SMIT –DATA Science and Machine Learning Project

Data Analysis and Machine Learning on Spotify Tracks Dataset

– **Objective:**
The goal is to predict whether a song will be popular based on audio features such as danceability, energy, and loudness using classification techniques.

– **Assignment Breakdown:**

Step 1: Load the Data

Task:

- Download the dataset from Kaggle. Load it into a Jupyter notebook using Pandas.

Expected Output:

- A loaded dataset (spotify_data) with an overview using .info(), .head(), and .describe() to familiarize yourself with the columns.
- Hint: Look for columns like popularity, danceability, energy, loudness, and speechiness.

Step 2: Data Cleaning

Task:

- Handle missing values (NaN) in the dataset.
- Drop irrelevant columns like track_name, artist_name, or release_date (as they may not contribute to predicting popularity).
- Convert categorical features (if any) into numerical representations.

Expected Output:

- Cleaned dataset ready for analysis.
- Handle missing values using .dropna() or by filling in missing data.
- Categorical features (if any) encoded into numerical values.

Step 3: Exploratory Data Analysis (EDA)

Task:

- Use visualizations to explore the dataset.
- Check the distribution of song features like danceability, energy, tempo, etc.

- Investigate the relationship between the features and the target variable (popularity).
- Use histograms, scatter plots, and boxplots to visualize the relationships.

Expected Output:

- Visualizations (at least 4) showing trends, such as:
- Correlation between energy and popularity.
- Boxplot of popularity across different loudness levels.
- Pairplot to visualize feature relationships.

Step 4: Feature Engineering

Task:

- Create a new binary column based on the popularity feature, which classifies songs as popular (popularity > 60) or not popular.
- Create additional features that might be useful for prediction (e.g., normalize tempo or combine energy and loudness into a new feature).

Expected Output:

- A new binary is_popular column (1 if popular, 0 otherwise).
- Additional engineered features that could improve model performance.

Step 5: Splitting the Data

Task:

- Split the dataset into a training set and test set (80% training, 20% test) using train_test_split from sklearn.model_selection.

Expected Output:

- Two datasets: one for training and one for testing the model.

Step 6: Applying a Machine Learning Model

Task:

- Apply a machine learning classification model. Start with Logistic Regression or Random Forest.
- Train the model on the training set and predict on the test set.

Expected Output:

- Model predictions on the test set.
- Summary of the model training process.

Step 7: Model Evaluation

Task:

- Evaluate the model's performance using metrics like accuracy, precision, recall, F1-score.
- Use a confusion matrix to assess model predictions.
- Plot a ROC curve to check the performance at different thresholds.

Expected Output:

- Accuracy score, precision, recall, F1-score for the model.
- Confusion matrix and ROC curve.

Step 8: Model Improvement

Task:

- Tune the model's hyperparameters using GridSearchCV to improve performance.
- Try different algorithms like Gradient Boosting or XGBoost.

Expected Output:

- Improved model performance metrics compared to the base model.
- Insights from hyperparameter tuning (if applicable).

Submission Requirements:

- A Jupyter notebook file (.ipynb) with all steps documented and explained.
- A PDF or markdown report summarizing the key findings from the analysis and model evaluation.
- Include visualizations and interpretation of model results.

Deliverables:

Jupyter Notebook with detailed code, comments, and explanations for each step.

Summary report with a brief explanation of the findings from EDA and model results.