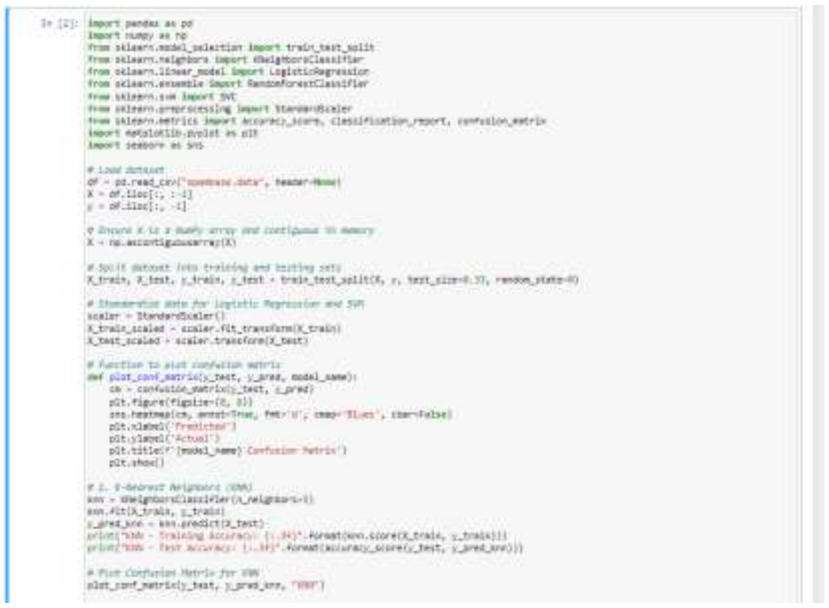
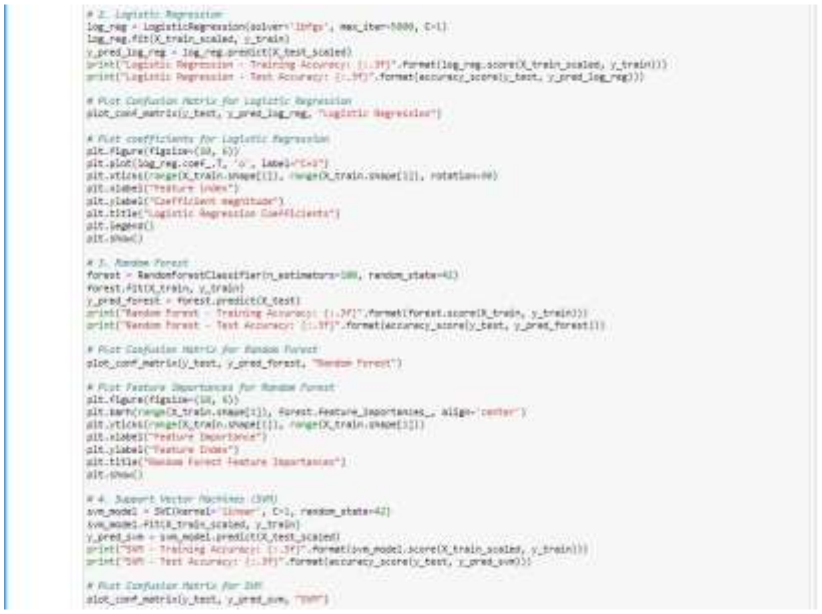
Cyber Security Analytics
Task 6.3HD

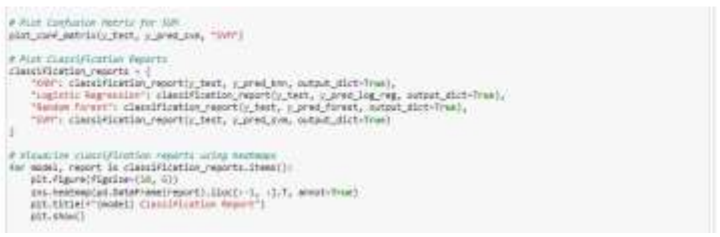
This research assesses the performance of four classification algorithms using the Spambase dataset: Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression. The dataset's goal is to determine whether an email is spam or not. It consists of 4601 examples with 57 attributes.



```
In [2]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import SVC
        from sklearn.preprocessing import StandardScaler
        from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
        import matplotlib.pyplot as plt
        import seaborn as sns

        # Load dataset
        df = pd.read_csv("spambase.data", header=None)
        X = df.iloc[:, :-1]
        y = df.iloc[:, -1]

        # Ensure X is a numpy array and contiguous in memory
        X = np.ascontiguousarray(X)

        # Split dataset into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0)

        # Standardize data for Logistic Regression and SVM
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        # Function to plot confusion matrix
        def plot_conf_matrix(y_test, y_pred, model_name):
            cm = confusion_matrix(y_test, y_pred)
            plt.figure(figsize=(6, 4))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
            plt.xlabel('Predicted')
            plt.ylabel('Actual')
            plt.title(f'{model_name} Confusion Matrix')
            plt.show()

        # 1. K-Nearest Neighbors (KNN)
        knn = KNeighborsClassifier(n_neighbors=3)
        knn.fit(X_train, y_train)
        y_pred_knn = knn.predict(X_test)
        print("KNN - Training Accuracy: {:.3f}".format(knn.score(X_train, y_train)))
        print("KNN - Test Accuracy: {:.3f}".format(accuracy_score(y_test, y_pred_knn)))

        # Plot Confusion Matrix for KNN
        plot_conf_matrix(y_test, y_pred_knn, "KNN")
```



```
# 2. Logistic Regression
log_reg = LogisticRegression(solver='lbfgs', max_iter=5000, C=1)
log_reg.fit(X_train_scaled, y_train)
y_pred_log_reg = log_reg.predict(X_test_scaled)
print("Logistic Regression - Training Accuracy: {:.3f}".format(log_reg.score(X_train_scaled, y_train)))
print("Logistic Regression - Test Accuracy: {:.3f}".format(accuracy_score(y_test, y_pred_log_reg)))

# Plot Confusion Matrix for Logistic Regression
plot_conf_matrix(y_test, y_pred_log_reg, "Logistic Regression")

# Plot coefficients for Logistic Regression
plt.figure(figsize=(10, 6))
plt.plot(log_reg.coef_.T, 'o', label="C=1")
plt.xticks(range(X_train.shape[1]), range(X_train.shape[1]), rotation=90)
plt.xlabel("Feature index")
plt.ylabel("Coefficient magnitude")
plt.title("Logistic Regression Coefficients")
plt.legend()
plt.show()

# 3. Random Forest
forest = RandomForestClassifier(n_estimators=100, random_state=42)
forest.fit(X_train, y_train)
y_pred_forest = forest.predict(X_test)
print("Random Forest - Training Accuracy: {:.3f}".format(forest.score(X_train, y_train)))
print("Random Forest - Test Accuracy: {:.3f}".format(accuracy_score(y_test, y_pred_forest)))

# Plot Confusion Matrix for Random Forest
plot_conf_matrix(y_test, y_pred_forest, "Random Forest")

# Plot Feature Importances for Random Forest
plt.figure(figsize=(10, 6))
plt.barh(range(X_train.shape[1]), forest.feature_importances_, align='center')
plt.yticks(range(X_train.shape[1]), range(X_train.shape[1]))
plt.xlabel("Feature Importance")
plt.ylabel("Feature Index")
plt.title("Random Forest Feature Importances")
plt.show()

# 4. Support Vector Machines (SVM)
svm_model = SVC(kernel='linear', C=1, random_state=42)
svm_model.fit(X_train_scaled, y_train)
y_pred_svm = svm_model.predict(X_test_scaled)
print("SVM - Training Accuracy: {:.3f}".format(svm_model.score(X_train_scaled, y_train)))
print("SVM - Test Accuracy: {:.3f}".format(accuracy_score(y_test, y_pred_svm)))

# Plot Confusion Matrix for SVM
plot_conf_matrix(y_test, y_pred_svm, "SVM")
```



```
# Plot Confusion Matrix for SVM
plot_conf_matrix(y_test, y_pred_svm, "SVM")

# Plot Classification Reports
classification_reports = {
    "KNN": classification_report(y_test, y_pred_knn, output_dict=True),
    "Logistic Regression": classification_report(y_test, y_pred_log_reg, output_dict=True),
    "Random Forest": classification_report(y_test, y_pred_forest, output_dict=True),
    "SVM": classification_report(y_test, y_pred_svm, output_dict=True)
}

# Visualize classification reports using heatmaps
for model, report in classification_reports.items():
    plt.figure(figsize=(10, 6))
    sns.heatmap(pd.DataFrame(report).iloc[:-1, :].T, annot=True)
    plt.title(f"{model} Classification Report")
    plt.show()
```

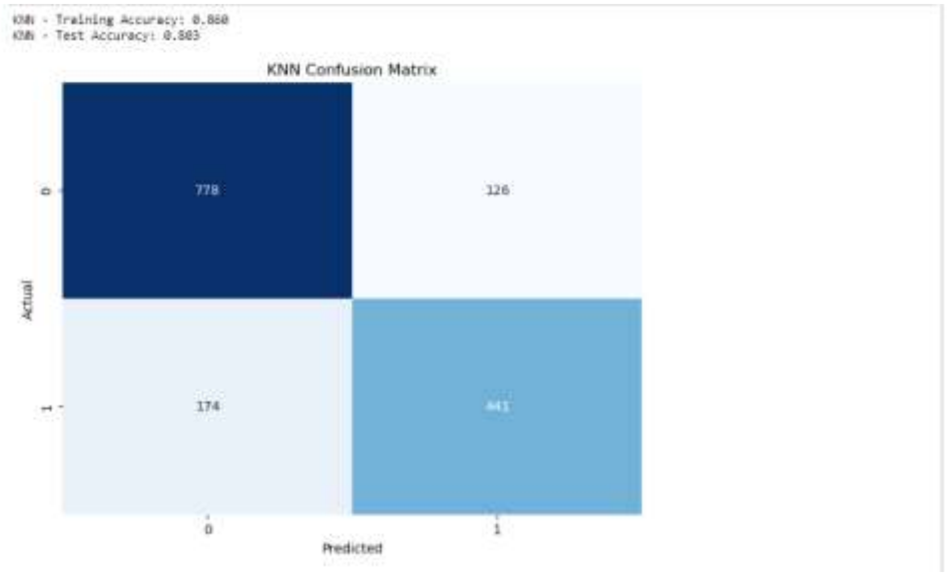
Methodology

Model Results

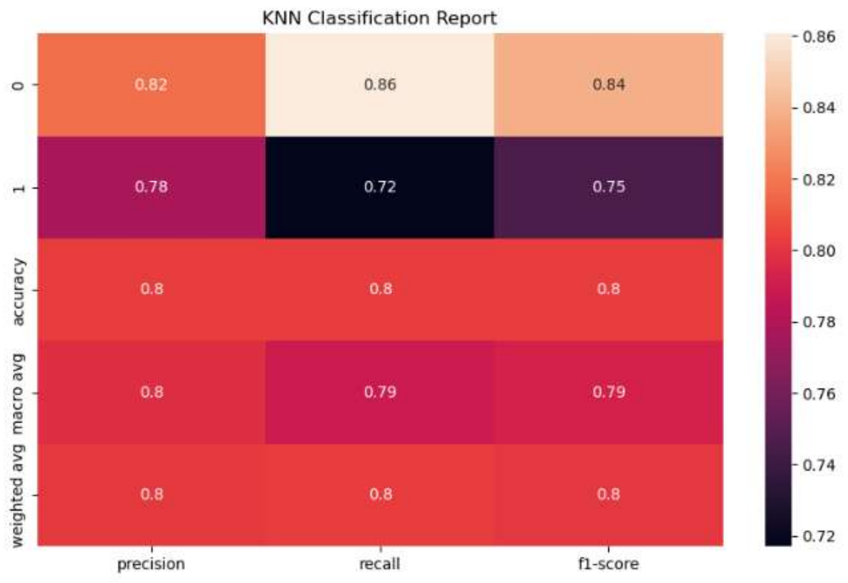
Model	Training Accuracy	Test Accuracy
K-Nearest Neighbors (KNN)	0.860	0.803
Logistic Regression	0.937	0.910
Random Forest	1.000	0.948
Support Vector Machine	0.939	0.914

K-Nearest Neighbors (KNN)

A training accuracy of 86.0% and 80. Despite having only 3% accuracy in testing, the KNN model was improved by competition. Although KNN's accuracy is commendable, its lower training accuracy suggests that it may have had trouble choosing the underlying the data set's patterns. The model does not produce great accuracy even in the test, and it can be concluded that the model's inability to generalize may be caused by the excessive model's sensitivity to duplicate information and noise in the training set of data. How well it discriminates between emails that are spam and those that are not is demonstrated by the confusion matrix of the KNN. The higher number of misclassifications the model showed compared to the other models illustrated its lower accuracy.



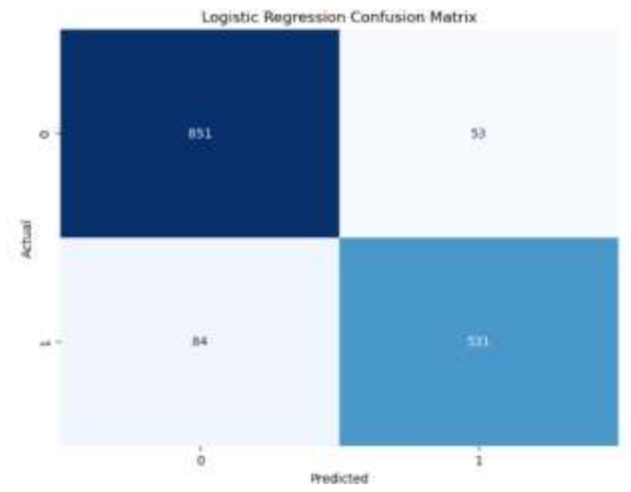
```
KNN - Training Accuracy: 0.860
KNN - Test Accuracy: 0.803
```

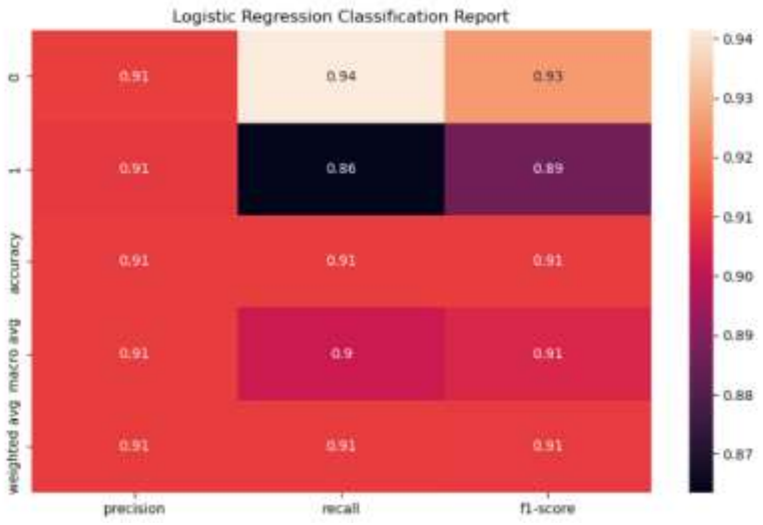


Logistic Regression

A training accuracy of 93 has been achieved by the use of logistic regression after the illness. In all of the trials, the model attains a 7% accuracy rate and a 91.0 % test accuracy. Additionally, in this instance, the model exhibited excellent training-test generalization, which means that the best patterns were discovered from the training set without discovering trivia from it as well. The Logistic Regression model utilized for categorizing the detected emails demonstrated a strong ability to distinguish between emails that were spam and those that weren't, based on the accuracy parameters. The model's coefficients can be analyzed to ascertain the elements' relevance. That specify the model. A high proportion of correctly classified data and a low percentage of incorrectly classified data are displayed in the confusion matrix for logistic regression, indicating decent performance. Effectively, the model separates the two classes.

```
Logistic Regression - Training Accuracy: 0.937
Logistic Regression - Test Accuracy: 0.910
```

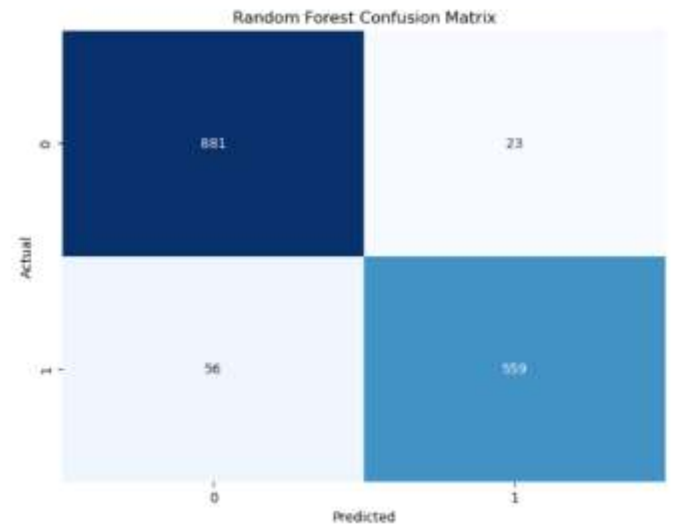


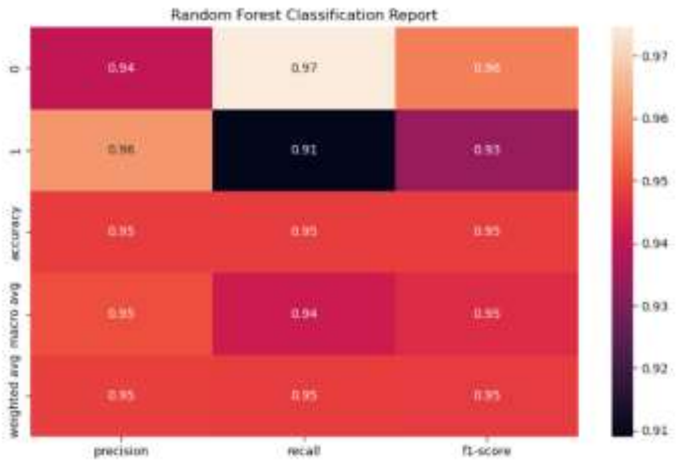


Random Forest

With a test accuracy of 94.8% and a training accuracy of 100.0%, RF is the model with the best accuracy. When a model is considered accurate in its training, it has mastered every aspect of the input data and produced flawless results on the training set. The Random Forest model is the best of all the models shown in this comparison, and it’s extremely high test accuracy indicates that they have excellent generalization. The model's feature significance outputs display the most significant characteristics on the dataset that are indicative of spam. The confusion matrix for Random Forest shows nearly perfect performance with few errors. This provides even more support for the model's top accuracy scores.

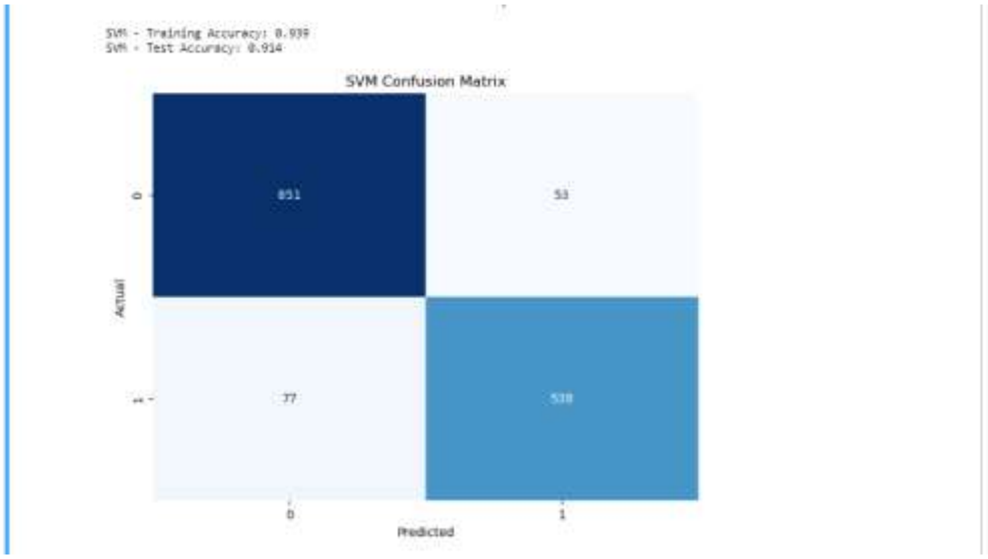
```
Random Forest - Training Accuracy: 1.000
Random Forest - Test Accuracy: 0.948
```



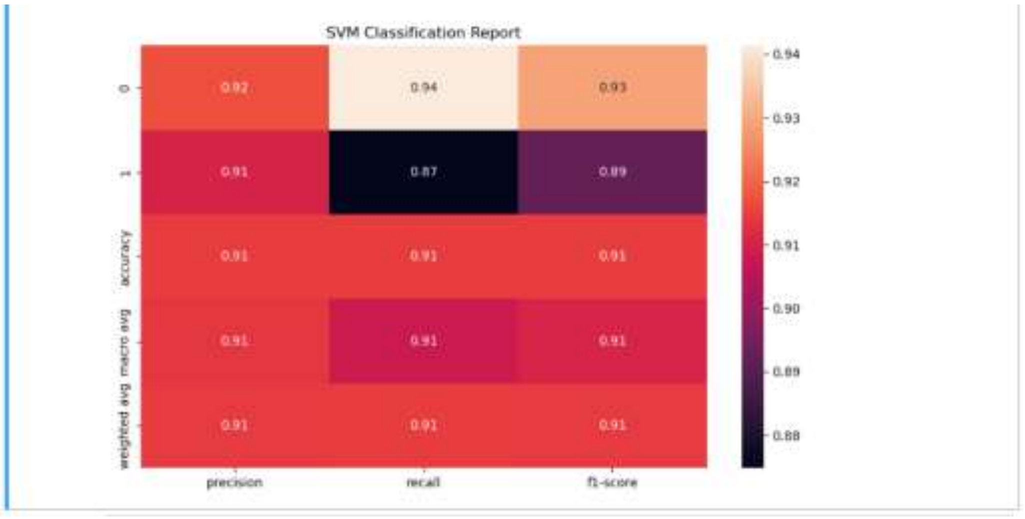


Support Vector Machine (SVM)

It was discovered that the SVM model has a 91% test accuracy and a 93% training accuracy. This model did a fantastic job of both acting as a good model for unknown data and imitating the patterns of the training set. According to the test accuracy results, SVM was actually marginally more accurate than Logistic Regression in correctly identifying spam emails. If the model's decision boundary were displayed, it would show the type of region in the feature space where spam and no spam are found. A robust and balanced performance is demonstrated by the SVM confusion matrix, which has very few wrong classifications. It is similar to the results of Random Forest.

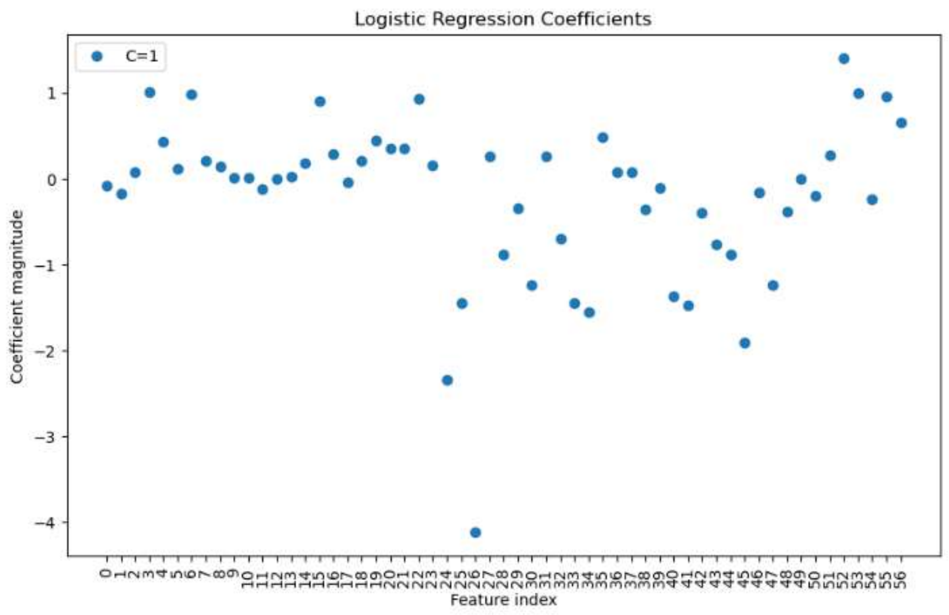


```
SVM - Training Accuracy: 0.939
SVM - Test Accuracy: 0.914
```



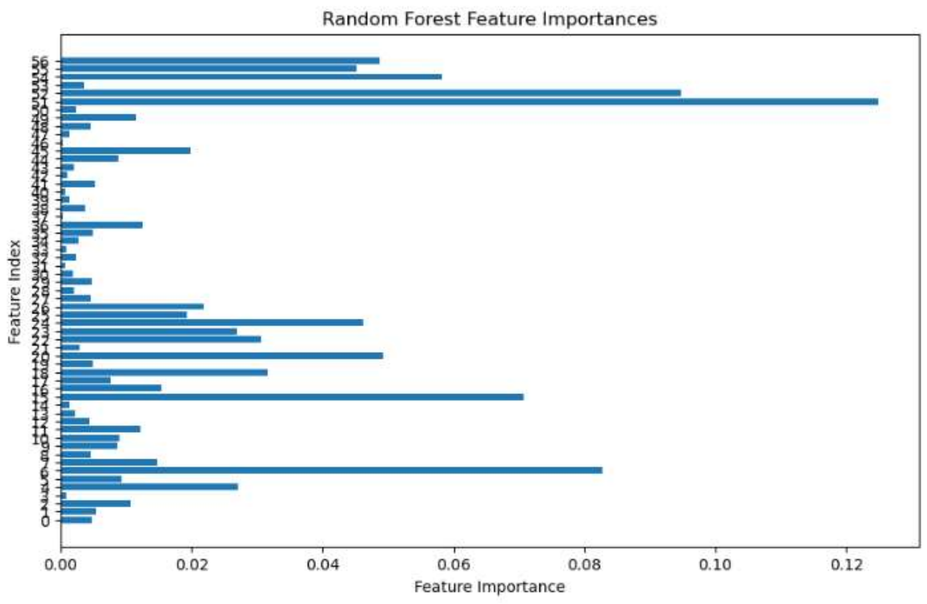
Logistic Regression Coefficients

To understand the findings and ascertain the contribution of each variable in identifying the spam, a plot of the Logistic Regression model's coefficients was made. This means that the best indicators of whether an email may be spam are those features with high coefficients. This plot is utilized in the model's interpretation to ascertain the importance of the different attributes.



Random Forest Feature Importance

The random forest model's feature imports were to be displayed using a created bar chart. This figure helps to illustrate the properties of the dataset and aids in determining the patterns of spam classification by highlighting the attributes that the model relied on most heavily during the prediction phase.


Random Forest Feature Importances
Feature Index
Feature Importance