FIRST PROBLEM STATEMENT :
AI-POWERED FRAUD DETECTION SYSTEM

DESCRIPTION:

Develop an AI model capable of detecting fraudulent transactions in real-time. Use historical transaction data to train the model to identify anomalies and flag suspicious activities.

OBJECTIVE:

Successfully implement and deploy a model that can accurately detect fraudulent transactions with minimal false positives.

OUTCOME:

QUANTITATIVE (70 POINTS) :

A. **Precision (30 POINTS):**
   Measures how many of the transactions flagged as fraud are actually fraudulent. It's crucial to minimize false positives.

B. **Recall (Sensitivity) (30 POINTS):**
   Indicates how well the model detects actual fraudulent transactions. This metric ensures that most fraud cases are caught.

C. **F1 Score (5 POINTS) :**
   Balances precision and recall, providing a single metric to evaluate the model's performance.

D. **AUC-ROC (5 POINTS):**
   Measures the trade-off between true positive rate and false positive rate. A higher AUC indicates better model performance.

QUALITATIVE (30 POINTS):

- **Feature Importance (5 POINTS):**

  Teams should analyze which features contribute most to the model's predictions.

- **Insights from Data (10 POINTS):**

  Participants should provide a detailed analysis of the dataset, highlighting interesting patterns or insights that informed their model development.

- **Progress During Hackathon (5 POINTS):**

  Evaluate the progress made by the team throughout the hackathon.

- **Innovation & Documentation (10 POINTS):**

  Assess the quality and effectiveness of the presentation.

SUBMISSIONS:

Jupyter notebook with the code and the prediction csv file.

DOCUMENTATION:

The evaluation metrics, the model details & configurations

SECOND PROBLEM STATEMENT:
API-DRIVEN FINANCIAL DATA AGGREGATOR

DESCRIPTION:
Develop an API that aggregates and normalizes financial data from various sources, offering a unified interface (dashboard) for accessing and analyzing this information.

OBJECTIVE:
Create an API that simplifies access to diverse financial data, enabling developers to build comprehensive financial applications. The solution should include a dashboard that visualizes the data through multiple graphs. Dashboard should be configurable in real time.

OUTCOME:

- An API or set of APIs capable of fetching financial data from the provided datasets.
- A dashboard that utilizes these APIs to generate and display various graphs.

SUBMISSION:

- GitHub repository link
- Screenshots or videos of the dashboard, or a deployment link (e.g., Vercel or Netlify)

EVALUATION METRICS:

1. **User-Friendliness (40 POINTS) :** The dashboard should be intuitive and easy to navigate for users.
2. **Update and Time Responsiveness (10 POINTS):** The system should handle updates efficiently and respond quickly to user interactions.
3. **Authentication and Secure API (10 POINTS):** Brownie Point
   The application should implement robust authentication methods and ensure that the API is secure against unauthorized access.
4. **Database (20 POINTS):** Use your personal accounts
5. **Deployment (10 POINTS):** Brownie Point
   Local host works but try deploying to github pages/netlify/vercel

6. **Innovation & Documentation (5 POINTS)**:
   Assess the clarity and effectiveness of the presentation, including the ability to communicate the features and functionality of the solution.
7. **Progress During Hackathon (5 POINTS):**
   Evaluate the progress and development achieved by the team throughout the hackathon.