# Assignment 1: - Project Fraud Detection Using Machine Learning Algorithms

**What does each row in the dataset represent?**

Each row corresponds to a single credit card transaction. This could be any purchase or payment made using a card. The row includes various details recorded for that specific transaction.

**Which columns are clearly understandable, and which ones look encrypted or coded? Can you guess what any of the V1, V2, ..., V28 columns might represent?**

* Clearly understandable columns:
  + Time: Represents the time elapsed since the first recorded transaction (in seconds).
  + Amount: Value of the transaction in currency units.
  + Class: Indicates whether the transaction is fraud (1) or not (0).
* Encrypted or coded columns:
  + V1 to V28: These are anonymized features produced by a Principal Component Analysis (PCA) transformation. Their exact meanings are unknown, as they were obfuscated for privacy.
* Numeric features derived from a PCA transformation to protect sensitive data.
* They are kept anonymous for privacy reasons, and PCA was applied to reduce dimensionality and remove correlations.

**What is the purpose of the Class column? What do the values 0 and 1 mean, and why is this important for a fraud detection project?**

The Class column is the target label used in machine learning:

* 0: The transaction is legitimate (not fraud).
* 1: The transaction is fraudulent.

This column is important because it teaches the model to identify patterns in past transactions that separate fraud from non-fraud. It helps the model understand what to look for when trying to detect future fraud.

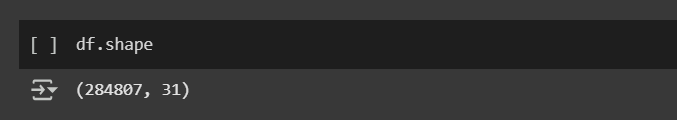
**Look at the values in the Amount and Time columns for the first 5 rows. What kind of patterns or differences do you notice? Do any amounts look unusually high or low?**

* Amount:
  + You might notice a wide range in Amount values—some are very low (like $0.89), while others are significantly higher. This suggests high variability in transaction sizes.
* Time:
  + The Time column usually shows increasing values, indicating that transactions are recorded sequentially. If the time doesn’t increase, it could mean multiple transactions occurred at nearly the same time.

The Amount column shows both very low and very high transaction values, which may indicate normal or suspicious activity. The Time column generally increases, showing transactions are recorded sequentially, with some occurring close together. These patterns help in spotting potential fraud.

**How many rows (entries) and columns (features) are there in the dataset?**

The dataset contains 2,84,807 rows (transactions) and 31 columns (features), meaning it holds information for over 280,000 individual credit card transactions, with 31 data points for each.



**Do any columns have missing values? How do you know?**

No, there are no missing values in the dataset. This is confirmed by checking the pd.isnull(df).sum(). output—each column shows the same number of non-null entries.

**What type of data (number or category) is stored in most of the columns?**

Most columns store numerical data, specifically of types float64 and int64. This includes PCA-transformed features (V1 to V28), the transaction Amount, and the Class label.

**Which column do you think represents the final outcome—whether a transaction is**

**fraudulent or not? Why?**

The Class column is the target variable representing the outcome. It holds binary values: 0 for legitimate transactions and 1 for fraudulent ones, which the model will learn to predict.

**Why do you think it is important to check this information before building a model?**

Checking this information early helps ensure the data is complete, clean, and correctly formatted. Skipping this step can lead to errors, poor model performance, or incorrect conclusions—just like cooking without knowing your ingredients.

**How many columns in your dataset are of type float64? What does this type represent?**

There are 30 columns of type float64 in the dataset. This data type represents decimal numbers, allowing for more precision than integers. These columns include features like Amount, Time, and all PCA-transformed variables (V1 to V28). These floating-point values help capture small variations in transaction behavior.

**What is the data type of the ‘Class’ column? Why do you think it is different from the**

**other columns?**

The Class column is of integer (int64) type, representing binary outcomes: 0 for non-fraud and 1 for fraud. It differs from the other columns because it’s a label or target variable, not a continuous input feature. Since the values are categorical (either fraud or not), using integers makes it easy for models to distinguish classes.

**Why is it important to know the data type of each column before building a machine**

**learning model?**

Knowing the data types ensures that you apply the correct preprocessing steps. For example:

* Models need numerical values—strings or mixed types can cause errors.
* You can decide whether a column needs scaling, encoding, or dropping.
* It prevents type mismatches and improves model accuracy by treating data appropriately.

**Why did we increase the number of rows shown in the output using the display setting?**

**How did it help in understanding the dataset?**

Increasing the number of visible rows (e.g., pd.set\_option(‘display.max\_rows’, 100)) allowed us to see more of the dataset at once, especially when dealing with many columns. If only 10 or 20 rows are shown, we might miss hidden values, spotty patterns, or columns with unusual formats. It improves visibility and confidence in the dataset’s structure.

**Did you find any column with a data type that surprised you? If yes, which one and**

**why? If not, explain why the types make sense.**

The Class column is int64, suitable for binary classification. Since there are no text or categorical columns, the dataset is well-prepared for numeric-based models.

**How many total records (rows) are present in the dataset? What does each row**

**represent?**

The dataset includes 284,807 rows, where each row represents a single credit card transaction made by a user.

**How many columns (features) are included in the dataset? What kind of information do**

**you think these columns might contain?**

There are 31 columns in total, capturing different features of each transaction. These include understandable ones like Time, Amount, and Class, as well as 28 anonymized features (V1 to V28) derived through PCA to protect sensitive data.

**Why is it important to know the number of rows and columns before starting data**

**analysis?**

Knowing the number of rows and columns beforehand is important because it affects processing time, memory usage, and the tools or techniques you'll need for cleaning and modeling.

**If this dataset had only 100 rows and 5 columns, how would your approach to analysis**

**change compared to working with 284,807 rows and 31 columns?**

If the dataset had only 100 rows and 5 columns, the approach would be different—models would need to be simpler, and results would be less reliable due to limited information.

**How can the size of a dataset help us identify potential challenges in cleaning,**

**visualization, or building models?**

With large datasets like this, we must also be prepared for challenges such as missing values, slow visualizations, and the need for proper scaling, sampling, or feature engineering to ensure the machine learning models perform well.

**What are missing values, and why is it important to check for them before starting**

**analysis?**

Missing values are empty or null entries in a dataset, similar to blank spots on a school attendance sheet. It’s important to check for them before analysis because they can lead to incorrect conclusions or errors during processing.

**go wrong if we use incomplete data in decision-making?**

**Based on your analysis, did your dataset contain any missing values? If yes, mention the**

**columns and how many were missing. What would you do if a column had a lot of missing values? Why?**

In this dataset, after using .isnull().sum(), we found that there were no missing values—all columns had complete data. However, if a column had a large number of missing values, we would need to make a decision based on its importance: we might drop the column if it’s not useful, or fill in the missing data using methods like mean, median, or forward fill.

**If there were missing values, how might they affect the result of a machine learning**

**model?**

If missing values are left untreated, they can negatively impact a machine learning model by causing inaccurate predictions or model failure.

**Why is it important to have clean and complete data in real-world projects like fraud**

**detection?**

In real-world applications like fraud detection, having clean and complete data is critical because missing or incorrect information could result in undetected fraud, false alarms, or financial losses, much like overlooking important details in medical or banking records.

**What does the mean value of the Amount column tell you about the typical**

**transaction size?**

The mean value of the Amount column is relatively low, suggesting that the average transaction size in the dataset is modest, which is expected for typical daily purchases.

**Which column in the dataset shows the highest variation, and what might that tell**

**you?**

When comparing standard deviations, the Amount column shows the highest variation, indicating a wide spread in transaction values, from very small purchases to large transactions.

**From the min and max values of the Amount column, what can you say about the**

**range of spending in the dataset?**

The minimum and maximum values in the Amount column reveal a significant gap—from close to zero up to very large amounts—highlighting the diverse spending behavior in the dataset. Looking at the Class column, the mean is very close to 0, with a min of 0 and a max of 1, confirming that fraudulent transactions are extremely rare in this dataset, and most entries are non-fraudulent.

**What does the Class column's summary (mean, min, max) tell you about how**

**common fraud is in this dataset?**

Looking at the Class column, the mean is very close to 0, with a min of 0 and a max of 1, confirming that fraudulent transactions are extremely rare in this dataset, and most entries are non-fraudulent.

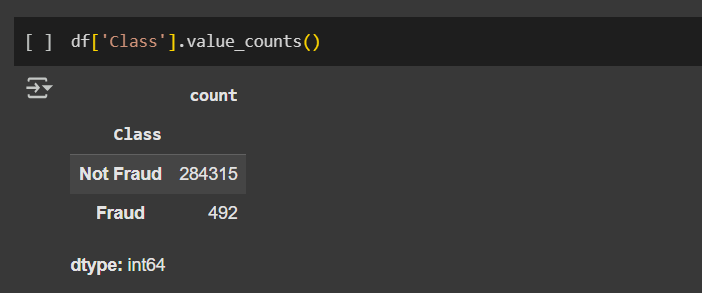
**Are there any columns that seem unusual or surprising to you when looking at their**

**summary statistics? Explain why.**

One surprising detail is the presence of very large values in the Amount column, which could represent outliers or high-risk transactions worth investigating further.

**How many total transactions in the dataset are labeled as “Fraud” and “Not Fraud”?**

There are 492 fraud transactions and 2,84,315 non-fraud transactions in the dataset, based on the Class column.



**What does the distribution of fraud vs. non-fraud transactions tell you about the data?**

The data is highly imbalanced, with fraud cases making up less than 0.2% of all transactions. Most of the data represents normal activity.

**Why is it important to know that fraud cases are very few in number compared to normal transactions?**

It’s important because models can easily ignore the rare fraud cases and still appear accurate, but fail to actually detect fraud, which is the goal.

**Why might that be misleading?**

the model would get over 99% accuracy, but this is misleading because it would miss all fraud cases, which defeats the purpose of the model.

**How would you handle the challenge of working with a dataset where fraud is rare? Can you think of one possible strategy?**

One strategy is to balance the data using techniques like SMOTE (oversampling fraud cases), or focus on metrics like recall and F1-score instead of accuracy, to better detect rare fraud cases.

**Why do we split the data into training and testing sets?**

We split the data to train the model on one part (training set) and evaluate it on another unseen part (testing set). This helps check if the model has truly learned patterns or is just memorizing the data.

**What is the purpose of using the 'Class' column as the output (target)?**

The Class column tells us whether a transaction is fraud (1) or not (0). It is used as the target variable, which the model tries to predict based on input features.

**Why do we keep the input features (like Time, V1–V28, Amount) separate from the**

**output label?**

Input features are the data the model uses to make predictions, while the output (Class) is what the model is trying to predict. Keeping them separate avoids data leakage and ensures proper learning.

**What is the importance of setting the 'random\_state' while splitting the data?**

Setting random\_state ensures the same split happens every time you run the code, which makes results consistent and reproducible.

**Why is it important to use the 'stratify' option when splitting the data in this case?**

Since the dataset is imbalanced, using stratify=y ensures the same ratio of fraud and non-fraud cases in both training and testing sets, giving the model a fair representation during learning and evaluation.

**Why do we need to scale the features in our dataset before training the model?**

We scale the features to ensure that all input values are on a similar range, so that large numbers like Amount don’t dominate smaller ones like V1 or V2. This helps the model learn fairly and efficiently.

**What would happen if we scaled the test data separately, without using the scaler**

**trained on the training data?**

If we scale the test data separately, it could lead to inconsistent transformations, because the model will be comparing inputs on different scales, resulting in wrong or unfair predictions.

**After scaling, what kind of values do we expect to see in our dataset?**

After using something like StandardScaler, the values are usually centered around 0 with a standard deviation of 1, meaning most values lie roughly between -3 and +3.

**Why is it important to apply the same scaling method to both training and testing data?**

Using the same scaler ensures that both training and testing data are treated equally, which makes the evaluation fair, accurate, and consistent.

**Can you think of a real-life example where using values of different scales might create**

**confusion or unfair results?**

Yes, comparing a test score out of 100 with another out of 10 without converting them would be misleading. Similarly, a model comparing features on different scales could give biased results.

**What is the purpose of using a Logistic Regression model in this fraud detection**

**project?**

Logistic Regression is used for binary classification, which is perfect for this project since we are predicting whether a transaction is “Fraud” (1) or “Not Fraud” (0)—two clear categories.

**Why is it important to split the dataset into training and testing parts?**

Splitting the data lets us train the model on one part and test it on unseen data. If we use the same data for both, the model might just memorize it, and the evaluation wouldn’t be fair or realistic.

**3. What does it mean when your model has a high accuracy score? Is accuracy always**

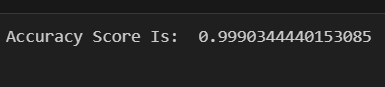
**the best way to evaluate a fraud detection model? Why or why not?**

A high accuracy score means the model is correct most of the time. But in fraud detection, where fraud cases are very rare, accuracy can be misleading—the model might predict “Not Fraud” for everything and still look good.

**4. If your model predicted “Not Fraud” for all transactions, what would the accuracy**

**look like? Would this be a good model? Why or why not?**

The accuracy would be over 99%, but this wouldn’t be a good model. It would miss all the fraud cases, which is the main thing we’re trying to detect, so it’s useless for real-world fraud prevention.



**5. Suppose your model's accuracy is lower than expected. List two things you might try**

**to improve the model’s performance.**

* Balancing the dataset using oversampling or under sampling.
* Using a different model like Random Forest improving feature scaling and preprocessing.

**What is a Random Forest model, and why do we use it for fraud detection?**

A Random Forest is an ensemble model made up of many decision trees, where each tree gives a prediction, and the final output is based on the majority vote. It works well for fraud detection because it handles complex patterns, reduces overfitting, and performs better on imbalanced datasets.

**How does the accuracy of the Random Forest model compare to the Logistic Regression**

**and LDA models you built earlier? Which one performed the best?**

Typically, Random Forest gives higher accuracy than Logistic Regression and LDA because it captures non-linear relationships better. If your accuracy scores showed this trend, then Random Forest likely performed the best among them.

**What might be the advantage of using multiple models (like Random Forest) instead of**

**just one decision tree?**

Using multiple trees helps reduce the risk of overfitting and improves stability. A single tree might make biased or poor decisions, but a forest of trees offers better generalization by averaging out errors.

**If your Random Forest model made wrong predictions, what could be the possible**

**reasons for that?**

Wrong predictions may occur due to class imbalance, irrelevant or unscaled features, or the model failing to capture rare patterns in fraud cases. Even strong models can struggle with very limited fraud samples.

**Do you think Random Forest is a good model for this project? Why or why not?**

Yes, Random Forest is a good choice for fraud detection because it offers high accuracy, handles imbalanced and noisy data, and doesn't require heavy feature engineering. However, it should be paired with proper data preprocessing and evaluation metrics (like precision and recall) to be truly effective.

**Which model achieved the highest accuracy in your results table?**

The Random Forest model achieved the highest accuracy in the results table, showing strong performance in identifying patterns in the data.

**What could be the reason this model performed better than the others?**

Random Forest performed better because it is a complex ensemble model that combines multiple decision trees. It can handle large datasets, capture non-linear patterns, and is less prone to overfitting than single models.

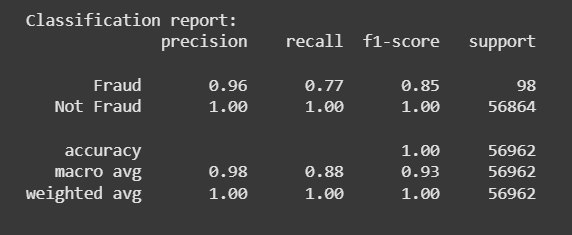
**Which model performed the worst in your results? Why do you think that happened?**

The Logistic Regression or LDA model likely performed the worst. This could be due to the dataset being imbalanced or containing complex patterns that simple linear models cannot capture effectively.

**If you had to recommend one model to a company for fraud detection, which one**

**would you choose and why?**

I would recommend the Random Forest model because it offers a good balance of accuracy, robustness, and interpretability. It works well with imbalanced data, requires less tuning, and is reliable in real-world scenarios.



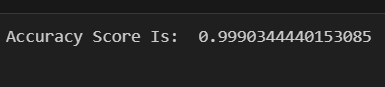
**What is one thing you learned from comparing all the models together?**

I learned that accuracy alone is not enough—especially in imbalanced problems like fraud detection. Model choice depends on the problem, and it's important to also consider metrics like recall, precision, and F1-score for a complete evaluation.

**Why this step is important:**

**A model is only useful if it makes good decisions. The confusion matrix shows you exactly**

**how well your model is doing, and helps you figure out if you can trust it for real-life use.**

****

Evaluating the confusion matrix is crucial because it shows how well your model predicts fraud vs. non-fraud, not just overall accuracy. It helps decide whether the model is trustworthy enough for real-life use, like in banking systems.

**What do the four numbers in the confusion matrix represent in your model’s results?**

**The confusion matrix shows:**

|  | **Predicted: Not fraud (0)** | **Predicted: Fraud (1)** |
| --- | --- | --- |
| **Actual: No fraud (0)** | True Negatives (TN) | False Positives (FP) |
| **Actual: Fraud (1)** | False Negatives (FN) | True Positives (TP) |

* True Positives (TP): Correctly predicted fraud
* True Negatives (TN): Correctly predicted non-fraud
* False Positives (FP): Normal transactions wrongly predicted as fraud
* False Negatives (FN): Fraud transactions wrongly predicted as normal

**Did your model correctly identify most of the fraud cases? Why is this important in real**

**life?**

If the model correctly identified most fraud cases (high TP, low FN), that’s important because missing fraud (False Negatives) can lead to huge financial losses or security breaches for banks and customers.

**Were there any False Positives in your model? What could be the impact of these in a**

**real business?**

Yes, False Positives can occur, and in a business, this means flagging genuine customer transactions as fraud—which could annoy customers, cause delays, or lead to loss of trust in the system.

**What steps can you take if your model is missing too many fraud cases (False**

**Negatives)?**

To reduce False Negatives, you can try:

* Adjusting the classification threshold
* Using models with better recall (e.g., Random Forest)
* Oversampling fraud cases using techniques like SMOTE
* Using cost-sensitive learning where missing fraud has a higher penalty

**Based on your confusion matrix, would you say your model is ready to be used in a real**

**bank? Why or why not?**

If your model shows low False Negatives and reasonable False Positives, then it's likely suitable for real use. But if fraud cases are often missed, then it’s not ready, and further tuning or advanced techniques are needed to make it reliable and safe for real-world banking.