PROGRAMMING MACHINE LEARNING APPLICATION

1. Problem You Are Trying to Solve

The main goal of this project is to build machine learning models that can accurately classify and predict the outcome based on the given dataset features. By analyzing the dataset, we aim to understand the relationship between input variables and the target variable and identify which machine learning algorithms perform best on this data.

2. How You Set Up Your Data for Analysis

We began by importing the dataset and checking for missing values or incorrect data entries. The dataset was cleaned to ensure it was ready for analysis. Features and the target variable were separated for modeling purposes. To improve model performance, we applied **feature scaling** using **StandardScaler**, especially important for distance-based models like KNN and SVM. After preprocessing, the data was split into **training (70%) and testing (30%) sets** to evaluate the models properly.

3. Pre-Exploratory Data Analysis (EDA)

In our EDA, we analyzed the basic structure of the dataset, including data types, value counts, and distribution of the target classes. We checked for class imbalances and plotted correlation heatmaps to understand feature relationships. Visualizations such as pair plots and histograms were created to observe the spread of data and possible patterns. This step helped us select the most relevant features for our models and informed the choice of algorithms.

4. Models Used and the Results

We implemented multiple machine learning classification models to compare their performance:

- **Logistic Regression:** A linear model that provided a baseline accuracy. It performed well but struggled with complex patterns.
- **K-Nearest Neighbors (KNN):** A distance-based model that classified data based on nearby samples. It worked well but was sensitive to the choice of 'K' and scaling.
- **Decision Tree:** A tree-based model that was easy to interpret and provided good results but tended to overfit on the training data.

- **Support Vector Machine (SVM):** Offered high accuracy with kernel tricks, effectively handling non-linear data.
- **Multi-Layer Perceptron (MLP):** A neural network that captured complex relationships and performed decently on the dataset.
- **Voting Classifier (Ensemble Model):** Combined predictions from the above models using soft voting. This ensemble approach improved the overall accuracy and reduced model bias and variance.

  Results: Among all models, the ensemble Voting Classifier performed the best, providing balanced accuracy and robustness by leveraging the strengths of individual models. Accuracy scores were compared using visualizations to showcase model performances.

5. How You Could Have Done Better / Future Improvements

If the results were not as expected or could be improved, the following steps could be considered:

- Perform **hyperparameter tuning** using Grid Search or Randomized Search to optimize each model.
- Apply **feature engineering** to create new meaningful features that could improve model predictions.
- Try **dimensionality reduction techniques** like PCA to reduce noise and improve performance.
- Use **cross-validation** for more reliable model evaluation and to avoid overfitting.
- Test with more complex ensemble techniques like **Stacking** or **Boosting (XGBoost/LightGBM)** for potential accuracy improvements.
- Collect more data or balance the dataset further if class imbalance exists.