Documentation for Phishing Email Classification Code

Steps in the Code

1. Importing Libraries

Key Python libraries are imported to handle data, visualization, preprocessing, and machine learning tasks:

- pandas: For data manipulation.
- matplotlib.pyplot: For data visualization.
- numpy: For numerical computations.
- collections.Counter: To count occurrences of data.
- sklearn.preprocessing: For scaling data.
- sklearn.model_selection: For splitting datasets.
- sklearn.ensemble: For machine learning models like Random Forest and Gradient Boosting.
- sklearn.feature_extraction.text: For converting text data to numerical data using TF-IDF.
- sklearn.pipeline: For creating machine learning pipelines.
- sklearn.metrics: For evaluating model performance.

2. Data Loading and Initial Inspection

The dataset is loaded using:

pd.read_csv(r"E:\Courses\...ML\Gradiant descent\Grediant\Phishing_Email.csv")

- The Unnamed: 0 column is inspected and later removed as it does not contribute to the analysis.
- Missing values are checked using isna() and dropped using dropna().

3. Data Exploration

- The distribution of the Email Type column is analyzed using value_counts().
- A bar chart is created to visualize the distribution with custom colors for Phishing Email and Safe Email.

4. Data Balancing

- To handle class imbalance, undersampling is performed:
    - All instances of Safe Email are sampled to match the count of Phishing Email.

- The balanced dataset is created by concatenating the sampled data.

5. Data Preprocessing

- The Unnamed: 0 column is dropped.
- Features (X) are the email text, and labels (y) are the email types.
- The dataset is split into training and testing sets using train_test_split() with 70% data for training and 30% for testing.

6. Random Forest Classifier

- A pipeline is created to combine TF-IDF vectorization and the Random Forest model:
- classifier = Pipeline([
- ("tfidf", TfidfVectorizer()),
- ("classifier", RandomForestClassifier(n_estimators=50, max_depth=40))
-])
- The pipeline is trained using fit() and predictions are made on the test set using predict().
- Performance metrics such as classification report, accuracy score, and confusion matrix are calculated.

7. Gradient Boosting Classifier

- A similar pipeline is created for the Gradient Boosting Classifier:
- GBClassifier = Pipeline([
- ("tfidf", TfidfVectorizer()),
- ("classifier", GradientBoostingClassifier(n_estimators=100, random_state=0))
-])
- The model is trained and tested, and the accuracy is printed using metrics.accuracy_score().

Visualization and Outputs

1. **Bar Chart for Email Type Distribution:**
    - A custom color-coded bar chart is displayed to show the counts of phishing and safe emails.
2. **Performance Metrics:**
    - The classification report and accuracy scores for both models are printed.

- Key metrics include precision, recall, F1-score, and overall accuracy.

Key Takeaways

- **Data Imbalance Handling:** Undersampling ensures that the model learns from a balanced dataset.
- **TF-IDF Vectorization:** Converts textual data into numerical representations for machine learning models.
- **Pipeline Usage:** Simplifies the workflow by combining preprocessing and model training.
- **Model Comparison:** Two different classifiers (Random Forest and Gradient Boosting) are implemented and their performance is compared.
