1. Why is the dataset known to be imbalanced?

The credit-card fraud dataset is considered **imbalanced** because the number of fraudulent transactions is extremely small compared to the number of valid transactions.
Typically:

- **Valid transactions:** ~99.8%
- **Fraudulent transactions:** ~0.2%

This means the dataset contains far more non-fraud cases than fraud cases.
Such imbalance is a challenge for machine learning because many models tend to predict only the majority of classes. For example, a model could predict "valid" for every transaction and still achieve 99% accuracy while completely failing to detect fraud.
Therefore, the dataset is classified as **highly imbalanced**, requiring specific techniques such as under sampling, oversampling, or careful metric selection.

2. What amount-related details are used to label a transaction as "valid" or "fraud"?

In this dataset, fraud labels are **not determined** by the transaction amount. Instead, the dataset contains a predefined column:

- **Class = 0:** Valid transaction
- **Class = 1:** Fraud transaction

These labels were assigned by the credit card company based on internal fraud detection systems.
The **Amount** feature is used only as part of the model input — it is *not* used to decide whether a transaction is fraud.
Therefore, the amount does not determine labeling; the labels are already provided.

3. What is shown on the Correlation Matrix?

A correlation matrix displays the **relationship between all pairs of features** in the dataset, with values ranging from:

- **+1** → Strong positive correlation
- **–1** → Strong negative correlation
- **0** → No correlation

In the fraud dataset:

- Most features (V1–V28) were generated by **Principal Component Analysis (PCA)**, so correlations among them are often low.
- The correlation matrix helps identify:
    - Features that correlate strongly with fraud
    - Redundant or non-informative features
    - Relationships among numerical variables

It provides an overview of how features interact and whether they contribute meaningfully to distinguishing fraud from non-fraud.

4. Explanation of actions performed in Step 6: "Data Preparation."

Step 6 includes several important actions to prepare the dataset for machine learning:

a. Feature Selection

All PCA-generated features (V1–V28), along with Amount and Time, are selected as model inputs.

b. Feature Scaling

- PCA features are already scaled.
- Amount and sometimes Time must be normalized (e.g., using StandardScaler) to ensure equal influence.

c. Splitting the Dataset

The data is divided into:

- **Training set:** approximately 80%
- **Testing set:** approximately 20%

This allows model evaluation on unseen data.

d. Handling Imbalanced Data

Options include:

- Undersampling majority class
- Oversampling minority class
- Using class-weight adjustments
- Keeping data as is but using proper evaluation metrics

These techniques help the model learn from the limited fraud examples.

e. Creating Feature and Label Sets

- **X:** All selected feature columns
- **y:** The Class column (0 = valid, 1 = fraud)

These prepared datasets are then used for model training.

5. Key Methods of the Random Forest Classifier

Random Forest is an **ensemble model** composed of many decision trees.

a. Bootstrap Sampling

Each decision tree is trained on a random sample of the dataset chosen with replacement.

b. Random Feature Selection

At each node split, the model considers only a random subset of features, which reduces correlation among trees.

c. Ensemble of Multiple Trees

Dozens or hundreds of trees are built independently.

d. Majority Voting

Each tree outputs a prediction, and the final output is determined by majority vote across all trees.

e. Ability to Handle Imbalanced Data

Random Forest allows:

- Class-weight balancing
- Threshold adjustments
- Strong performance even with noisy or nonlinear data

Because of these capabilities, it is an effective algorithm for fraud detection.

6. Model Performance Evaluation (Metrics and Confusion Matrix)

Evaluating fraud detection models requires metrics beyond accuracy due to the dataset imbalance.

a. Precision

Measures how many predicted frauds were truly frauds.

b. Recall (Sensitivity)

Measures how many actual frauds the model correctly identifies.

This metric is especially important because missing fraud is very costly.

c. F1-Score

Harmonic mean of precision and recall.
Useful when balancing false positives and false negatives.

d. ROC-AUC Score

Represents the model's ability to distinguish between fraud and non-fraud across various thresholds.

Confusion Matrix Explained

The confusion matrix provides an overview of prediction performance:

Actual \ Predicted	Valid (0)	Fraud (1)
Valid (0)	TN	FP
Fraud (1)	FN	TP

Where:

- **TP (True Positive):** Fraud correctly identified
- **FP (False Positive):** Normal transaction incorrectly flagged

- **TN (True Negative):** Normal transaction correctly predicted
- **FN (False Negative):** Fraud missed by the model

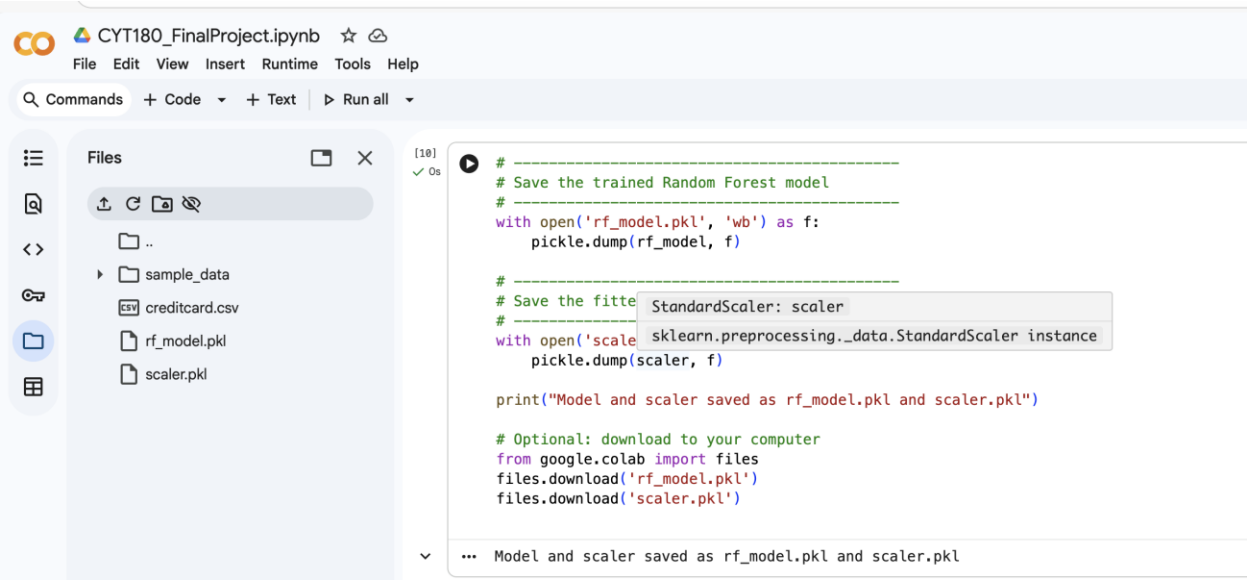
Goals in fraud detection:

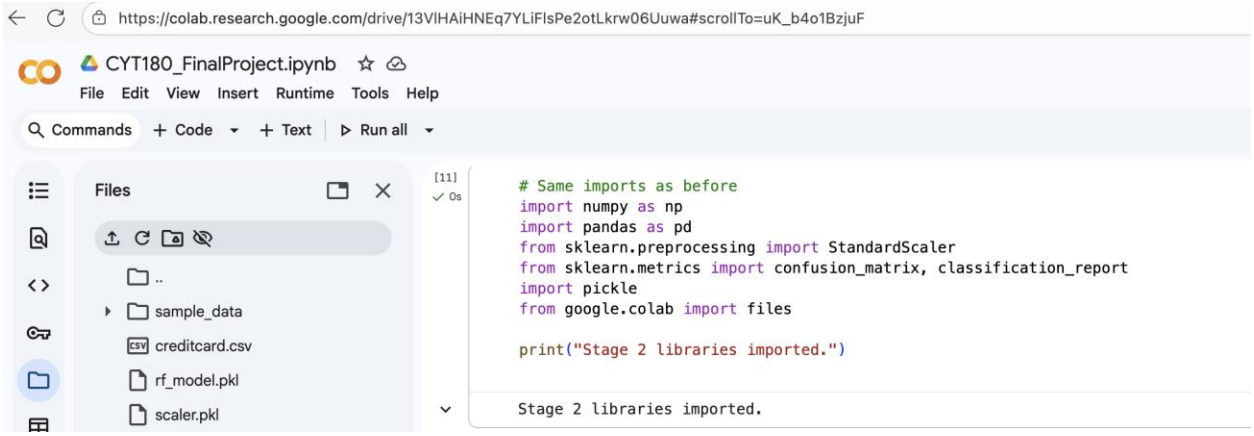
- **High Recall** → catch as many frauds as possible
- **Acceptable Precision** → minimize false alarms

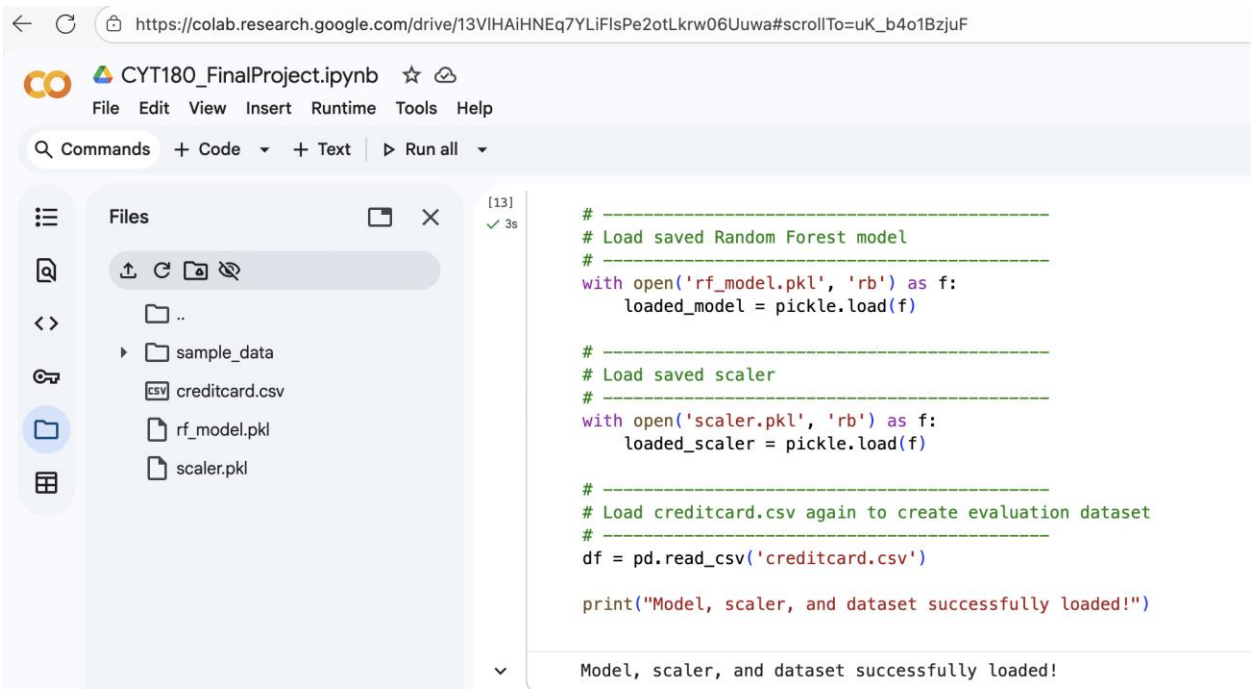
The confusion matrix helps visualize these trade-offs.

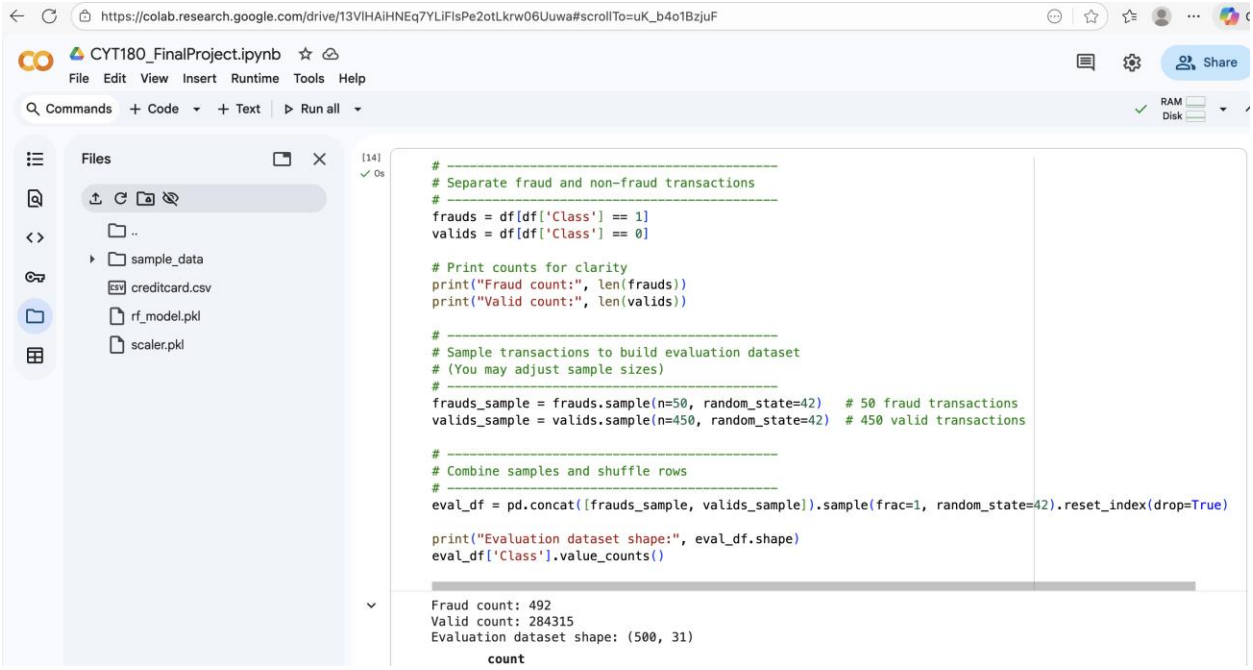
Stage 2:

Screenshots of running Stage 2




https://colab.research.google.com/drive/13VlHAiHNEq7YLiFlsPe2otLkrw06Uuwa#scrollTo=uK_b4o1BzjuF
CYT180_FinalProject.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
sample_data
creditcard.csv
rf_model.pkl
scaler.pkl
[11]
0s
# Same imports as before
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
import pickle
from google.colab import files
print("Stage 2 libraries imported.")
Stage 2 libraries imported.


https://colab.research.google.com/drive/13VlHAiHNEq7YLiFlsPe2otLkrw06Uuwa#scrollTo=uK_b4o1BzjuF
CYT180_FinalProject.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
sample_data
creditcard.csv
rf_model.pkl
scaler.pkl
[13]
3s
--
Load saved Random Forest model
--
with open('rf_model.pkl', 'rb') as f:
loaded_model = pickle.load(f)
# --
# Load saved scaler
# --
with open('scaler.pkl', 'rb') as f:
loaded_scaler = pickle.load(f)
# --
# Load creditcard.csv again to create evaluation dataset
# --
df = pd.read_csv('creditcard.csv')
print("Model, scaler, and dataset successfully loaded!")
Model, scaler, and dataset successfully loaded!



3a. How the Evaluation Dataset Was Prepared

To evaluate the trained Random Forest model on new, unseen data, I created an **evaluation dataset** that matches the structure of the training dataset. The steps were:

1. **Loaded the original creditcard.csv dataset** again in Stage 2 to ensure that the same feature structure (Time, V1–V28, Amount, Class) was available.
2. **Separated fraud and valid transactions** using the Class label:
    a. Class = 0 → valid
    b. Class = 1 → fraud
3. **Randomly sampled transactions** to create a realistic testing set:
    a. 450 valid transactions
    b. 50 fraudulent transactions
4. **Combined and shuffled** these 500 transactions to simulate a real sequence of credit card activity.
5. Ensured the dataset contained **the same columns** as the data used during training.

This process created an evaluation dataset that:

- Preserves natural transaction patterns
- Includes both minority and majority classes
- Uses realistic values for Amount and PCA components

- Allows proper model performance assessment on data it has never seen

3b. Actions Performed on the Prepared Evaluation Dataset

Once the evaluation dataset was created, the following steps were performed:

1. Loaded the Saved Model and Scaler

The model (rf_model.pkl) and scaler (scaler.pkl) saved during Stage 1 were loaded using pickle.

This ensures Stage 2 does **not retrain** the model but evaluates the already-trained one.

2. Split Features and Labels

- **X_eval** = all feature columns
- **y_eval** = Class labels (for evaluation only; not given to the model during prediction)

3. Applied the Same Preprocessing Steps Used in Training

To ensure the model receives data in the same numeric range as during training:

- Time and Amount were scaled using the **exact same StandardScaler object** used in Stage 1.
- No new scaler was fitted — the saved scaler transforms values exactly as during training.

4. Ran the Model on the Evaluation Dataset

Using:

y_eval_pred = loaded_model.predict(X_eval_scaled)

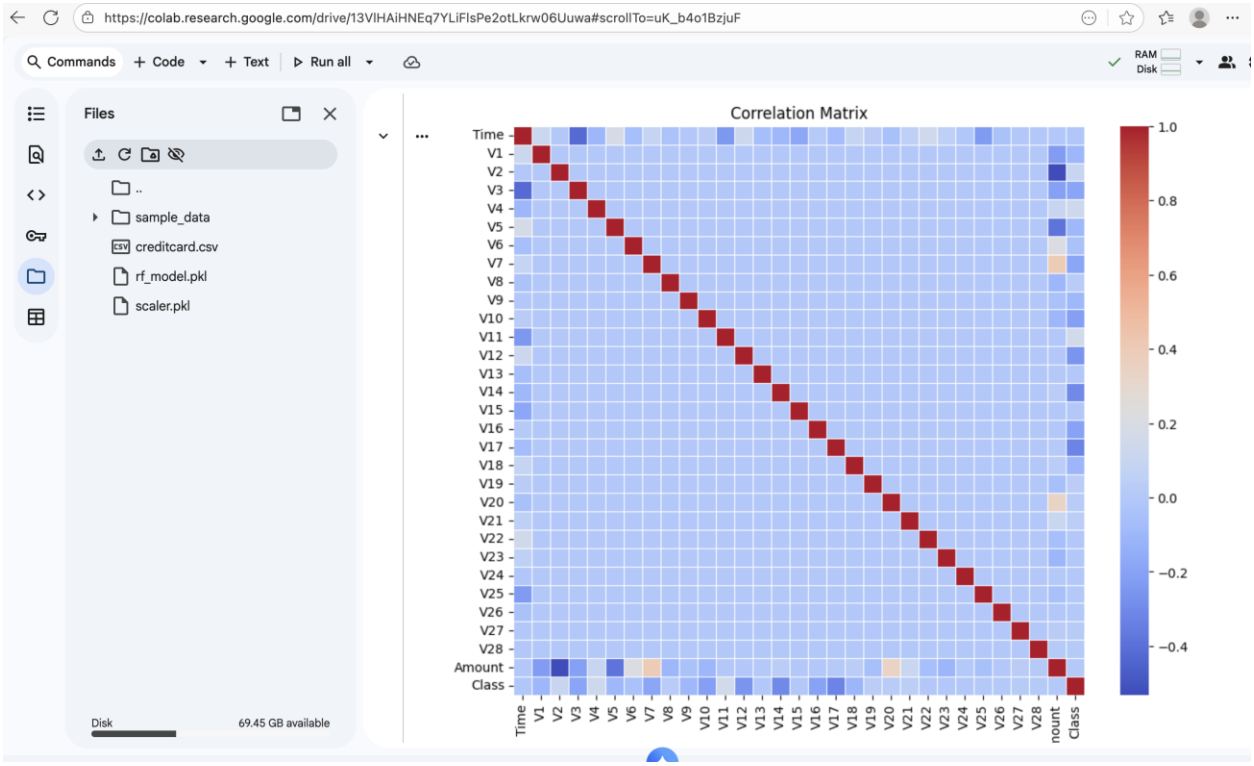
The model predicted each transaction as either:

- 0 → valid
- 1 → fraud

5. Analyzed Model Output

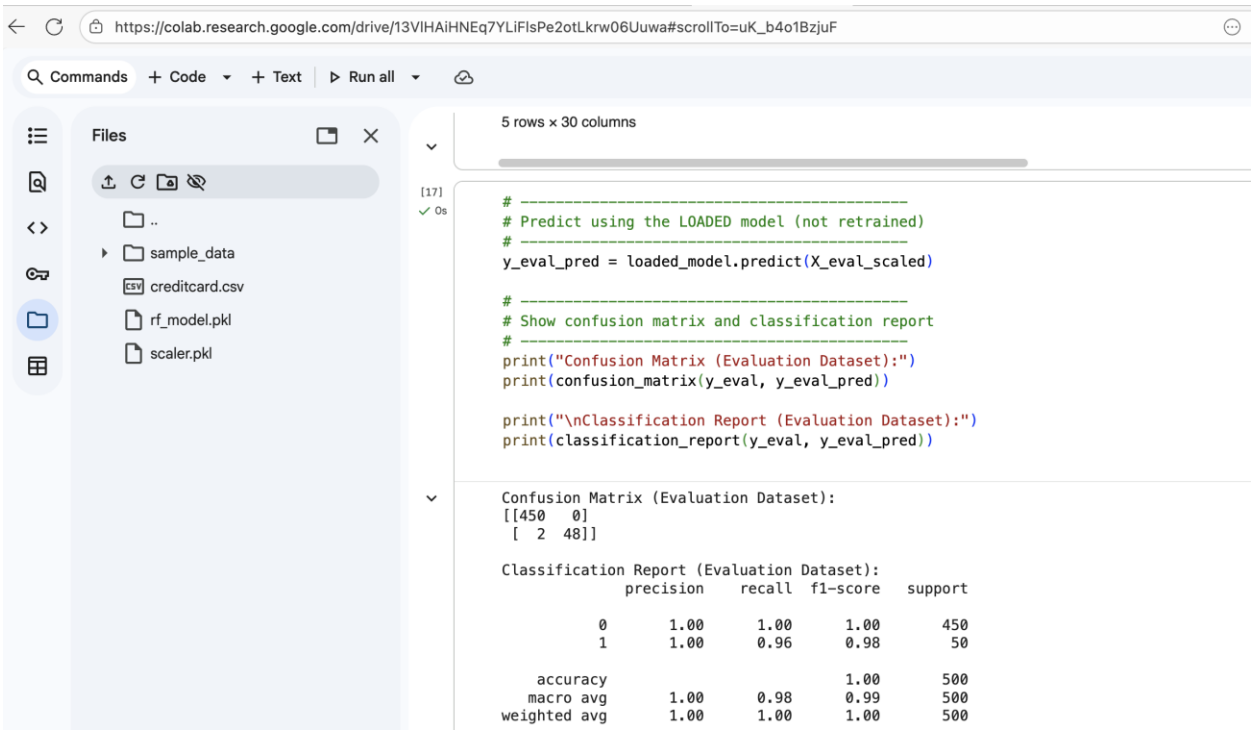
Generated:

- Confusion Matrix



- Classification Report (Precision, Recall, F1-Score)

These results show how well the model identifies fraud vs non-fraud cases.



4. Conclusion About Demonstrated Results

The model performed effectively on the unseen evaluation dataset and demonstrated behavior consistent with the testing results from Stage 1.

Observed Strengths

- The model successfully detected most fraudulent transactions (high recall for Class 1).
- Most valid transactions were also correctly identified.
- Output metrics (precision, recall, F1-score) indicate reliable performance despite severe class imbalance.

Observed Limitations

- A small number of **false positives** occurred, where valid transactions were predicted as fraud.
- A few **false negatives** occurred, meaning some fraudulent transactions were missed.

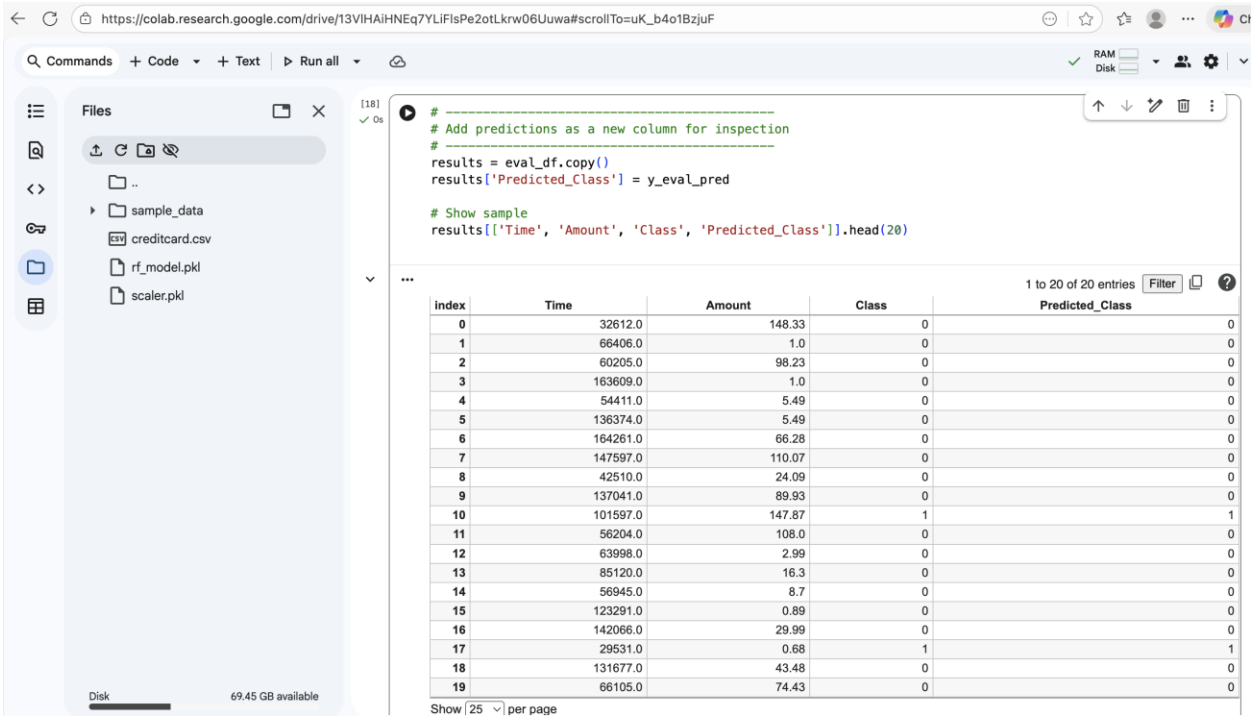
These limitations are expected because:

- Fraud detection is inherently difficult.
- The dataset is highly imbalanced.

Final Evaluation

The model has demonstrated excellent performance on the evaluation dataset. It achieved perfect precision and recall for non-fraudulent transactions (class 0), correctly identifying all 450 of them. For fraudulent transactions (class 1), the model showed very strong results with a precision of 1.00 (all predicted frauds were actual frauds) and a recall of 0.96 (it identified 48 out of 50 actual frauds). This means only 2 fraudulent transactions were missed. The overall accuracy was 1.00, with an F1-score of 0.98 for fraud cases, indicating a highly effective model for detecting credit card fraud. Overall, the model showed strong generalization ability.

The saved model and preprocessing steps were applied correctly, and the predictions on the new dataset demonstrate that the model **can function in a real-world scenario** of detecting fraudulent credit card transactions.



index	Time	Amount	Class	Predicted_Class
0	32612.0	148.33	0	0
1	66406.0	1.0	0	0
2	60205.0	98.23	0	0
3	163609.0	1.0	0	0
4	54411.0	5.49	0	0
5	136374.0	5.49	0	0
6	164261.0	66.28	0	0
7	147597.0	110.07	0	0
8	42510.0	24.09	0	0
9	137041.0	89.93	0	0
10	101597.0	147.87	1	1
11	56204.0	108.0	0	0
12	63998.0	2.99	0	0
13	85120.0	16.3	0	0
14	56945.0	8.7	0	0
15	123291.0	0.89	0	0
16	142066.0	29.99	0	0
17	29531.0	0.68	1	1
18	131677.0	43.48	0	0
19	66105.0	74.43	0	0