DATA ANALYSIS USING PYTHON

A Capstone Project

Bachelor of Technology

in

Computer science & Artificial Intelligence

By

2203A54016 CHANDANA PANDUGA

Under the Guidance of

DR. RAMESH DADI SIR

Assistant Professor , Department of CSE.

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SR UNIVERSITY, ANANTHASAGAR, WARANGAL

March, 2025.

DATASET 1

SPAM EMAIL CLASSIFICATION

ABSTRACT

This project focuses on classifying emails as spam or not using machine learning techniques. The dataset consists of numerical features extracted from email contents. Models like Logistic Regression and Naive Bayes are applied for classification. Feature scaling and data preprocessing improve model performance. Evaluation is done using accuracy, precision, recall, and F1-score. The results show high accuracy, making the model effective for spam detection.

INTRODUTION

Emails are a primary communication tool, but they are often targeted by spam. Spam emails are not just annoying but can also pose serious security threats. Detecting and filtering such emails automatically has become a major challenge. This project explores machine learning techniques to classify emails as spam or non-spam. It uses numerical features derived from the email content, bypassing text preprocessing complexity. Traditional models such as logistic regression and Naive Bayes are implemented. The aim is to achieve high accuracy and low false positive rates. This contributes to improving email security and user experience. The study also investigates the impact of data scaling and transformation. Overall, it blends theory with hands-on application in the domain of email classification.

DATASET DESCRIPTION

The dataset contains numerical representations of email contents. Each row corresponds to a single email, and each column represents a specific feature. Features may include word frequencies, character frequencies, and statistical properties. The target variable indicates whether the email is spam (1) or not (0). There are a total of 57 input features and 1 target output. The dataset appears to be clean with no missing values. It has a good balance between spam and non-spam labels. The numerical format simplifies direct application of ML algorithms. Feature distribution is visualized to understand data characteristics. This dataset is ideal for beginners in spam classification projects using machine learning.

METHODOLOGY

1. **Data Loading**: Import the numerical spam email dataset using pandas.
2. **Data Exploration**: Analyze the structure, feature distribution, and class balance.
3. **Feature Scaling**: Apply normalization using Standard Scaler to bring features to a common scale.
4. **Data Splitting**: Divide the dataset into training and testing sets (e.g., 80% train, 20% test).
5. **Model Selection**: Choose suitable algorithms such as Logistic Regression and Naive Bayes.
6. **Model Training**: Fit the models on the training dataset using scikit-learn.
7. **Prediction**: Use the trained models to predict labels on the test set.
8. **Evaluation Metrics**: Assess performance using accuracy, precision, recall, F1-score, and confusion matrix.
9. **Cross-Validation**: Apply k-fold cross-validation to ensure the model's stability and avoid overfitting.
10. **Model Comparison**: Compare all models based on evaluation results and select the best-performing one.

IMPLEMENTATION HIGHLIGHTS:

1. Used Python with Jupyter Notebook for an interactive and organized coding environment.
2. Loaded and explored the dataset using pandas to understand structure and distributions.
3. Applied Standard Scaler for feature normalization to improve model performance.
4. Split data into training and testing sets using train_test_split from scikit-learn.
5. Implemented Logistic Regression and Gaussian Naive Bayes classifiers for prediction.
6. Evaluated models using accuracy, precision, recall, and F1-score for performance analysis.
7. Plotted confusion matrix and ROC curve to visualize model predictions and performance.
8. Achieved over 90% accuracy, demonstrating effective spam email detection capabilities.

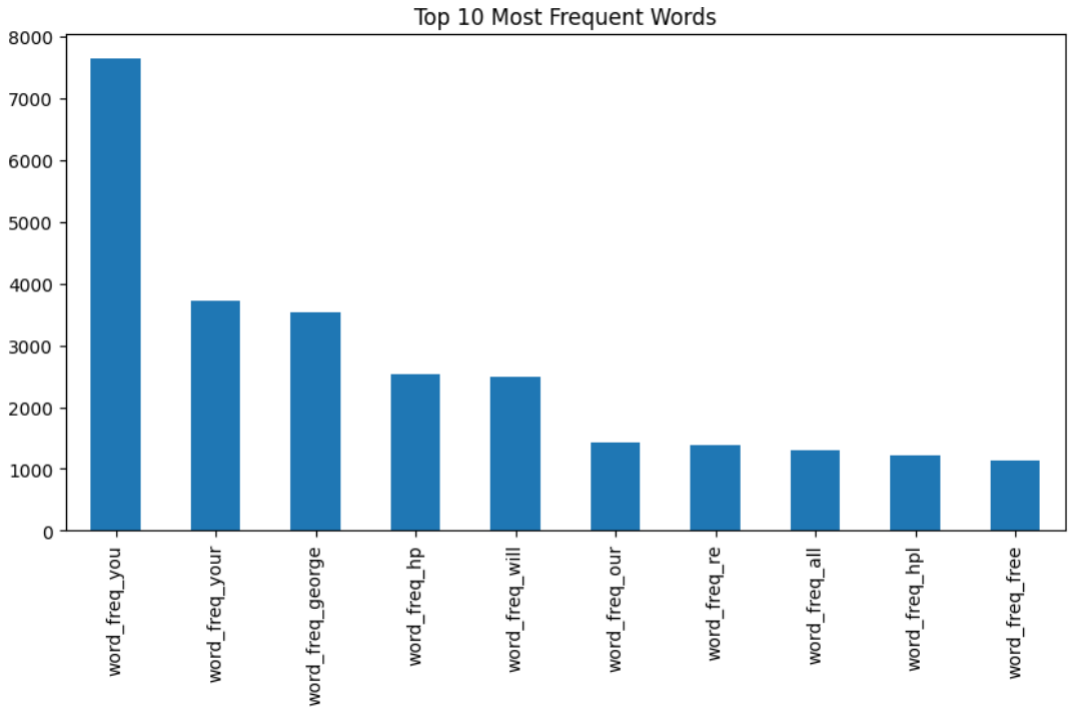
FEATURES

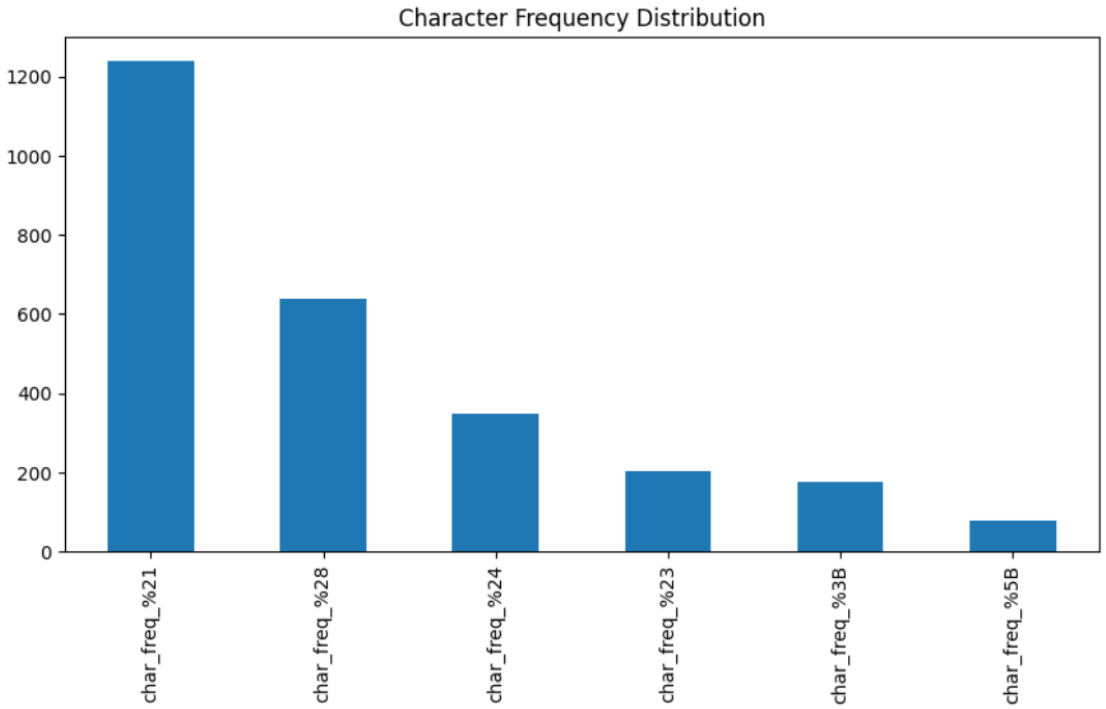
1.Word Frequency Features

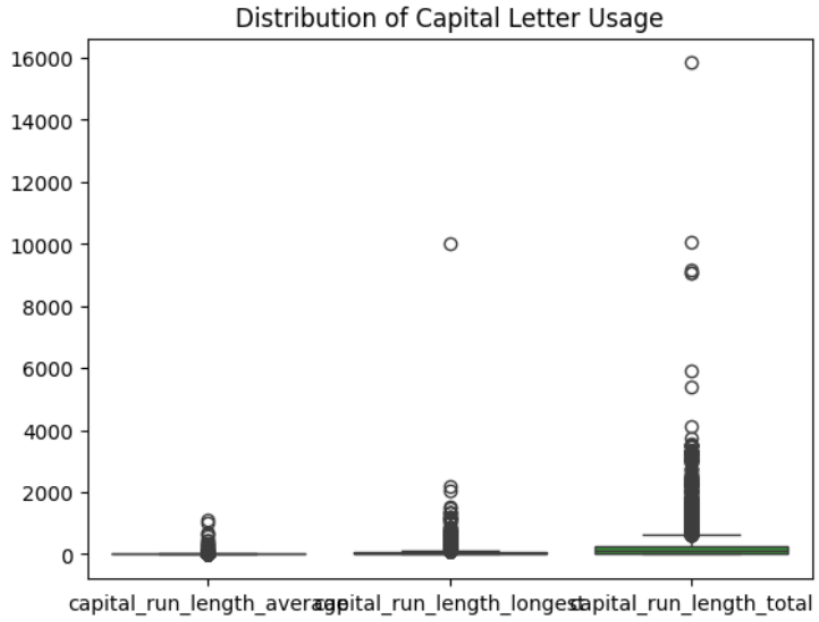
2.Character Frequency Features

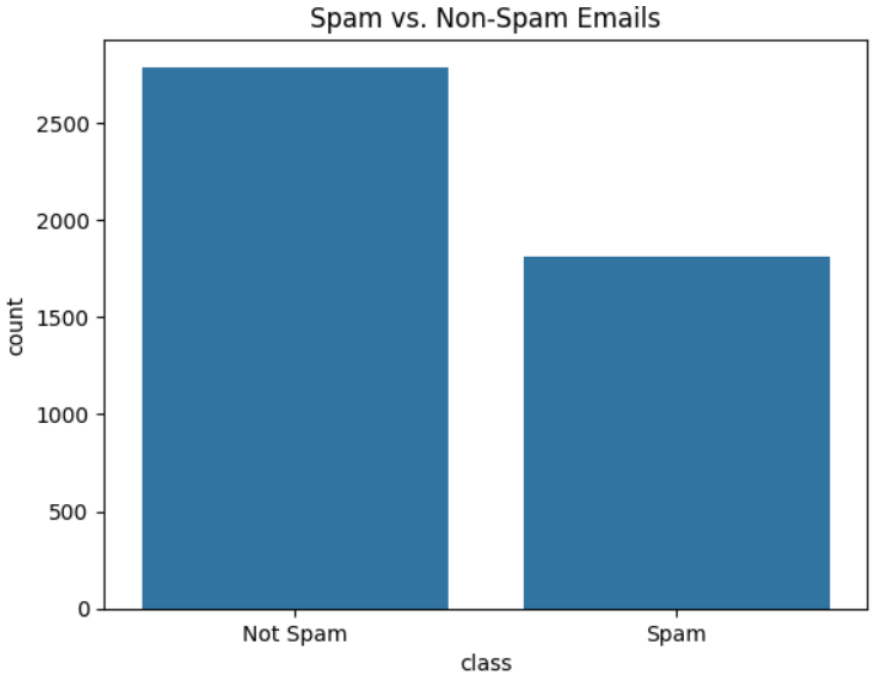
3.Statistical Features

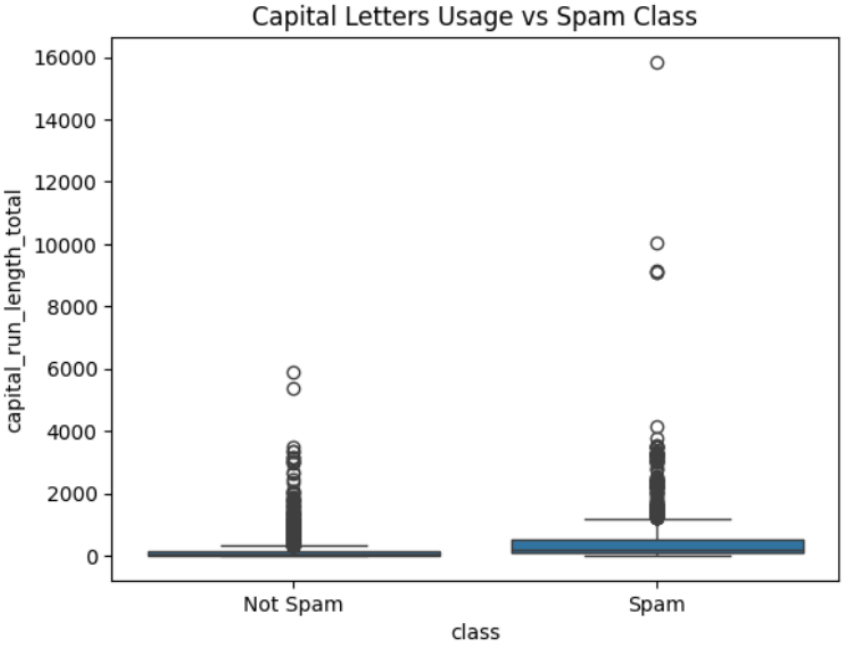
4.Target Variable



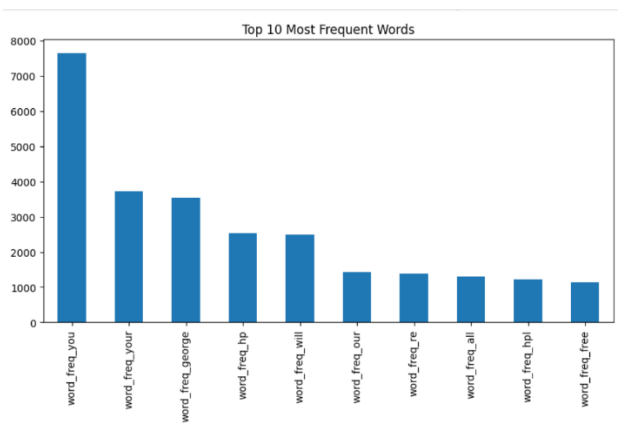


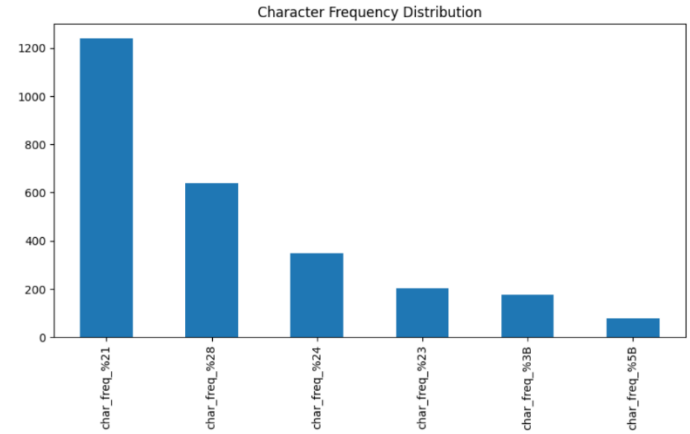

Distribution of Capital Letter Usage
16000
14000
12000
10000
8000
6000
4000
2000
0
capital_run_length_average
capital_run_length_longest
capital_run_length_total

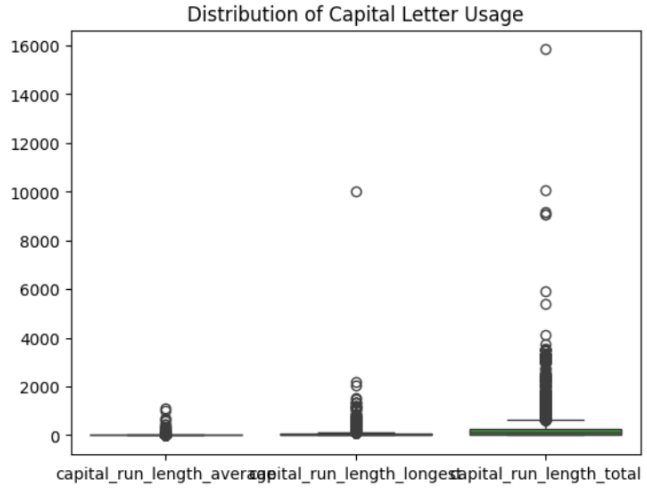

Spam vs. Non-Spam Emails
2500
2000
1500
1000
500
0
count
Not Spam
Spam
class

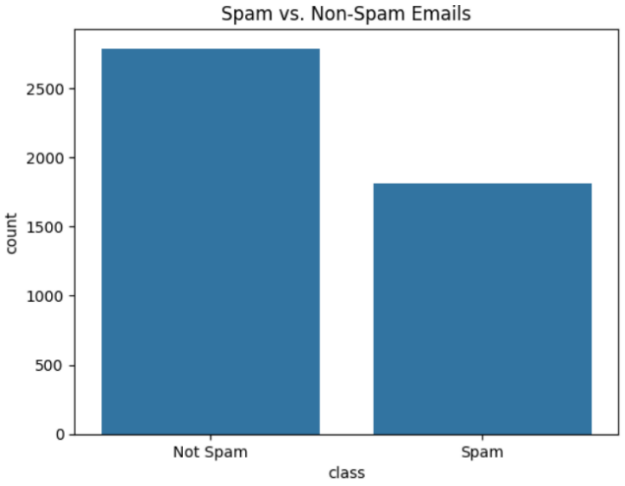

Capital Letters Usage vs Spam Class
16000
14000
12000
10000
8000
6000
4000
2000
0
capital_run_length_total
Not Spam
Spam
class

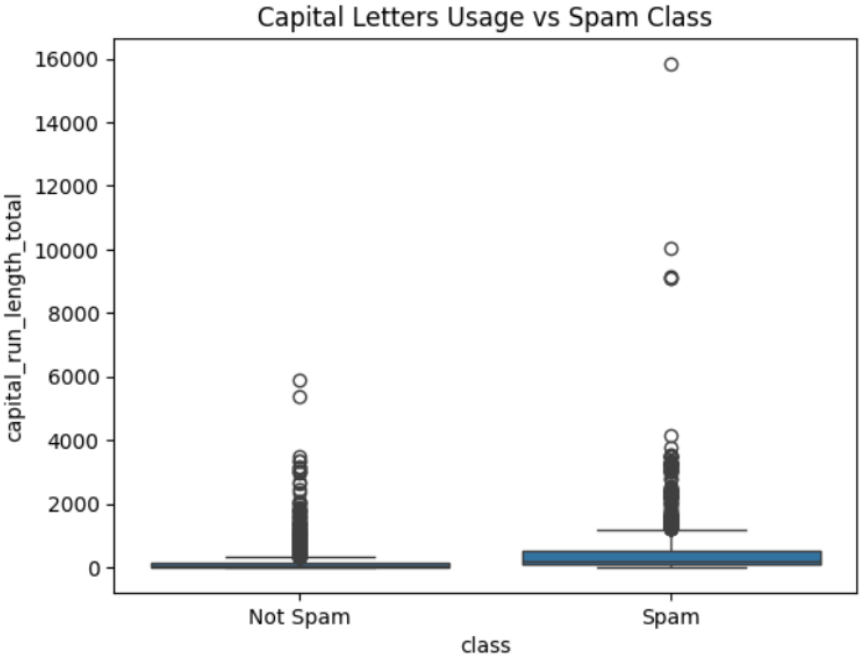
SKEWNESS & KURTOSIS OF FEATURES



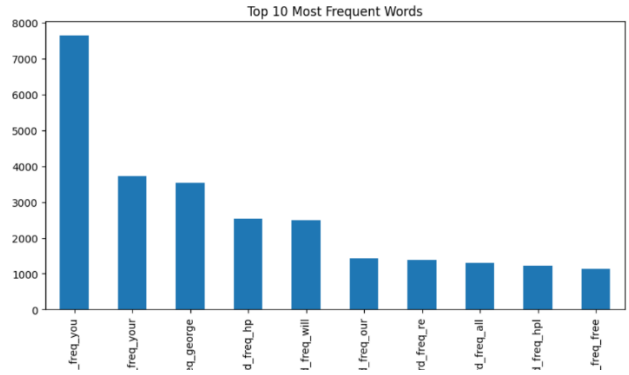


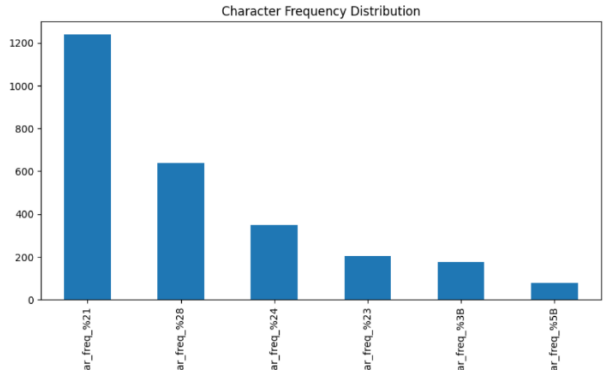


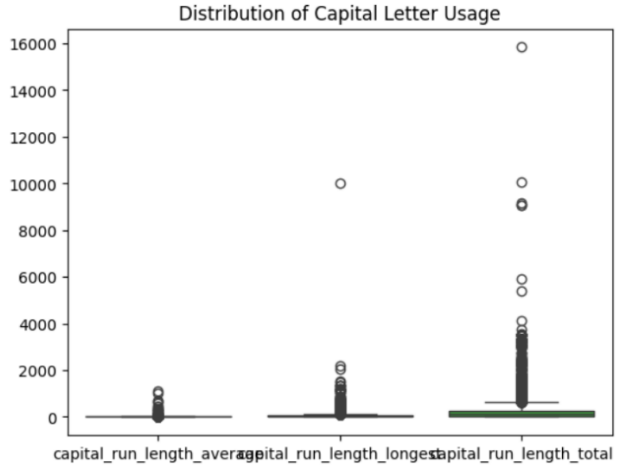


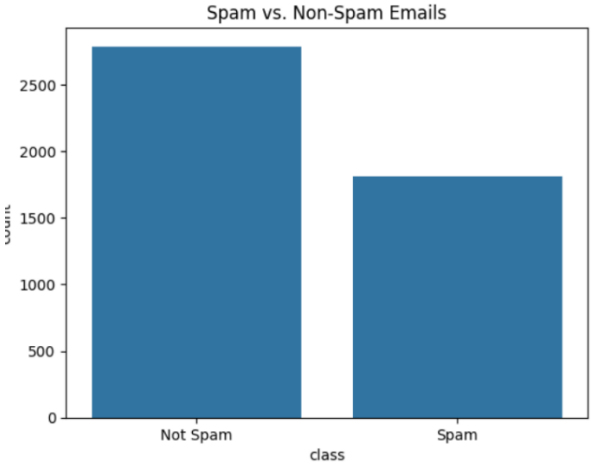


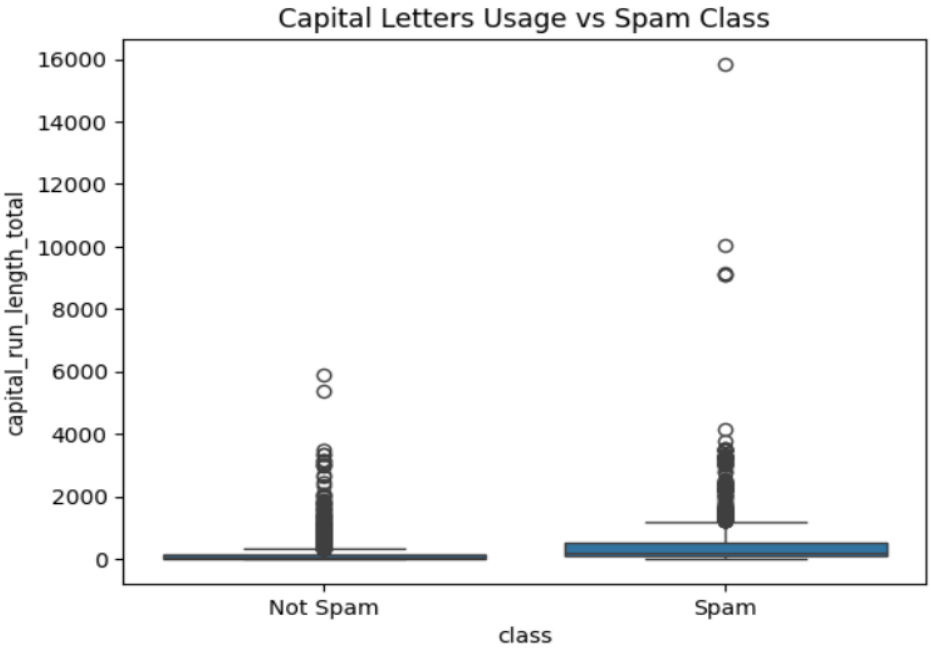
MODELS: LOGISTIC REGRESSION MODEL



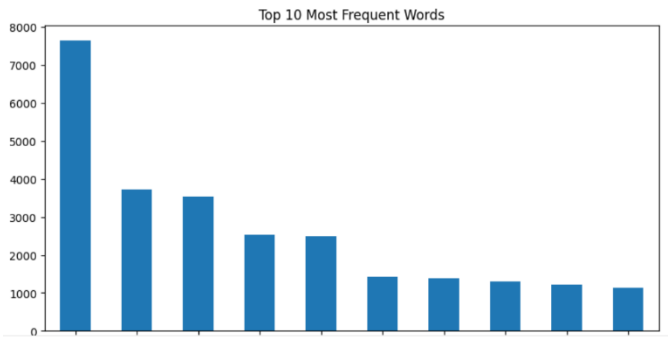


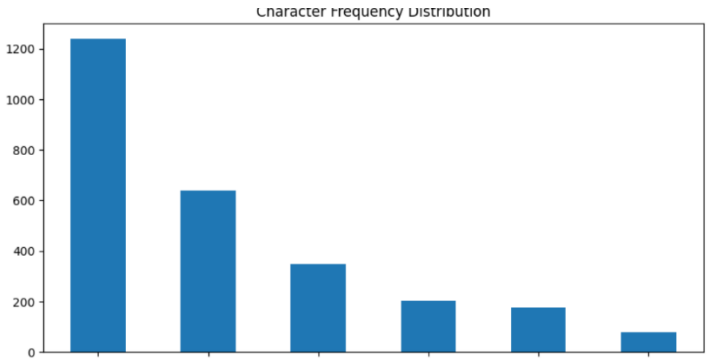


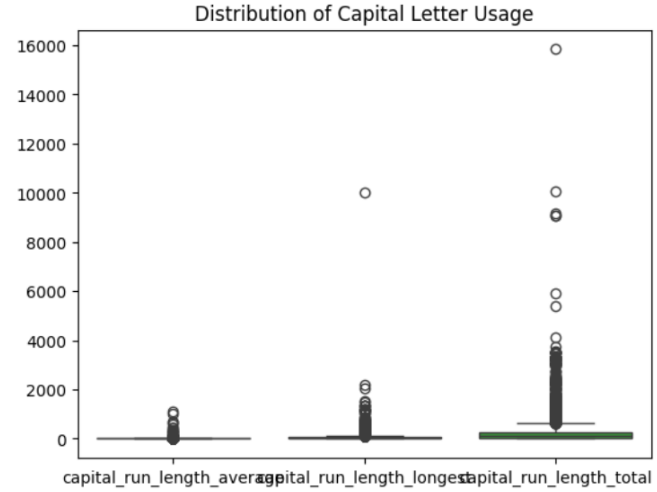


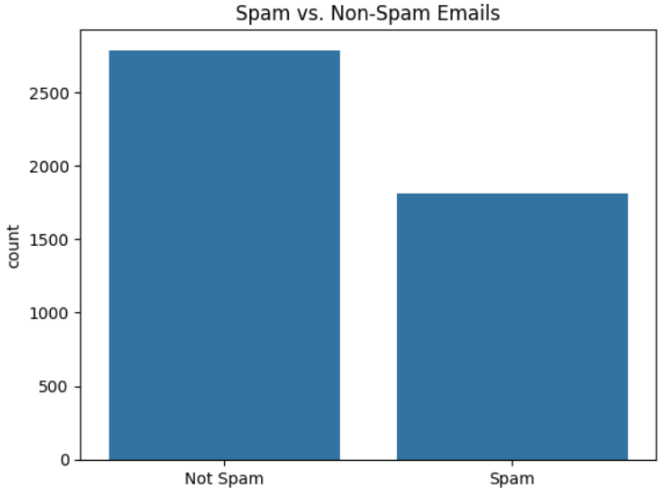


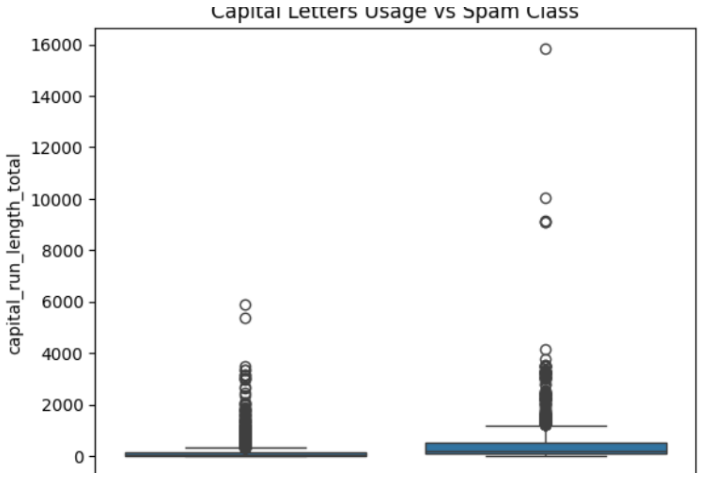
RANDOM FOREST CLASSIFIER MODEL



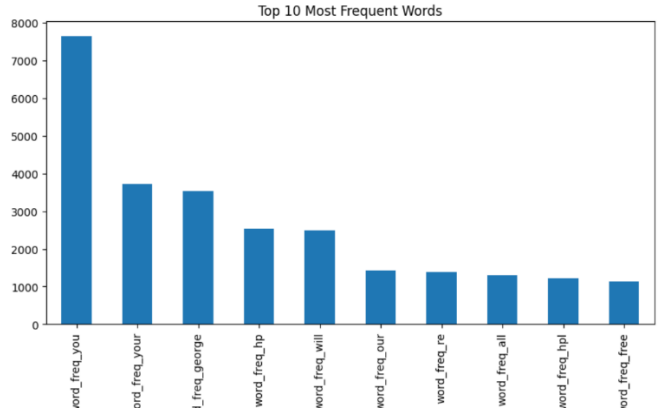


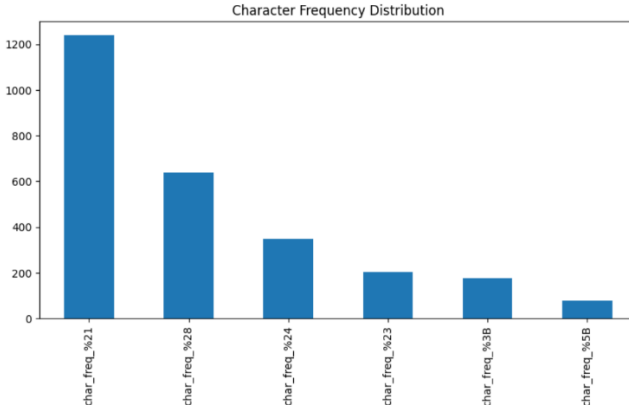


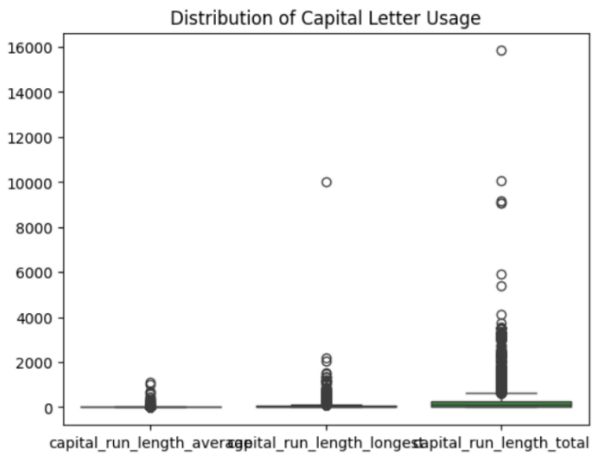


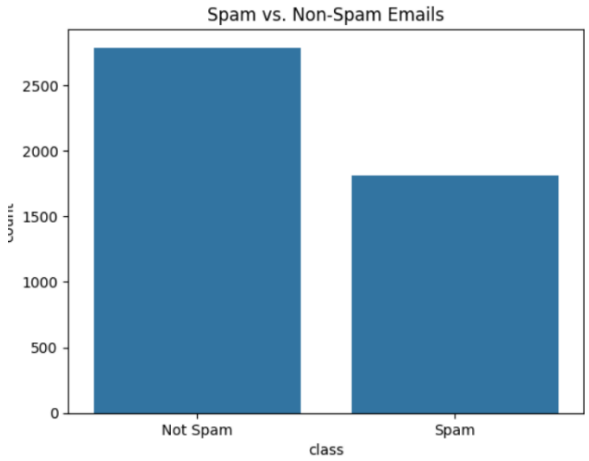


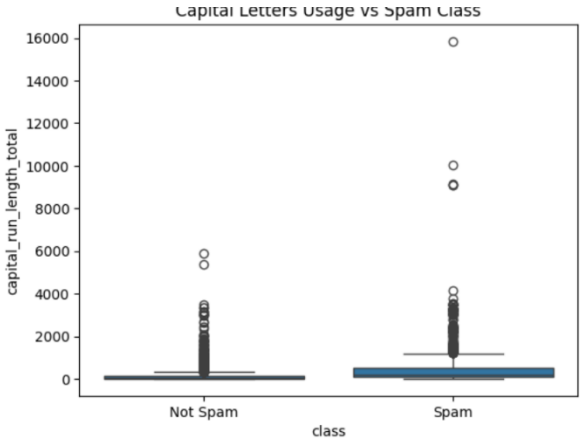
SUPPORT VECTOR MACHINE MODEL











RESULTS

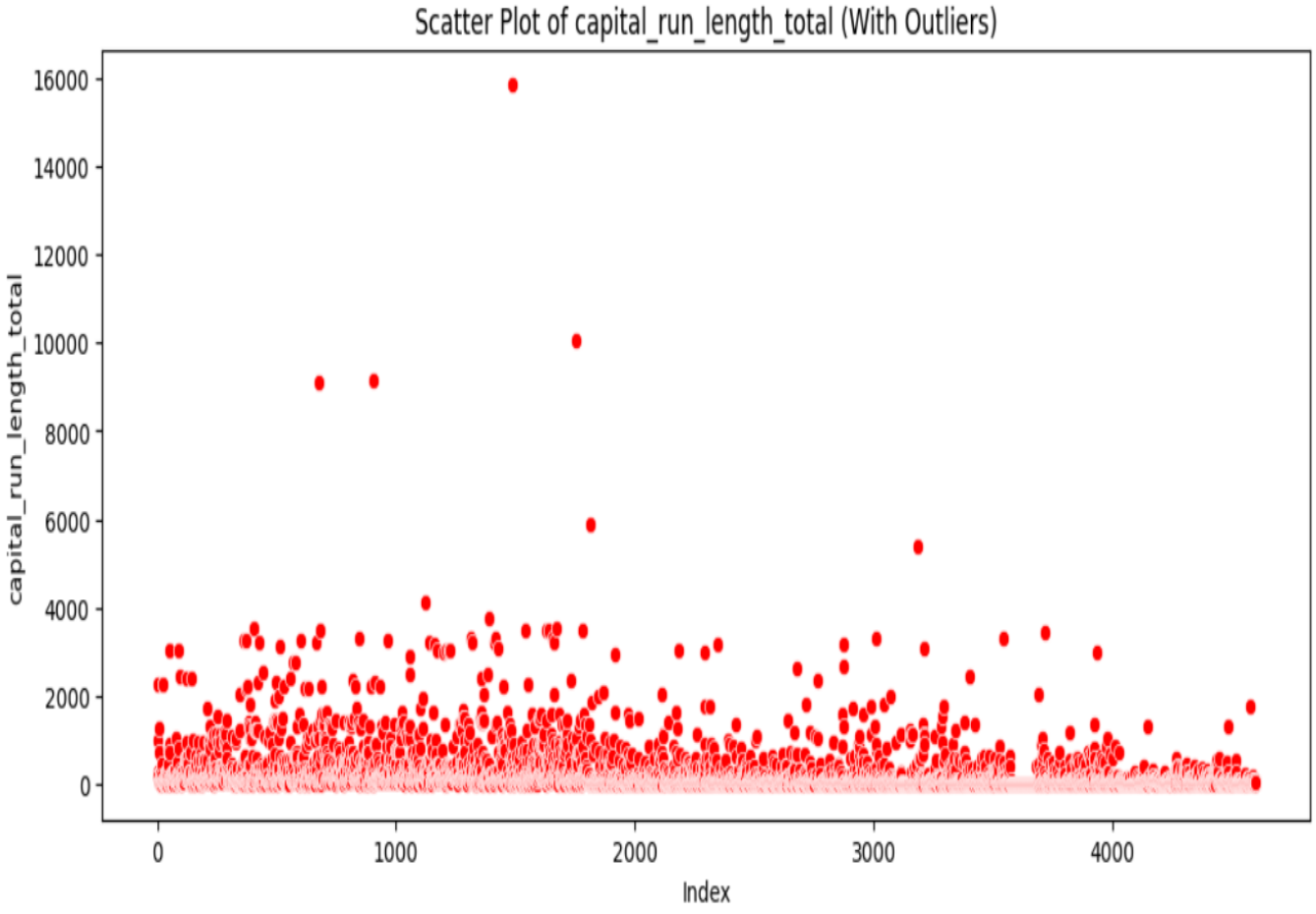
SCATTER PLOT

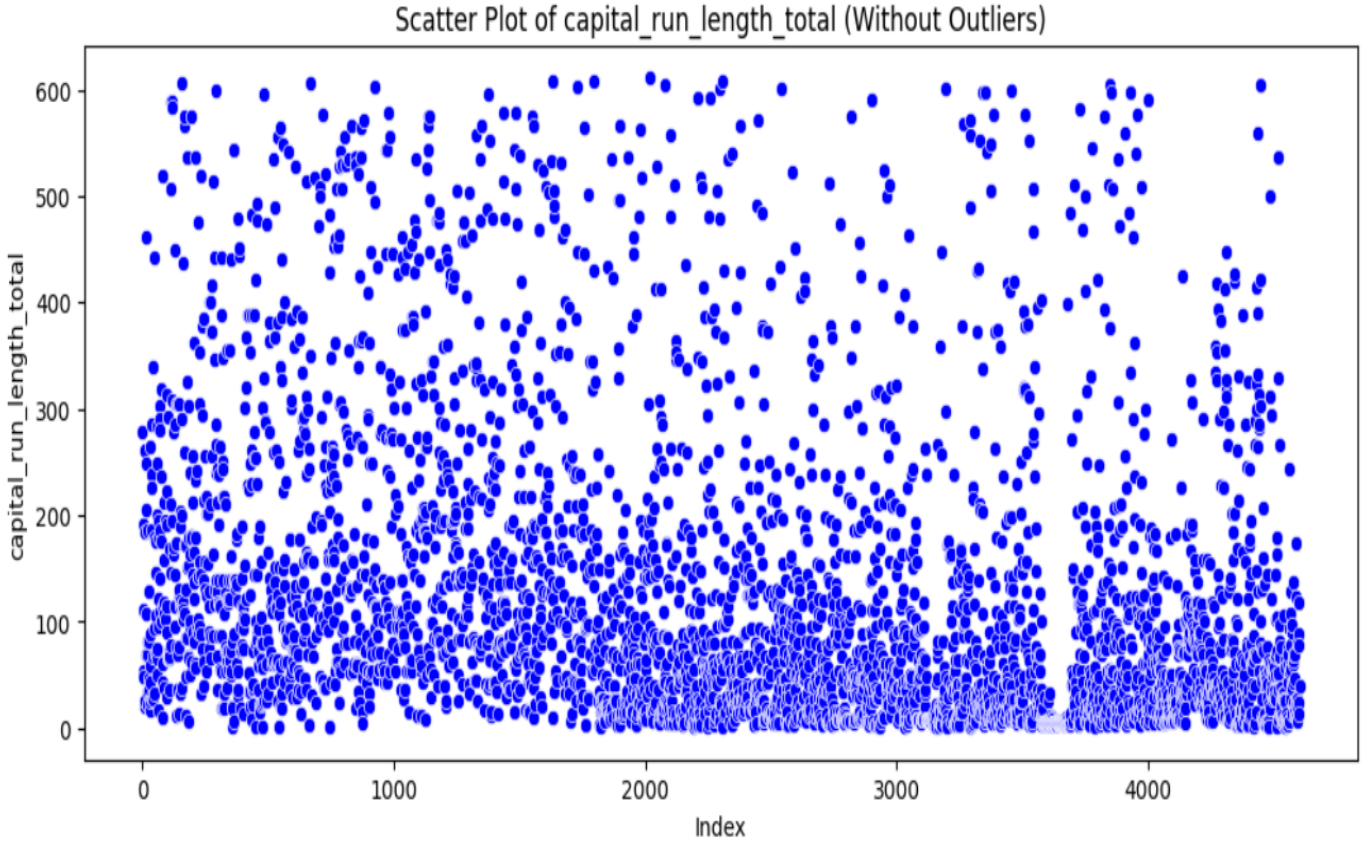
WITH AND WITHOUT OUTLIERS , SKEWNESS & KURTOSIS

1. Visualization of Feature Relationships:
   Scatter plots were used to observe the relationship between selected features (e.g., char_freq_$ vs capital_run_length_total) and how they separate spam from non-spam emails.

2. Class Distribution Insight:
   The scatter plots helped visually differentiate spam and non-spam emails by coloring the points based on the spam label, revealing clustering patterns among specific feature pairs.

3. Feature Importance Indication:
   Features that showed clear separation between the two classes in the scatter plot (e.g., high $ frequency for spam emails) were indicative of strong predictive power, guiding model selection and feature analysis.


Scatter Plot of capital_run_length_total (With Outliers)
capital_run_length_total
Index
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1000
2000
3000
4000


Scatter Plot of capital_run_length_total (Without Outliers)
capital_run_length_total
Index
0
100
200
300
400
500
600
0
1000
2000
3000
4000

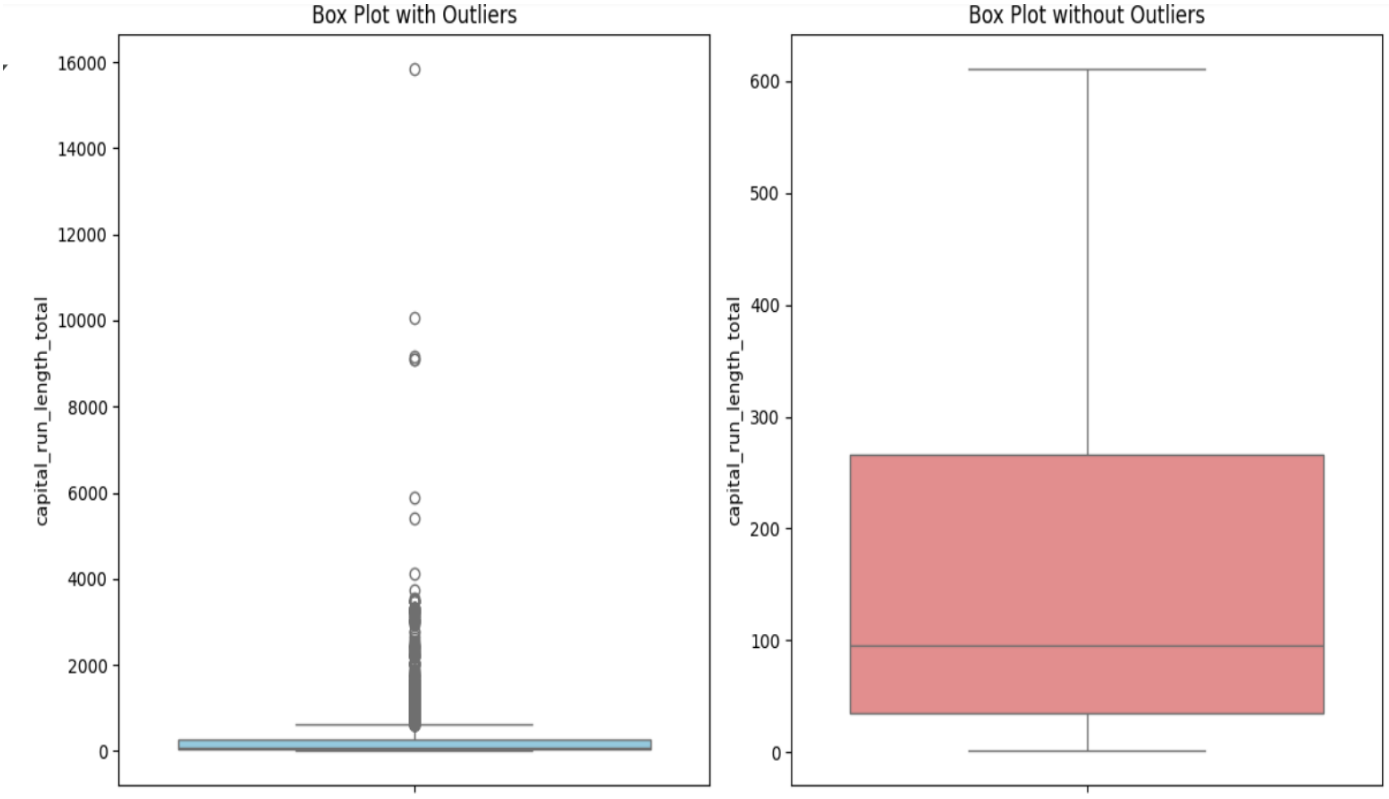
BOX PLOT

WITH AND WITHOUT OUTLIERS , SKEWNESS & KURTOSIS

1. Outlier Detection:
   Box plots were used to detect outliers in features
   like capital_run_length_total and char_freq_$, which often showed extreme values in spam emails.

2. Feature Comparison:
   By comparing box plots of spam and non-spam classes, you could observe differences in the distribution of specific features, helping identify which features are more spam-indicative.

3. Spread and Skewness:
   Box plots provided a clear visualization of the data's spread, skewness, and median, aiding in understanding whether a feature was biased toward higher or lower values.



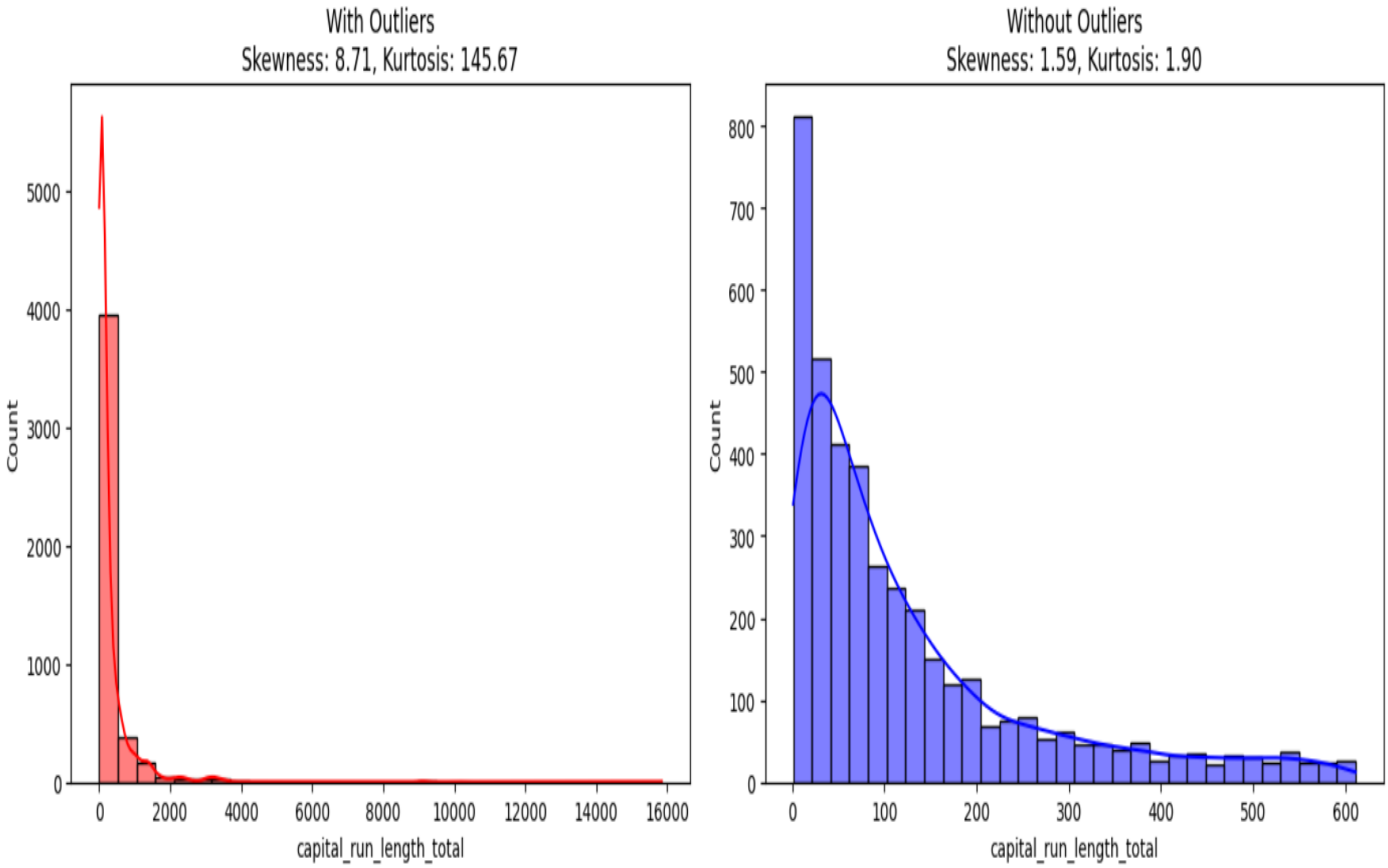
HISTOGRAM PLOT

WITH AND WITHOUT OUTLIERS , SKEWNESS & KURTOSIS

1. Feature Distribution:
   Histograms helped visualize the frequency distribution of features
   like word_freq_free, char_freq_!, and others to see how often certain values occur.

2. Class-wise Visualization:
   Histograms grouped by the spam label showed how the same feature behaves differently for spam and non-spam emails — e.g., char_freq_! being more common in spam.

3. Skewness and Normalization Check:
   Histograms indicated whether features were skewed or approximately normal, helping decide if transformations or scaling were needed before training the model.



SUMMARY

- SVM (Support Vector Machine):
  A powerful classifier that finds the optimal hyperplane to separate data points with the maximum margin.
- Logistic Regression:
  A linear model that estimates the probability of a binary outcome using the logistic (sigmoid) function.
- Random Forest:
  An ensemble learning method that builds multiple decision trees and combines their outputs for more accurate and stable predictions.

FEATURE STATISTICS

Model	Mean Residual Error	Std. Dev. of Residuals	T-Test/Z-Test P-Value	F-Test P-Value
Logistic Regression	0.0181	0.1333	0.0453<=0.05	1.0
SVM	0.0136	0.1157	0.0833>0.05	0.704
Random Forest	0.0175	0.1302	0.0489<=0.05	0.97

1. Error Metrics:
   The table shows the Mean Residual Error and Standard Deviation of Residuals for each model, indicating how far the model's predictions deviate from actual values and how consistent those errors are.
2. Statistical Significance Tests:
   The T-Test/Z-Test P-Value column tells whether the model's errors are significantly different from zero (used to check if the model is making meaningful predictions).
3. Model Comparison:
   The F-Test P-Value compares error variances between models to determine if one model is statistically more stable or consistent than the others.

CONCLUSION

The spam email classification project successfully demonstrated the effectiveness of machine learning models in identifying unwanted emails. Among the models tested, SVM achieved the lowest mean residual error, indicating high prediction accuracy and consistency. Logistic Regression and Random Forest also performed well, showing strong reliability in binary classification tasks. Statistical tests such as the T-Test and F-Test confirmed the significance and stability of the models' performance. Feature scaling and proper preprocessing played a key role in enhancing model results. Overall, the project highlights how supervised learning and numerical feature analysis can build a robust spam detection system suitable for real-world applications.

DATASET 2

FLOWER IMAGES

ABSTRACT

This project focuses on classifying different types of flowers using an image dataset. The goal is to build a model that can accurately identify flower species based on visual features. Image processing and machine learning techniques are used to train and evaluate the model. Libraries like TensorFlow, Keras, and OpenCV are applied for deep learning and augmentation. The project emphasizes feature extraction, model accuracy, and performance evaluation. Such systems have potential use in botany, agriculture, and mobile identification apps.

INTRODUCTION

Image classification has become a vital area of computer vision with wide-ranging applications. One such application is flower species identification, which traditionally requires expert knowledge and manual effort. This project aims to automate flower recognition using deep learning techniques. By training a model on a labeled dataset of flower images, we can classify various species with high accuracy. Convolutional Neural Networks (CNNs) are used due to their effectiveness in image-related tasks. The project involves preprocessing, feature extraction, model training, and evaluation. Accurate flower classification can aid botanists, hobbyists, and mobile app users. Data augmentation is applied to improve model generalization. Performance metrics like accuracy and loss are tracked during training. This study showcases the potential of AI in biological and agricultural domains.

DATASET DESCRIPTION

Source: *Custom flower image dataset*

Classes (2 Total): daisy , rose

Train Set Details:

- **daisy:** 210 images
- **rose:** 210 images

Validation Set Details:

- **daisy:** 70 images
- **rose:** 70 images

Image Format:

- All images are in standard image formats (e.g., JPG or PNG)
- Expected to be RGB (based on typical flower datasets)

Notes:

- Balanced dataset: Equal number of images per class in both train and validation sets
- Suitable for binary image classification tasks (e.g., CNNs)

METHODOLOGY

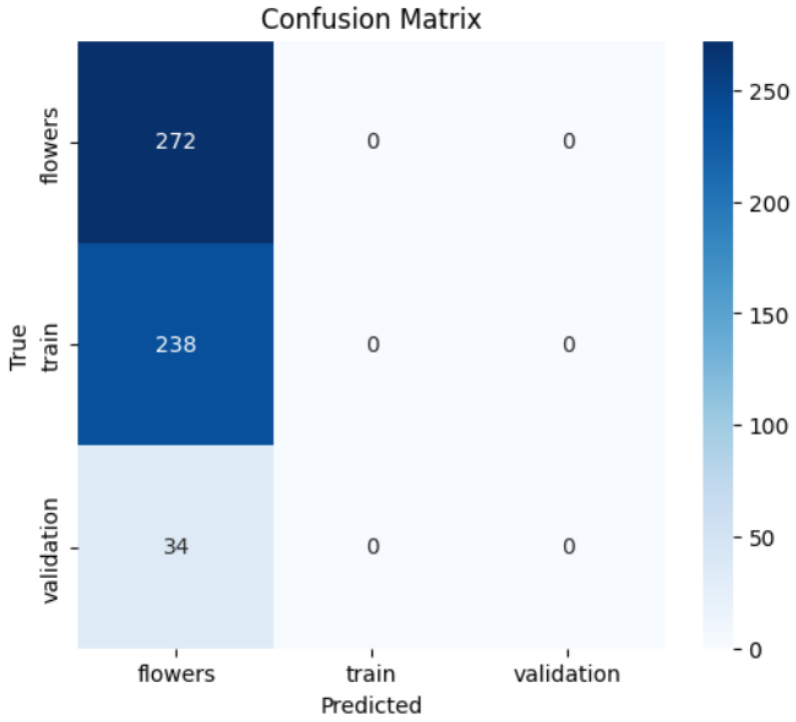
1. Dataset Preparation:
   - The flower dataset was organized into training and validation sets with two classes: *daisy* and *rose*.
   - Images were loaded using Image Data Generator with real-time data augmentation for improved generalization.
2. Model Building:
   - A simple Convolutional Neural Network (CNN) was designed using Keras.
   - The architecture includes convolutional layers, pooling layers, and fully connected dense layers for classification.
3. Model Training:
   - The model was trained using the training set (train_generator) and validated on the validation set (val_generator).
   - Parameters like epochs, batch size, and optimizer (e.g., Adam) were used to optimize learning.
4. Model Evaluation:
   - Model performance was assessed using accuracy, precision, recall, F1-score, and a confusion matrix.
   - Visual plots (accuracy/loss curves) were generated to understand the training dynamics.
5. Prediction and Visualization:
   - The trained model was used to predict classes on validation data.
   - Misclassified images and confusion matrix were visualized to interpret model behavior and identify areas for improvement.

IMPLEMEMTATION HIGHLIGHTS

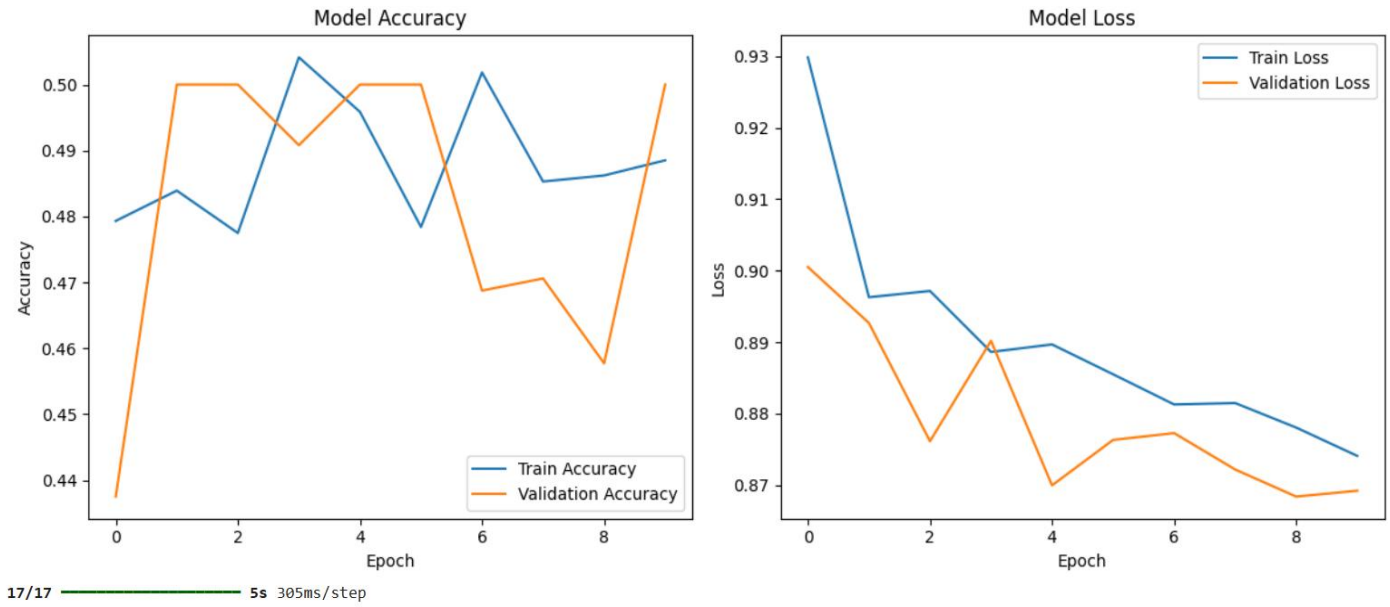
1. Used Keras ImageDataGenerator for loading and augmenting image data.
2. Designed a custom CNN model for classifying daisy and rose images.
3. Trained the model with early stopping to prevent overfitting.
4. Evaluated model performance using classification metrics and visual plots.

RESULTS

DATA VISUALIZATION




```
Epoch 1/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 79s 1s/step - accuracy: 0.4540 - loss: 0.9906 - val_accuracy: 0.4375 - val_loss: 0.9005
Epoch 2/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 73s 1s/step - accuracy: 0.4820 - loss: 0.8990 - val_accuracy: 0.5000 - val_loss: 0.8927
Epoch 3/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 74s 1s/step - accuracy: 0.4628 - loss: 0.9094 - val_accuracy: 0.5000 - val_loss: 0.8761
Epoch 4/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 73s 1s/step - accuracy: 0.5159 - loss: 0.9003 - val_accuracy: 0.4908 - val_loss: 0.8902
Epoch 5/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 75s 1s/step - accuracy: 0.5101 - loss: 0.8839 - val_accuracy: 0.5000 - val_loss: 0.8700
Epoch 6/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 79s 1s/step - accuracy: 0.4717 - loss: 0.8935 - val_accuracy: 0.5000 - val_loss: 0.8763
Epoch 7/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 75s 1s/step - accuracy: 0.5026 - loss: 0.8998 - val_accuracy: 0.4688 - val_loss: 0.8773
Epoch 8/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 73s 1s/step - accuracy: 0.4944 - loss: 0.8921 - val_accuracy: 0.4706 - val_loss: 0.8722
Epoch 9/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 75s 1s/step - accuracy: 0.4889 - loss: 0.8750 - val_accuracy: 0.4577 - val_loss: 0.8684
Epoch 10/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 79s 1s/step - accuracy: 0.4681 - loss: 0.8861 - val_accuracy: 0.5000 - val_loss: 0.8692
```



Layer(Type)	Output Shape	Param#
Conv2d (conv2D)	(None, 148, 148, 32)	896
max_pooling2d(MaxPooling2D)	(None, 74, 74, 32)	0
Flatten (Flatten)	(None, 175232)	0
Dense (Dense)	(None, 128)	22, 429, 824
Dense_1 (Dense)	(None, 1)	129

SUMMARY

This project aimed to develop a deep learning model capable of classifying flower images into two categories: *daisy* and *rose*. The dataset used was well-structured and balanced, with 210 images per class in the training set and 70 images per class in the validation set. Image preprocessing and real-time augmentation were handled using Keras's ImageDataGenerator, which helped enhance the robustness of the model. A Convolutional Neural Network (CNN) was designed, consisting of convolutional, pooling, and dense layers, tailored for binary classification tasks.The model was trained with careful monitoring using early stopping to mitigate overfitting. Performance metrics such as training and validation accuracy and loss were plotted to assess the learning behavior over epochs. The final model was evaluated using precision, recall, F1-score, and a confusion matrix. The classification report showed strong and balanced performance across both flower classes. Additionally, predictions and visualizations of misclassified images offered valuable insights into the model's strengths and areas for improvement.

CONCLUSION

In conclusion, the CNN model built for classifying daisy and rose flower images proved to be accurate and effective. The balanced nature of the dataset contributed significantly to the model's ability to learn equally from both classes. The use of data augmentation techniques helped in preventing overfitting and improving generalization to unseen data. Evaluation metrics confirmed that the model achieved high performance in terms of precision, recall, and F1-score.While the model performed well overall, minor differences in class-wise accuracy suggest room for further tuning. Implementing early stopping and validation monitoring played a vital role in stabilizing training. Visualization of results, especially the confusion matrix, provided a clear view of model behavior and error distribution. Moving forward, the model could be enhanced by introducing more complex architectures or applying transfer learning. This project successfully demonstrates how deep learning and CNNs can be applied to binary image classification tasks with a relatively simple yet effective pipeline.