NLP Project Report: Sentiment Analysis & Spam Detection

1. Introduction

Project Objective: This project aims to perform a multi-class classification task on a social media sentiment dataset. The initial exploration revealed a highly complex and granular set of over 270 unique sentiment labels. To simplify the problem and make it more tractable for modeling, the project was strategically pivoted to a binary classification task: distinguishing between legitimate messages (`ham`) and irrelevant or promotional messages (`spam`) based on the sentiment text.

Libraries Used: The project leverages a powerful stack of Python libraries for data manipulation, text processing, and machine learning:

- **Data Handling & Computation:** `pandas`, `numpy`, `joblib`
- **Natural Language Processing (NLP):** `nltk` (for tokenization, stopwords, lemmatization), `re` (regex)
- **Machine Learning & Feature Engineering:** `scikit-learn` (for models, metrics, preprocessing, pipelines, grid search)
- **Visualization & Utilities:** `wordcloud`, custom functions from a local `Custom_func` module.

---

2. Data Understanding

Source: The dataset is loaded from a CSV file named `"3) Sentiment dataset.csv"`.

Initial Inspection: The dataset contains **732 entries** and **15 columns** initially. After removing two redundant index columns (`Unnamed: 0.1`, `Unnamed: 0`), the working dataset has **13 features**.

Key Features:

- `Text`: The primary feature containing the social media post text for analysis.
- `Sentiment`: The original target variable with 279 unique, granular sentiment labels (e.g., "Joy", "Excitement", "Anger", "Fear").
- `Timestamp`: Date and time of the post (converted to datetime format).
- `User`, `Platform`, `Hashtags`: Metadata about the post.
- `Retweets`, `Likes`: Engagement metrics.
- `Country`, `Year`, `Month`, `Day`, `Hour`: Temporal and geographical metadata.
- `Spam`: **The new binary target variable** created during preprocessing (`ham` vs. `spam`).

Initial Data Quality:

- **Missing Values:** The `df.info()` output shows no null values in any column.
- **Duplicates:** The dataset contains 20 duplicate entries (`df.duplicated().sum()`). The decision to keep or remove these would be based on further analysis but is noted here.

---

3. Data Preprocessing & Feature Engineering

Several critical preprocessing steps were undertaken to prepare the data for analysis and modeling:

1. **Column Cleaning:** Redundant index columns were dropped.
2. **Data Type Correction:** The `Timestamp` column was converted to a datetime object for proper time-series analysis.
3. **Target Variable Transformation:** This was the most significant preprocessing step.
   - The original `Sentiment` column contained 279 unique, often noisy, labels (e.g., `' Positive '` with extra spaces, `'Curiosity'`, `'Boredom'`).
   - A custom function `map_emotion()` (imported from `Custom_func.Functions`) was applied to categorize these granular sentiments.
   - Another custom function, `check_spam()`, was used to map these broader categories into a new binary column, **`Spam`**.
   - **Class Distribution:**
     - **ham:** 652 entries (89.1%)
     - **spam:** 80 entries (10.9%) This indicates a significant class imbalance, which will need to be addressed during modeling (e.g., using class weights, SMOTE).
4. **Text Preprocessing Setup:** The NLTK library was configured for downstream text processing tasks, including:
   - Downloading `punkt` for tokenization.
   - Downloading `stopwords` to remove common, uninformative words.
   - Downloading `wordnet` for lemmatization (reducing words to their base form).

---

4. Exploratory Data Analysis (EDA)

Target Analysis (`Spam` vs. `ham`): The key insight is the severe class imbalance, with `ham` messages outnumbering `spam` messages by a ratio of approximately 8:1.

Text Analysis (Word Clouds): Word clouds were generated to identify frequent keywords in each class, revealing distinct linguistic patterns:

- **Frequent Keywords in Spam:** `[('new', 14), ('excitement', 9), ('surprise', 7), ('school', 7), ('adventure', 5), ...]`
  - **Insight:** Spam messages often contain words associated with novelty, promotion, and events, like "new," "excitement," "school," and "adventure."
- **Frequent Keywords in Ham:** `[('life', 34), ('new', 29), ('like', 25), ('dreams', 25), ('feeling', 24), ('heart', 24), ...]`
  - **Insight:** Legitimate (`ham`) messages use more personal, emotional, and introspective language, such as "life," "feeling," "heart," "joy," and "world."

This stark contrast in vocabulary provides a strong linguistic signal that a model can learn to distinguish between the two classes effectively.

---

5. Modeling Methodology

Intended Approach (As prepared in the notebook): The notebook code sets up a robust, reproducible machine learning pipeline for text classification.

1. **Text Preprocessing Pipeline:** A custom function (presumably defined in `Custom_func.Functions`) will be used within a `FunctionTransformer` to clean the text. This likely includes:
    - Lowercasing
    - Removing punctuation/special characters
    - Tokenization
    - Removing stopwords
    - Lemmatization
2. **Feature Extraction:** `TfidfVectorizer` will be used to convert the cleaned text into a TF-IDF matrix, representing the importance of words in the corpus.
3. **Dimensionality Reduction (Optional):** `TruncatedSVD` is prepared to reduce the dimensionality of the large TF-IDF matrix, which can help with computational efficiency and sometimes improve model performance.
4. **Model Selection & Training:** The code prepares a `ColumnTransformer` and a `Pipeline` to handle the text feature. Multiple classifiers are imported for experimentation:
    - `LogisticRegression` (Linear model)
    - `LinearSVC` (Support Vector Machine)
    - `RandomForestClassifier`, `DecisionTreeClassifier` (Ensemble methods)
    - `KNeighborsClassifier`
5. **Model Evaluation:** The project is set up to use comprehensive metrics from `sklearn.metrics`:
    - `accuracy_score`
    - `precision_score`
    - `recall_score`
    - `f1_score` (Crucial for imbalanced datasets)
    - `classification_report`
6. **Validation Strategy:** `StratifiedKFold` and `GridSearchCV` are imported, indicating the intention to use k-fold cross-validation and hyperparameter tuning to find the best model and avoid overfitting. `train_test_split` will be used for creating hold-out validation sets.

---

6. Conclusion and Future Work

Conclusion: This project successfully transformed a complex, multi-class sentiment analysis problem into a clear and practical binary classification task of identifying spam based on sentiment content. The EDA revealed a strong textual distinction between `ham` and `spam` classes, providing a solid foundation for modeling. A robust machine learning pipeline has been constructed for TF-IDF feature extraction and model training/evaluation.

Future Work:

1. **Address Class Imbalance:** Implement techniques like SMOTE (Synthetic Minority Over-sampling Technique) or adjust class weights in the models to improve performance on the minority `spam` class.
2. **Advanced Feature Engineering:** Experiment with word embeddings (Word2Vec, GloVe, FastText) to capture semantic meaning instead of just TF-IDF statistics.
3. **Deep Learning Models:** Explore deep learning architectures like RNNs (LSTMs, GRUs) or Transformer-based models (BERT, DistilBERT) which can capture complex contextual relationships in text and potentially yield state-of-the-art results.
4. **Hyperparameter Tuning:** Execute the prepared `GridSearchCV` to thoroughly optimize the hyperparameters for each chosen model.
5. **Deployment:** Package the best-performing model into a scalable application (e.g., a REST API using Flask/FastAPI) for real-time spam filtering.