Name: Vaishnavi Vishnu Udanshiv
Student Number: R00224406

Applied Machine Learning Report

Libraries Used:

1. **os**: os library is used in this assignment to read in the dataset and check file paths.
2. **random**: To ensure the unbiased distribution in training and testing data we need to shuffle the data before splitting it and this library is used to shuffle the dataset
3. **numpy**: To get the summary statistics of the data, this library has been used.
4. **sklearn.model_selection(train_test_split)**: This library has been used to split the data into training and testing sets.
5. **pandas**: This library is used to read in the dataset and store it in a pandas dataframe.
6. **seaborn**: This library has been used to visualize the email length and relative frequency of the words used in spam and ham emails with the help of boxplot and histogram respectively.
7. **sklearn.feature_extraction.text**: from this library CountVectorizer and TfidfTransformer classes have been used to convert text data into a numerical format to work with machine learning algorithms, in simpler terms this library provides tools for feature extraction. The TfidfVectorizer class from this library has also been used as a feature extraction method for spam classification.
8. **sklearn.naive_bayes**: From this library MultinomialNB class has been used as a baseline model for spam classification.
9. **sklearn.metrics**: This library has been used for evaluating the performance of machine learning models. Accuracy, Precision, Recall, and F1 score has been calculated for the model with the help of this library.

10. **pickle**: This library has been used to save the trained model to disk so that it can be used later without needing to retrain.
11. **sklearn.linear_model**: From this library the LogisticRegression class has been used to train a model for spam classification.
12. **sklearn.svm**: The SVC class from this library has been used to build a model for spam classification.
13. **matplotlib.pyplot**: This library has been used to create a confusion matrix plot.

1. **Loading the dataset:**

   There are two directories here, one for spam emails and other for ham emails. os.path.join() Function joins main directory path with sub directory path to provide full path. open() Function has been used to read the contents from each file in these directories. Email contents are then stored in the form of strings into two separate lists for spam and ham. Here os module has been used to handle the file paths and directories. Encoding parameter is provided to specify the character encoding for each file which needs to be read. The end result is two lists of strings containing contents of emails from spam and ham directory.

2. **Preprocessing:**

   Next part of the code is about pre-processing the data for building a classification model. First step is to remove duplicates. For this i have converted the list into sets and then again back to list. Next with the strip function, checked if the string is empty and removed empty emails from the list if it existed. Next labeled 1 for spam emails and 0 for the ham emails. Next both the lists for spam and ham emails have been concatenated into a single list and labels are concatenated into a single numpy array. Our final result after preprocessing is input data which is a combined list of emails and target labels which we will be using for training a supervised classification model.

3. **Training and test splits:**

   Next step is to split the dataset into training and testing sets. Here the train_test_split function from sklearn.model_selection package has been used to split the emails and labels into four separate arrays, x_train, x_test, y_train, y_test. 30 percent of the data will be used for testing and remaining 70 percent will be used for training. random_state parameter is used to ensure the reproducibility. Next the count of spam and ham emails have been printed for training and testing datasets and these arrays have been stored in separate compressed files using np.savez_compressed function from the numpy module in order to load them whenever needed without repeating the splitting process.

   Following is the output after splitting
   Training set: 3495 emails (1027 spam, 2468 ham)
   Test set: 1499 emails (436 spam, 1063 ham)

4. **Feature Extraction:**

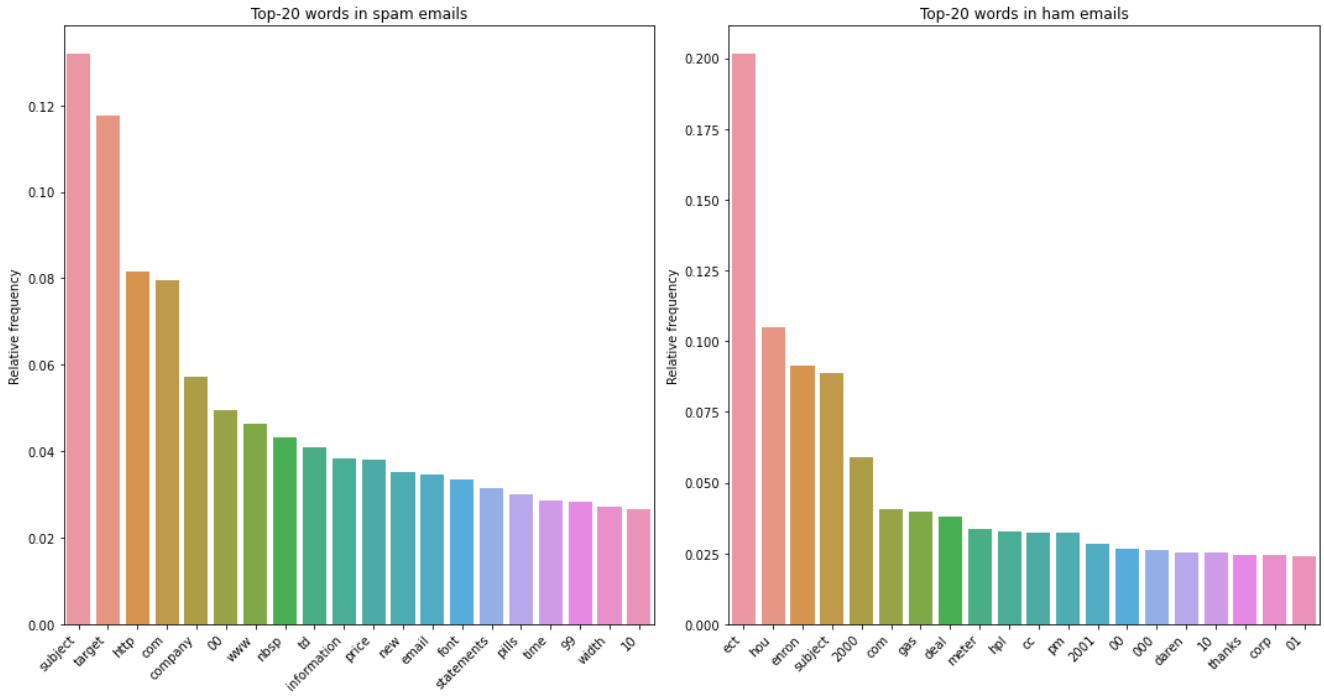
   For this step loaded the training data that we saved in the previous step using np.load() function. Once Emails from x_train and its corresponding labels have been loaded, move to create a bag of word representation. To create a bag of word representation, the CountVectorizer() function from sklearn.feature_extraction.text package has been used. Bag of word representation is a way in which we convert text documents into numerical vectors where each element of the vector corresponds to a specific word from the predefined vocabulary and the value of the vector represents the number of times the word is repeated in the document. Here parameter stop_words have been used to remove common english words like ‘the’, ‘and’ and parameter max_feature is set to limit the size of vocabulary to the 5000 most frequent words.

   Further using fit.transform()method, created a matrix of bags of words and printed the shape of the matrix. The output generated is as follows:
   Shape of x_train_bag_of_words: (3495, 5000)

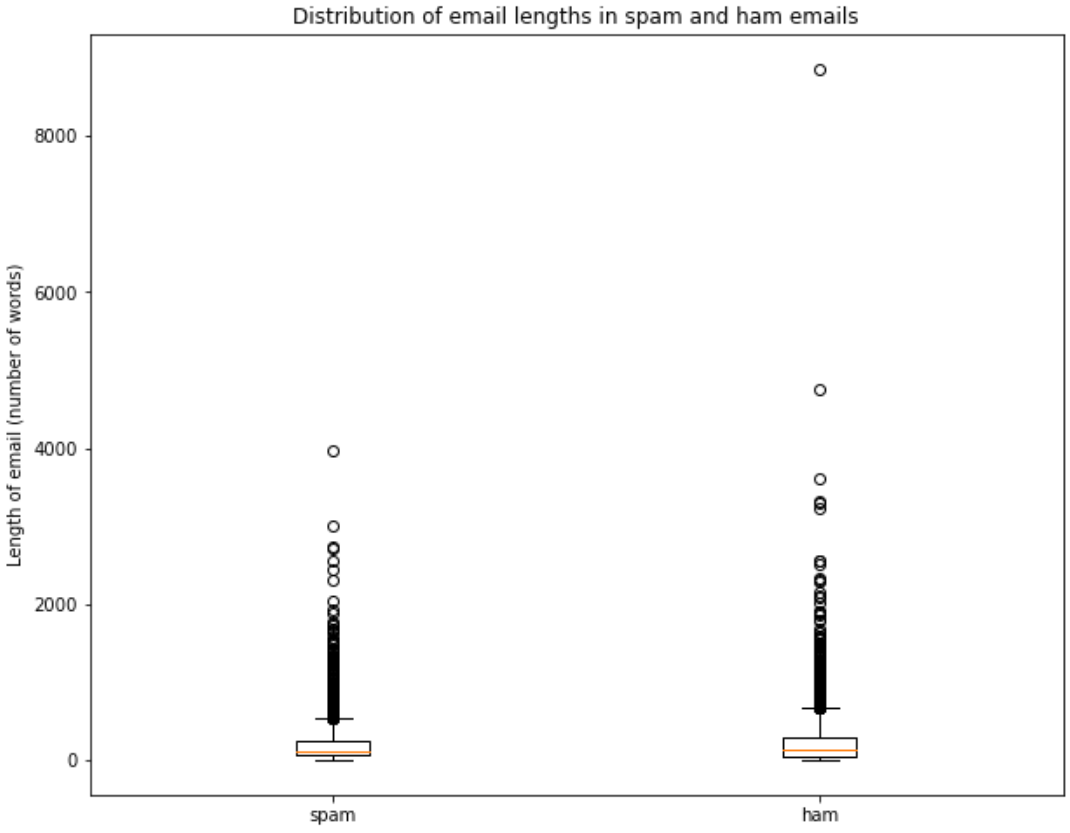
5. **Exploratory data analysis:**

The first step is to generate 20 most frequent words in spam emails and non spam emails. The bag of word representation has been first converted from sparse matrix to pandas dataframe using pd.DataFrame.sparse.from_spmatrix() method and then added target variable to the data frame which is in y_train. To calculate the top 20 most frequent words, use iloc to select all the columns except the last one which is a target variable and sum function to calculate the frequency of each word and the result is sorted in descending order to select the top 20 values.

Once the top 20 words are found, the relative frequency of those words is plotted using sns.barplot() function. And here is the output:



Next created a box plot that compares the distribution of email lengths, depending upon the number of words in each email in spam and ham. And here is the output:



Looking at the boxplot we can say that the distribution of email lengths is more spread out in ham emails than in spam. Boxplot also indicates that ham emails can tend to be longer than spam emails. There are several outliers in ham emails which indicate extremely long emails which are not present in spam emails. Therefore, we can infer that the email lengths can be considered as the distinguishing factor between spam and non spam emails.

6. **Supervised classification:**

   Further, trained multinomial naive bayes classifier on bag of words representation of training data, and then evaluated the model performance on test set.

Output on the test set is as follows:
Accuracy: 0.961 which means the model has correctly classified 96 percent emails on test set.
Confusion matrix shows that out of 1063 ham emails, 1028 have been correctly classified and 35 are incorrectly classified. Similarly out of 436 spam emails , the model correctly classified 412 emails and incorrectly classified 24 emails.

Classification report shows the precision, recall and f1-score for spam and ham:

Classification Report:

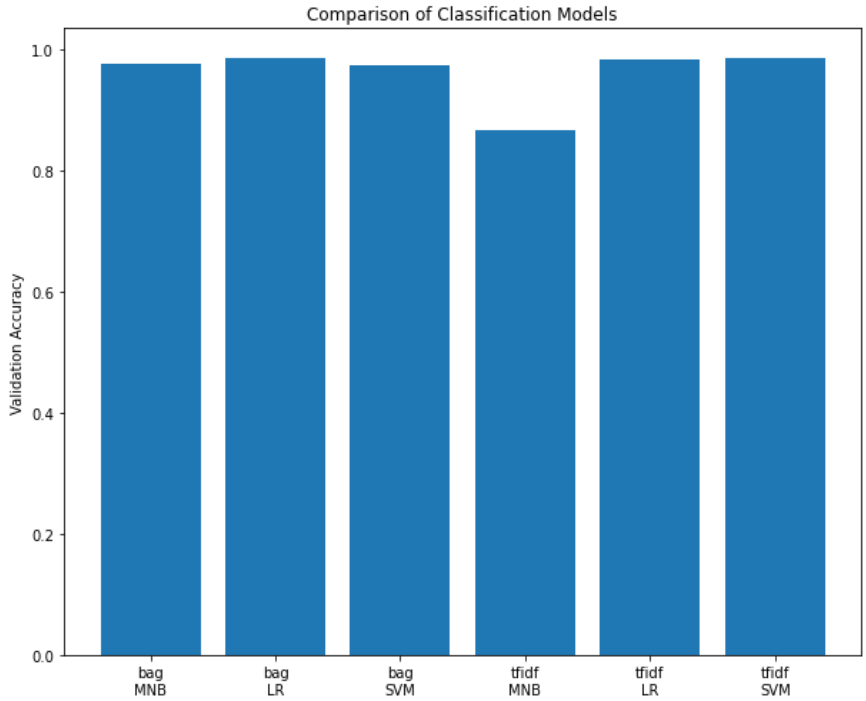
	precision	recall	f1-score	support
0.0	0.98	0.97	0.97	1063
1.0	0.92	0.94	0.93	436

In this case all the three scores for ham emails are greater than spam emails therefore the model is correctly classifying ham emails than the spam emails.

Lastly, saved the model using the pickle module.

7. **Model Selection:**
   Next compared different combinations of vectorizers such as bag of words and tf-idf and classifiers such as multinomial naive bayes, logistic regression and support vector machine. The loop iterates over all the possible combinations of vectorizers and classifiers and trains the model on the training set and evaluates the accuracy on the test set. Results are saved in the list of tuples which contains the name of the vectorizer, name of the classifier and the accuracy of the model. After testing all the possible combinations, to visualize the accuracy of each model plotted the bar plot. With this best model with the highest accuracy is found

Comparison of different vectorizers and classifiers:



After combining various combinations of vectorizers and classifiers on the training data, we found that the best model for classifying spam and ham emails is Support Vector Machine with the vectorizer TF-IDF. The model achieved accuracy of 98 percent on the testing dataset.

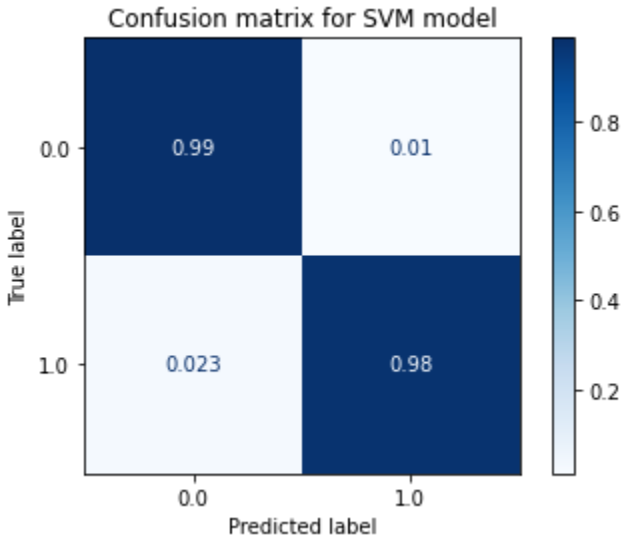
8. Model Evaluation:

Further printed classification report and plotted confusion matrix for the best model.

Following is the output for the classification report:

	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	1063
1.0	0.97	0.98	0.98	436

Further also plotted the confusion matrix for the best model which gives the following result:



The confusion matrix depicts that out of 1063 actual ham emails it correctly classified 1052 as ham and misclassified 11 emails. And out of 436 spam emails it correctly classified 426 spam emails and mis-classified 10 spam emails.

How to Run code to reproduce results:

Random seed is generated to ensure reproducibility. In order to run the code data should be stored in the correct directory. Once the dataset is stored in the working directory under the enron1 folder and necessary packages are imported, Results can be easily reproduced.

Conclusion: Overall the results suggest that SVM classifier with tf-idf vectorizer is a reliable and effective method for identifying spam emails and plot confirms that the model has the highest highest accuracy and performs well for both the classes.