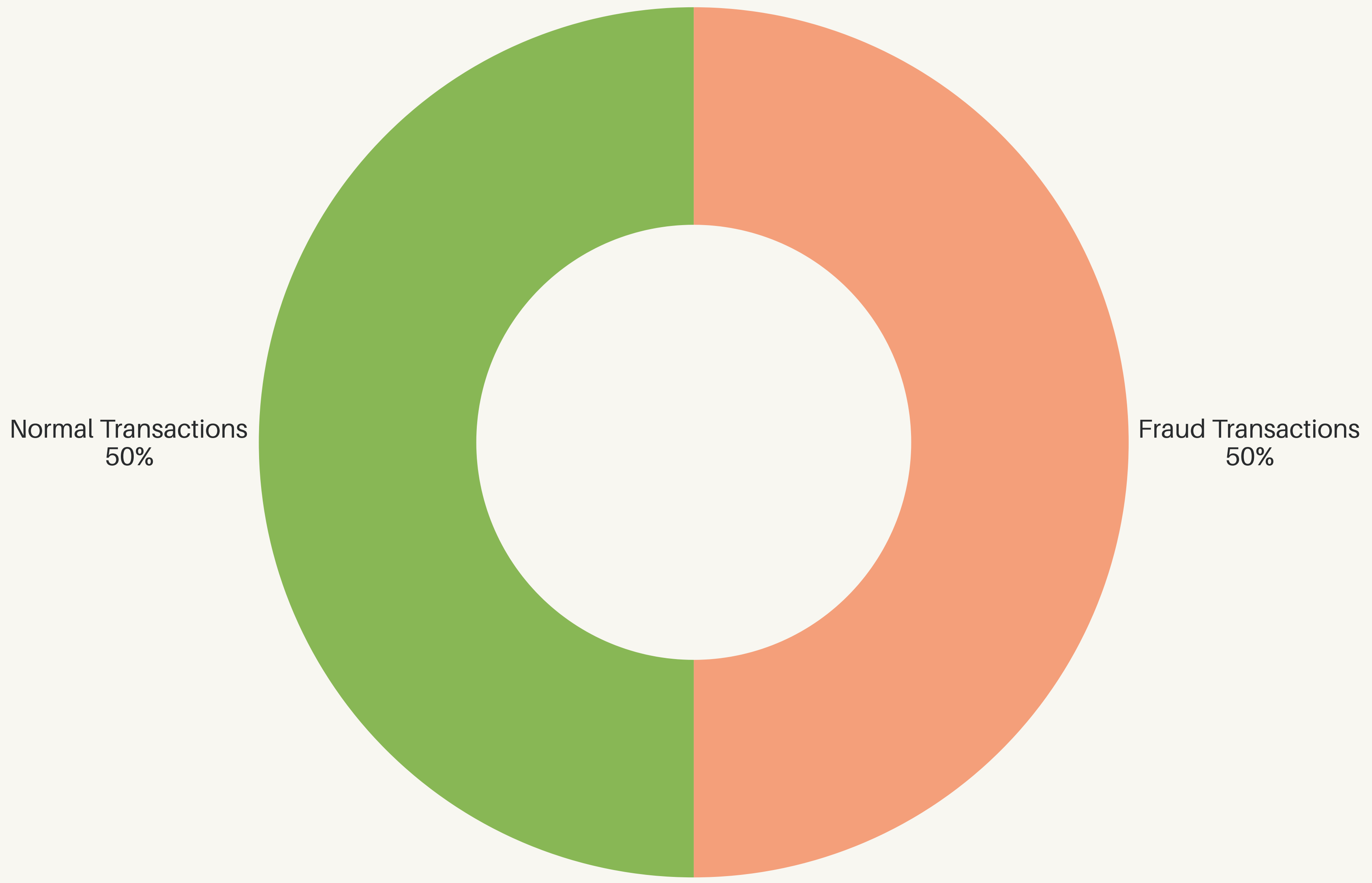
Data Collection & Preperation

Fraud cases were extremely rare, so a balanced dataset was created:

- Took all **492 fraud** cases
- Sampled **492 normal** cases randomly
- Combined into a new dataset of **984 transactions**, 50/50 balanced



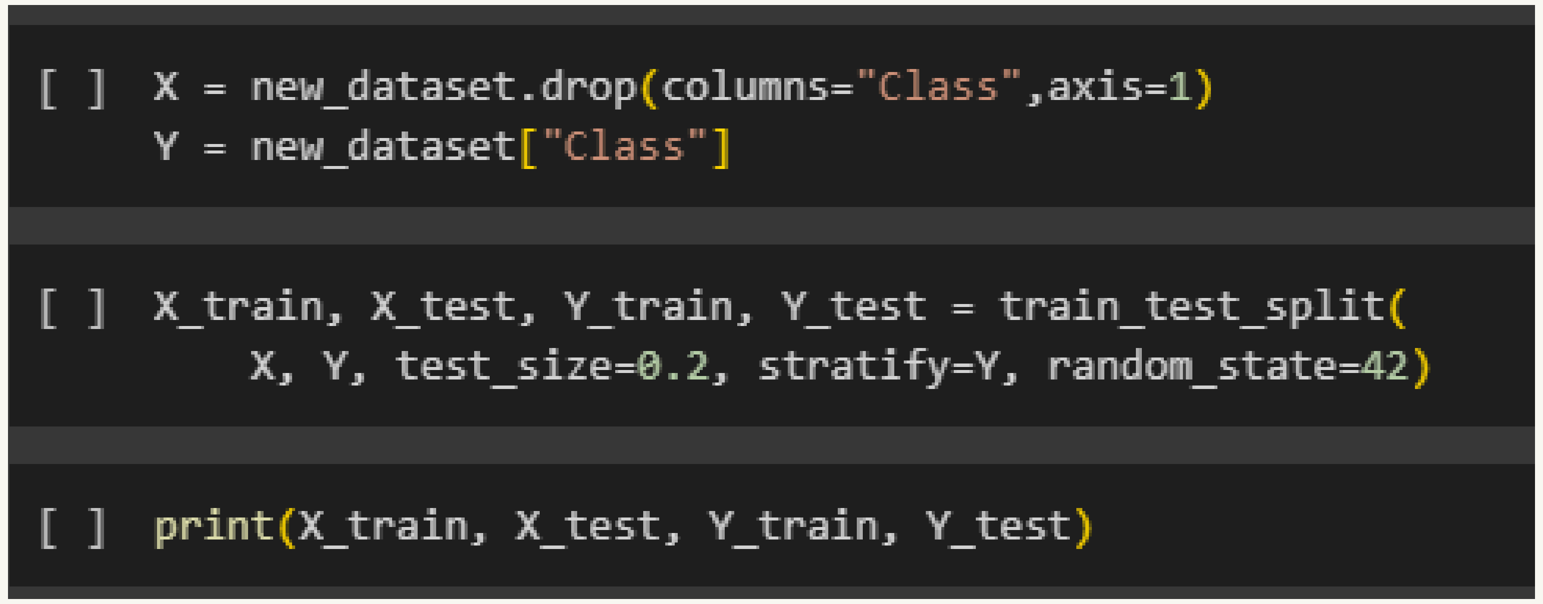
```
[10] normal_sample = normal.sample(n=492)

[11] #Contcatenating the two datasets
     new_dataset = pd.concat([normal_sample,fraud],axis=0)
```



Preprocessing

- No missing values in the dataset
- No scaling or transformation applied
- Features used as-is for model input
- Dataset ready for training in the next step

Training & Optimization



```
[ ] X = new_dataset.drop(columns="Class",axis=1)
    Y = new_dataset["Class"]
```

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(
        X, Y, test_size=0.2, stratify=Y, random_state=42)
```

```
[ ] print(X_train, X_test, Y_train, Y_test)
```

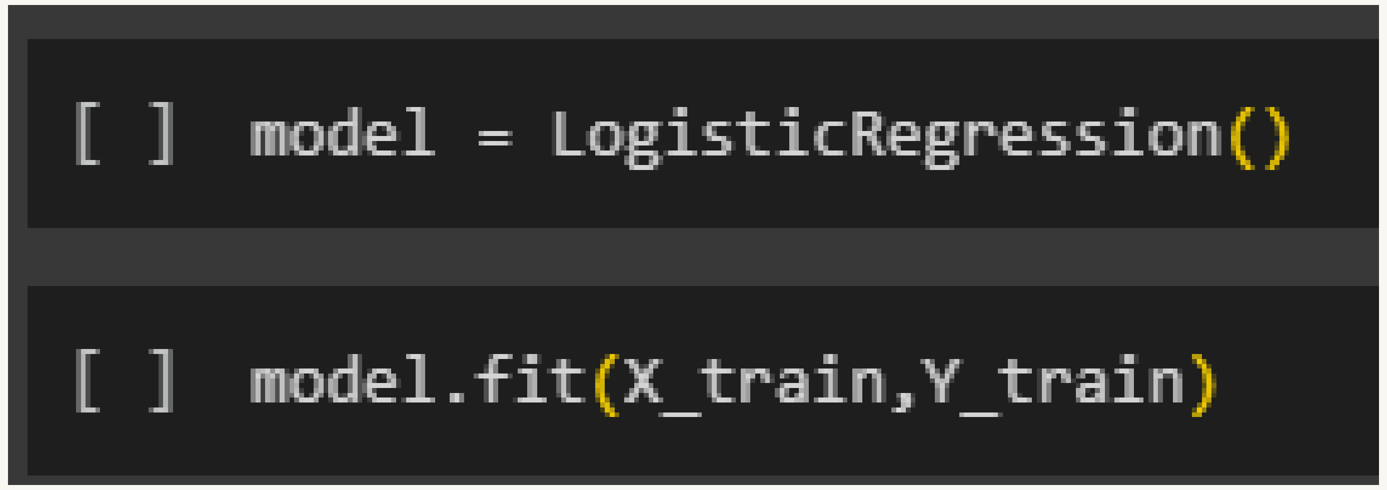
Train-Test Split

- After constructing a balanced dataset containing 492 fraud and 492 non-fraud transactions (total 984 records), we split the data for training and testing the model.
- 80% Training Data (787 records)
- 20% Testing Data (197 records)
- To ensure that both classes (fraud and non-fraud) were proportionally represented in both sets, we used stratified sampling. This helps maintain the class distribution during model evaluation.

To ensure that both classes (fraud and non-fraud) were proportionally represented in both sets, we used stratified sampling. This helps maintain the class distribution during model evaluation.

Using stratify=Y ensures that both the train and test sets have a 50/50 split between fraud and non-fraud.

Training & Optimization



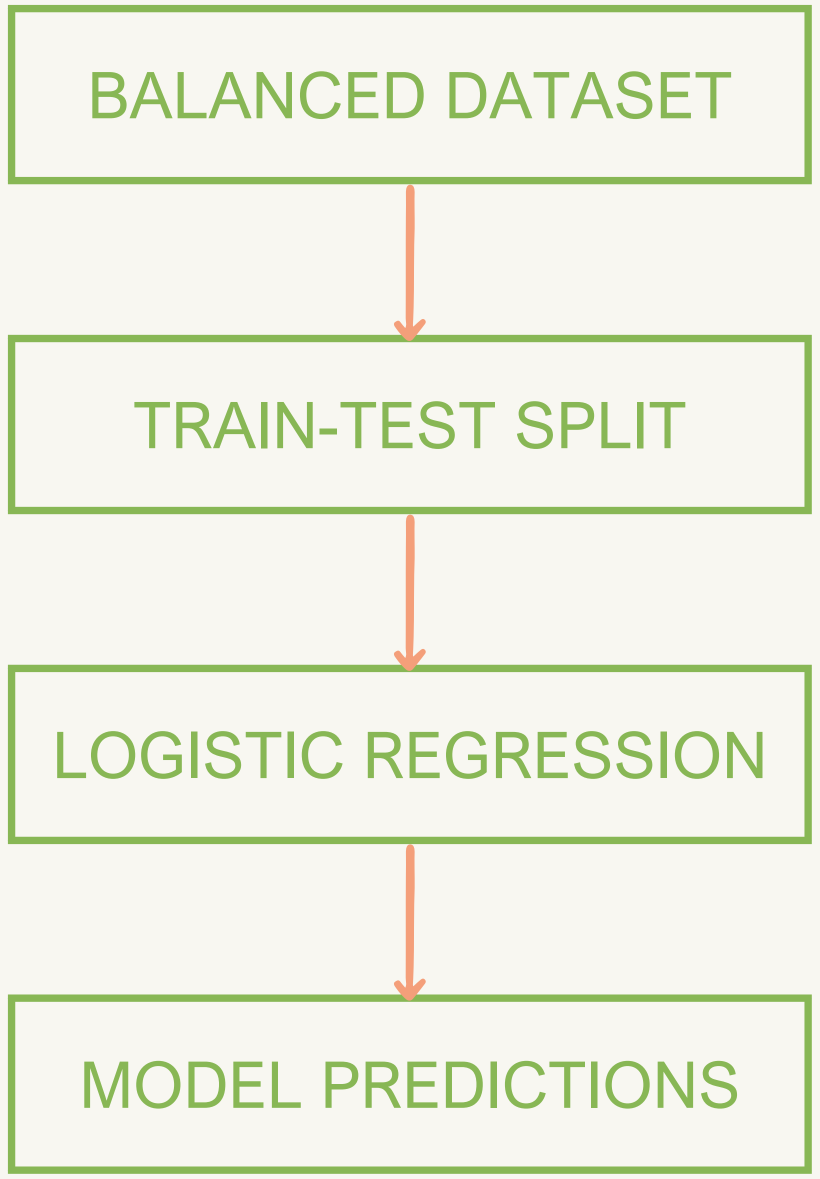
```
[ ]  model = LogisticRegression()
```

```
[ ]  model.fit(X_train,Y_train)
```

Model Selection: Logistic Regression

- I chose Logistic Regression as my initial classification model due to its:
- Simplicity and interpretability
- Fast training and low computational cost
- Ability to output probability scores, which is useful in fraud detection for setting thresholds
- Effectiveness in binary classification problems like this one

Training & Optimization

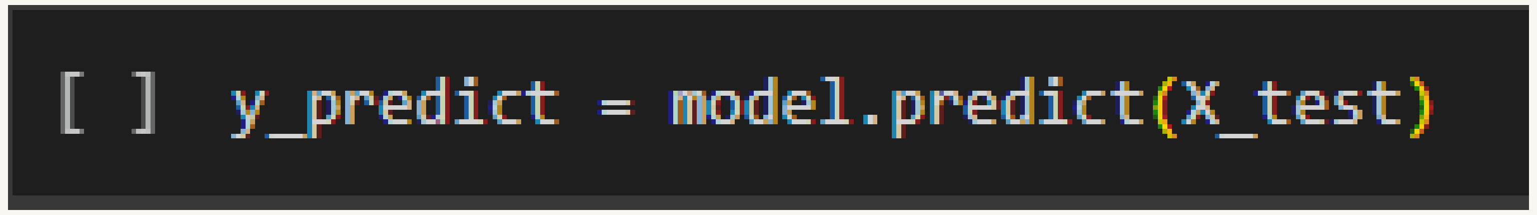


Optimization Approach

In this project:

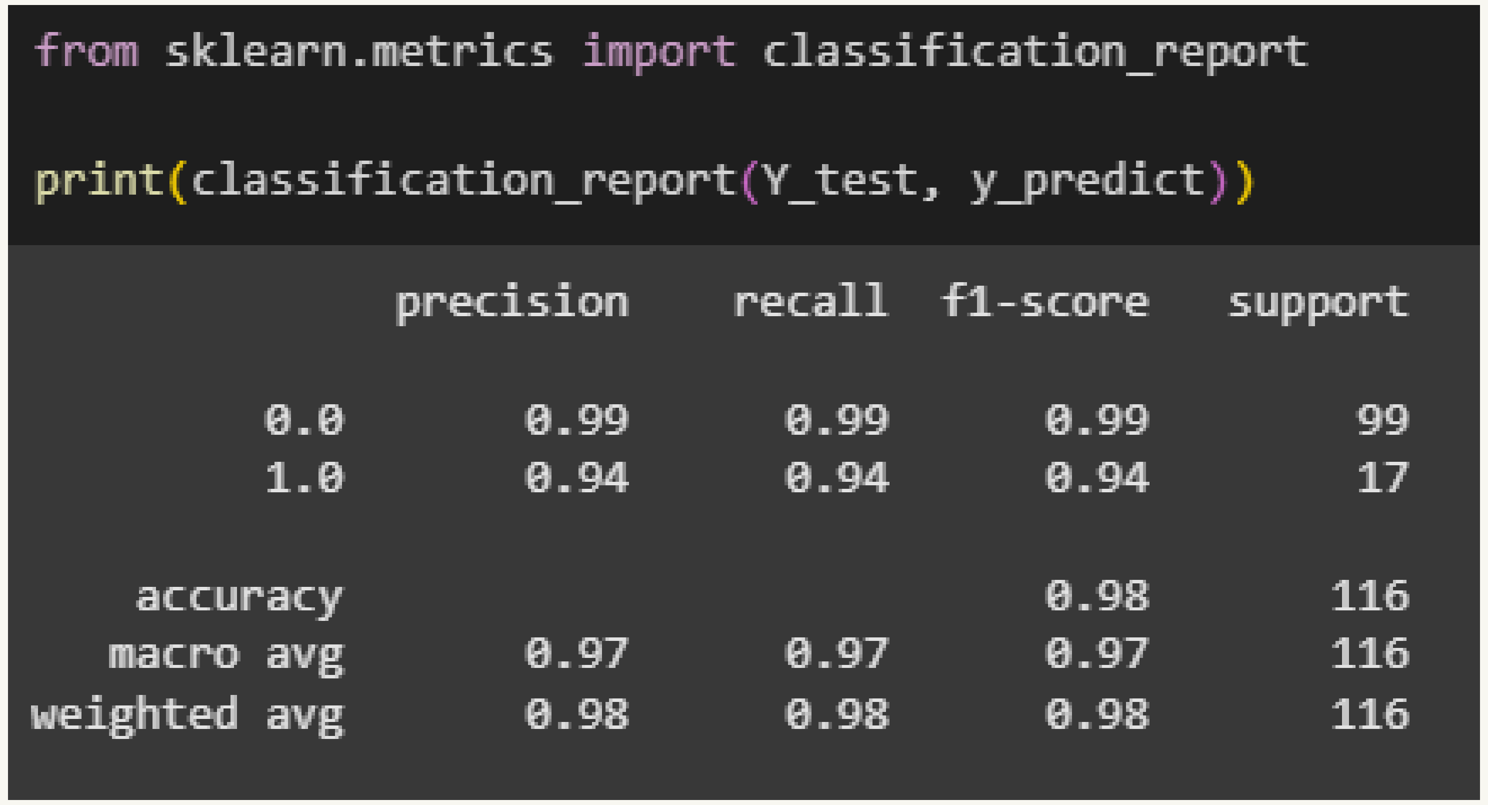
- No feature scaling or transformation was applied since most features are already PCA-transformed
- We did not apply hyperparameter tuning (like grid search or regularization changes) to keep the model simple
- Our focus was on training a working model and evaluating its base performance

The model was then used to predict outcomes on the test set:



```
[ ]  y_predict = model.predict(X_test)
```

Model Evaluation & Results



```
from sklearn.metrics import classification_report

print(classification_report(Y_test, y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99        99
         1.0       0.94      0.94      0.94        17

    accuracy                           0.98       116
   macro avg       0.97      0.97      0.97       116
weighted avg       0.98      0.98      0.98       116
```

Evaluation Metrics

After training the Logistic Regression model on a balanced dataset, we evaluated it using the test set (116 records). The results demonstrate strong overall performance, especially in identifying fraudulent transactions.

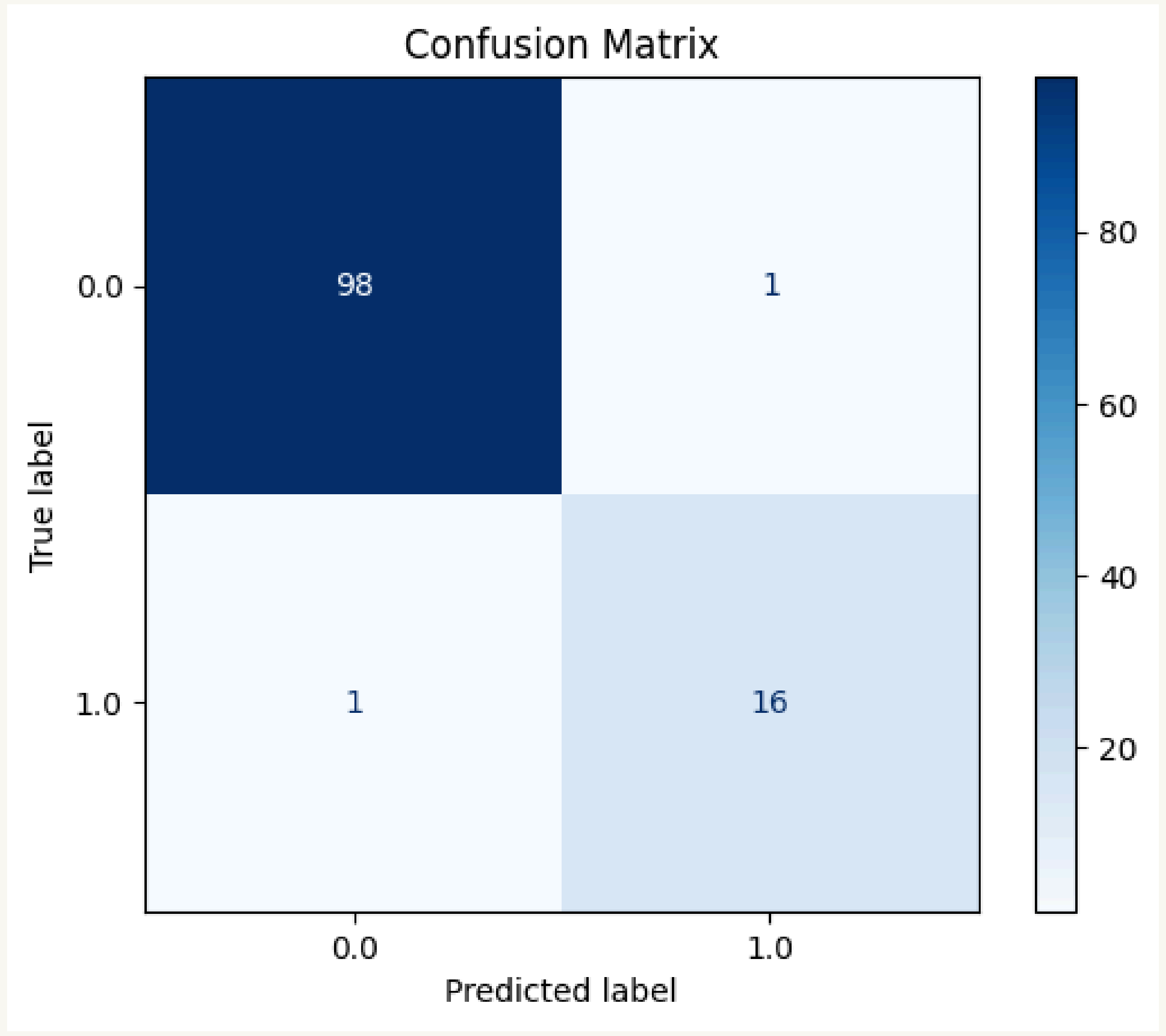
- Accuracy: 98%
- Macro Avg (balanced class view): 97% across all metrics
- Weighted Avg: 98% (adjusted by class size)

Interpretation

- High Precision (0.94) on frauds: Most flagged frauds were actually fraudulent
- High Recall (0.94): The model correctly identified 94% of actual frauds — excellent performance for detection
- F1 Score (0.94) balances the precision-recall tradeoff very well

This means the model is both accurate and aggressive in catching fraud without generating too many false alarms.

Model Evaluation & Results



Confusion Matrix

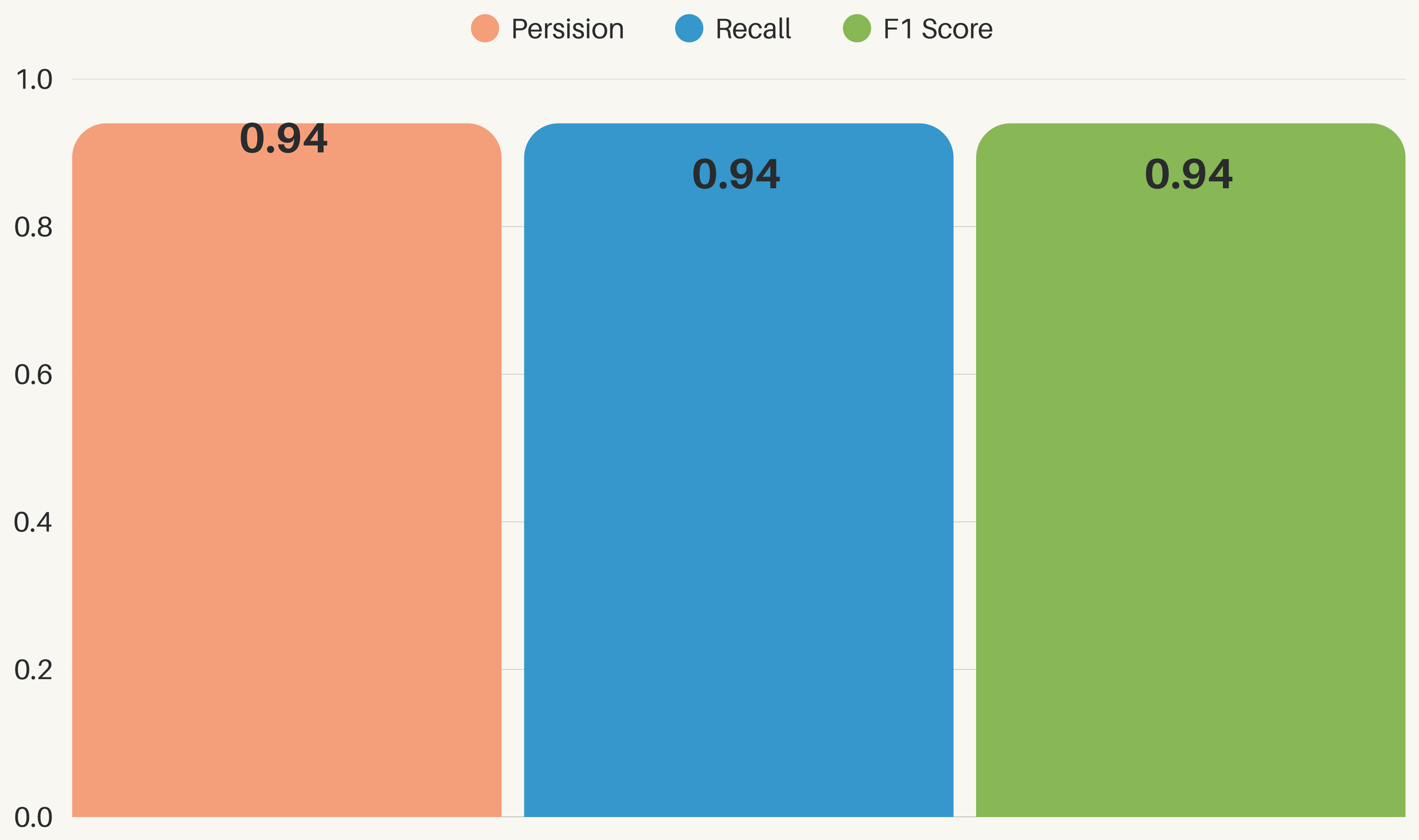
- True Positives (TP): 16
- False Negatives (FN): 1
- False Positives (FP): 1
- True Negatives (TN): 98

This shows only 2 incorrect predictions out of 116, a very strong performance.

What These Results Mean

- Extremely strong performance on both classes
- Only 1 fraud missed (FN) and 1 false alarm (FP)
- Balanced recall and precision for Class 1 (fraud), meaning the model is not only aggressive at catching fraud, but also rarely flags legit transactions by mistake
- In a real-world deployment, false positives can lead to blocked customers, so high precision helps avoid this

Conclusion



Summary of the Project

- Built a **Logistic Regression model** to detect credit card fraud using a publicly available dataset.
- Handled the **class imbalance** by creating a **balanced dataset** with equal parts fraud and non-fraud transactions.
- Achieved an **accuracy of 98%** and a **recall of 94%** for fraud cases on the test set — demonstrating that the model is highly effective in identifying fraudulent behavior.

Key Takeaways

- **Balanced data** (through strategic sampling) allowed us to train an effective model without complex resampling techniques like SMOTE.
- **Logistic Regression**, despite being a simple model, performed exceptionally well — showing that baseline models can be powerful with the right data preparation.
- **High recall and precision** make the model practical for fraud detection, reducing both **missed frauds** and **false alarms.**

Conclusion

Limitations

- The model was trained on a small subset of the original dataset (balanced to only 984 samples). In practice, models should be trained on larger, real-world imbalanced data using techniques like:
  - SMOTE / ADASYN oversampling
  - Cost-sensitive learning
  - Advanced ensemble models (e.g., XGBoost, LightGBM)

Future Work

- Scale up the model using the full, imbalanced dataset and apply **oversampling** or **weighting.**
- Experiment with **more advanced algorithms** (e.g., Random Forest, XGBoost).
- Integrate the model into a **real-time monitoring system** using Python + Flask or Streamlit.
- Add dashboards and alert systems for practical deployment.

Real-World Impact

A well-performing fraud detection model like this can:

- Save financial institutions **millions in losses**
- Increase **customer trust**
- Automate and **accelerate risk mitigation**

Contact Me

Thank you for your attention!
Feel free to reach out if you have any questions, feedback, or would like to collaborate.

Let’s connect.

Yara Lihawa
✉ **yaralihawa@gmail.com**
Yara Lihawa