Spam Mail Prediction Documentation Updated

1 Introduction

This project detects spam emails using machine learning techniques. The goal is to classify emails as spam or legitimate (ham) using text data and engineered features.

2 Exact Imports Used

The notebook uses the following exact imports

- import numpy as np
- import pandas as pd
- from sklearn.model_selection import train_test_split
- from sklearn.feature_extraction.text import TfidfVectorizer for text to numerical data
- from sklearn.linear_model import LogisticRegression
- from sklearn.metrics import accuracy_score

3 Dataset Overview

Typical spam datasets contain email text and a target label where 1 represents spam and 0 represents ham.

- Emails require cleaning to remove noise.
- Datasets may have class imbalance with fewer spam samples.

4 Data Preprocessing

- Convert all text to lowercase.
- Remove punctuation stopwords and special characters.
- Apply tokenization and optionally stemming or lemmatization.
- Transform text into TF IDF vectors using TfidfVectorizer.
- Split dataset into training and test sets using train_test_split.

5 Feature Engineering

- Use TF IDF for body and subject text.
- Add features like email length number of links and presence of keywords.
- Use n grams for phrase level signals.

6 Exploratory Data Analysis

- Analyze class distribution to understand spam ratio.
- Visualize word frequencies for spam vs ham.
- Compare average email length across classes.

7 Model Used

Logistic Regression is used as the main classifier in the notebook.

- Reason: Simple interpretable baseline works well with TF IDF features and is fast to train.

8 Model Training

- Vectorize text using TfidfVectorizer fit on training data.
- Train LogisticRegression on vectorized training set.
- Predict labels on the test set.

Sample code snippet

- vectorizer = TfidfVectorizer()
- X_train_vec = vectorizer.fit_transform(X_train_text)
- model = LogisticRegression()
- model.fit(X_train_vec y_train)
- preds = model.predict(vectorizer.transform(X_test_text))

9 Model Evaluation

- Use accuracy_score to measure overall correctness.
- Also consider precision recall and F1 score for class imbalance.
- Use confusion matrix to inspect types of errors.

10 Key Results

- Logistic Regression with TF IDF often gives solid baseline performance.
- Text cleaning and ngrams heavily impact predictive accuracy.
- Adding simple metadata features can boost results.

11 Conclusion

A simple pipeline with TF IDF and Logistic Regression provides a reliable and interpretable approach to spam detection. Accuracy is a helpful metric but should be complemented by precision recall for production readiness.

12 Future Improvements

- Experiment with transformer based models for deeper text understanding.
- Use cross validation and hyperparameter tuning.
- Address class imbalance with sampling or class weights.
- Deploy the model as an API using Flask or Streamlit.

13 Author

Satyam Gajjar