<u>Description:</u>

This experiment is about implementing the support vector machine (Supervised learning algorithm) and feature selection. In this algorithm, training and testing are done on ‘Spambase’ dataset. The spambase dataset contains 4601 sample example of spam and not spam emails. Support vector machine is a supervised learning algorithm which is used in this experiment for the classification of spam and not spam emails. The spam-base dataset is divided into training set and test set, each has the same portion of the positive and negative examples in the entire set. The training set is scaled using a preprocessing technique and using mean and standard deviation of training data, we scaled the test data for testing. In this algorithm, data item as a data point in n-dimensional space (where n is a number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiates the two classes. To see the effect of different features on the accuracy, we choose the feature with the magnitude of the highest weight of the support vector. In one experiment we choose the feature randomly. The results are shown for each experiment below.

<u>Experiment [1]:</u>

In this experiment, after reading the dataset from the file, the data is divided into attributes and labels. The model_selection library from scikit-learn library contains the test_train_split method to divide the dataset into training and testing data. The training and testing of spambase dataset are done using scikit-learn package (SVM). SVC (C-support vector classification) class is used for classification. The implementation of SVC class is based on the libsvm library of scikit-learn package. SVC class handled the multiclass using one to one scheme. In this experiment, the SVC class takes one parameter, which is kernel type, we used the ‘Linear’ kernel parameter for training and testing the data.

Accuracy: 92.30 %

Precision: 0.92096

Recall: 0.8815

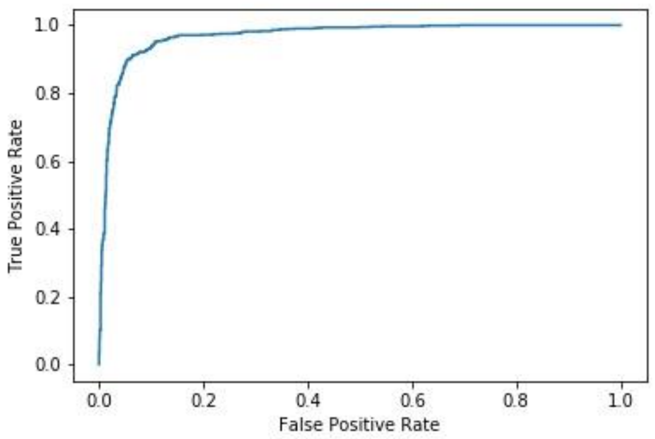

Figure. 1: ROC curve for test data

The fit method is used to train the training data and predict method is used for testing the test data. Based on the predicted and actual labels, calculated the accuracy, precision, and recall for our test data. The roc_curve method of matrices library from scikit-learn library is used for calculating the true positive rate and false positive rate for ROC curve.

Experiment [2]: Feature selection with linear SVM.

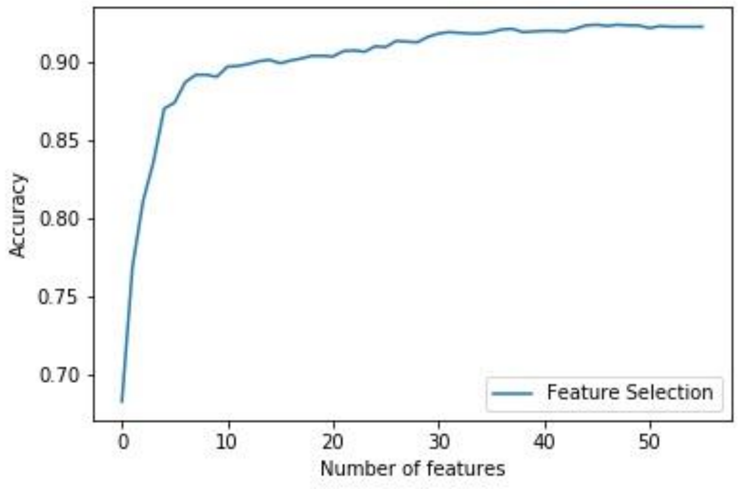

Figure.2: Accuracy (on test data) vs number of features for feature selection

For this experiment, we selected the number of features based on the highest magnitude of weight (feature). We can observe from the above graph that as we increase the number of features, the accuracy also increased to a certain point after that it starts decreasing. For the above graph, the accuracy is 68.21% for a number of feature =2, the accuracy is 92.30% for the number of feature =57, and we get the maximum accuracy of 92.43% for the number of feature =45. Thus, there are an only certain number of features that are important for the prediction (For example in this experiment only 45 features are important) and if we use more number of features, the accuracy starts decreasing.

Feature	Weight
word_freq_remove	9.67391151652852
word_freq_free	9.419686863687085
word_freq_hp	-13.882161204323303
word_freq_george	-16.131117308923354
char_freq_!	10.9860170889894

Figure.3: Top 5 features with highest weight magnitude from SVM

Here, the negative weight for the particular feature indicates that the email is less likely to classify as spam email. For this top 5 features, the emails which contain word ‘george’, is less likely to predict as spam email and the emails which contains word ‘free’, is more likely to predict as spam email.

Experiment [3]: Random feature selection

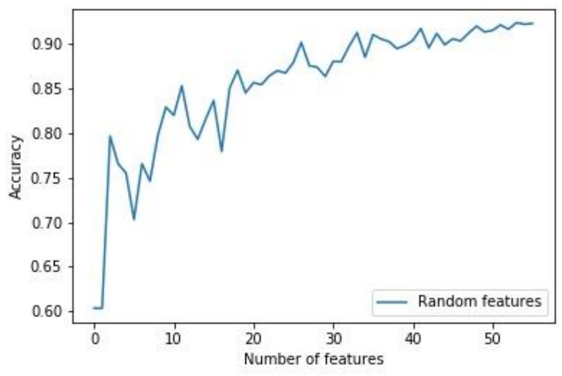

Figure.4: Random feature selection

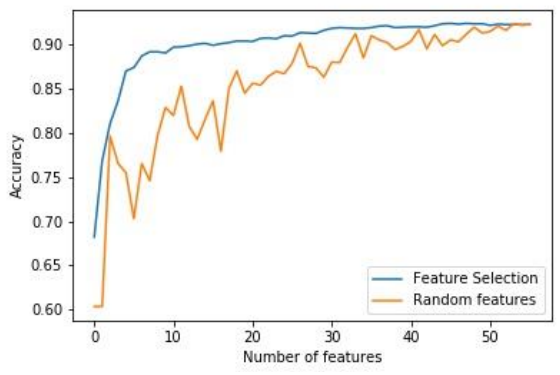

Figure.5: Feature selection vs Random features

For this experiment, features are selected randomly. We can observe from the graphs that there is too much oscillation (noise) in accuracy for less number of feature and accuracy is low as compare to weighted feature selection. Random features give an accuracy 60.34% for a number of features=2 and it is not fixed. It might get lower or higher, that depends on the randomly generated two features because every time it generates two random features, it is different from previously generated two random features. When two random features are generated, which are less likely to predict as spam, in that case, an accuracy will even get lower. Where weighted feature selection gives an accuracy of 68.21% for a number of features=2, which is fixed because of the weighted feature selection. Thus, here selection of features based on the highest magnitude of weight (feature) is more important than random feature selection. Feature selection helps in filtering the noise in dataset and benefits the algorithm. Weighted feature selection might help in avoiding the overfitting when we don’t have lots of data for training.