Project Report:
Author -KANNAN KHOSLA
1668217

Name: Email Spam-Ham Classifier

1. Introduction:

The goal of this project is to develop an Email Spam-Ham Classifier using three different classifiers: Linear Regression, Logistic Regression, and Multinomial Naive Bayes (MNB). The dataset used for this project was obtained from Kaggle in the form of a CSV file named "spam.csv."

Problem Formulation: Spam Classification

Input:

- The input to the model consists of text messages.
- Each text message serves as a sample for the model to analyze.

Output:

- The output is a binary classification indicating whether a given text message is spam (1) or not spam (0).

Dataset:

- The dataset used for training and testing the model was obtained from Kaggle, specifically the 'spam.csv' dataset.
- The 'spam.csv' dataset contains labeled examples of text messages, where each message is tagged as either spam or not spam (ham).

Task Objective:

- The primary goal of the task is to build a machine learning model capable of accurately classifying text messages as spam or not spam.
- The model's performance is evaluated based on accuracy, precision, recall, and F1 score.

2. Dataset Overview:
The dataset consists of labeled email messages, categorized as spam or ham (non-spam). It includes both the text content of the emails and their corresponding labels. The dataset was preprocessed to extract relevant features and prepare it for training the classifiers.

Data Preprocessing Description:

The preprocess_data function reads the "spam.csv" dataset, drops unnecessary columns, and renames relevant columns to 'target' and 'text' for clarity. It encodes the 'target' column using Label Encoding, transforming categorical labels into numerical values. Duplicate rows are removed to ensure data integrity.

The function then calculates additional features related to the text data:

- num_characters: Number of characters in each email message.
- num_words: Number of words in each email message.
- num_sentences: Number of sentences in each email message.

Text transformation is applied using the transform_text function, which performs the following steps:

- Converts text to lowercase.
- Tokenizes the text into words.
- Removes non-alphanumeric characters.
- Eliminates stop words and punctuation.
- Applies stemming using the Porter Stemmer.

The transformed text is then used to create a TF-IDF (Term Frequency-Inverse Document Frequency) representation with a maximum of 3000 features. The dataset is split into training, validation, and test sets using the train_test_split function, ensuring an appropriate distribution of instances in each set.

Data Shapes:

Shape of x_train: (3618, 3000)

Shape of x_test: (776, 3000)

Shape of y_train: (3618,)

Shape of y_test: (776,)

Shape of X_Valid (775, 3000)

Shape of y_Valid (775,)

—----------

Problem formulation : We input any message or email and we run our model which classifies it to be spam or ham.

—----------

3. Classifiers Implemented:

The classifiers were implemented from scratch using mathematical formulas. The training-validation-test infrastructure was established to evaluate the performance of each classifier systematically.

The classifiers were trained and evaluated using the provided dataset. The performance metrics, including MSE, accuracy, precision, recall, and F1 score, were calculated for each classifier on the validation and test sets. The results were used to compare the effectiveness of each algorithm in classifying spam and ham emails.

Linear Regression: Linear regression is a regression algorithm used for predicting a continuous output. In this project, it is adapted for binary classification by setting a threshold of 0.5 on the predicted values.

Logistic Regression: Logistic regression is a widely used algorithm for binary classification. It models the probability of an instance belonging to a particular class.

Multinomial Naive Bayes (MNB): Naive Bayes is a probabilistic algorithm based on Bayes' theorem. The Multinomial Naive Bayes variant is suitable for text classification, making it my first choice for spam-ham classification.

—----------

Hyperparameter Tuning

The models were fine-tuned using a systematic approach for hyperparameter tuning. The key hyperparameters, such as learning rates or alpha and epochs, were optimized to enhance the models' predictive accuracy, precision, recall and f1 scores.

Performance Metrics

The following metrics were used to evaluate the classifiers:

Mean Squared Error (MSE): For regression-based classifiers (Linear Regression).

Accuracy: Percentage of correctly classified instances.

Precision, Recall, F1 Score: Metrics providing insights into the classifier's performance in terms of false positives, false negatives, and overall accuracy.

Results:

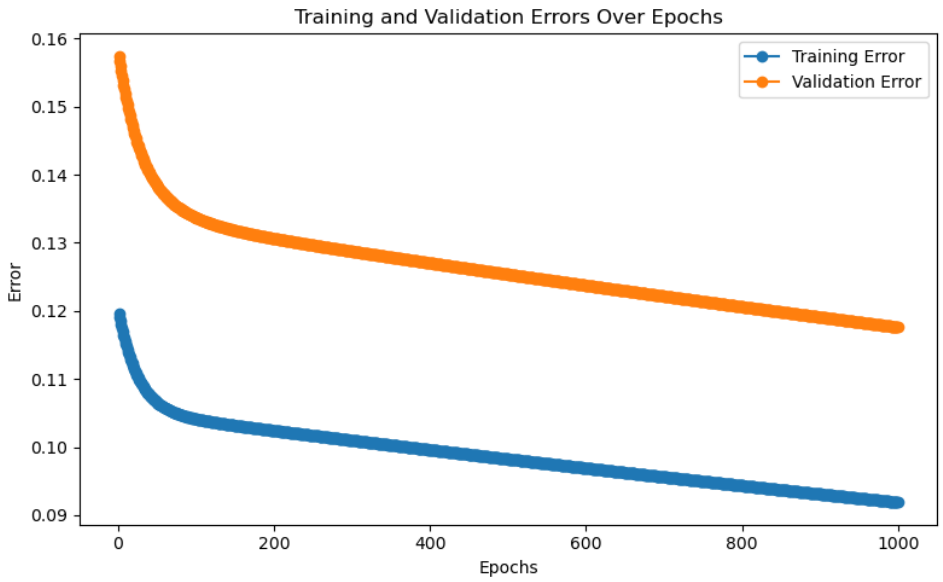
The classifiers were trained and evaluated using the provided dataset. The performance metrics, including MSE, accuracy, precision, recall, and F1 score, were calculated for each classifier on the validation and test sets. The results were used to compare the effectiveness of each algorithm in classifying spam and ham emails.

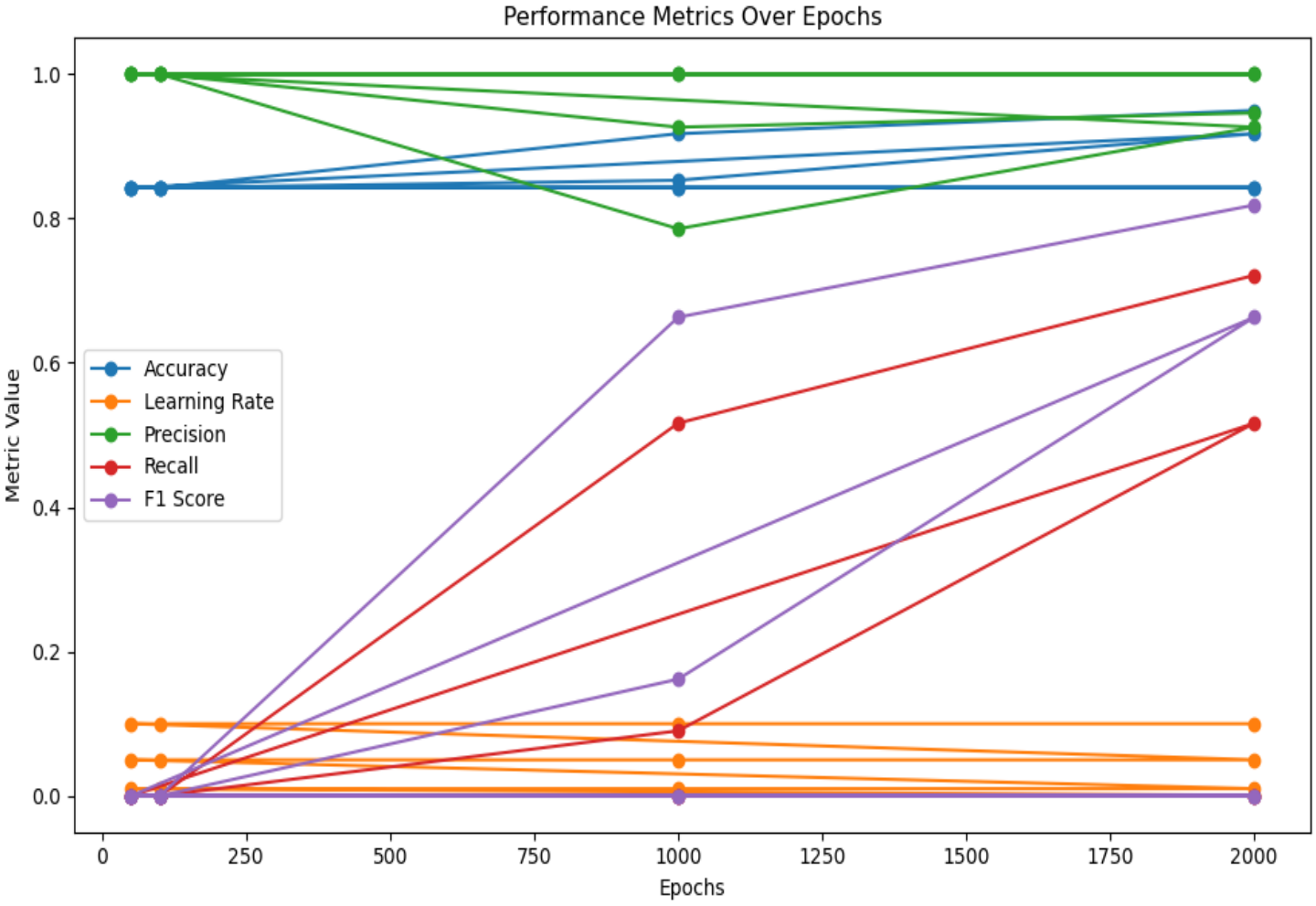
Linear Regression Results

Test Accuracy using linear regression 0.8737113402061856
Validation Accuracy using linear regression 0.8425806451612903
Test MSE using linear regression 0.09635930300098218

After Hyperparameter Tuning
Best Epoch Value: 2000
Best Learning Rate: 0.1
Best Accuracy: 0.9496774193548387
Precision: 0.946236559139785
Recall: 0.7213114754098361
F1 score: 0.8186046511627908

Learning Rate	Epochs	Accuracy	Precision	Recall	F1 Score
0.001	50	0.8426	1.0	0.0	0.0
0.001	100	0.8426	1.0	0.0	0.0
0.001	1000	0.8426	1.0	0.0	0.0
0.001	2000	0.8426	1.0	0.0	0.0
0.01	50	0.8426	1.0	0.0	0.0
0.01	100	0.8426	1.0	0.0	0.0
0.01	1000	0.8426	1.0	0.0	0.0
0.01	2000	0.8426	1.0	0.0	0.0
0.05	50	0.8426	1.0	0.0	0.0
0.05	100	0.8426	1.0	0.0	0.0
0.05	1000	0.8529	0.7857	0.0902	0.1618
0.05	2000	0.9174	0.9265	0.5164	0.6632
0.01	50	0.8426	1.0	0.0	0.0
0.01	100	0.8426	1.0	0.0	0.0
0.01	1000	0.9174	0.9265	0.5164	0.6632
0.01	2000	0.9497	0.9265	0.7213	0.8186


Training and Validation Errors Over Epochs
Training Error
Validation Error
Error
Epochs
0.16
0.15
0.14
0.13
0.12
0.11
0.10
0.09
0
200
400
600
800
1000


Performance Metrics Over Epochs
Accuracy
Learning Rate
Precision
Recall
F1 Score
Metric Value
Epochs
1.0
0.8
0.6
0.4
0.2
0.0
0
250
500
750
1000
1250
1500
1750
2000

Logistic Regression Results

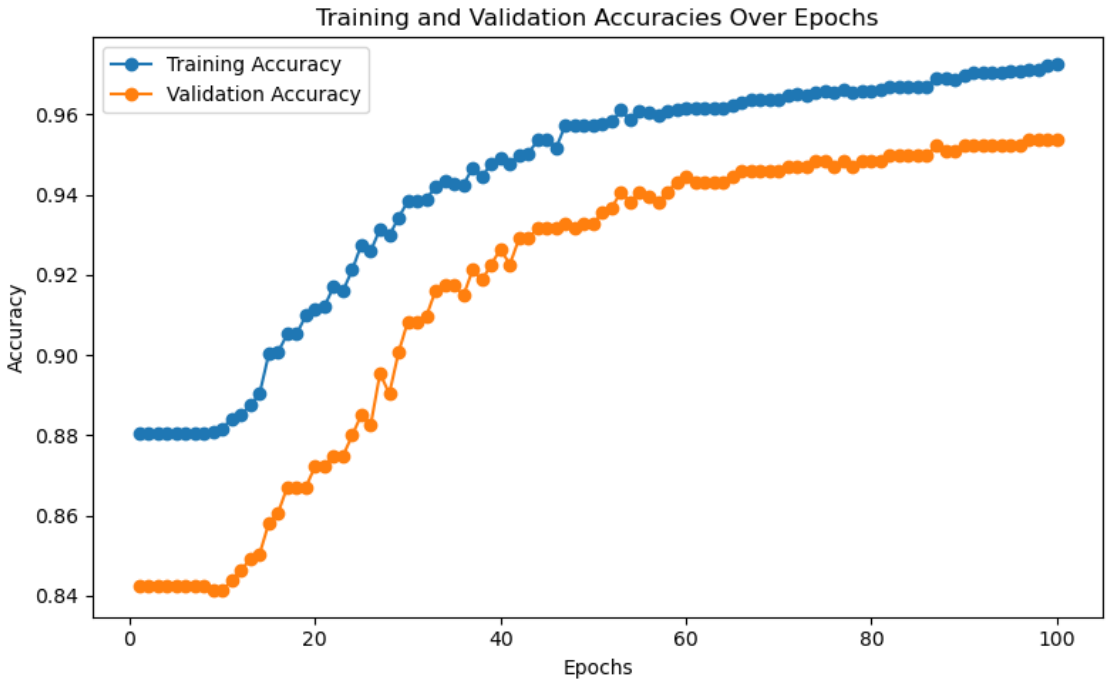
Test Accuracy using logistic regression 0.97036082474226

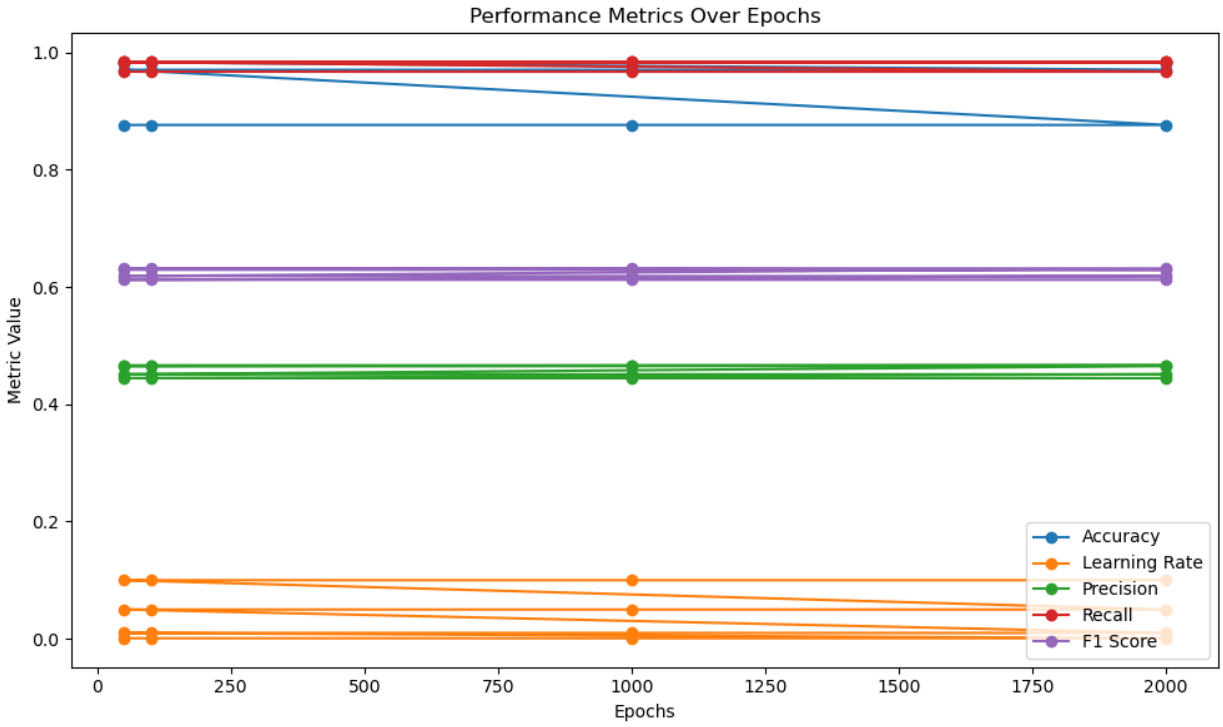
After Hyperparameter Tuning

Best Learning Rate: 0.05
Best Epoch Value: 50
Best Accuracy: 0.9832474226804123
Precision: 0.45112781954887216
Recall: 0.9836065573770492
F1 score: 0.6185567010309279

Learning Rate	Epochs	Accuracy	Precision	Recall	F1 Score
0.001	50	0.8762886597938144	0.466403162055336	0.9672131147540983	0.6293333333333334
0.001	100	0.8762886597938144	0.466403162055336	0.9672131147540983	0.6293333333333334
0.001	1000	0.8762886597938144	0.466403162055336	0.9672131147540983	0.6293333333333334
0.001	2000	0.8762886597938144	0.466403162055336	0.9672131147540983	0.6293333333333334
0.01	50	0.970360824742268	0.46511627906976744	0.9836065573770492	0.631578947368421
0.01	100	0.970360824742268	0.46511627906976744	0.9836065573770492	0.631578947368421
0.01	1000	0.970360824742268	0.46511627906976744	0.9836065573770492	0.631578947368421
0.01	2000	0.970360824742268	0.46511627906976744	0.9836065573770492	0.631578947368421
0.05	50	0.9832474226804123	0.45112781954887216	0.9836065573770492	0.6185567010309279
0.05	100	0.9832474226804123	0.45112781954887216	0.9836065573770492	0.6185567010309279
0.05	1000	0.9832474226804123	0.45112781954887216	0.9836065573770492	0.6185567010309279
0.05	2000	0.9832474226804123	0.45112781954887216	0.9836065573770492	0.6185567010309279
0.1	50	0.9832474226804123	0.4444444444444444	0.9836065573770492	0.6122448979591837
0.1	100	0.9832474226804123	0.4444444444444444	0.9836065573770492	0.6122448979591837
0.1	1000	0.9832474226804123	0.4444444444444444	0.9836065573770492	0.6122448979591837
0.1	2000	0.9832474226804123	0.4444444444444444	0.9836065573770492	0.6122448979591837

Graph showing behavior of train and valid accuracies over epochs


Training and Validation Accuracies Over Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.84
0.86
0.88
0.90
0.92
0.94
0.96
0
20
40
60
80
100


Performance Metrics Over Epochs
Metric Value
Epochs
Accuracy
Learning Rate
Precision
Recall
F1 Score
0.0
0.2
0.4
0.6
0.8
1.0
0
250
500
750
1000
1250
1500
1750
2000

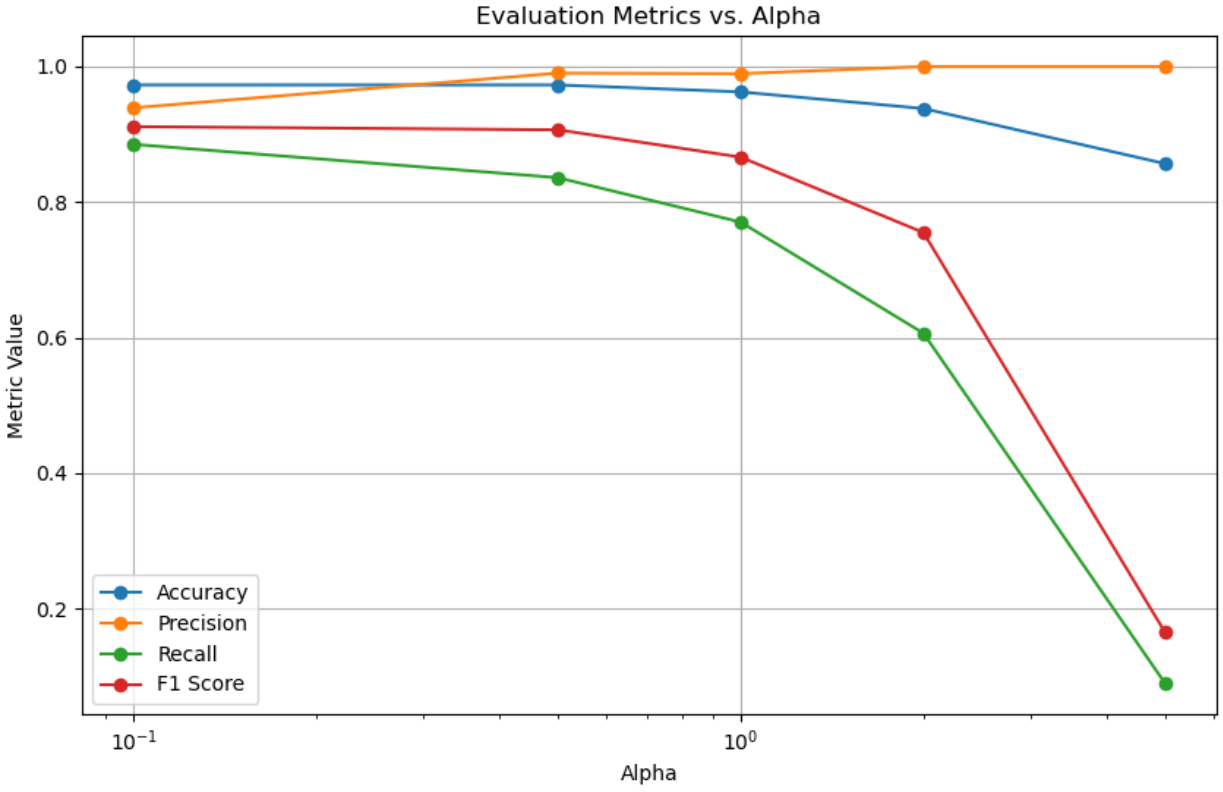
Multinomial Naive Bayes Results

Test Accuracy using multinomial naive bayes 0.9845360824742269

After hyperparameter tuning

Best alpha: 0.1
Best accuracy: 0.9729032258064516
Precision: 0.9391304347826087
Recall: 0.8852459016393442
F1 score: 0.9113924050632911

Alpha	Accuracy	Precision	Recall	F1 score
0.1	0.9729032258064516	0.9391304347826087	0.8852459016393442	0.9113924050632911
0.5	0.9729032258064516	0.9902912621359223	0.8360655737704918	0.9066666666666666
1.0	0.9625806451612903	0.9894736842105263	0.7704918032786885	0.8663594470046082
2.0	0.9380645161290323	1.0	0.6065573770491803	0.7551020408163265
5.0	0.8567741935483871	1.0	0.09016393442622951	0.16541353383458648


Evaluation Metrics vs. Alpha
Metric Value
Alpha
Accuracy
Precision
Recall
F1 Score
1.0
0.8
0.6
0.4
0.2
10^{-1}
10^{0}

Approaches and Baselines:

Linear Regression:

- Hyperparameters:
  - Learning Rate: [0.001, 0.01, 0.05, 0.1]
  - Epochs: [50,100,1000,2000]
- Tuning Process:
  - Systematic search over learning rates and epochs.
  - Evaluated model performance on validation set.
  - Selected hyperparameters based on best validation performance.

Logistic Regression:

- Hyperparameters:
  - Learning Rate: [0.001, 0.01, 0.05, 0.1]
  - Epochs: [50,100,1000,2000]
- Tuning Process:
  - Systematic search over learning rates and epochs.
  - Evaluated model performance on validation set.
  - Selected hyperparameters based on best validation performance.

Multinomial Naive Bayes (MNB):

- Hyperparameters:
  - Alpha Values : [0.1, 0.5 , 1.0, 2.0, 5.0]
- Tuning Process:
  - Systematic search over Alphas
  - Evaluated model performance on validation set.
  - Selected hyperparameters based on best validation performance.

Evaluation Metric

In evaluating the success of my spam classification task, I use several measures to understand how well my model performs. The key metrics I focus on are precision, recall, F1 score, and accuracy.

In my case, I've prioritized accuracy as the main metric for success. Accuracy gives me an overall sense of how well my model performs in distinguishing between spam and non-spam messages.

After experimenting with different classifiers, I found that Multinomial Naive Bayes (MNB) consistently provides the highest accuracy. As a result, I've chosen MNB as my classifier for obtaining the final results.

TESTING

To test the code, run main.py and input any text message when prompted. The model will then predict whether the given message is a real (non-spam) or spam message based on the trained classifier.

Open a terminal or command prompt.
Navigate to the directory containing your main.py file.
Run the file

When prompted, enter a text message you'd like to test.

The model will make a prediction and display the result, indicating whether the input message is classified as real or spam.

Ensure that your environment has the necessary dependencies installed and that the model has been trained before running the testing script.