**Spam Email Detection Project Documentation:**

Github link:

*https://github.com/soban39/Email\_Spam\_Detection*

**Project Concept:**

The goal of this project is to develop a machine learning model that automatically classifies emails as spam or non-spam. The model is trained using a labeled dataset to recognize patterns commonly found in spam emails. By automating spam detection, organizations and individuals can reduce distractions and improve efficiency in handling email communications.

**Problem Statement:**

Spam emails are a persistent issue, causing productivity losses and security risks. This project addresses the problem by leveraging Artificial Intelligence to:

* Detect patterns in email text.
* Accurately classify emails as spam or ham using a trained model.

By using machine learning, the project automates spam detection and enhances email filtering systems.

**Dataset:**

* Name: spam.csv
* Source: [UC Irvine](https://archive.ics.uci.edu)
* Description: Contains labeled email messages, where spam denotes unwanted emails and ham denotes legitimate emails.

**Data Statistics:**

* Total Records: 5572
* Spam Emails: 747
* Ham Emails: 4825
* No missing values or null entries were found.
* The dataset was critically evaluated for duplicates and missing values, ensuring a clean and reliable data source for the model.

**Project Steps:**

**1. Data Acquisition**

* Obtained the spam.csv dataset from Kaggle.
* Evaluated the dataset for missing values, duplicates, and class balance.
* Cleaned the data by removing unnecessary columns (Unnamed: 2, Unnamed: 3, Unnamed: 4).

**2. Exploratory Data Analysis (EDA)**

* **Visualizations:**
  + Count plot showing the distribution of spam vs. ham emails.
  + Word clouds highlighting frequently used words in spam and ham emails.
* **Insights:**
  + Common words in spam emails include "win," "offer," and "free."
  + Ham emails often include terms like "project," "meeting," and "team."

**3. Feature Engineering**

* **Techniques:**
  + Cleaned text data by tokenizing and stemming using the Porter Stemmer.
  + Extracted numerical features using TfidfVectorizer.
  + For deep learning models, tokenized and padded sequences with:
    - Vocabulary Size (input\_dim): 8921
    - Maximum Sequence Length (maxlen): 171

*This step ensured compatibility with both traditional ML models and advanced architectures.*

**4. Model Training**

**Logistic Regression:-**

* **Accuracy:**
  + Training Data: 98.5%
  + Test Data: 96.2%

**Random Forest Classifier:-**

* **Accuracy:**
  + Training Data: 98.7%
  + Test Data: 97.4%

**Deep Learning Model (LSTM):-**

* **Architecture:**
  + Embedding Layer: input\_dim=8921, output\_dim=128, input\_length=171
  + Two LSTM layers with 64 and 32 units respectively.
  + Dense Layer with Sigmoid activation.
* **Accuracy:**
  + Test Data: 95.8%

**5. Model Evaluation**

* **Assessed model performance using:**
  + Accuracy: Overall correctness of the predictions.
  + Precision: Proportion of correctly identified spam emails.
  + Recall: Ability to detect all actual spam emails.
  + Specificity: Correct classification of non-spam emails.
* **Confusion Matrix:**
  + Provided a visual comparison of predicted vs. actual classifications.

**6. Bonus Task: Evaluation**

* **Explored potential improvements by:**
  + Training additional models, including Random Forest and LSTM.
  + Experimenting with hyperparameter tuning for deep learning models.

**7. Deep Dive**

* **Deep Dive 4: Feature Engineering**
* Focused on advanced feature extraction using ***TfidfVectorizer.***
* Preprocessed text data by tokenizing, stemming, and removing stop words.
* **Deep Dive 5: Model Training**
* Built and trained a deep learning model (LSTM) for unstructured text data.
* Experimented with a custom architecture to handle sequence data effectively.

**Tools and Libraries:**

* **Programming Language:**
* Python 3.x
* **Libraries**
* Data Handling: pandas, numpy
* Visualization: matplotlib, seaborn
* NLP: nltk, scikit-learn
* Deep Learning: TensorFlow, Keras
* Miscellaneous: WordCloud

*These tools were chosen for their efficiency in handling text data, building models, and visualizing insights.*

**Results and Insights:**

* Logistic Regression provided a robust baseline with high accuracy and precision.
* Random Forest improved test accuracy slightly by leveraging ensemble learning.
* The deep learning model performed comparably but is more suited for larger datasets or unstructured text.

**Trade-offs:**

* Logistic Regression: Simpler and faster, ideal for this dataset.
* LSTM: More computationally expensive, better for more complex data. Fine-tuning could improve performance in real-world applications.

**Resources and References:**

1. [Understanding Spam Filtering and Detection](https://www.geeksforgeeks.org/detecting-spam-emails-using-tensorflow-in-python/)
2. [UCI SMS Spam Dataset](https://archive.ics.uci.edu/datasets)
3. [Text Preprocessing in NLP](https://medium.com/analytics-vidhya/text-preprocessing-for-nlp-natural-language-processing-beginners-to-master-fd82dfecf95)
4. [Logistic Regression in Machine Learning](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
5. [Introduction to LSTMs](https://colah.github.io/posts/2015-08-Understanding-LSTMs/)
6. [Python Libraries Documentation](https://www.tensorflow.org/api_docs)