Samatrix.io

Detecting Financial Fraud Using Machine Learning: A Predictive Approach to Safeguard Digital Transactions

Project Goal:

The goal of this project is to **develop a machine learning–based fraud detection system** that can accurately identify whether a financial transaction is legitimate or fraudulent. Using anonymized real-world credit card transaction data, the project aims to:

- **Analyze patterns in large-scale financial transactions**
- **Handle extreme class imbalance** (only ~0.17% fraud cases)
- **Build and compare predictive models** such as Logistic Regression, LDA, Decision Trees, Random Forest, SVM, and others
- **Evaluate models using metrics beyond accuracy**, such as **precision, recall, and confusion matrix**, to ensure low false-negative rates
- **Support real-time fraud detection efforts** in financial institutions by identifying suspicious activities early

By the end of this project, learners will understand how to apply supervised learning to high-stakes, imbalanced classification problems, and how to optimize models for **business-critical outcomes** rather than generic accuracy.

Learning Objectives

Data Acquisition & Exploration

1. **Load and interpret** real-world anonymized credit card transaction data containing imbalanced classes.
2. **Inspect and summarize** dataset structure using shape, data types, and statistical descriptions.
3. **Identify class imbalance** and **discuss its impact** on model learning and evaluation.
4. **Verify data quality** by checking for missing values and nulls, and explaining why additional cleaning may or may not be required.

Feature Engineering & Preprocessing

5. **Separate input features (X)** from the target variable (Class) representing fraud or normal transactions.
6. **Apply standardization or scaling** to numeric features where needed.
7. **Split the dataset** into training and test sets using stratified sampling to preserve class proportions.
8. **Apply dimensionality checks** and **validate the readiness** of the dataset for modeling.

Model Building & Comparison

9. **Train multiple classification models**, including:
    - Logistic Regression
    - Linear Discriminant Analysis (LDA)
    - Gaussian Naive Bayes
    - Decision Tree

- Random Forest
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)

10. **Generate predictions** using trained models on unseen test data.

Model Evaluation & Selection

11. **Compute and interpret** model evaluation metrics such as:
    - Accuracy
    - Precision
    - Recall
    - F1-score
    - Confusion Matrix
12. **Visualize the confusion matrix** to examine true/false positives and negatives.
13. **Compare model performance** using a unified score table to identify the most effective algorithm.
14. **Discuss the trade-offs** between precision and recall in the context of fraud detection.

Critical Thinking & Application

15. **Analyze the business impact** of false positives and false negatives in fraud detection systems.
16. **Justify the model choice** based on evaluation results and business constraints (e.g., minimizing missed frauds).
17. **Propose real-world use cases** where the selected model could be deployed in financial transaction systems.

Dataset

https://samatrix-data.s3.ap-south-1.amazonaws.com/ML/creditcard.zip

What to Submit:

Submission Type: Individual

Each student must submit the following:

1. **Jupyter Notebook (.ipynb file) or python filr (.py file)**
    a. Filename: YourFullName_CreditCard.ipynb (e.g., AnanyaKumar_ CreditCard.ipynb)
    b. Your notebook must follow the steps and structure discussed in class following the instructions in the submission guideline
2. **Word or PDF file**
    a. Answers “Questions for Report” in separate file

Samatrix.io

Required Sections and Instructions:

1. Getting Started with Data Analysis and Model Evaluation

Goal of the Lab:
In this lab, you will learn how to:

- Import and organize your dataset
- Explore the data using graphs and tables
- Prepare the data for analysis
- Understand how well your model performs using accuracy as a measure

Let's begin!

Step-by-Step Instructions:

Step 1: Set Up Your Lab Notebook
Begin by opening your Python environment or Jupyter notebook where you'll do all your work. Think of it as your digital lab notebook.

Step 2: Load Tools You Will Use
Before working with your data, you'll bring in the tools you need.
These tools are like items in your science lab toolbox.
In this project, the tools include:

- **Numpy** – a calculator that helps with numbers and math operations
- **Pandas** – helps you read, clean, and organize data tables (like spreadsheets)
- **Seaborn and Matplotlib** – help you draw beautiful graphs and charts
- **Scikit-learn tools** – used to prepare your data for modeling and check how good your model is

---

2. Exploring the Credit Card Fraud Dataset

Objective:
In this lab, you will learn how to load a real-world dataset and view the first few records to understand what kind of data you're working with.

Step-by-Step Instructions

Step 1: Understand the Goal
We are working with a **credit card transactions dataset**. This dataset includes both **normal and fraudulent transactions**. Our goal is to **load the data** and take a **first look** at what it contains.

Step 2: Load the Dataset

- You will be working with a file named **creditcard.zip**. It contains the data in a compressed format.
- When we load it, we are telling the computer:
  "Please open this file and show me the content in table form (just like Excel)."

Think of this like opening your school report card to see your marks—it's the very first look at the data inside!

Step 3: Set the Display Width

- By default, long tables may look messy or get cut off on the screen.
- To make it **more readable**, we adjust the **width of the display window** so that more columns can fit in one line.
- This doesn't change the data—just how it **looks** when shown.

Step 4: View the First Few Records

- Once the data is loaded, we will **ask the computer to show the first 5 rows** (just like the top of a spreadsheet).
- This helps us **get a feel for the data**—what columns are there, what types of values we're dealing with, and how the data is structured.

What to Look For in the Output

When you see the first 5 rows of the table, look for:

- A **column named Time** – tells when the transaction happened.
- A **column named Amount** – tells how much money was involved.
- A **column named Class** – this is very important. It tells if a transaction was normal (0) or fraud (1).
- Many columns with names like V1, V2, etc. – these are encoded features used to protect customer privacy.

Why This Step is Important

- **This is your first impression of the data**. Just like when you meet someone new, the first few details help you understand who they are.
- It helps you **plan your next steps**, like cleaning data, finding patterns, and building your fraud detection model.

Activity Checklist

- Before you move on to the next lab:
- Make sure the dataset loads without any error
- Ensure the table shows at least 5 rows
- Try to describe what each column might represent
- Notice how many digits or decimals appear in the numbers
- Look at the Class column — do you see any fraud cases?

Questions for Report

What does each row in the dataset represent?
Hint: Think of a real-world action, like a payment made using a credit card. Each row usually stands for a single event.

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?
Hint: Columns like Amount, Time, and Class seem readable. But V1 to V28 don't tell us much directly—why might they look like that?

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?
Hint: A machine learning model needs to know what counts as "fraud" or "not fraud." This column helps the model learn that difference.

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?
Hint: Try comparing the Amount values. Is one much higher than the others? Are the Time values increasing or repeating?

Why is it helpful to look at the first few rows of a dataset before doing any analysis or modeling? What kind of problems or surprises could this help you avoid later?
Hint: You might spot missing values, strange formats, or unexpected column names that need fixing before analysis.

---

3. Understand the Structure of Your Fraud Dataset

Objective:
To **explore the structure** of the credit card fraud dataset—find out what kind of data you have, how complete it is, and how it's organized.

Step-by-Step Instructions

Step 1: Run the Info Command
You're going to use a simple command that gives you a **summary** of the entire dataset. Think of it like checking the **table of contents and condition report** of a big book before reading it. This step gives you:

- The number of rows and columns,
- The names of all columns (features),
- The type of data in each column (numbers, text, etc.),
- Whether any data is missing.

Step 2: Read the Output Like a Detective
Once the summary appears, take a moment to read it carefully. Here's what to focus on:

- **Total Entries**: This tells you how many records (rows) are in the dataset. In our case, it's around 284,807 transactions.
- **Column Names**: These are the different features of each transaction (like Time, Amount, and V1 to V28).
- **Data Types**: Notice if the column holds decimal numbers (called float64), whole numbers (int64), or text (object).
- **Non-Null Counts**: This shows how many values are present in each column. If it's less than the total number of rows, then some data is missing.

Step 3: Write Down Key Observations
As a beginner, don't worry about understanding every single detail right away. Focus on writing down:

- How many columns are in the data?
- Do you see any missing data?
- What types of values are in the dataset?

- Which column do you think might represent **fraud or not-fraud**?

Why This Step is Important:
Just like a doctor needs to check a patient's vitals before starting treatment, a data analyst needs to **check the health of the dataset** before building models.
If you skip this step:

- You might miss columns with missing data,
- You might treat numbers as text (which breaks your code later),
- Or you might build a model using the wrong target!

Real-Life Analogy:
Imagine you're opening a big Excel file of student grades. Before analyzing it:

- You first check if every student's name and marks are there,
- You see if all columns are filled,
- And you check if any column is in the wrong format (like marks written as "eighty-five" instead of 85).

This is exactly what fraud_data.info() helps you do—but for a dataset of credit card transactions.

Questions for Report

1. How many rows (entries) and columns (features) are there in the dataset?
Hint: Look at the top line of the output. The number of entries shows total transactions, and the column count tells you how many types of information are stored for each transaction.

2. Do any columns have missing values? How do you know?
Hint: Check if the "Non-Null Count" for each column matches the total number of entries. If a column has fewer non-null values, that means some data is missing.

3. What type of data (number or category) is stored in most of the columns?
Hint: Look under the "Dtype" column. Are they mostly float64, int64, or something else?

4. Which column do you think represents the final outcome—whether a transaction is fraudulent or not? Why?
Hint: Think about what the column named "Class" might be used for. What kind of values does it hold?

5. Why do you think it is important to check this information before building a model?
Hint: Imagine trying to cook a recipe without checking your ingredients. What could go wrong if you skip this step when working with data?

---

4. Check Data Types

Goal of This Step:
You will explore **what type of data is present** in each column of the dataset and **increase the number of rows shown in the output**, so that nothing important gets hidden when you print large tables.

Step-by-Step Instructions:

Step 1: Open your dataset
Make sure you have already **loaded the dataset** (credit card fraud data) into your project.
This step assumes you have your data saved in a variable like fraud_data.

Step 2: Adjust the display setting
By default, Python might **hide some rows** when showing long outputs (like column names or data types). In this step, you will **increase the number of rows shown** on the screen to 500.

Think of it like telling your screen: "Hey! Don't hide anything. I want to see more!"
This will help you clearly **view all 31 columns** (features) in your dataset when checking their data types.

Step 3: Check the data types
Now, you will use a command that tells you the **type of data in each column**.
Why is this useful?
Because it tells you:

- Which columns have numbers with decimals (like 19.99) — these are called float64
- Which columns have whole numbers (like 0, 1) — these are called int64

This step is important because **machine learning models work differently with numbers and text**. Knowing the data types early will help you **prepare the data correctly** later.

What You Will Learn from This Step:

- You will find out if all your features are numeric or if there's anything that needs fixing.
- You'll be able to check that the "Class" column (which shows whether a transaction is fraud or not) is in the correct format.
- You'll become more confident in understanding your dataset before moving into deeper analysis.

Tip:
If you skip this step, you might run into errors later like:

- "This column is not numeric."
- "This column type is not supported."

So always check your data types before doing any analysis or model training.

Questions for Report

How many columns in your dataset are of type float64? What does this type represent?
Hint: Count the number of columns labeled as float64. Think about what kind of values (like decimal numbers) fall into this category.

What is the data type of the 'Class' column? Why do you think it is different from the other columns?
Hint: Is it a number with decimal points or a whole number? How does that make sense for classifying fraud vs. not fraud?

Why is it important to know the data type of each column before building a machine learning model?
Hint: Think about whether the model needs numbers, and how text or mixed types might confuse the model.

Why did we increase the number of rows shown in the output using the display setting? How did it help in understanding the dataset?
Hint: What problem might happen if only 10 or 20 rows were visible when the dataset has many columns?

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.
Hint: Were you expecting all numeric columns? Did any column look different from what you thought?

---

5. Understanding Your Dataset's Size

Objective:
To learn how to check **how many records (rows)** and **how many columns (features)** are present in your dataset using a simple command.

Instructions: Step-by-Step
Step 1: Make sure your dataset is loaded
Before checking the shape, ensure that you've already successfully loaded your dataset into memory (for example, as fraud_data). This dataset should contain all the information we want to analyze.

Step 2: Ask a simple question – "How big is my data?"
Think of your dataset like a large table or spreadsheet.

- Each **row** in that table is one transaction (like one payment made by credit card).
- Each **column** is a type of information (like time, amount, etc.).

In this step, you're asking your computer:
"Hey, how many rows and how many columns does this dataset have?"

Step 3: Read the result
You'll get an answer that looks like this:
(284807, 31)
Here's how to understand it:

- **284,807** means you have **284,807 transactions** (or rows).
- **31** means you have **31 different details or variables** recorded for each transaction (or columns).

Why is this important?

- It tells you whether your data loaded properly.
- It helps you plan the next steps (like cleaning, visualizing, or modeling the data).
- Knowing the size of the data helps you manage your time and computer memory better.

Real-World Example:
Imagine you're working at a bank, and you're analyzing credit card purchases to catch fraud. You first want to know:

- **How many total purchases** are there in your data?
- **What details** do you have about each purchase?

This command gives you that quick overview.

What you should learn from this step:

- How to **check the size of any dataset** in terms of rows and columns.
- How to **interpret** what the number of rows and columns mean.
- Why it's important to know your data's shape **before you build a machine learning model**.

Questions for Report

How many total records (rows) are present in the dataset? What does each row represent?
Hint: Think of each row as a single transaction made using a credit card.

How many columns (features) are included in the dataset? What kind of information do you think these columns might contain?
Hint: Columns usually include time, transaction amount, and other coded details.

Why is it important to know the number of rows and columns before starting data analysis?
Hint: Consider how size affects memory, analysis time, and decision-making.

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?
Hint: Think about reliability, performance, and the time needed to explore or model.

How can the size of a dataset help us identify potential challenges in cleaning, visualization, or building models?
Hint: Think of large datasets having more chances of missing data, slow visualizations, or need for preprocessing.

---

6. Checking for Missing Values

Objective:
To check if any data is missing in our dataset. This is an important step before we do any kind of analysis or model building.

Step-by-Step Instructions

1. **Understand what "missing values" are**
   Imagine a class attendance sheet where some student names or dates are blank. That's what a missing value looks like in a dataset—something important is left empty or is not recorded.
2. **Run the command to find missing values**
   This step will give you a full list of all the columns and show how many missing values each one has.
3. **Interpret the result**
   - If a column shows "0", it means it's completely filled—great!

- If a column shows any number greater than 0, that many values are missing, and you'll need to decide how to handle it (e.g., delete or fill in the gaps).

4. **Record your findings**
   Write down what you observe. Are there missing values? If yes, in which columns?

Tip:

- Always check for missing values before analyzing the data.
- Missing values can affect the accuracy of your machine learning model.
- Think of this step as "quality checking" your data—like making sure all ingredients are ready before cooking.

Questions for Report

What are missing values, and why is it important to check for them before starting analysis?
Hint: Think of a school attendance sheet where some names or dates are missing. What could 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.
Hint: Look at the output you got after running .isnull().sum().

What would you do if a column had a lot of missing values? Why?
Hint: Think practically—would you remove that column, try to fill it, or something else?

If there were missing values, how might they affect the result of a machine learning model?
Hint: Consider if the model is learning from incomplete data. Could this cause wrong predictions?

Why is it important to have clean and complete data in real-world projects like fraud detection?
Hint: Imagine the consequences of missing key transaction details in banking or medical records.

---

7. Descriptive Statistics of the Dataset

Objective of This Step:
The goal is to **summarize the main features** of the dataset. You will find the average, minimum, maximum, and other useful statistics for every column. This helps you get a quick understanding of what your data looks like and whether it needs cleaning or preparation.

Step-by-Step Instructions

1. **Set precision to make numbers easier to read**
   You will now make the output easier to read by limiting the number of decimal places shown. Instead of showing long decimals like 2.4873947393, we will show numbers like 2.49.

2. **Generate the summary table**
   Use the built-in .describe() function to generate a summary of all numeric columns. This summary will include:
   - Count: Number of rows
   - Mean: Average value
   - Standard Deviation: How spread out the values are
   - Minimum and Maximum values
   - 25th, 50th (median), and 75th percentiles
3. **Read the output carefully**
   - Look at which columns have large values or very small ones.
   - Spot outliers by checking min and max values.
   - Check if any columns have weird values like 0 or negative numbers.
   - Understand the scale of each column — are they similar or different?
4. **Write down your observations**
   In your notebook or worksheet, write 2-3 sentences about what you observe from this summary.
   Example: "The Amount column has an average of ₹88, but the maximum amount is more than ₹25,000. That means some purchases are very large and rare. The Class column has mostly 0s, so fraud is rare."

Why This Step is Important:

- It helps you get a **quick health check** of your dataset.
- You can **spot problems** like extreme values, imbalances, or weird data early.
- It sets the stage for **data cleaning** or **scaling**, if needed, before machine learning.

Questions for Report

1. What does the mean value of the Amount column tell you about the typical transaction size?
Hint: Think about whether the average amount spent is low, moderate, or high. Is it what you expected?

2. Which column in the dataset shows the highest variation, and what might that tell you?
Hint: Look at the std (standard deviation) values. A high standard deviation means the values in that column are spread out.

3. From the min and max values of the Amount column, what can you say about the range of spending in the dataset?
Hint: How much do the smallest and largest transactions differ? Is there a big gap?

4. What does the Class column's summary (mean, min, max) tell you about how common fraud is in this dataset?
Hint: This column has only 0s and 1s. Think about what a value of 0 or 1 means in this context.

5. Are there any columns that seem unusual or surprising to you when looking at their summary statistics? Explain why.
Hint: Maybe some columns have extreme values, or strange averages. Pick one that caught your attention.

8. Understanding Class Distribution in a Fraud Detection Dataset

Objective of this Step:
To count and understand how many transactions in the dataset are labeled as **fraudulent** and how many are **not fraudulent**. This helps us see how common or rare fraud is, which is very important for building a smart and fair model.

Step-by-Step Instructions for Students:

1. **Use the label column ("Class") to identify frauds:**
    - The dataset has a special column named **"Class"**.
    - It contains two types of values:
        - **0** means the transaction is **Not Fraud**.
        - **1** means the transaction is **Fraud**.
2. **Count how many times each class appears:**
    - You will now count how many **0s** and how many **1s** are there.
    - This tells us how **balanced or imbalanced** the dataset is.
3. **Rename the class labels for easy reading:**
    - Instead of showing 0 and 1, rename them to:
        - 0 → "Not Fraud"
        - 1 → "Fraud"
    - This makes your output more readable and beginner-friendly.
4. **Look at the results and reflect:**
    - You'll likely see that there are **way more "Not Fraud" cases than "Fraud" ones**.
    - That means fraud is **rare**, and your model must work harder to catch those few suspicious transactions.

Why this step is important:
Understanding how your data is distributed is like knowing your battlefield before a mission. If you ignore this and just jump into model building, your model might fail to detect fraud because it's overwhelmed by all the normal data. This step helps you prepare for smarter analysis and better decision-making.

Questions for Report

How many total transactions in the dataset are labeled as "Fraud" and "Not Fraud"?
Hint: Use the class label counts you observed in the previous step.

What does the distribution of fraud vs. non-fraud transactions tell you about the data?
Hint: Think about whether one class is much larger than the other and what that might mean.

Why is it important to know that fraud cases are very few in number compared to normal transactions?
Hint: Consider how this might affect your model's ability to learn or make accurate predictions.

If a model predicted all transactions as "Not Fraud", would it still get a high accuracy? Why might that be misleading?

Hint: Think about the imbalance in the data and how a model could be "right" most of the time but still not useful.

How would you handle the challenge of working with a dataset where fraud is rare? Can you think of one possible strategy?
Hint: Think creatively—could you balance the data? Or focus more on precision or recall?

---

9. Splitting Data into Training and Testing Sets

In this step, you will **separate your dataset** into two parts:

- One part for **training the model**
- Another part for **testing how well the model works**

Think of it like studying for an exam: you first **learn from your notes (training data)** and then **test yourself with new questions (testing data)** to see how well you've learned.
Follow the steps below carefully:

Step-by-Step Instructions:

1. **Identify your target column (output)**
   Look at your dataset and find the column that shows the outcome you want to predict.
   In this case, it is the **"Class"** column, which tells whether a transaction is fraud (1) or not fraud (0).
   Save this column as your **output labels**.
2. **Select all other columns as input features**
   The rest of the columns (like Time, Amount, V1 to V28) are the **inputs** that will help the model predict the output.
   These should be saved separately as your **input data**.
3. **Split the data into two groups**
   Divide your data into:
   - **Training Data** (about two-thirds of the dataset):
     This part will be used to teach the model how fraud looks like in data.
   - **Testing Data** (about one-third of the dataset):
     This part will be used to check how well the model performs on new, unseen data.
4. **Make sure the fraud cases are fairly distributed**
   Because fraud cases are rare, you need to make sure both the training and testing data have a similar mix of fraud and non-fraud cases.
   This keeps your model fair and balanced.
5. **Fix the random seed for repeatability**
   When you split the data, the rows are selected randomly.
   To make sure the same data is used every time you run the project, set a fixed random number.
   This makes your results **reproducible**.

What You've Learned
By the end of this step, your data will be neatly divided into four groups:

- Input data for training
- Output labels for training
- Input data for testing
- Output labels for testing

Now your machine learning model can start **learning from past data** and **testing its ability to make smart predictions**!

Questions for Report

Why do we split the data into training and testing sets?
Hint: Think about how we teach a model and how we check if it has learned well.

What is the purpose of using the 'Class' column as the output (target)?
Hint: Consider what the model is trying to predict—fraud or not fraud.

Why do we keep the input features (like Time, V1–V28, Amount) separate from the output label?
Hint: Remember, inputs are used to make predictions, and the output is what we want to predict.

What is the importance of setting the 'random_state' while splitting the data?
Hint: Think about getting the same split every time you run your code.

Why is it important to use the 'stratify' option when splitting the data in this case?
Hint: Think about the imbalance in fraud vs. non-fraud cases, and how we keep that balance in both training and test data.

10.Scaling the Data for Fair and Balanced Learning

In this step, you'll prepare your data so that all the features (like V1, V2, Amount, etc.) are treated fairly by the machine learning model. Some columns might have very large numbers, while others have small decimals. If we don't fix this, our model might get confused and give more importance to certain features just because of their size. That's why we scale the data.

Instructions:
Step 1: Create a tool to scale your data

- Think of this as a **measuring machine** that helps put all your data values on the same scale.
- It works by finding the **average (mean)** and how spread out (standard deviation) each feature is.

Step 2: Apply this measuring tool to your training data

- You'll **teach the scaling tool** using your training data only. This way, it learns how to center and scale each column correctly.
- Then you **transform** the training data using this knowledge so that all values are balanced around 0 and fall within a similar range.

Step 3: Use the same scaling method on your test data

- Now that your scaling tool knows how to adjust data, you'll use the **same tool** to scale the test data.
- It's important to **not teach the scaler again** using test data. Just apply what it already knows.
- This keeps the test process fair—just like using a fixed set of exam rules for every student.

Why This Step Matters:

- **Models need fairness**: Some numbers might seem more important just because they're big—but size doesn't mean importance. Scaling fixes this.
- **Smooth learning**: Scaling makes training smoother and faster for the model.
- **Better results**: Especially for certain models, like logistic regression or support vector machines, scaling is critical for accurate predictions.

Real-Life Example:
Imagine you're comparing the weight of apples (in kilograms) and the price (in rupees). If you don't convert them to a common scale, the model might think price is more important just because it has bigger numbers. Scaling is like converting everything into the same unit—so your model focuses on **patterns, not just sizes**.

Questions for Report

Why do we need to scale the features in our dataset before training the model?
Hint: Think about what happens when some numbers are very big (like Amount) and others are small (like V1, V2, etc.).

What would happen if we scaled the test data separately, without using the scaler trained on the training data?
Hint: Imagine using a different measuring scale for test data—will it be fair or consistent?

After scaling, what kind of values do we expect to see in our dataset?
Hint: Are they all very big, very small, or centered around something?

Why is it important to apply the same scaling method to both training and testing data?
Hint: Think about fairness and consistency during model evaluation.

Can you think of a real-life example where using values of different scales might create confusion or unfair results?
Hint: Try comparing scores out of 100 and out of 10 without converting them.

11.Build and Test a Logistic Regression Model for Fraud Detection

Step-by-Step Instructions:
Step 1: Load the Logistic Regression Tool
Start by loading the Logistic Regression tool from your machine learning toolbox (scikit-learn). This tool helps us predict outcomes where there are only two possible choices, like fraud vs. not fraud.
Think of it like loading a calculator that only solves yes-or-no questions.

Step 2: Prepare the Model
Now, create a new Logistic Regression model. While doing this, tell it to take enough time to learn properly (by setting the number of training attempts to 1000). This ensures it doesn't give up too early and learns the best pattern from the data.
Imagine you're giving a student 1000 chances to solve a puzzle. The more they try, the better they can learn.

Step 3: Train the Model

Use your training data to teach the model. Show it many examples of past transactions—some that were fraud and some that weren't. The model will try to find patterns that help it make good guesses in the future.
Like teaching a guard dog what a suspicious person looks like by showing lots of photos and stories.

Step 4: Test the Model
Now it's time to check if the model really learned something. Give it new, unseen examples (test data), and ask it to predict whether they are fraud or not. This tests its memory and understanding.
Just like giving a quiz to a student after study time to see if they actually learned.

Step 5: Measure the Accuracy
Finally, compare the model's answers to the correct answers. Find out how many times the model was right, and turn that into a percentage score.
It's like checking how many questions a student got right out of 100 in a test to get their score.

Step 6: Print or Write Down the Result
Display the model's accuracy so you can track its performance. If the percentage is high, your model is doing well! If it's low, don't worry—later you can improve it with more training, better features, or smarter techniques.

What You'll Learn from This Task:

- How to create a machine learning model for binary classification.
- How to teach the model using real-world data.
- How to test the model's ability to make predictions.
- How to understand and measure accuracy in machine learning.

Questions for Report

1. What is the purpose of using a Logistic Regression model in this fraud detection project?
Hint: Think about the type of prediction you are making — is it a number or a category like "fraud" or "not fraud"?

2. Why is it important to split the dataset into training and testing parts?
Hint: What would happen if you trained and tested the model on the exact same data? Would it be fair?

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?
Hint: Think about how rare fraud cases are in the dataset. Can a model still be "wrong" even if the accuracy looks 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?
Hint: Remember how many fraud and non-fraud cases there are. What if the model just guessed the most common class?

5. Suppose your model's accuracy is lower than expected. List two things you might try to improve the model's performance.
Hint: Think about the data or the model settings. Can we prepare the data better? Use a different model?

12.Train and Test a Gaussian Naive Bayes Model

Step 1: Select the Naive Bayes Model
Start by choosing the **Gaussian Naive Bayes** model from your machine learning toolkit. This model is good when your data follows a bell-shaped curve, like many things in real life (such as height, weight, etc.).

Step 2: Create the Model
Now imagine taking a brand-new notebook and writing "Naive Bayes" on the cover. You are preparing it to take notes and learn from your data. This step sets up your model and gets it ready for training.

Step 3: Train the Model with Training Data
Feed the model with your **training data**. This data has examples of past transactions—some were fraud and some were not. The model will now study these examples and try to learn patterns. It's like a student learning from solved examples before taking a test.

Step 4: Test the Model with New Data
Now, give the model **new unseen data** (called test data). Ask it to guess whether each transaction in the new data is fraud or not. This is like giving the student a quiz to see how much they've learned.

Step 5: Check How Well It Did
Check how many answers the model got right by **comparing its predictions with the actual answers**. Calculate the accuracy of the model as a percentage. This shows how good the model is at spotting fraud.

Step 6: Note Down the Result
Finally, write down the model's accuracy result. This number will help you later when you compare it with the results of other models like Logistic Regression or Random Forest.

Tip:
Think of the model like a weather forecaster. It has seen past weather data (training) and now it's trying to predict whether it will rain tomorrow (testing). You evaluate it based on whether its prediction comes true (accuracy).

Questions for Report

What kind of data did you use to train the Gaussian Naive Bayes model? Why is this step important before making predictions?
Hint: Think about what the model needs to learn from.

In your own words, how would you explain the job of the Naive Bayes model in this fraud detection project?
Hint: Imagine explaining it to a friend who is not from a technical background.

What was the accuracy of the Gaussian Naive Bayes model on your test data? Do you think this is good enough for detecting fraud? Why or why not?
Hint: Think about how many fraud cases the model might miss or catch.

---

13. Training a Fraud Detection Model Using LDA (Linear Discriminant Analysis)

Objective:
To help students understand how to train a model that learns patterns in credit card transaction data and then uses that learning to detect fraud using a technique called **Linear Discriminant Analysis (LDA)**.

Step-by-Step Instructions:

Step 1: Understand What You're Doing
You're going to build a smart fraud detector. First, you'll train it using known data—this is like teaching it what fraud and non-fraud transactions look like. Then you'll check if it can accurately predict fraud on new, unseen data.

Step 2: Bring in the Right Tool
Think of LDA as a smart divider. It helps draw a clear boundary between two groups: "Fraud" and "Not Fraud". This tool is especially helpful when your data has lots of variables (features), and you want to reduce the confusion between the two classes.

Step 3: Create the Fraud Detector
Once you've chosen LDA as your tool, your next step is to create the model. This is like building a blank fraud detection device. It doesn't know anything yet, but it's ready to learn.

Step 4: Train the Model (Let It Learn)
Now feed your model the training data. This data has all the details of previous transactions, including which ones were fraud and which ones were safe. The model will study these examples to learn patterns that separate fraud from normal transactions.

Step 5: Test the Model's Skills
Once the model has learned, give it some new transactions that it hasn't seen before and ask it to classify them. This tests how well it learned during training.

Step 6: Check the Score
Now calculate how many times your model got the prediction right. This will give you a percentage—called the **accuracy**. The higher the number, the better the model is at spotting fraud correctly.

Step 7: Record and Reflect
Write down the accuracy score of your model. Think about what the number means. Did your model do a good job? Would you trust it to help a bank detect fraud?

Reminder:

- You're not expected to understand all the math behind LDA.
- Focus on **how the model learns**, **makes predictions**, and **how well it performs**.
- The goal is to practice training and testing a real machine learning model on real data.

Questions for Report

1. What does the LDA model learn from the training data, and why is this learning useful in detecting fraud?
Hint: Think about how the model sees the patterns in transactions marked as fraud or not fraud.

2. What is the accuracy of your LDA model, and what does this number tell you about its performance?
Hint: Look at the percentage shown after the model made predictions on the test data.

3. Was your dataset balanced between fraud and not-fraud cases? Why is this important when training a model?
Hint: You already counted how many fraud vs not-fraud cases you had earlier in the project.

4. Imagine you are a bank manager. Based on the accuracy score of your model, would you feel confident using it to flag suspicious transactions? Why or why not?
Hint: Think like a decision-maker. What would make you trust a model?

5. If you could improve the model, what is one thing you would try next time?
Hint: Think about adding more data, trying a different model, or changing how the model learns.

---

14. Training and Evaluating a Fraud Detection Model Using Decision Tree Classifier

In this step, you will build a new machine learning model to detect fraudulent transactions using a **Decision Tree Classifier**. A decision tree works by asking a series of "yes" or "no" questions about the data to arrive at a final decision — in this case, whether a transaction is fraud or not fraud.

Follow these instructions step by step:

1. **Create a Decision Tree Model**
   Start by creating a blank Decision Tree model. Think of this like preparing a blank chart that will learn how to make decisions based on the training data.
2. **Fit the Model on the Training Data**
   Train the model using the scaled training data (X_train_scaled) and corresponding class labels (y_train). This step is the same as how you trained the models in:
   - **Step 11: Build and Test a Logistic Regression Model for Fraud Detection**, and
   - **Step 12: Training a Fraud Detection Model Using LDA (Linear Discriminant Analysis)**
3. **Predict with the Test Data**
   Once the model is trained, use it to predict outcomes on the test dataset

(X_test_scaled). This means you are asking the model to guess whether each new transaction is fraud or not.

4. **Check Model Accuracy**
   Measure how accurate your model's predictions are by comparing them with the actual results (y_test). Use the same method shown in the Logistic Regression and LDA steps to calculate and display the accuracy percentage.
5. **Write Down Your Accuracy Result**
   Note the accuracy you get from the Decision Tree model. You will later compare it with the other models to decide which one performs the best.

Why This Step Is Important:
This step helps you explore a different type of algorithm (Decision Tree) and understand how it performs in detecting fraud. Decision Trees are simple and explainable, which means banks or companies can easily understand the model's decision logic.

---

15. Build and Test a Random Forest Model for Fraud Detection

In this step, you will train a **Random Forest Classifier**, a powerful model that uses many decision trees working together to make accurate predictions.

What You'll Do:

You'll now build a machine learning model that predicts whether a credit card transaction is **fraudulent or not**, using the **Random Forest algorithm**. This model is known for its **high accuracy** and **reliability**, especially when handling complex data.

Step-by-Step Instructions:

1. **Use the same scaled training and test datasets** (X_train_scaled and X_test_scaled) you created in:
   - Step **11: Build and Test a Logistic Regression Model for Fraud Detection**
   - Step **12: Training a Fraud Detection Model Using LDA (Linear Discriminant Analysis)**
2. **Build a Random Forest model** using the appropriate ML tool or function. Think of it as assembling a team of decision trees that will each vote on whether a transaction is fraud or not.
3. **Train your model** on the training data.
   - This is the learning phase, where your model sees past examples of fraud and not fraud.
4. **Make predictions** on the test data using the trained Random Forest model.
   - Now, your model sees new, unseen transactions and decides which ones are likely fraud.
5. **Calculate the accuracy** of your predictions.
   - This tells you how well your model is performing.
   - Accuracy is the percentage of correct predictions.
6. **Record your accuracy result** and compare it with the models from Step 11 (Logistic Regression), Step 12 (LDA), Step 13 (Decision Tree)

Why This Step Is Important:

- Random Forest is like asking **many smart people for their opinion**—then going with the majority. It helps reduce mistakes made by any one decision tree.
- It's very effective in real-world problems like fraud detection, where patterns are complex and tricky to detect.
- Comparing this model's accuracy with your earlier models helps you **evaluate which approach works best** for your project.

Questions for Report

What is a Random Forest model, and why do we use it for fraud detection?
Hint: Think of how multiple decision trees work together like a team giving majority votes.

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?
Hint: Look at the percentage accuracy values you calculated for each model.

What might be the advantage of using multiple models (like Random Forest) instead of just one decision tree?
Hint: Consider the benefit of combining multiple opinions rather than relying on a single one.

If your Random Forest model made wrong predictions, what could be the possible reasons for that?
Hint: Think about things like data imbalance, or the way the features are used by the model.

Do you think Random Forest is a good model for this project? Why or why not?
Hint: Use your understanding of accuracy and decision-making from this step to answer.

16.Train and Evaluate a Support Vector Machine (SVM) Model

In this step, you will build a machine learning model called **Support Vector Machine (SVM)**. This model helps us figure out whether a transaction is fraudulent or not, based on patterns it learns from the training data.

Step-by-Step Instructions:

1. **Get the Right Tool (SVM Classifier):**
    - You'll first bring in a tool called Support Vector Machine from your machine learning library.
    - This tool is great at finding a clear boundary between fraud and non-fraud transactions.
2. **Create the Model:**
    - Imagine you're designing a smart assistant who can tell if a transaction looks suspicious.
    - You'll now create that assistant by setting up an SVM model.
3. **Teach the Model Using Training Data:**
    - Next, you'll train this model by feeding it real examples of transactions.
    - It will learn what makes a transaction normal and what makes it suspicious (fraudulent).
4. **Test the Model on New Data:**

- After training, you will test how well your assistant performs on new, unseen data.
- This is like checking if your assistant can correctly identify fraud in new cases.

5. **Measure How Good the Model Is:**
   - You'll now calculate how many correct answers the model gave out of the total.
   - This result is called "accuracy" and is usually shown as a percentage (like a test score).
6. **Display the Accuracy:**
   - Finally, you will print out this accuracy score so you know how well your model performed.
   - A high accuracy means your model is good at detecting fraud. A low accuracy means you might need to improve your model or data.

Example:
Think of your model like a **security guard at a mall**. You teach the guard how to spot shoplifters based on past behaviors. After the training, the guard watches new people entering the mall and tries to predict who might be a shoplifter. You then check how often the guard was right. That's your accuracy score!

Questions for Report

What is a Support Vector Machine (SVM) and how does it help in detecting fraud in this project?
Hint: Think about how the model separates fraud and non-fraud cases based on patterns.

How did the SVM model perform on the test data? What was the accuracy score?
Hint: Look at the output you got after running the model and find the accuracy percentage.

17.Using K-Nearest Neighbors (KNN)

Step 1: Load the KNN Tool
You will begin by loading the K-Nearest Neighbors (KNN) algorithm from your machine learning toolbox. KNN is a classification technique that looks at the "nearest neighbors" to make a decision. In our case, it will help us predict whether a transaction is fraudulent or not.

Step 2: Decide How Many Neighbors to Check
When setting up the KNN model, you'll specify how many nearby data points (neighbors) the model should check before making a decision. In this task, we use 5 neighbors. That means it will look at the 5 most similar past transactions to decide whether the new transaction is fraud.

Step 3: Train the Model on Known Data
Now, it's time to teach the model using data we already know — that is, transactions labeled as fraud or not fraud. The model will "remember" the patterns found in this training data.

Tip: Make sure the data you're using has been scaled — this helps ensure all features (like time or amount) are treated equally.

Step 4: Use the Model to Make Predictions
After training, you'll give the model new transactions it hasn't seen before. It will use what it learned to predict whether each new transaction is fraudulent or not.

Step 5: Check How Accurate the Predictions Are
To find out how well the model performed, you'll compare its predictions to the actual results. This will give you a percentage accuracy score — for example, 97% accuracy means the model got 97 out of 100 predictions correct.

Step 6: Display the Results
Finally, show the model's accuracy on screen. This helps you understand how reliable the KNN model is when used for fraud detection.

Outcome
By completing this step, students will successfully build a KNN model, use it for prediction, and measure how well it performs. This teaches them how similarity-based learning can help detect fraud.

Questions for Report

What does the KNN model do when it needs to classify a new transaction?
Hint: Think about how the model looks at similar past transactions before deciding.

What value of 'K' (number of neighbors) did we use in our project, and how might changing this number affect the results?
Hint: More neighbors can mean more generalization, fewer neighbors can mean more sensitivity.

18.Compare the Accuracy of All Machine Learning Models

Objective:
You have trained several machine learning models (like Logistic Regression, Decision Tree, Random Forest, etc.) to detect fraud. Now it's time to **summarize and compare** how well each model performed.

Step-by-Step Instructions:

Step 1: Create a Summary Table
- Start by making a table (like a marksheet).
- In the **first column**, write the names of all the machine learning models you have built.
- In the **second column**, write the accuracy score of each model. These are the results you got after testing each model using your test dataset.

Step 2: Fill In the Scores
- Go back to each model you've trained.
- Copy the accuracy result (the one printed after print("Accuracy of...")).

- Place each accuracy value next to the correct model name in the table.

Step 3: Sort the Table
- Arrange your table in **descending order**—from the model with the highest accuracy to the one with the lowest.
- This helps you quickly see which model is **performing the best**.

Step 4: Interpret the Results
- Now that the models are sorted, look at the top model.
- This is the **best performing model** in your project. It is most accurate in detecting fraudulent transactions.
- Make a note of which model it is and its accuracy.

Step 5: Reflect
- Think about why that model might be performing better than others.
- Is it because it's a more complex model like Random Forest?
- Or maybe a simpler one like Logistic Regression worked better due to the data?

Tip:
This final table is very useful when you **write your project report**. You can include it in your conclusion section to show a clear comparison and your final recommendation.

Questions for Report

1. Which model achieved the highest accuracy in your results table?
Hint: Look at the top of your sorted table where the accuracy is highest.

2. What could be the reason this model performed better than the others?
Hint: Think about whether the model is simple or complex. Does it handle large data well?

3. Which model performed the worst in your results? Why do you think that happened?
Hint: Consider if the model struggled with imbalanced data or didn't generalize well.

4. If you had to recommend one model to a company for fraud detection, which one would you choose and why?
Hint: Don't just pick the most accurate—also think about speed, simplicity, and ease of use.

5. What is one thing you learned from comparing all the models together?
Hint: Think about how accuracy alone doesn't tell the full story or how model choice depends on the problem.

19.Evaluate Your Model Using a Confusion Matrix

Objective:
Learn how to check how many correct and incorrect predictions your model made — especially for fraud detection.

Step-by-Step Instructions:
Step 1: Check model predictions

- After training your model and predicting the test data, you now have a list of predictions (model guesses) and a list of actual answers (ground truth).
- You are going to compare these two lists to see where the model was right and where it was wrong.

Step 2: Use the confusion matrix tool

- Use the special function called *confusion matrix* to compare actual values and predicted values.
- This function will organize the results into four groups:
  - **Correct Not-Fraud predictions**
  - **Wrong Fraud predictions**
  - **Missed fraud cases**
  - **Correctly caught frauds**

Step 3: Read the confusion matrix output

- The matrix looks like a 2x2 table.
- It shows the number of times the model was right or wrong in predicting fraud and not fraud.
- This helps you understand if the model is:
  - Catching fraud cases well (True Positives),
  - Missing fraud cases (False Negatives),
  - Raising false alarms (False Positives),
  - Or safely skipping non-fraud cases (True Negatives).

Step 4: Think like an investigator

- Are too many fraud cases being missed?
- Is your model too cautious?
- Would a real company be happy with these results?

Step 5: Plan improvements

- If your model missed many frauds, consider trying other models, changing parameters, or balancing the dataset better.
- Use this step as a checkpoint to decide what to try next.

Questions for Report

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.

What do the four numbers in the confusion matrix represent in your model's results?
Hint: Think of them as correct/incorrect guesses about fraud and not fraud.

Did your model correctly identify most of the fraud cases? Why is this important in real life?
Hint: Think about the consequences of missing a fraud.

Were there any False Positives in your model? What could be the impact of these in a real business?
Hint: A False Positive is when your model wrongly labels a normal transaction as fraud.

What steps can you take if your model is missing too many fraud cases (False Negatives)?
Hint: Consider trying different models or adjusting model settings.

Based on your confusion matrix, would you say your model is ready to be used in a real bank? Why or why not?
Hint: Think about both accuracy and mistakes in predictions.
