Introduction:

The advent of spam emails has created a sustained issue for user privacy and safety while using email platforms. Spam emails not only clutter the user's inbox but can be a conduit of several attacks on user security. Phishing, malicious links to hijack user sessions, and the sending of malicious software all occur through spam emails. If a user is not careful about the emails they are opening, or are unaware of the risk, they easily can compromise not only their email account, but also their entire device. These emails are often automated or contain improper grammar but have become more sophisticated in recent years, making them harder to detect.

The goal of this project is to create a machine learning model that accurately classifies spam emails as such and reduce false positives to create a cleaner, safer inbox. By leveraging machine learning algorithms, the model will be able to classify spam emails as such with efficiency and a low rate of false positives.

Method:

The dataset that is being used is the Enron spam dataset that was collected in 2006 and contains 55% spam emails. The dataset is structured as such: six folders, labeled 'enron{n}', where n is one through six. Within each folder there are two sub folders, ham and spam, as well as a summary file that details each. Each subfolder is comprised of their respective email text files.

The model was trained on a feature vector derived from a pandas dataframe. The dataframe was created by the os.walk() method, leaving us with a dataframe that is structured as three columns; 'file_name', 'content' and 'label'. The content is the subject and body of the email, and the labels are 'spam' and 'ham'.

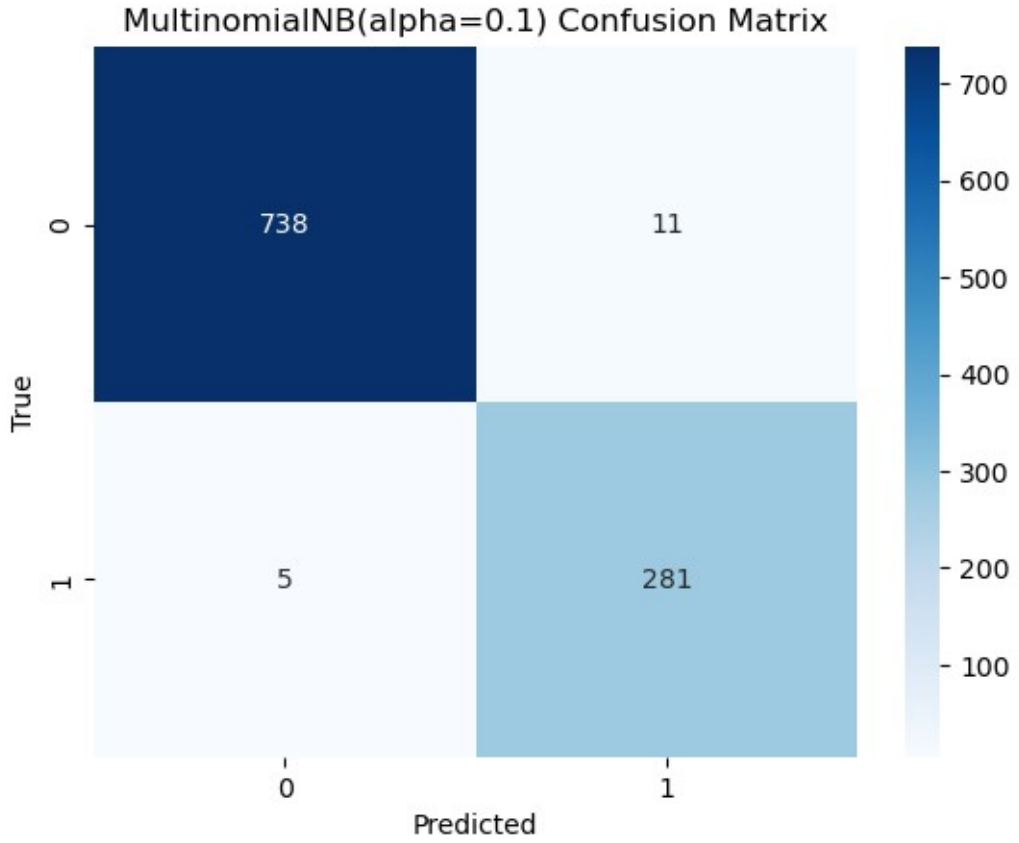
Text preprocessing is done by the function 'preprocess'. Here the text is formatted to be lowercase, removed of punctuation, newlines and stopwords. These filtered tokens are then returned as the processed email body into a separate column in the dataframe. During the 'train function', vectorization is used to convert the email body into floats. Data was split using the train_test_split function in the sklearn library. The vectorizer here was the TF-DIF vectorizer in the sklearn libray. This completes the preprocessing.

Model selection was done by utilizing a five fold cross validation grid. There were 4 possible models that could have been chosen: KneighborsClassifier,

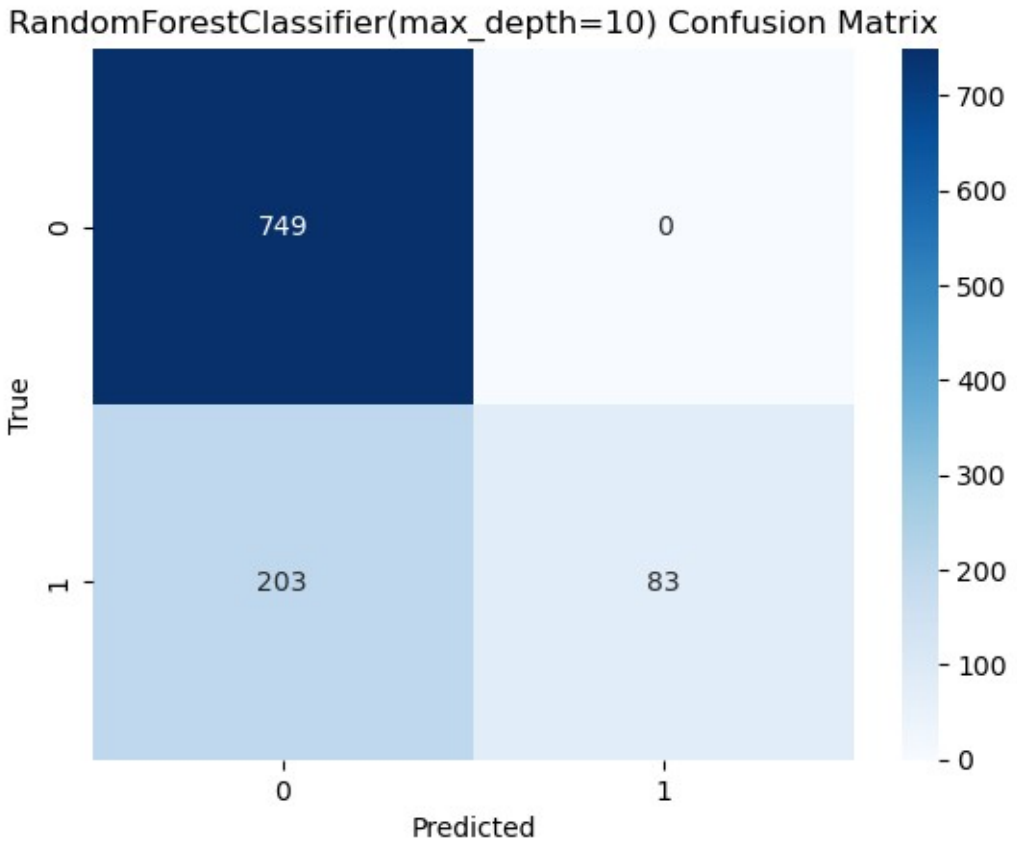
DecisionTreeClassifier, RandomForestClassifier and Multinomial Naive Bayes. Each model is fitted in a loop. After, the models are looped over a separate time, now printing out their accuracy score, f1 score and their optimal parameters. The model selected was Multinomial Naive Bayes (referred to as MNB), which was chosen based on its high accuracy (0.984541) and F1 score (.972318). It was also selected over the RandomForest model, which had suspected overfitting issues based on the fact the best depth for the model was 'none', meaning the tree could grow unrestricted to fit whatever data was given to it. Subsequent restrictions on the RandomForestModel's depth showed a decrease in score, proving these suspicions correct.

Results:

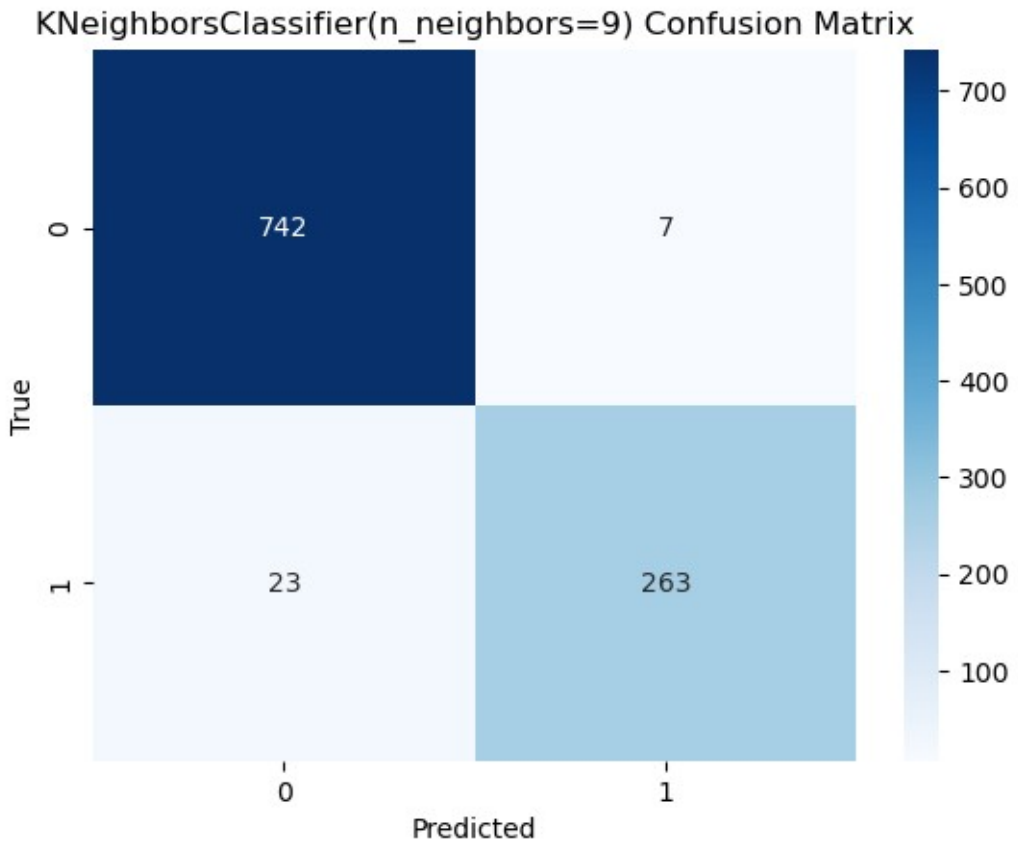
The MNB model was highly accurate at detecting spam emails, with minimal false positives and false negatives. The accuracy on testing data was 0.984541, and the confusion matrix shows a low degree of error.



Random forest, when limited to a depth of 10, loses accuracy significantly. The confusion matrix below emphasises this.



K-Nearest Neighbors also was slightly inferior to the MNB model, as shown in the confusion matrix below:



Conclusions to be drawn are that the RandomForestModel at first seemed like a good fit, both judging by score and intuition. This was dispelled by the overfitting issue that was revealed by the cross validation testing. Knearest neighbors failed slightly by being to lenient on classification. The MNB model was highly accurate and a good choice of model, lending its simplicity and accuracy in text based classification.

In the future it may be beneficial to implement a neural network to classify spam emails, although this might face the same issues of overfitting. This overfitting can be adjusted however and could be prove to be a better model than MNB.