Classification Algorithms for Spam Filtering

Sukhman Singh
18041216

Abstract – In this paper, I will begin with a brief introduction to Naïve Bayes, Neural Networks, and Support Vector Machine which are all classification algorithms. Then the two datasets used to test these algorithms will be described. The first dataset will have a collection of all the original dataset split into 70% training and 30% testing data while maintaining spam to non-spam ratio. The second dataset will use 3/5 folders for training and the other two will be used for testing. Each dataset will go through have pre-process tasks done which are removing unwanted characters, lemmatization, and vectorisation. The goal is to find out which algorithm performs the best in spam filtering. This will be determined by Precision, Recall, and F-value results. I found that Neural Network seemed to perform the best out of the three algorithms in spam filtering with both datasets.

Introduction

Email spam has been a problem since email was first introduced. Spam email is any email sent over the internet which is sent in a bulk amount. To overcome the issue of spam email, solutions using machine learning algorithms has been implemented and have worked well. Algorithms that perform the best with text classification tent to perform the best with this type of problem. Naïve Bayes has been the most popular machine learning algorithm used to sort spam emails and for good reason. Naïve Bayes has been around for the longest dating back all the way to the 1990s. This means that sufficient research and development has been done around the algorithm making it a great algorithm and works well with very large data sets [1]. This is ideal with email spam filtering because usually in this case there is a large set of data to work with. Although Naïve Bayes is the most popular, a new machine learning algorithm using Neural Networks has shown signs of improvement from Naïve Bayes and may even perform better. Neural Network is a system which it's design is inspired by the operation of neurons in the human brain [2]. Neural Networks have been around for a long time but only recently have been usable in problems such email spam filtering. Neural networks are made up of 3 layers, input layer, hidden layer, and output layer. The input layer receives all the data that needs to be processed. The hidden layer which is placed between the input layer and output layer is the inner workings of the network. The output layer has the role of doing the predicting and classifying [3]. Another machine learning algorithm which can be used for email spam filtering is Support Vector Machine (SVM). This algorithm in simple terms creates a boundary between different features of the dataset and classifies by seeing where each data entry falls. SVM would be a great machine learning algorithm for spam filtering because it is widely used in classification objectives [4]. In this paper, I will be comparing Naïve Bayes, Neural Networks, and Support Vector Machine (SVM) in the task of sorting spam and non-spam emails and concluding which one is the best for this task. First the dataset that is being used to test these algorithms will be covered in depth and reveal its characteristics. Next each algorithm will be compared in detail covering the differences, similarities, advantages, and disadvantages of each. Finally, each algorithm will be tested with the same dataset and the results of Precision, Recall and F-value will be compared to determine the best algorithm for this application. In this application, I hypothesize that Neural Networks will perform the best for sorting spam and non-spam emails.

Data Description

The data used in the experiment of comparing the algorithms is sorted into five different folders called "enron1", "enron2", "enron3", "enron4", "enron5". Each of these folders contain a collection of emails sent by a certain company over several years. Each folder contains two sub folders called "ham" and "spam". As the name suggests, the ham folder contains all emails that are considered non spam, and the spam folder contains all the emails that are considered spam. The data is formatted by a text document which represents an email. All the top-level folders are formatted in the exact same way by separated the emails by a ham and spam folder. An important thing to note is that in most cases, there is much more ham emails than there are spam emails. This will be important when splitting the data for training and testing for each algorithm. The dimensionality of the dataset is medium as there is many features to consider when doing spam and non-spam email sorting but there are many samples in our dataset which makes it medium dimensionality. Next the Sparsity of the dataset is quite high because there are a lot of emails that contain missing data and nothing meaningful. Last the Resolution of the data is mid-level where it is not too fine and is not too coarse which makes patterns easier to recognize for the algorithms.

The data in the state it is in right now cannot be used directly to test the algorithms. Further pre-processing must be done through Python in order to have it ready. The first pre-processing task done to the data was to first read all the data in and combine it into one dataset. First all the special and single characters have to be removed as they are not meaningful for the algorithms. Next all text was converted to lowercase and multiple spaces between each word were removed. The next crucial part of part of pre-processing was the

Lemmatization of the text. Lemmatization is the process of analysing words by looking through detailed dictionaries and link the form back to its lemma [5]. This allows the algorithms to make sense of what the different sentences mean in each email. The next essential part of processing the data is the Vectorization of the text. Vectorization in simple terms is converting a body of text and numerically representing it. Algorithms do not understand bodies of text as we humans do. What algorithms do understand well is numbers and that's why the process of the Vectorization to text data is essential in order for the algorithm to correctly operate. Now the data is ready to be split into training and testing sets. This will be done in two different methods. The first method is to split 70% of the data into training and 30% into testing while maintaining the spam and non-spam email ratio of the data. This data will be used first to test the algorithms capabilities. The other method will be to use enron1, enron3, and enron5 datasets for training and enron2 and enron4 for testing. This will be the second data used to test the algorithms. Now we have two sets of data that can be used to test the algorithms.

Classification Machine Learning Algorithms

I'll discuss the three classifier algorithms that will be tested. First classifier I will be discussing will be the Naïve Bayes. As mentioned in the introduction, this is the most popular spam filtering algorithm used and dates back to 1990s. Naïve Bayes algorithms have predictors that assume that a particular feature in a class has no relation to another [6]. This is where the naïve part of the algorithm comes from. The type of Naïve Bayes Classifier used in this application is Gaussian.

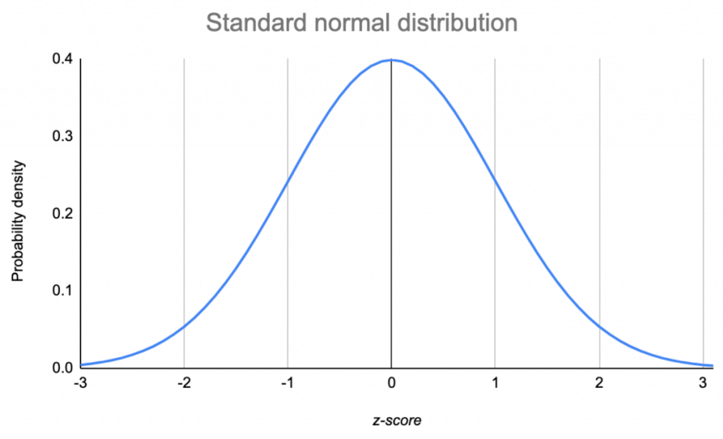


Fig. 1. Graph showing gaussian distribution

For Gaussian Naïve Bayes, the predictors take up a continuous value and are not discrete which are sampled off the gaussian distribution [7]. The advantage of this algorithm is that it works quickly because it is a very simple algorithm so even a not so powerful computer could complete a test with this algorithm within a reasonable time. Another advantage is that this algorithm is suited for categorical input rather than numerical variables which is great in our application because the task is to sort emails into spam and non-spam. Some disadvantages are that in some cases, the estimations can be wrong and that the probability outputs shouldn't be taken seriously. As mentioned before each attribute of the dataset is assumed to be independent for Naïve Bayes, which is rarely the case in the real world, which limits the cases it can be used in. Last disadvantage of this algorithm is that it suffers from a zero-frequency problem. This is when the algorithm assigns zero probability to a categorical variable whose category was in the test data but not in the training [6].

Next algorithm I'll be discussing is the Neural Network. In recent times the Neural Network has shown real promise in improved performance and results in modern problems. One of those problems that it does good in is text classification. As mentioned before, the Neural Network structure consists of three layers. The input layer, hidden layer, and output layer. Each layer is responsible for a different class, and they work together to produce a result. Some advantages of this algorithm are that even with data that has incomplete information, the algorithm should be able to be trained as normal and produce good results. This is a particularly good advantage in our case because there are many email entries with missing information. Another advantage is that all the information for the algorithm is stored on the entire network, which means if data were to be lost in one place, it would not affect functionality. Some disadvantages are that if the network produces unexpected results, it does not give any clue to why this is happening making it difficult to debug. Another one is that there is no standard method of setting up the network structure. This means setting the hidden layers and other parameters. They can only be determined through trail and error which can be time consuming [8].

Last algorithm I'll discuss is Support Vector Machine (SVM). In simple terms, this algorithm works by having a line running across a graph. There will be data samples that will fall under this line or above and that's how the classification happens.

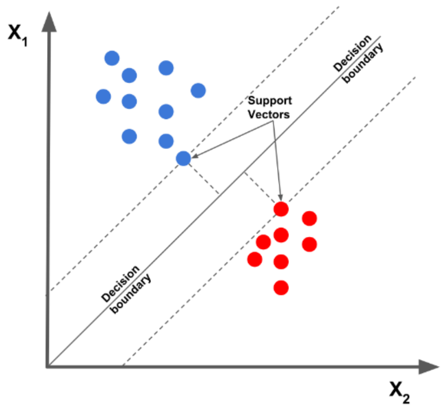


Fig. 2. Graph showing classification in SVMs

You can see in Figure 2 that the line is the decision boundary and has further dotted lines around it which act as a margin. The middle of the graph can be thought of as a road and the dots on the side are like pedestrians [9]. The goal for SVM is to maximize the margin in order to accurately classify each sample. Some advantages of SVM are that it works well with data that has high dimensionality which is the case with our email data. Next advantage is that it is memory efficient meaning computers with memory on the lower end will be able to complete test runs with this algorithm within a reasonable amount time. Some disadvantages are that SVM is not suitable for large datasets which means it will lose some performance in that aspect as our dataset is large. Another one is that SVM does not deal with noisy data well and performance is not always the best [10].

Methodology and Parameters

The way the testing will be done is using the same two sets of the data. The first set of data being used is the collection of all the data split into 70% training and 30% testing data. This split is done while maintaining the spam to non-spam ratio of the whole data. The data pre-processes will be consistent across each test for the algorithms. All algorithms will be tested within the same python run. Gaussian Naïve Bayes has no parameters to change so it will be run as is. Neural Network each layer will have 8 neurons, the activation function used is “relu” and the solver is “adam” with 500 iterations. SVM will have gamma set to auto while the rest of the algorithm will run as is. Performance of each algorithm will be compared in terms of Precision, Recall and F-value. Precision is a measure of the ratio of true positives to the sum of a true positive and false positive [11]. In simple terms precision tells what percentage of the predictions were correct. Next Recall is the measure of a classifier to find all positive instances [11]. Last the F1-value represents the percent of positive prediction that were correct [11]. The best score for this value is 1.0 and the worst is 0.0. These tests will be repeated with the dataset that uses enron1, enron3, enron5 for training and enron2, enron4 for testing. This will ensure that a more accurate conclusion can be performed because we can see which algorithm performs the best in both cases. The algorithm that gets the best Precision, Recall, and F-value will be declared the best performing algorithm in this application.

Results and Discussion

The dataset with the 70% training and 30% testing data will be used first to test each algorithm. The results are shown in Figure 3 below.

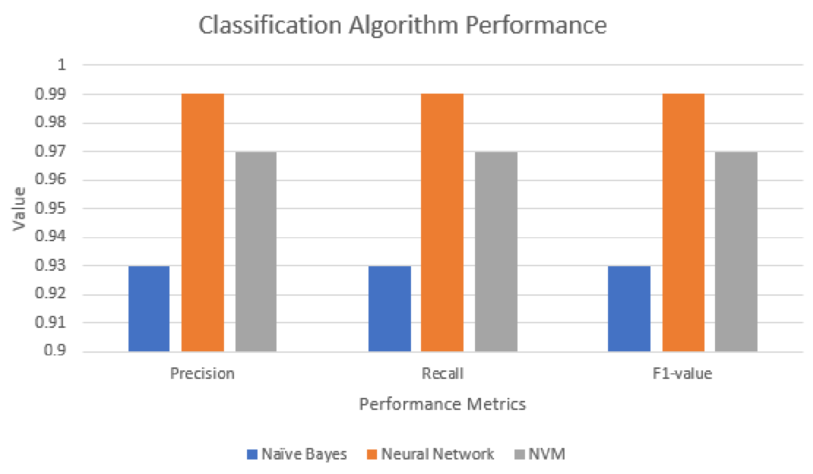


Fig. 3. Performance of algorithms with 70-30 training-testing data

You can see that in Figure 3 that Neural network seemed to be slightly better than NVM, while Naïve bayes is much worse. Note that the weighted average of Precision, Recall and F1-value. Extremely good results are expected from this run because we are using a lot of training data which has similar ratio of spam to non-spam emails as the whole dataset. Also, the data is randomly selected from each folder which means there is a good mix of data which trains the algorithm well and also testing data is similar to training data. Neural Network and NVM sufficiently delivered this by giving 0.99 and 0.97 in all values respectively. While Naïve Bayes under delivered by giving 0.93 in all values. The spam and non-spam ratio are maintained when splitting the data because that will give a scenario where the training and testing data is similar to the original dataset which makes training and testing easier and produces better results. This is basically simulating a situation where the data is ideal. On the other hand, to really test a algorithm, it should be put through difficult situations such as where the data is random and there is less training data and more testing data. This situation will be tested with the dataset which used the files from enron1, enron3, enron5 for training and enron2, enron4 for testing. This data is not ideal because the data can be quite different from each folder. This means that the training data is quite different from the testing data. This is a good way to test the ability of a algorithm because in the real world, this situation is where the algorithm will be used the most. Moving on to results of each algorithm shown in Figure 4 below.

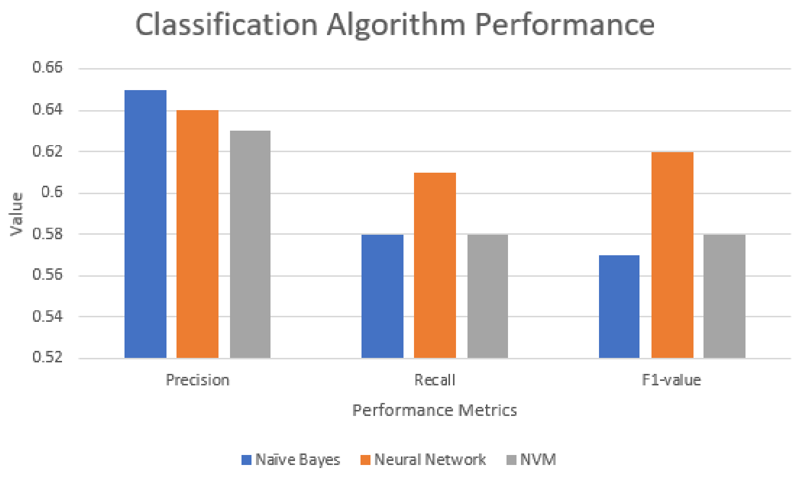


Fig. 4. Performance of algorithms using enron dataset

Notice how the value has drastically dropped for each algorithm compared to last dataset. This is expected because 60% training and 40% testing data is used. This means there is fewer training data than before and more testing data. This makes it more difficult to classify for the algorithms which is shown in the result. The training and testing data are different for each other because they are obtained from different folders. Also, the spam to non-spam ratio is not maintained so that means the training and testing data is different from the original dataset. You can see in Figure 4 that Naïve Bayes has the best Precision value of 0.65 but not far off is Neural Network with 0.64 and NVM not far behind with 0.65. All algorithms performed closely with the Precision, but Naïve Bayes seemed to be the best. Next the Recall you can see that Neural Network performed the best by far while Naïve Bayes and NVM were equal. Last is the f-value which once again Neural Network has the best performance while NVM has the second best and Naïve Bayes third.

Conclusion

To conclude we can see based off the results that Neural Network performed the best in both sets of data. Neural Network only was second in precision for the harder set of data but in every other aspect, it showed the best results. I learned that networks only perform as good as the data which was shown to my by the drastically different results for the two datasets. I also learned that there are many algorithms used for the spam filtering problem and each perform differently case by case basis. In the future, I would add another data set which was even harder where the training and testing data was 50-50. The spam to non-spam ratio will not be maintained to make it more difficult. This will show a good measure of how the algorithms perform with excellent data, average data, and poor data. To conclude, Neural Network is the best performing algorithm in spam filtering compared to Naïve Bayes and NVM.

References

[1] Sunil Ray, "6 Easy Steps to Learn Naïve Bayes Algorithm with codes in Python and R", 2017, https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[2] Ed Burns and John Burke, "What is a neural network? Explanation and examples", 2021, https://searchenterpriseai.techtarget.com/definition/neural-network

[3] S. Abirami and P. Chitra, in "Advances in Computers", 2020, https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron

[4] Rohith Gandhi, "Support Vector Machine – Introduction to Machine Learning Algorithms", 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[5] bitext, "What is the difference between stemming and lemmatization?", 2021, https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/

[6] Pavan Vadapalli, "Naïve Bayes Explained: Function, Advantages & Disadvantages, Applications in 2021", 2021, https://www.upgrad.com/blog/naive-bayes-explained/

[7] Rohith Gandhi, "Naïve Bayes Classifier", 2018, https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

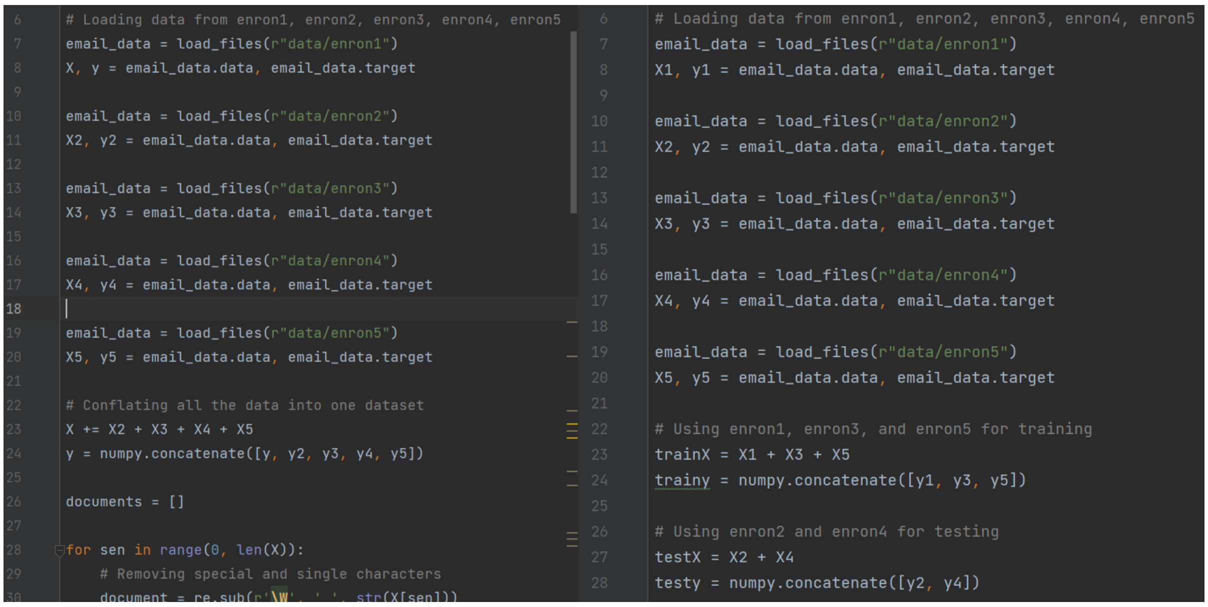
[8] Maad M. Mijwil, "Artificial Neural Networks Advantages and Disadvantages", 2018, https://www.linkedin.com/pulse/artificial-neural-networks-advantages-disadvantages-maad-m-mijwel

[9] Lilly Chen, "Support Vector Machine – Simply Explained", 2019, https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496

[10] Dhiraj K, "Top 4 advantages and disadvantages of Support Vector Machine or SVM", 2019, https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107

[11] Muthu Krishnan, "Understanding the Classification report through sklearn", 2018, https://muthu.co/understanding-the-classification-report-in-sklearn/

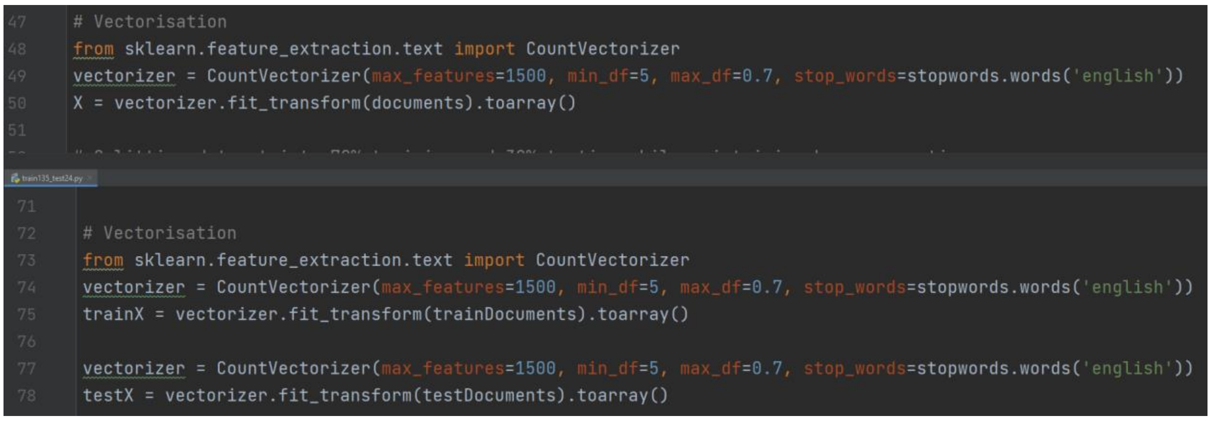
Appendix



```python
# Loading data from enron1, enron2, enron3, enron4, enron5
email_data = load_files(r"data/enron1")
X, y = email_data.data, email_data.target

email_data = load_files(r"data/enron2")
X2, y2 = email_data.data, email_data.target

email_data = load_files(r"data/enron3")
X3, y3 = email_data.data, email_data.target

email_data = load_files(r"data/enron4")
X4, y4 = email_data.data, email_data.target

email_data = load_files(r"data/enron5")
X5, y5 = email_data.data, email_data.target

# Conflating all the data into one dataset
X += X2 + X3 + X4 + X5
y = numpy.concatenate([y, y2, y3, y4, y5])

documents = []

for sen in range(0, len(X)):
    # Removing special and single characters
    document = re.sub(r'\W', ' ', str(X[sen]))
```

```python
# Loading data from enron1, enron2, enron3, enron4, enron5
email_data = load_files(r"data/enron1")
X1, y1 = email_data.data, email_data.target

email_data = load_files(r"data/enron2")
X2, y2 = email_data.data, email_data.target

email_data = load_files(r"data/enron3")
X3, y3 = email_data.data, email_data.target

email_data = load_files(r"data/enron4")
X4, y4 = email_data.data, email_data.target

email_data = load_files(r"data/enron5")
X5, y5 = email_data.data, email_data.target

# Using enron1, enron3, and enron5 for training
trainX = X1 + X3 + X5
trainy = numpy.concatenate([y1, y3, y5])

# Using enron2 and enron4 for testing
testX = X2 + X4
testy = numpy.concatenate([y2, y4])
```

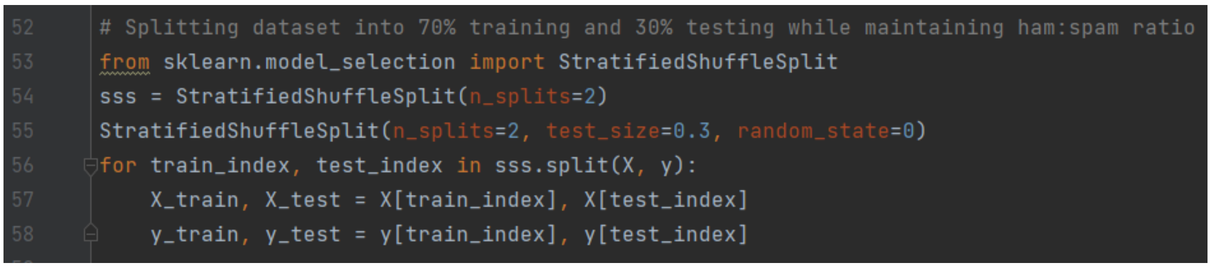
Fig. 5. Comparing loading data for 70-30 training-test data(left) vs enron data (right).



```python
# Vectorisation
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()
```

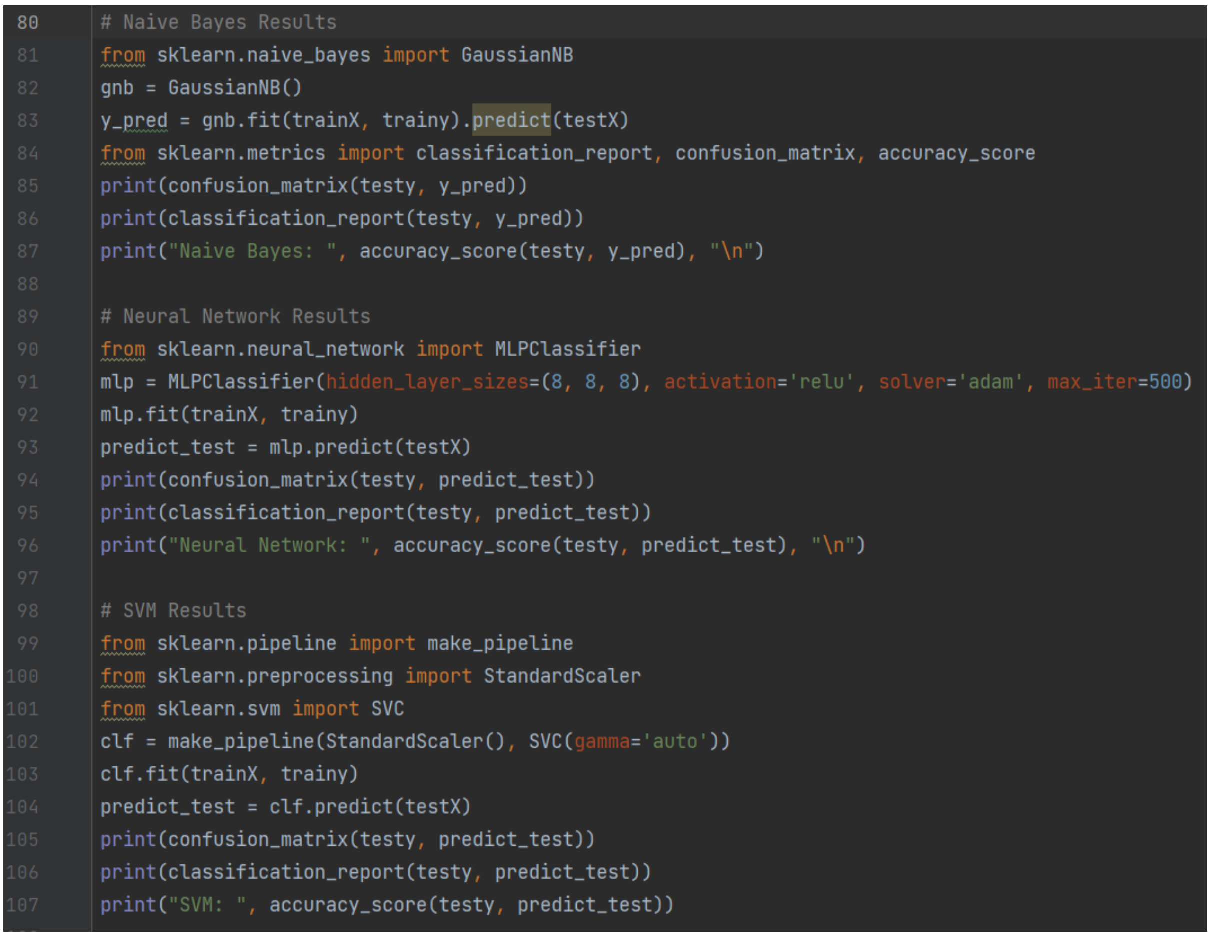
```python

# Vectorisation
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
trainX = vectorizer.fit_transform(trainDocuments).toarray()

vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
testX = vectorizer.fit_transform(testDocuments).toarray()
```

Fig. 6. Comparing Vectorisation for 70-30 data (top) and enron data (bottom).



```python
# Splitting dataset into 70% training and 30% testing while maintaining ham:spam ratio
from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=2)
StratifiedShuffleSplit(n_splits=2, test_size=0.3, random_state=0)
for train_index, test_index in sss.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Fig. 7. Splitting data into 70% training and 30% testing while maintained ham : spam ratio.



```
# Naive Bayes Results
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(trainX, trainy).predict(testX)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(testy, y_pred))
print(classification_report(testy, y_pred))
print("Naive Bayes: ", accuracy_score(testy, y_pred), "\n")

# Neural Network Results
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(8, 8, 8), activation='relu', solver='adam', max_iter=500)
mlp.fit(trainX, trainy)
predict_test = mlp.predict(testX)
print(confusion_matrix(testy, predict_test))
print(classification_report(testy, predict_test))
print("Neural Network: ", accuracy_score(testy, predict_test), "\n")

# SVM Results
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf.fit(trainX, trainy)
predict_test = clf.predict(testX)
print(confusion_matrix(testy, predict_test))
print(classification_report(testy, predict_test))
print("SVM: ", accuracy_score(testy, predict_test))
```

Fig. 8. Displaying classification results for each algorithm (same for both datasets)