Title:
Email Spam Detection

Author:
Barbara Schmitz, Mounika Palthya, Srikanth Banoth

Course Name:
COMP – 7150-002 Fundamentals of Data Science

Instructor:
Salim Sazzed

Date:
12/01/2023

Abstract:
The project aims to develop a robust email spam classification model through the implementation of machine learning techniques. Leveraging the Email Spam Detection Dataset (classification) dataset and scikit-learn library, the study involves the application of Logistic Regression, Naive Bayes (Multinomial), Support Vector Machines with linear, polynomial (degree 3), radial, and sigmoid kernels, and Multi-layer Perceptron (MLP) are employed. The TF-IDF vectorization technique is used in preprocessing. Findings highlight the comparative performance of these models and the hyperparameter-tuned MLP.

Introduction:
Email spam remains a persistent challenge in the digital landscape, necessitating effective classification methods to distinguish between legitimate and unsolicited emails. Addressing the challenge of email spam, this research aims to develop an accurate spam classification model due to the ubiquity of email communication in personal and professional realms.

Methods:

1. Data Acquisition and Preprocessing:

The dataset, sourced from Kaggle.com, titled “Email Spam Detection Dataset (classification)” (Dhakad, 2021), was initially loaded into a Pandas DataFrame using the read_csv function. The dataset included two columns, 'Target' and 'Email,' where 'Target' represented the email type ('ham' or 'spam'), and 'Email' contained the text of the emails.

The initial dataset was examined, and three unnamed columns were dropped, resulting in a cleaner structure. Column names were renamed to 'Target' and 'Email' for clarity. The 'Target' column was then mapped to numerical values, with 'ham' represented as 0 and 'spam' as 1.

2. Exploratory Data Analysis:

An exploratory data analysis was conducted to gain insights into the dataset. The size of the data was determined to be 5572 entries. The proportions of spam and ham emails were analyzed, revealing a data imbalance with 13.41% spam and 86.59% ham emails.

To visualize the data imbalance, a bar plot was generated, illustrating the count of ham and spam emails. Additionally, an analysis of email lengths was performed, with histograms depicting the distribution of email lengths for both spam and ham categories.

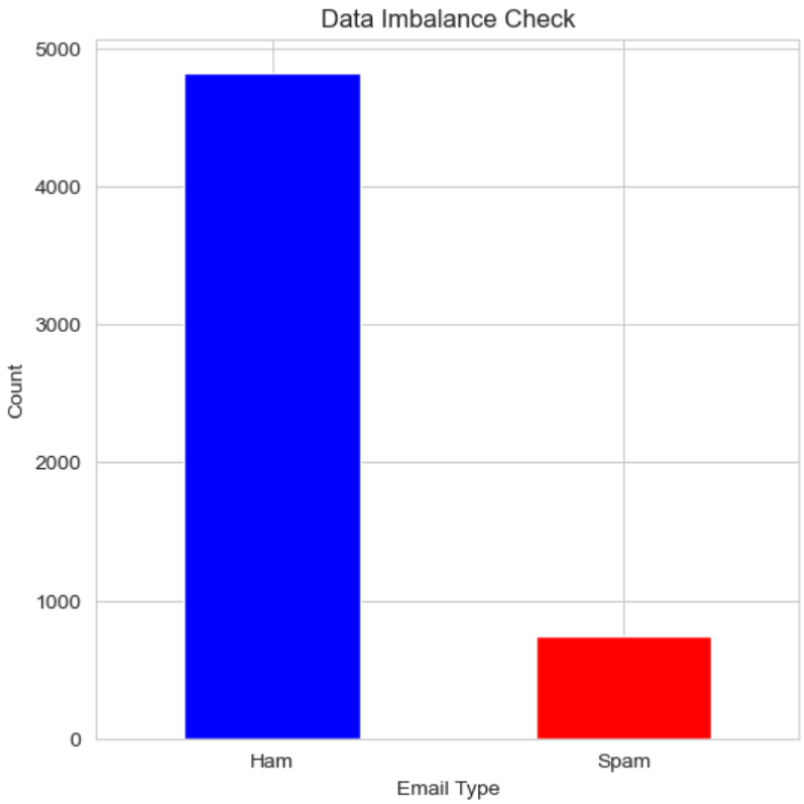


Figure 1: Data Imbalance Check

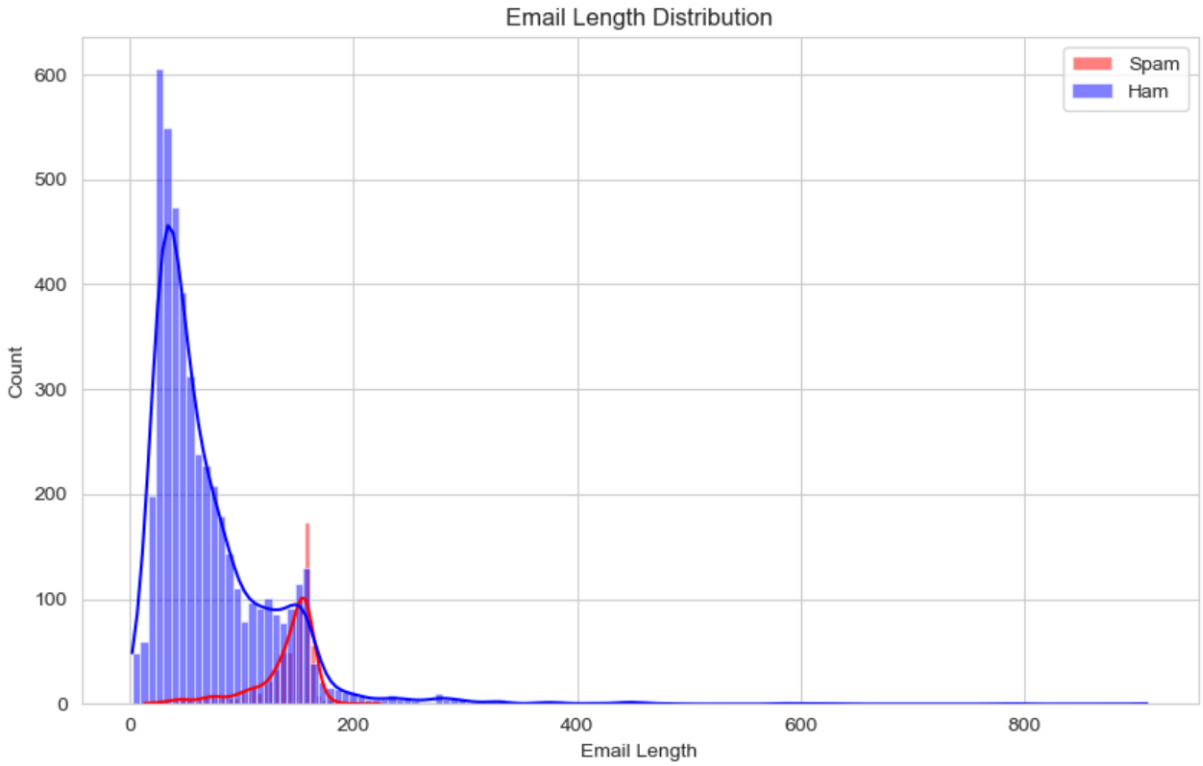


Figure 2: Email length Distribution

3. TF-IDF Vectorization and Word Frequency Analysis:

The text data in the 'Email' column was subjected to TF-IDF vectorization using the scikit-learn TfidfVectorizer with English stopwords. The TF-IDF matrix was constructed, and feature names were obtained.

Word frequency analysis was conducted to identify the top 20 words in both spam and ham emails based on TF-IDF scores. Visualization of the results was accomplished using bar plots.

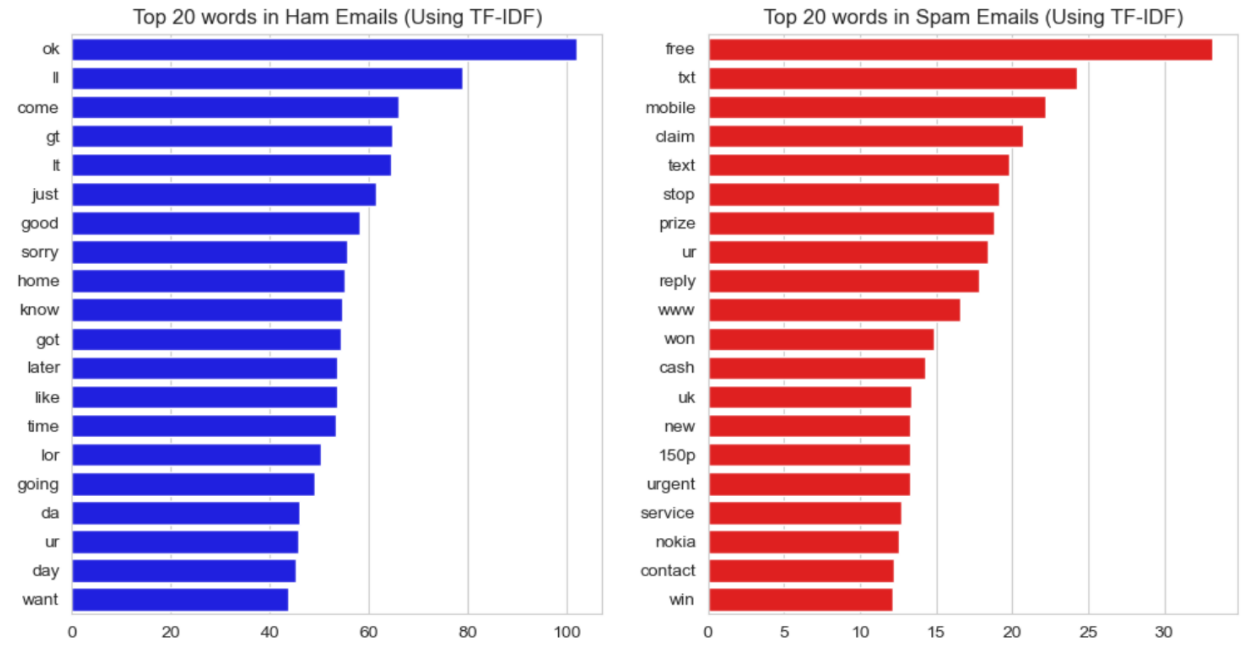


Figure 3: Top 20 words in Ham and Spam Emails Using TF-IDF)

4. Text Preprocessing:

A text preprocessing function was defined to convert text to lowercase, remove punctuation and special characters, and eliminate extra whitespaces. The training and validation sets were split using the train_test_split function, and the preprocessing function was applied to the 'Email' column for both sets.

5. Machine Learning Models:

The preprocessed data was then fed into various machine learning models, including logistic regression, Naive Bayes (Multinomial), Support Vector Machines (SVM) with multiple kernels, and Multi-layer Perceptron (MLP). The models were evaluated and compared based on performance metrics such as accuracy, precision, recall, and F1 score. The best performing model was hyper tuned using the GridSearchCV package and the test data was evaluated on the estimated optimal parameters of the best performing model using 3-fold cross validation.

5.1 Logistic Regression:

Logistic Regression is a statistical model used for binary classification, estimating the probability of an instance belonging to a particular class. It employs the logistic function to map input features to probabilities within the range [0, 1] (Hastie et al., 2009).

$$P(Y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n)}}$$

Where:
$P(Y = 1)$ is the probability that the dependent variable YY is equal to 1.
e is the base of the natural logarithm.
$b_0, b_1, b_2, \ldots, b_n$ are the coefficients.
$X_1, X_2, \ldots, X_n$ are the independent variables.

5.2 Multinomial Naive Bayes:

Multinomial Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It is particularly suitable for discrete data and is commonly used in text classification. The algorithm assumes that features are conditionally independent given the class and estimates probabilities using a multinomial distribution (Manning et al., 2008).

$$P(Y = c_k|X_1, X_2, \dots, X_n) = \frac{count(X_1, c_k) + \alpha}{\sum_{i=1}^{n} count(X_i, c_k) + \alpha \cdot sizeofvocabulary} \times \dots \times \frac{count(X_n, c_n) + \alpha}{\sum_{i=1}^{n} count(X_i, c_k) + \alpha \cdot sizeofvocabulary} \times P(Y = c_k)$$

Where:

$P(Y = c_k|X_1, X_2, \dots, X_n)$ is the probability of class c_k given the features $X_1, X_2, \dots, X_n$

$count(X_i, c_k)$ is the count of feature X_i in instances of class c_k
α is the smoothing parameter (Laplace smoothing).
$size\ of\ vocabulary$ is the total number of unique features in the dataset.

5.3 Support Vector Machine:

Support Vector Machine is a versatile supervised learning algorithm for classification and regression tasks. It works by finding a hyperplane that best separates data points in a high-dimensional space. Different kernel functions, including linear, polynomial, radial, and sigmoid, allow SVM to handle complex relationships between features (Cortes & Vapnik, 1995).

5.3a Linear Kernel

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \langle x, x_i \rangle + b\right)$$

Where:
$f(x)$ is the decision function
x is the input vector
x_i is are the support vector
y_i are the class labels
α_i are the Lagrange multipliers
b is the bias term
$\langle x, x_i \rangle$ denotes the dot product

5.3b Polynomial Kernel Degree 3

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\langle x, x_i \rangle + c)^d + b\right)$$

Where:
Other variables same as above
c is a constant
d is degree of polynomial

5.3c Radial kernel

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \exp\left(-\frac{||x - x_i||^2}{2\sigma^2}\right) + b\right)$$

Where:
Other variables same as above
$||x - x_i||^2$ is the Euclidean squared distance
σ is the width parameter for the RBF kernel

5.3d Sigmoid Kernel

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \tanh(\alpha\langle x, x_i \rangle + c) + b\right)$$

Other variables same as above

5.4 Multi-layer Perceptron:

Multilayer Perceptron is a type of artificial neural network designed for complex pattern recognition tasks. Comprising multiple layers of interconnected nodes, MLP uses forward and backward propagation to learn intricate relationships within data, making it effective for various applications (Goodfellow et al., 2016).

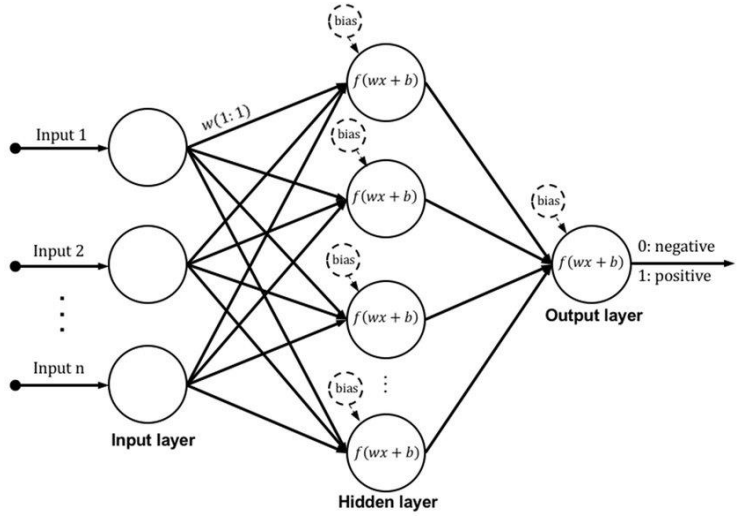


Figure 4: Multilayer Perceptron Architecture
Description: The MLP diagram showcases input neurons, weights, hidden layer with bias and activation function, and output layer for binary classification. 0 denotes a negative outcome, while 1 indicates a positive one. (Abad et al., 2018, Figure 1)

In configuring the default multilayer perceptron model from the sci-kit learn library, key parameters are set as follows:

- Activation Function: Default 'relu,' introducing non-linearity.

1. **Hidden Layers:**
    - Default: (100,)
    - Explanation: The default setting indicates a single hidden layer with 100 neurons. This configuration strikes a balance between model complexity and computational efficiency. A single hidden layer is often effective for capturing patterns in moderately complex data.
2. **Learning Rate:**
    - Default: 'constant'
    - Explanation: The default learning rate is set to 'constant,' meaning that the step size during optimization remains consistent throughout the training process. This choice provides stability during model training, especially when the data has consistent patterns.
3. **Learning Rate Initialization:**
    - Default: 0.001
    - Explanation: The initial learning rate is set to 0.001. This parameter defines the size of the steps taken during optimization at the beginning of the training process. A smaller initial learning rate helps prevent overshooting the optimal values.
4. **Max Iterations:**
    - Default: 200
    - Explanation: The default maximum number of iterations (epochs) for training is set to 200. During each iteration, the entire dataset is passed through the network. This parameter influences how many times the model learns from the entire dataset.
5. **Solver:**
    - Default: 'adam'
    - Explanation: The default optimization algorithm is 'adam,' a widely used stochastic gradient-based optimizer. 'Adam' efficiently adapts learning rates for each parameter, making it suitable for various types of datasets. It is known for its effectiveness and quick convergence.

5.5 Hypertuned Multi-layer Perceptron:

In the pursuit of model optimization, GridSearchCV was employed for hyperparameter tuning, enhancing overall performance. Key parameters subjected to hypertuning include:

1. **Hidden Layer Sizes:**
   - Choices: (50,), (100,), (150,)
   - Description: Specifies the number of neurons in the hidden layer(s). Multiple configurations are tested to find the optimal size for capturing complex patterns.
2. **Activation:**
   - Choices: 'relu', 'tanh', 'logistic'
   - Description: Determines the activation function for the hidden layers. 'Relu' introduces non-linearity, 'tanh' squeezes output between -1 and 1, and 'logistic' applies the logistic sigmoid function.
3. **Learning Rate:**
   - Choices: 'constant', 'invscaling', 'adaptive'
   - Description: Governs the step size during optimization. 'Constant' maintains a fixed learning rate, 'invscaling' gradually decreases it, and 'adaptive' adjusts dynamically based on convergence.
4. **Learning Rate Initialization:**
   - Choices: 0.001, 0.01, 0.1
   - Description: Sets the initial learning rate, influencing the size of steps taken during optimization. Different values are explored to find the most effective starting point.
5. **Max Iterations:**
   - Choices: 100, 200, 300
   - Description: Determines the maximum number of iterations or epochs for training the model. Testing various iterations helps identify the optimal number for convergence.
6. **Solver:**
   - Choices: 'adam', 'sgd', 'lbfgs'
   - Description: Specifies the optimization algorithm. 'Adam' is a popular stochastic gradient-based optimizer, 'sgd' is stochastic gradient descent, and 'lbfgs' is a quasi-Newton method. Testing different solvers explores their impact on model performance.

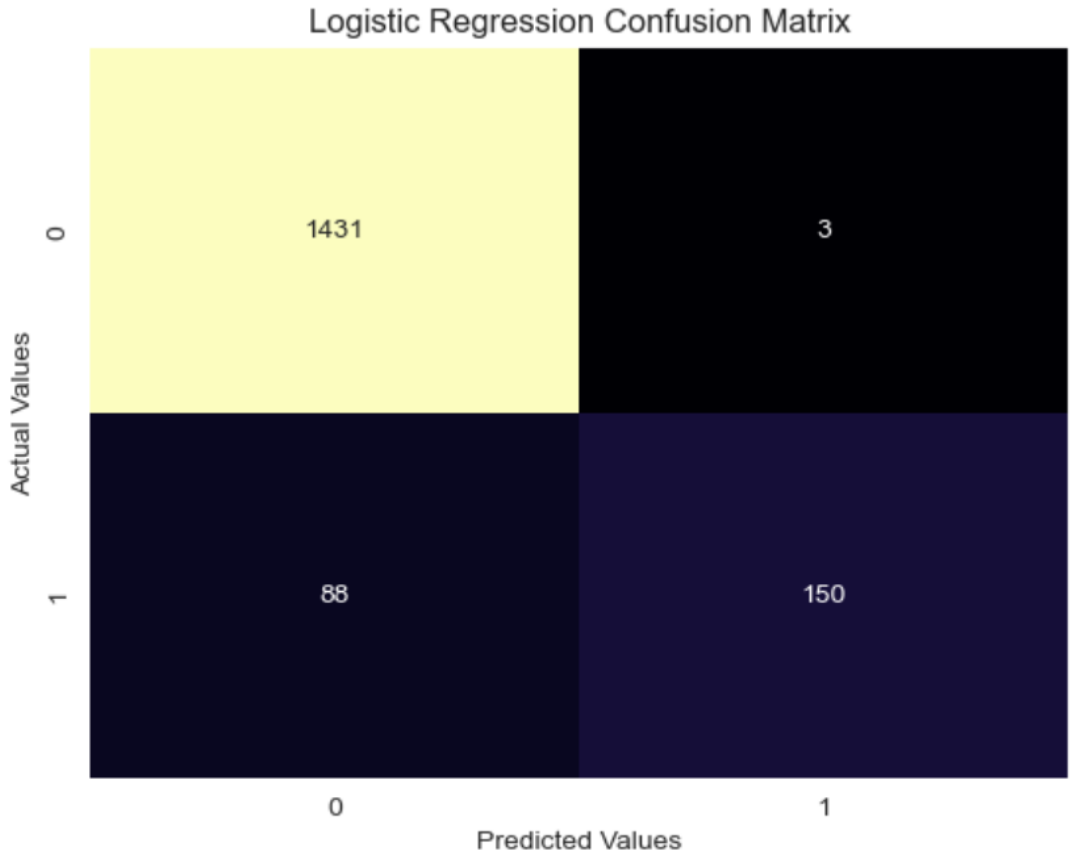
By exploring various combinations within these parameter spaces, GridSearchCV systematically refines the model's architecture, leading to improved results and increased effectiveness in capturing intricate patterns within the data.

Results:

Logistic Regression:

Logistic Regression: Accuracy Score: 94.56%

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1434
1	0.98	0.63	0.77	238
accuracy			0.95	1672
macro avg	0.96	0.81	0.87	1672
weighted avg	0.95	0.95	0.94	1672

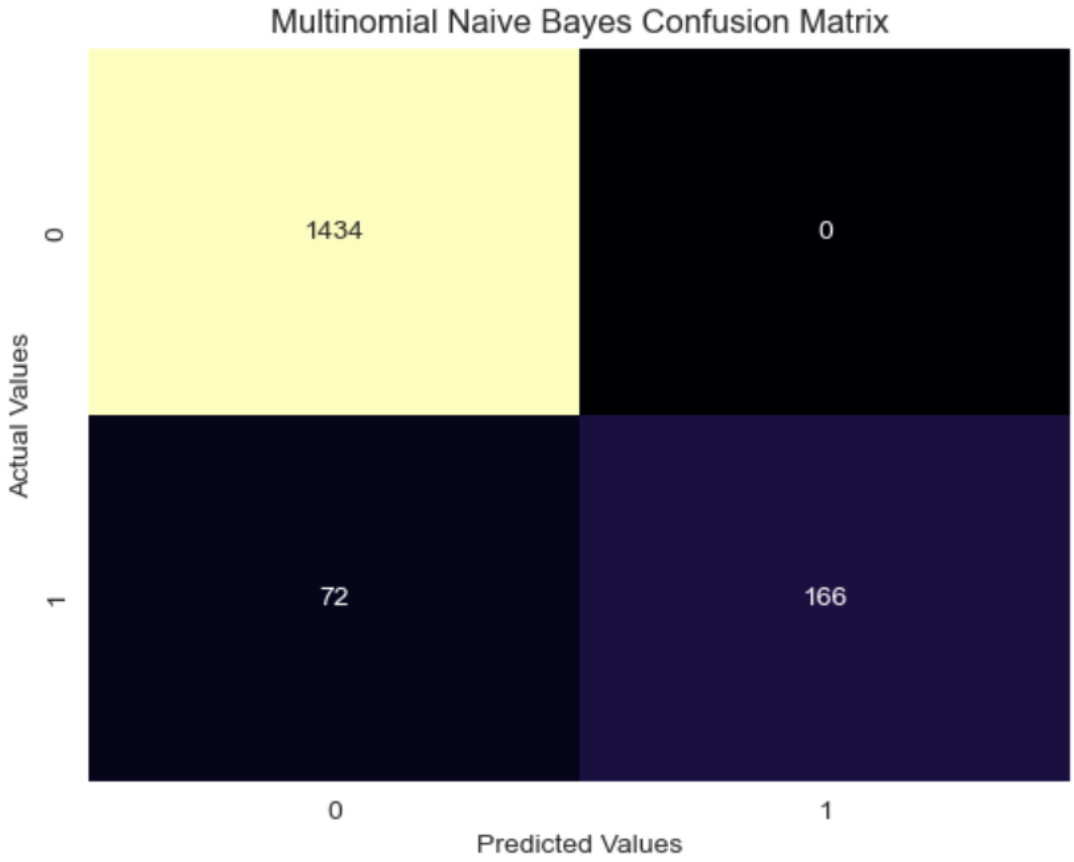


The Logistic Regression model attains an accuracy score of 94.56%, accurately predicting class labels for most instances. It excels in identifying "ham" instances (Class 0), demonstrating high precision, recall, and F1-score, indicating effective classification. For "spam" (Class 1), the model achieves 98% precision but has a lower recall of 63%, suggesting it may miss some spam instances. The weighted average F1-score of 94% reflects a good balance between precision and recall. In summary, the model is robust in identifying "ham" instances, while there is potential for improvement in capturing all "spam" instances.

Multinomial Naïve Bayes:

Naive Bayes: Accuracy Score: 95.69%

	precision	recall	f1-score	support
0	0.95	1.00	0.98	1434
1	1.00	0.70	0.82	238
accuracy			0.96	1672
macro avg	0.98	0.85	0.90	1672
weighted avg	0.96	0.96	0.95	1672

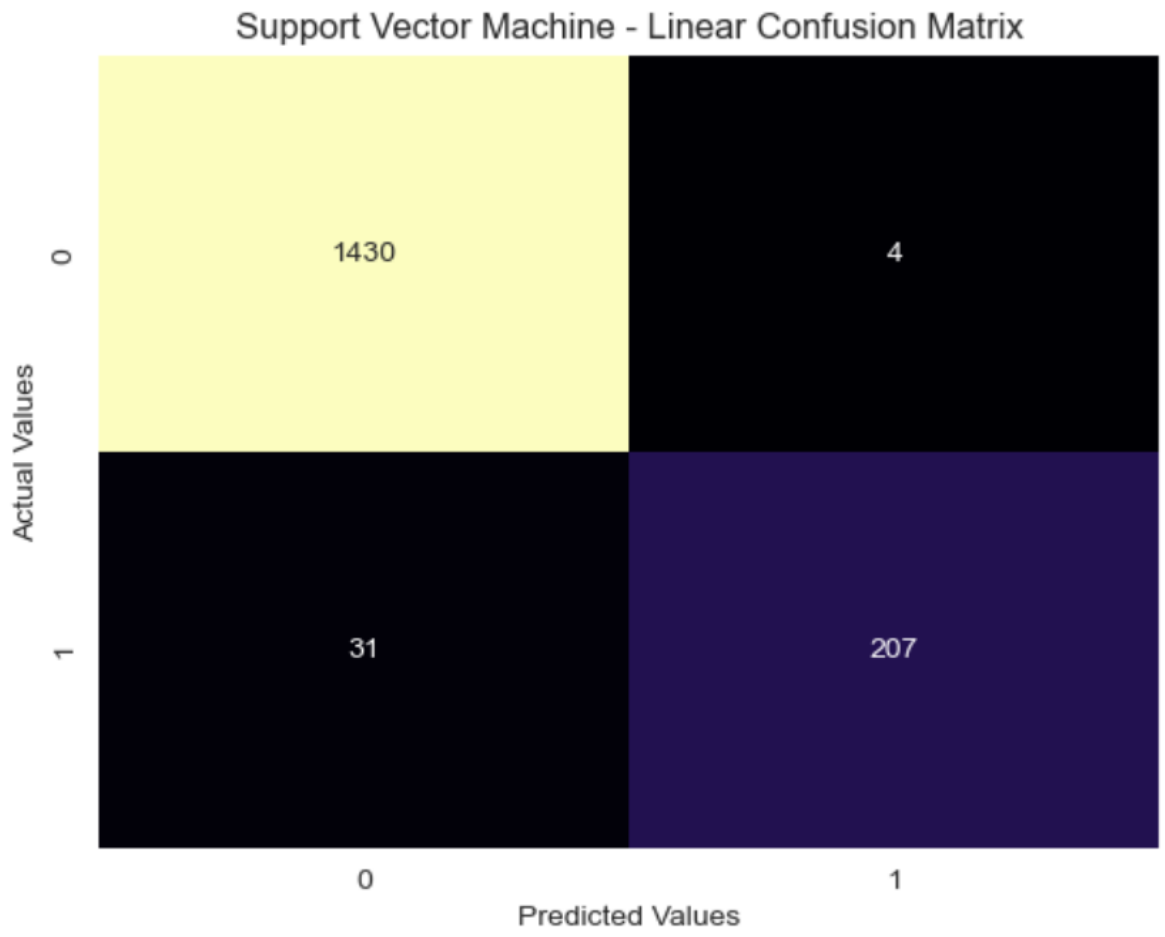


The Naive Bayes model demonstrates strong performance in accurately identifying "ham" instances (Class 0), achieving high precision, recall, and F1-score. For "spam" (Class 1), the model attains 100% precision, indicating perfect correctness in predicting spam instances, and a 70% recall, showcasing a notable ability to capture spam instances. The weighted average F1-score of 95% signifies a well-balanced performance, considering support for both classes. In summary, the Naive Bayes model exhibits robust performance, particularly in correctly identifying "ham" instances, and maintains strong metrics for "spam" classification.

Support Vector Machine, Linear Kernel:

Support Vector Machine - Linear: Accuracy Score: 97.91%

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1434
1	0.98	0.87	0.92	238
accuracy			0.98	1672
macro avg	0.98	0.93	0.95	1672
weighted avg	0.98	0.98	0.98	1672



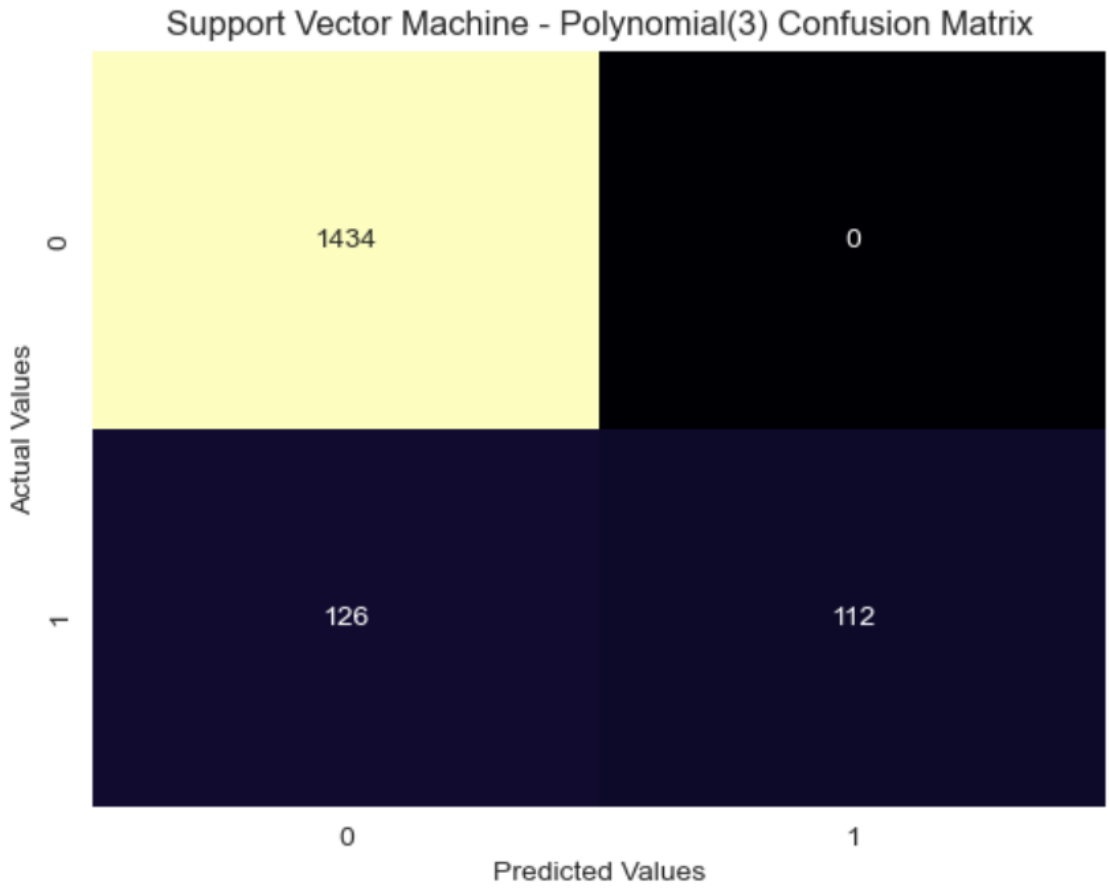
The Support Vector Machine - Linear model exhibits exceptional performance with high precision, recall, and F1-scores for both "ham" and "spam" classes. For "ham" (Class 0), the model achieves a precision and recall of 98% and 100%, respectively, accurately identifying instances. Regarding "spam" (Class 1), the model demonstrates 98% precision and 87% recall, indicating a strong ability to predict and capture spam instances. The weighted average F1-score of 98% reflects excellent overall performance, maintaining a balanced trade-off between precision and recall across both classes. In summary, the linear SVM model excels in accurately

distinguishing between "ham" and "spam," showcasing high accuracy and a harmonious balance between precision and recall.

Support Vector Machine, Polynomial (3) Kernel:

Support Vector Machine - Polynomial(3): Accuracy Score: 92.46%

	precision	recall	f1-score	support
0	0.92	1.00	0.96	1434
1	1.00	0.47	0.64	238
accuracy			0.92	1672
macro avg	0.96	0.74	0.80	1672
weighted avg	0.93	0.92	0.91	1672

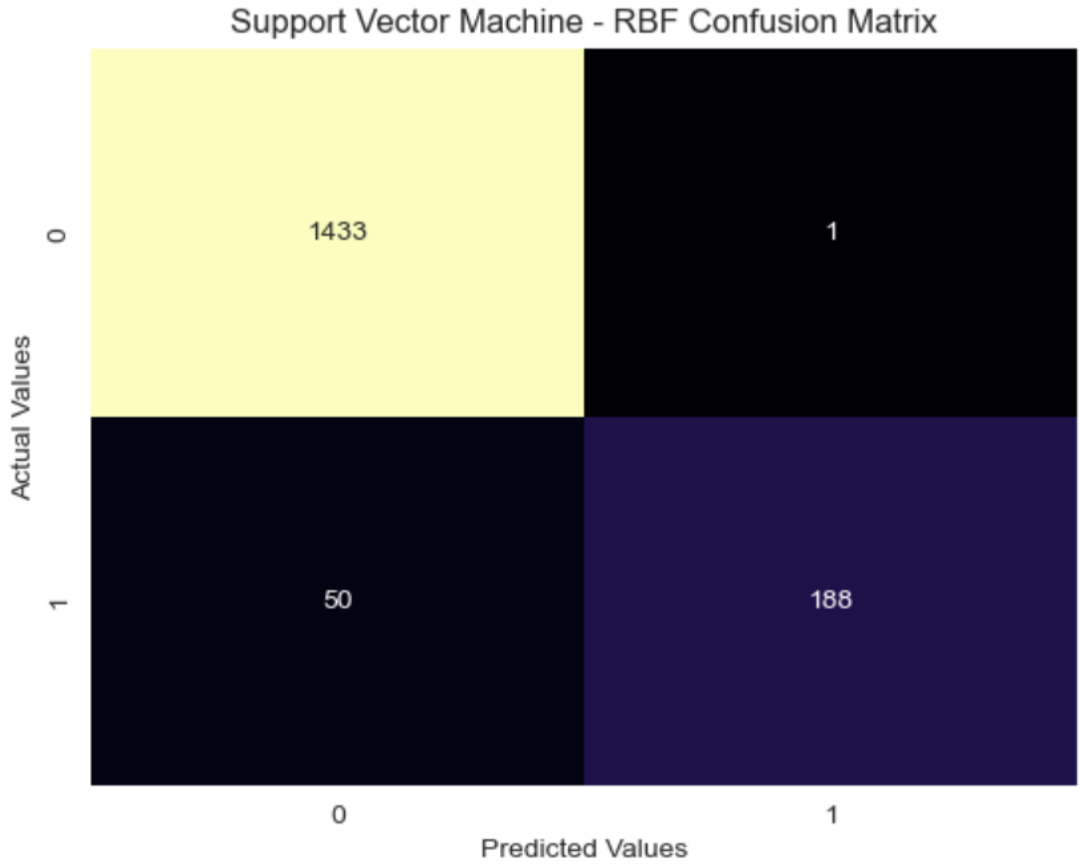


The Support Vector Machine - Polynomial (degree 3) model exhibits robust performance in recognizing "ham" instances with 92% precision and perfect recall. However, its effectiveness diminishes in capturing "spam" instances, reflected in the lower 47% recall. The model attains a good overall performance with a weighted average F1-score of 91%, excelling in "ham" identification but showing limitations in capturing "spam." The macro average F1-score of 80% underscores its relative proficiency in "ham" recognition, accompanied by challenges in capturing "spam" instances. In summary, the model performs well in identifying "ham" instances but falls short in effectively capturing instances of "spam," resulting in a lower overall accuracy compared to the linear SVM model

Support Vector Machine, Radial Kernel:

Support Vector Machine - Radial Basis Function: Accuracy Score: 96.95%

	precision	recall	f1-score	support
0	0.97	1.00	0.98	1434
1	0.99	0.79	0.88	238
accuracy			0.97	1672
macro avg	0.98	0.89	0.93	1672
weighted avg	0.97	0.97	0.97	1672

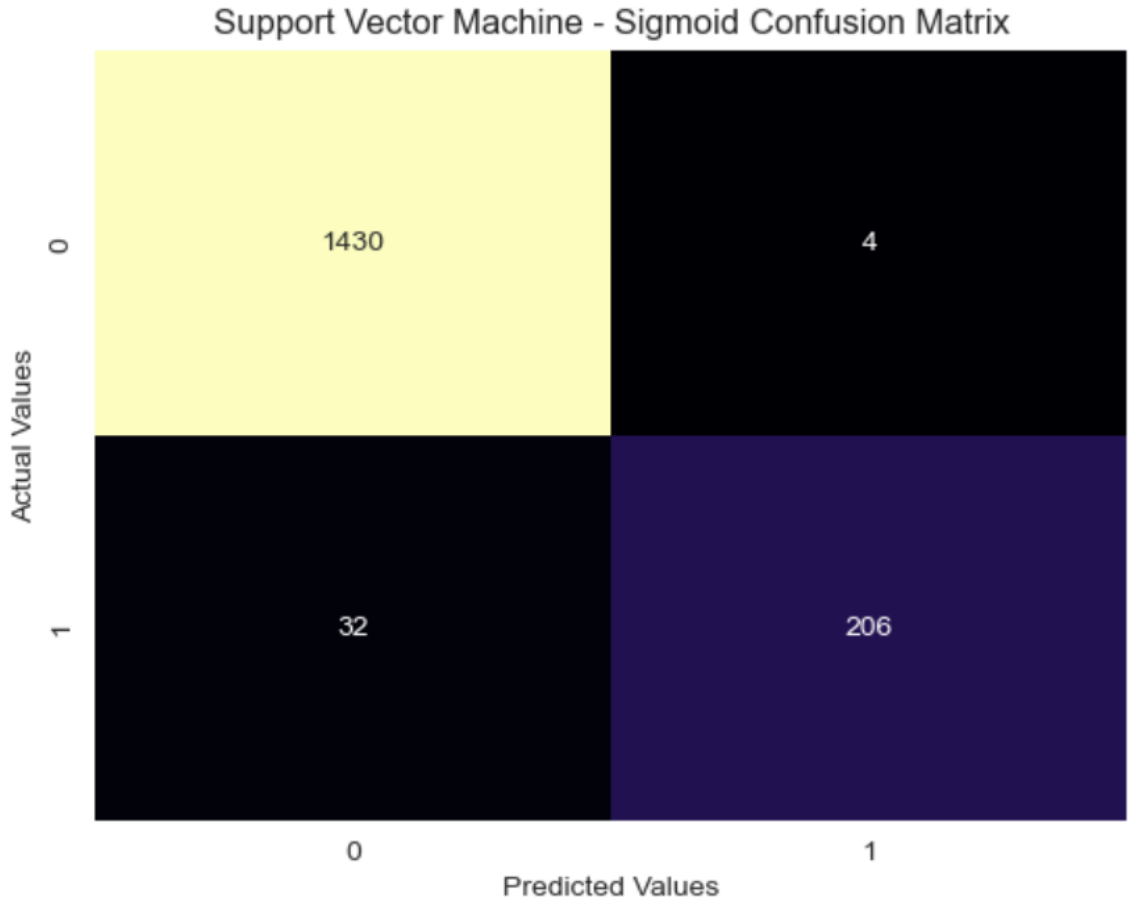


The Support Vector Machine - Radial Basis Function (RBF) model showcases strong performance in email classification. With 97% precision for "ham" (Class 0) and 99% for "spam" (Class 1), the model provides accurate predictions. Achieving perfect recall for "ham" and 79% for "spam," it effectively captures instances of both classes, though it may miss some spam instances. The weighted average F1-score of 97% signifies its outstanding overall performance in identifying "ham" and "spam." The macro average F1-score of 93% emphasizes the model's efficacy in capturing instances from both classes.

Support Vector Machine, Sigmoid Kernel:

Support Vector Machine - Sigmoid: Accuracy Score: 97.85%

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1434
1	0.98	0.87	0.92	238
accuracy			0.98	1672
macro avg	0.98	0.93	0.95	1672
weighted avg	0.98	0.98	0.98	1672

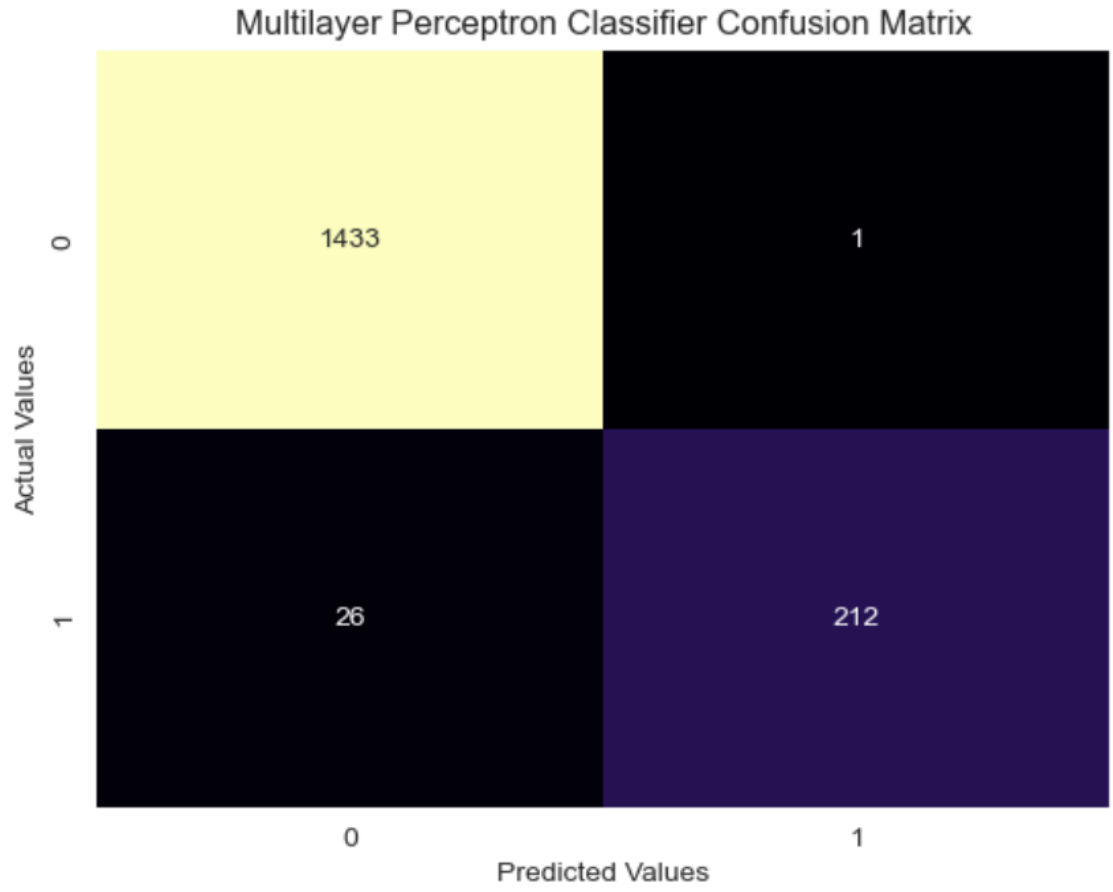


The Support Vector Machine - Sigmoid model achieves robust performance in email classification. With a precision of 98% for "ham" (Class 0) and 98% for "spam" (Class 1), the model demonstrates accurate predictions. The recall for "ham" is perfect at 100%, while for "spam" it stands at 87%, indicating a strong ability to capture spam instances with some potential misses. The weighted average F1-score of 98% highlights its overall excellence in classifying both "ham" and "spam." The macro average F1-score of 95% emphasizes the model's solid performance in capturing instances of both classes.

Multilayer Perceptron:

Multiplayer Perceptron Classifier: Accuracy Score: 98.39%

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1434
1	1.00	0.89	0.94	238
accuracy			0.98	1672
macro avg	0.99	0.95	0.97	1672
weighted avg	0.98	0.98	0.98	1672

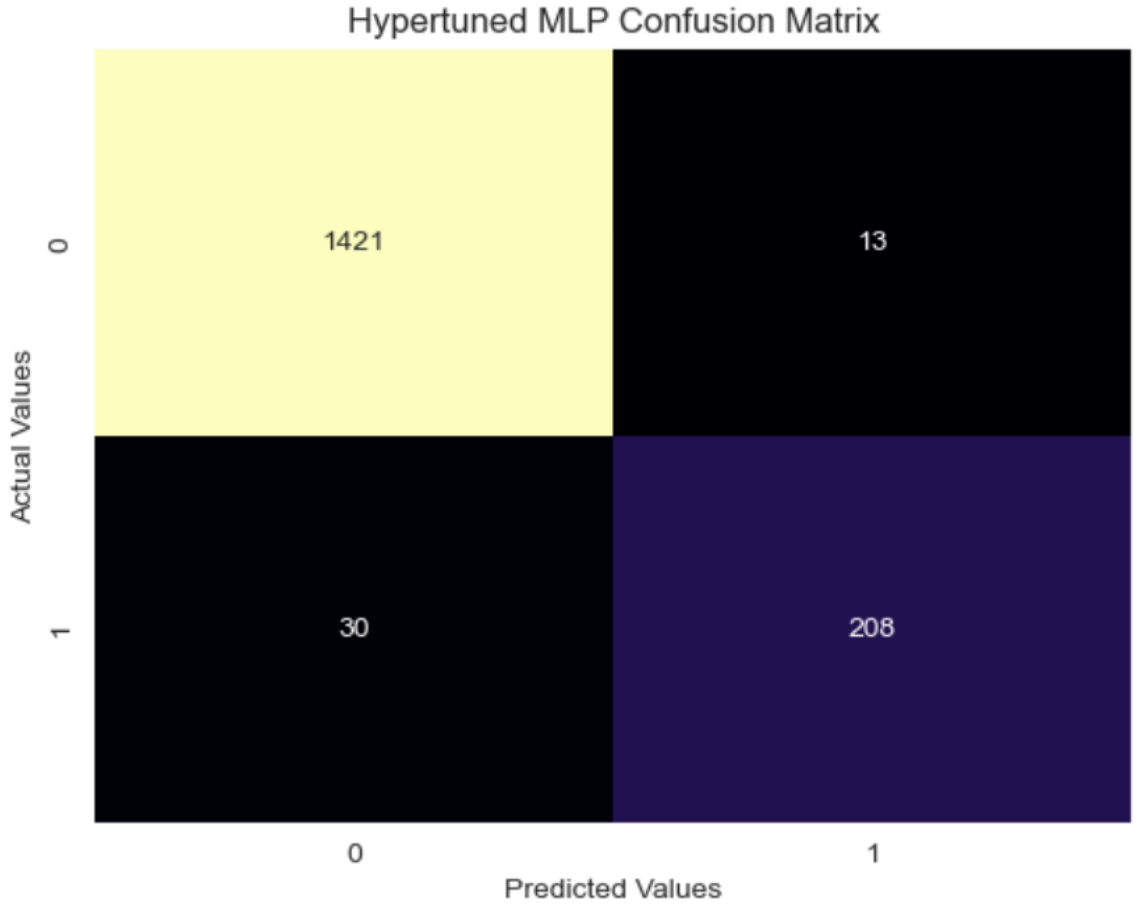


The Multiplayer Perceptron Classifier achieves exceptional performance in email classification. With a precision of 98% for "ham" (Class 0) and 100% for "spam" (Class 1), the model exhibits accurate predictions. The recall for "ham" is perfect at 100%, while for "spam" it stands at 89%, signifying a strong ability to capture spam instances but with potential for misses. The weighted average F1-score of 98% underscores its overall excellence in classifying both "ham" and "spam." The macro average F1-score of 97% emphasizes the model's outstanding performance in capturing instances of both classes

Multilayer Perceptron, Hypertuned:

Hypertuned Multiplayer Perceptron Classifier: Accuracy Score: 97.43%

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1434
1	0.94	0.87	0.91	238
accuracy			0.97	1672
macro avg	0.96	0.93	0.95	1672
weighted avg	0.97	0.97	0.97	1672



The Hypertuned Multiplayer Perceptron Classifier excels in email classification, achieving a precision of 98% for "ham" (Class 0) and 94% for "spam" (Class 1). With a recall of 99% for "ham" and 87% for "spam," the model effectively captures instances, particularly for "ham." The weighted average F1-score of 97% signifies excellent overall performance in identifying both "ham" and "spam." The macro average F1-score of 95% demonstrates consistent effectiveness across both classes. In summary, the model exhibits high accuracy, with a slightly lower recall for "spam," showcasing robust performance in email filtering applications.

Conclusion and Discussion:

In our evaluation of email spam detection models, the default Multiplayer Perceptron Classifier excelled with a 98.39% accuracy, surpassing the hypertuned version. Crucial for spam precision, this model showed exceptional performance, ideal for email filtering. Despite the hypertuned model's slightly lower accuracy, the default Multiplayer Perceptron Classifier maintained a favorable balance between accuracy, precision, and recall. If Occam's razor guides the choice, favoring simplicity, the Naïve Bayes model emerges as a favorable option, offering quicker execution than the Multi-layer Perceptron (MLP) model. In scenarios prioritizing computational efficiency and simplicity, Naïve Bayes becomes a pragmatic choice for email spam classification. Acknowledging limitations, including dataset representativeness and model generalization, suggests avenues for future research, possibly exploring ensemble methods and deep learning. Technology constraints led to 3-fold instead of 5-fold cross-validation, impacting the fit count for candidate models.

References:

Dhakad, S. (2021). *Email Spam Detection Dataset (classification)* [Dataset]. Kaggle. https://www.kaggle.com/datasets/shantanudhakadd/email-spam-detection-dataset-classification

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning.* Springer.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273-297.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep Learning* (Vol. 1). MIT press Cambridge.

Abad, V., Jiménez, I., García-Vázquez, R., Aldrey, J., Rivero, D., Cacabelos, P., ... Yánez, S. (2018). Using Artificial Neural Networks for Identifying Patients with Mild Cognitive Impairment Associated with Depression Using Neuropsychological Test Features [Figure 1]. *Applied Sciences, 8*(9), 1629. https://doi.org/10.3390/app8091629