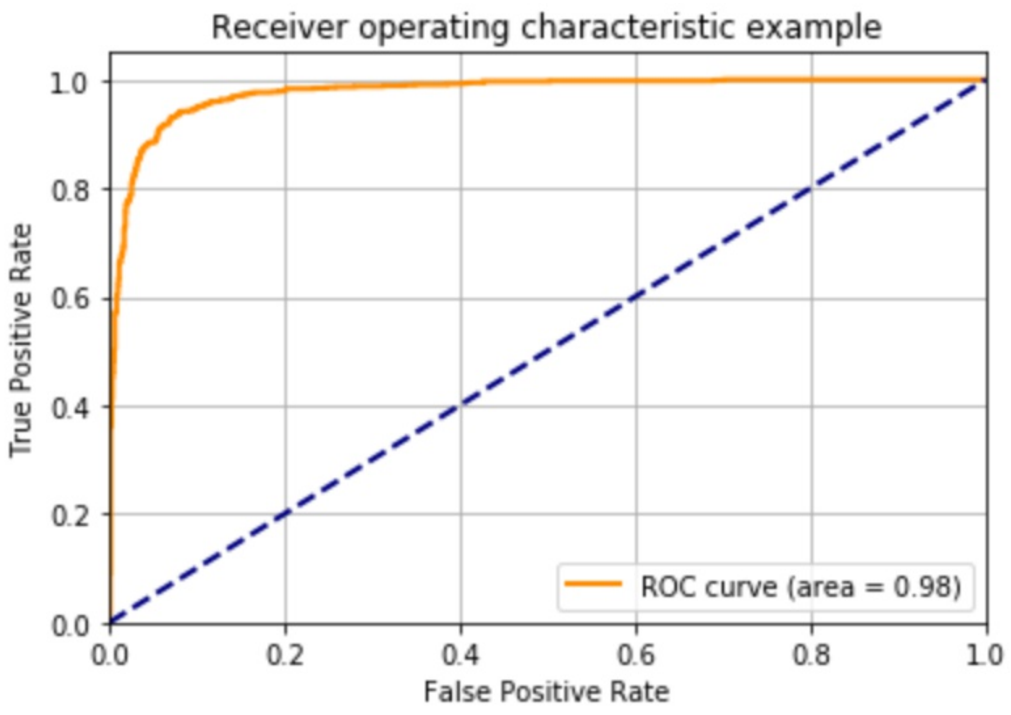
Experiment 1

I used the svm package provided by the scikit learn library

The following are the evaluation metrics I got for the above experiment

```
Accuracy: 0.9273913043478261
Precision: 0.937046004842615
Recall: 0.8706411698537683
```

The following is the ROC curve



Experiment 2

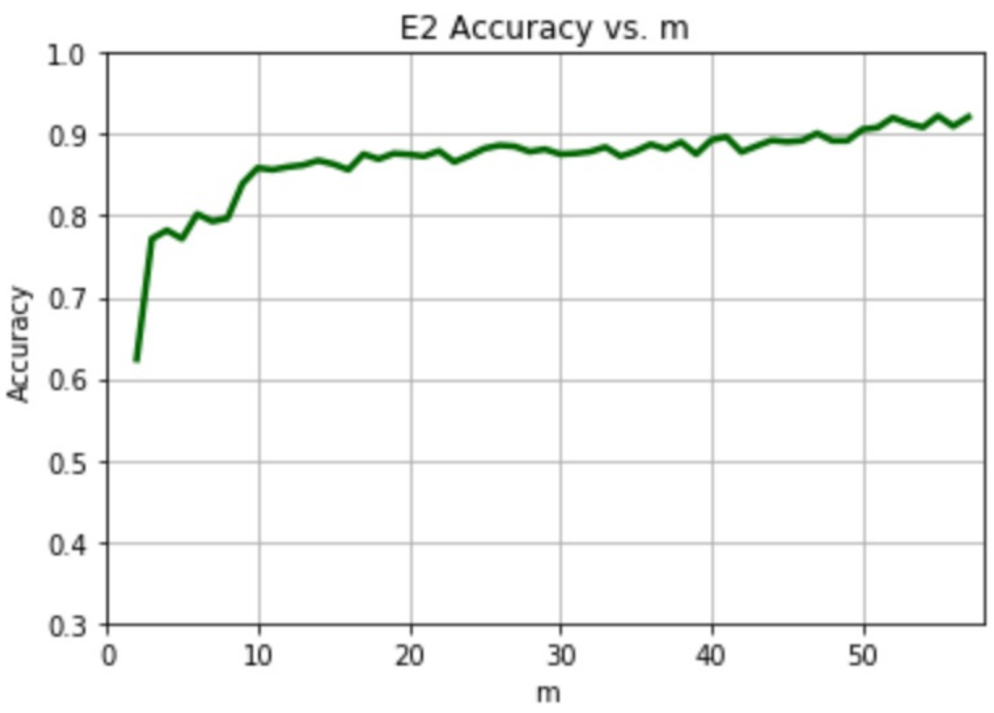
```
weights = np.reshape(clf.coef_, (57,))
L = [ (weights[i],i) for i in range(len(weights)) ]
L.sort(reverse=True)
weights_sorted,index = zip(*L)

X = spam_data.iloc[:,:-1]

m_list = []
a_list = []
features = []
features.append(weights_sorted[0])
```

```
for i in range(1, len(index)):
    m_list.append(i+1)
    features.append(index[i])
    X_m = spam_data.iloc[:, lambda spam_data: features]
    X_train, X_test, y_train, y_test = train_test_split(X_m, y, test_size=0.5)
    X_train_scaled = preprocessing.scale(X_train)
    scaler = preprocessing.StandardScaler().fit(X_train)
    X_test_scaled = scaler.transform(X_test)
    clf = LinearSVC()
    clf.fit(X_train_scaled, y_train)
    a_list.append(clf.score(X_test_scaled, y_test))
```

The following is the ROC curve



The following were the top 5 features picked by SVM:
george,cs,hp,meeting,415

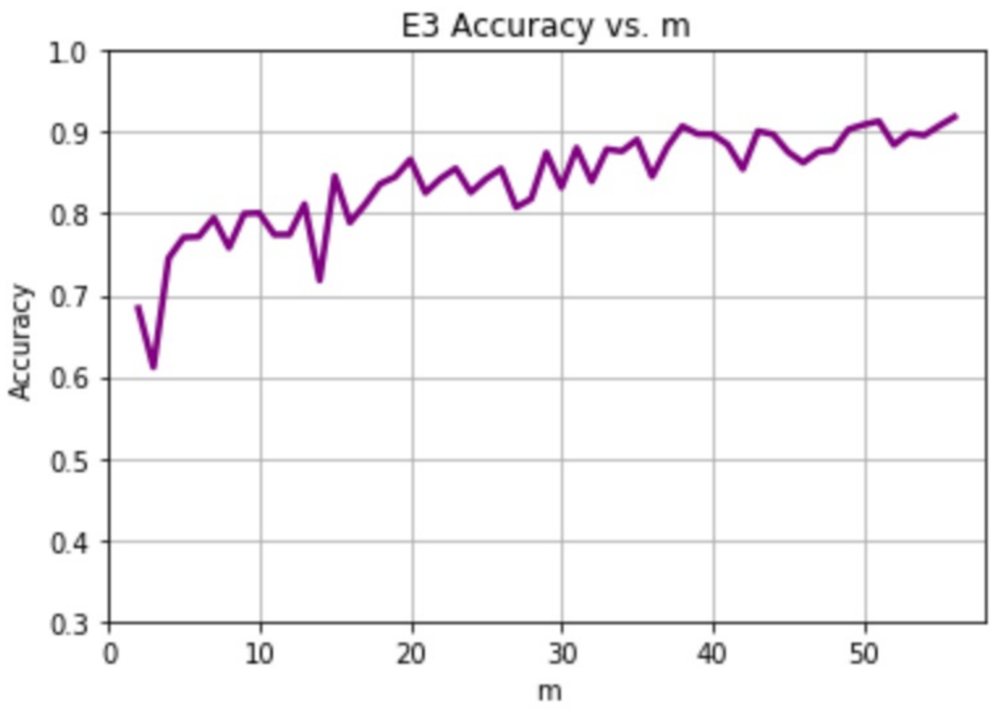
Going with the top features is not improving the models prediction, in fact it has low accuracy. Eventually when all the features get picked it reaches the same accuracy which I got in experiment 1. As you can see accuracy steadily increase when m increases and there are few cases when m < 57 the model reached highest accuracy, but it's not a very significant jump.

Experiment 3

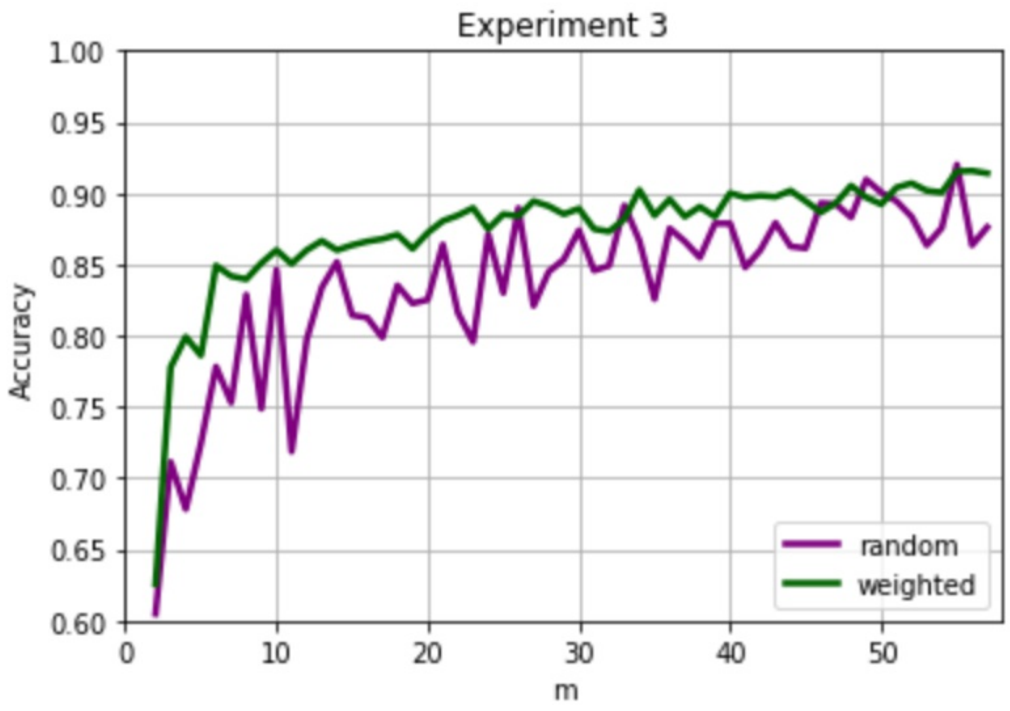
```
e3_a_list = []
```

```
for i in range(2, 58):
    features = np.random.uniform(0,57,i)
    X_m = spam_data.iloc[:, lambda spam_data: list(features)]
    X_train, X_test, y_train, y_test = train_test_split(X_m, y, test_size=0.5)
    X_train_scaled = preprocessing.scale(X_train)
    scaler = preprocessing.StandardScaler().fit(X_train)
    X_test_scaled = scaler.transform(X_test)
    clf = LinearSVC()
    clf.fit(X_train_scaled, y_train)
    e3_a_list.append(clf.score(X_test_scaled, y_test))
```

The following is the ROC curve



As one can see in the graph below. In both the experiments the accuracy seems to steadily improve with m value being increased. With random selection there is a lot of fluctuations. This seems to be so because, the random number generator might have picked features which are not that significant predicts for spam/not spam. Eventually as the m value gets close to 50s there was one instance where accuracy value shot up the accuracy value for that particular m on feature selection graph. But then it sharply dropped when the m value increased. Feature section is much reliable since it's consistent.



P.S. I have included my full code in the jupyter notebook.