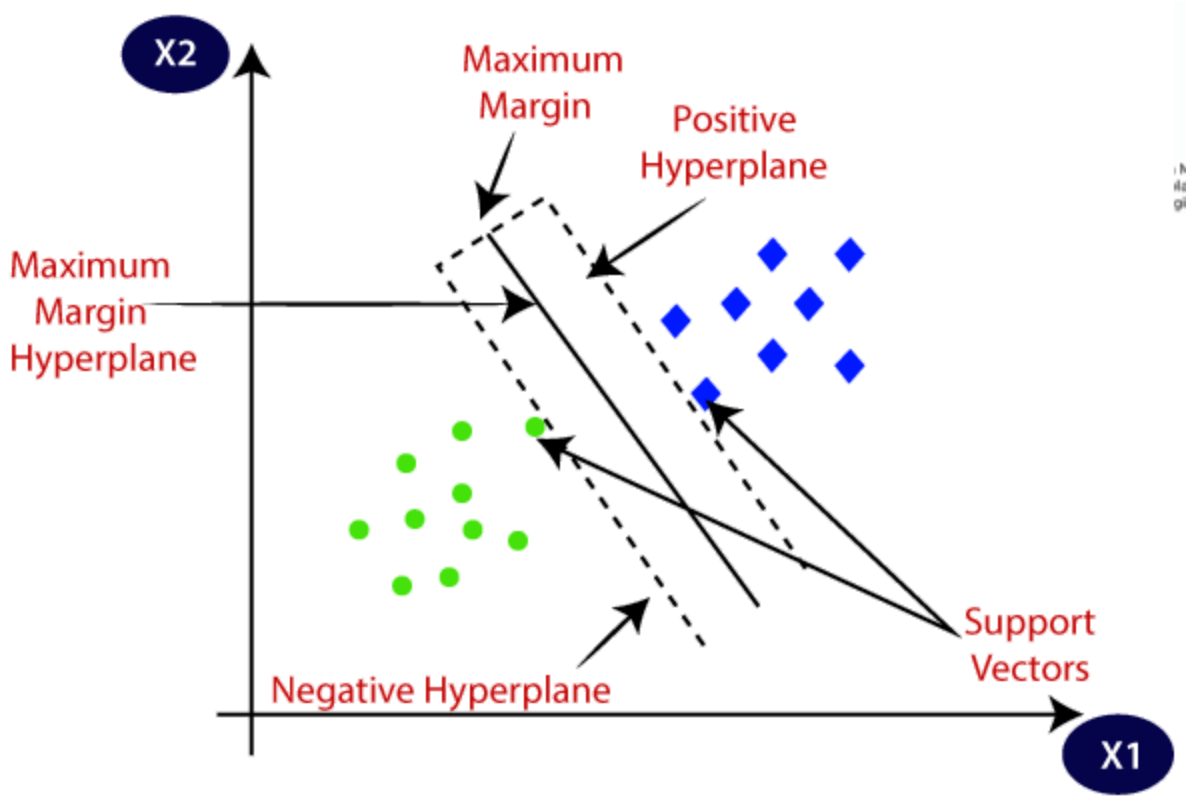
Pattern Recognition and Machine Learning

Support Vector Machine Classifier

Kartik Choudhary [B20CS025]



Data Preprocessing

An email is represented by various features like frequency of occurrences of certain keywords, length of capitalized words etc. A data set containing about 4601 instances and 58 columns.

The last column of the dataset denotes whether the Email was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

Out of about 4601 instances Spam emails were found to be 1813 and non-spam emails were found to be 2788.

70% of the dataset is chosen as training data and the remaining as test data.

In this report we are going to run a Support Vector machine classifier with different kernels(linear,gaussian,polynomial (degree 2)) and also tune the parameter C to find out the best performing model .

Methodology:

The SVM Package used by me for the following lab assignment is sklearn.svm.SVC.

It is a C-Support Vector Classification, if it comes to multiclass classification the package

handles it with one vs one classification

Parameters for SVC:

The 2 major parameters for SVC are C , Kernel, degree(if kernel = polynomial).

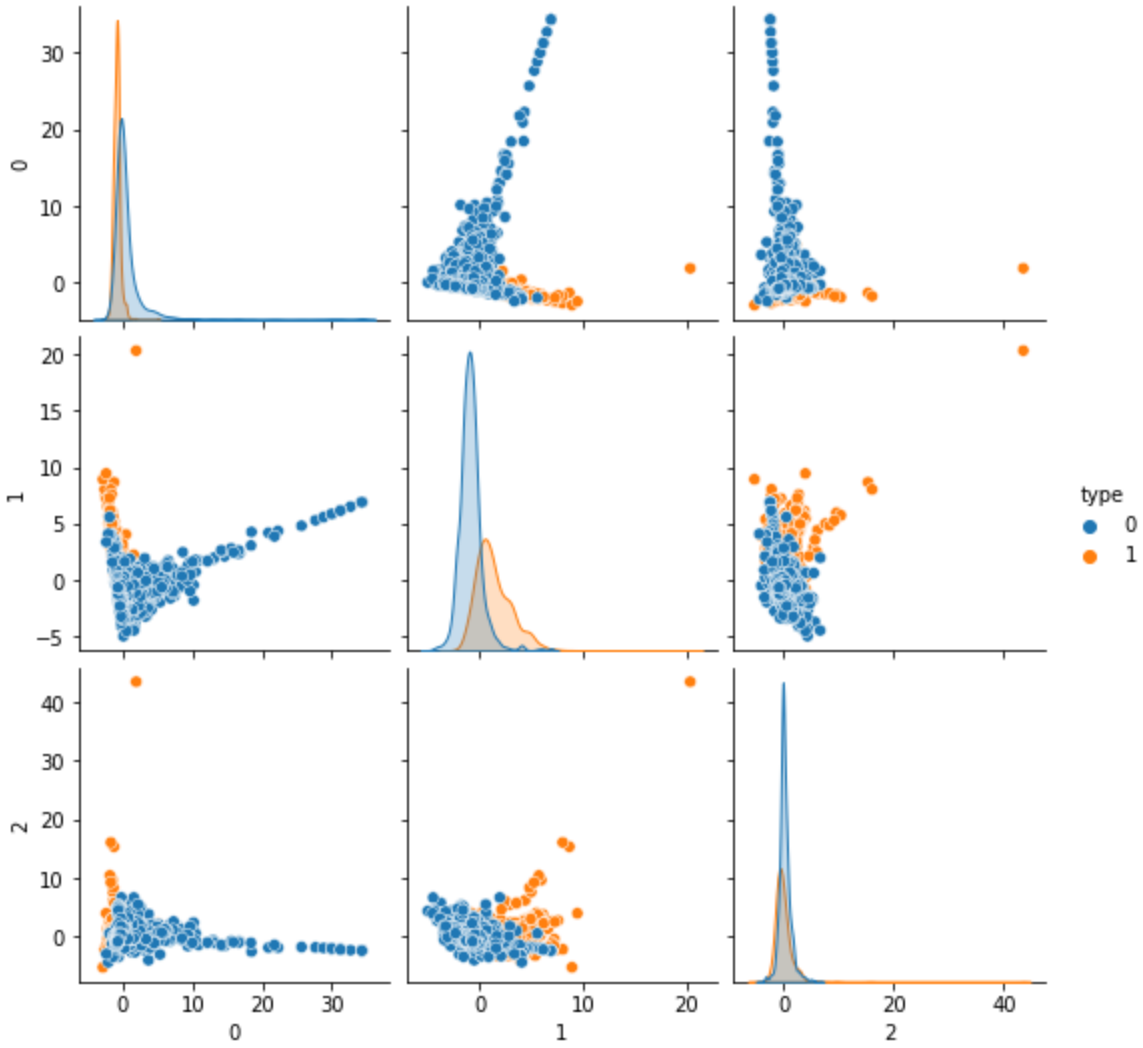
The three different types of kernels that was required to implement are:

- **Linear Kernel:** It is mainly used when the data is linearly separable or there are a large number of features.
- **Quadratic Kernel:** It is a non-linear SVM Kernel.
- **RBF:** Kernel Radial Basis Function Kernel, the unique thing about it is that its complexity increases with the size of the training set.

Experimental Results:

Visualization:

Making Pairplot by reducing the dimension using PCA to number of components as 3



Model Training and Evaluation:

On running SVM with default parameters: (Without modifying value of C)

```
Accuracy Score on testing data:
0.8731884057971014
```

1) Using Default Linear Kernel

```
svc=SVC(kernel='linear')
```

```
Accuracy Score on testing data:
0.8775362318840579
```

2) **Using Default polynomial Kernel of degree 2**

```
svc=SVC(kernel='poly', degree=2)
```

```
Accuracy Score on testing data:
0.731159420289855
```

3) **Using Default RBF Kernel**

```
svc=SVC(kernel='rbf')
```

```
Accuracy Score on testing data:
0.8731884057971014
```

We can conclude from above that svm by default uses rbf kernel as a parameter for kernel.

Polynomial kernel is performing poorly.The reason behind this maybe it is overfitting the training dataset.

Performing K-fold cross validation with different kernels.(Default)

CV on Default Linear kernel taking cv = 10 we get the mean score:

```
Mean Accuracy score on training Data for Default Linear SVM Kernel with CV = 10
0.8726086956521739
```

CV on Default rbf kernel taking cv = 10 we get the mean score:

```
Mean Accuracy score on training Data for Default RBF SVM Kernel with CV = 10
0.8706521739130434
```

CV on Default polynomial kernel with degree 2 taking cv = 10 we get the mean score:

```
Mean Accuracy score on training Data for Default polynomial(deg=2) SVM Kernel with CV = 10
0.7323913043478261
```

When K-fold cross validation is done we can see different score in each iteration.This happens because when we use train_test_split method,the dataset get split in random manner into testing and training dataset.Thus it depends on how the dataset got split and which samples are training set and which samples are in testing set. With K-fold cross validation we can see that the dataset got split into 10 equal parts thus covering all the data into training as well into testing set.This is the reason we got 10 different accuracy scores.

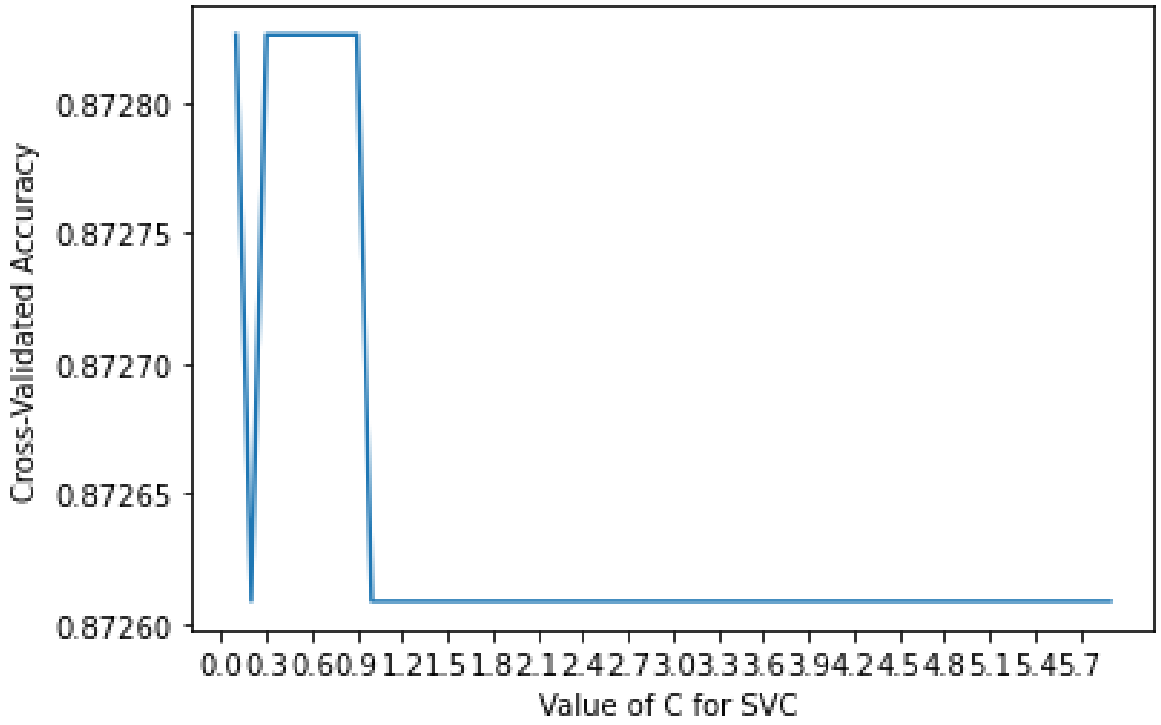
Taking all the values of C and checking out the accuracy score with different kernels.

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassified more points.

Thus for a very large values we can cause overfitting of the model and for a very small value of C we can cause underfitting.Thus the value of C must be chosen in such a manner that it generalized the unseen data well

.

For Linear Kernel:



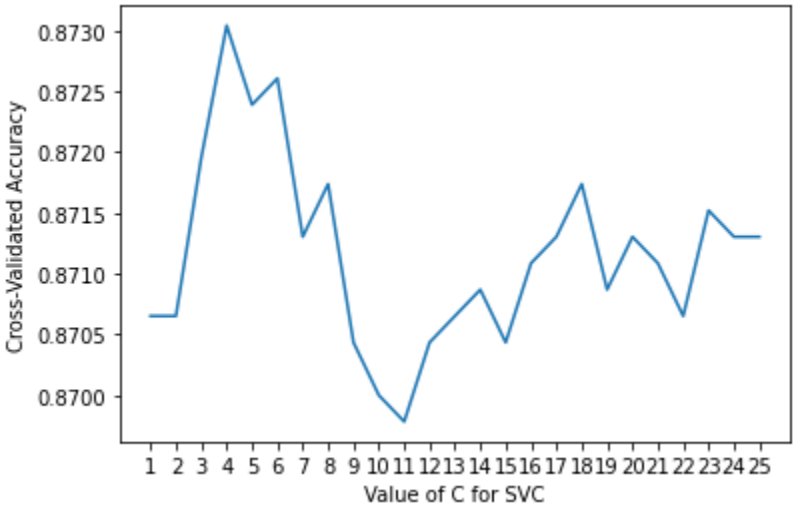
From above accuracy score for linear kernel is highest for C = 0.3.

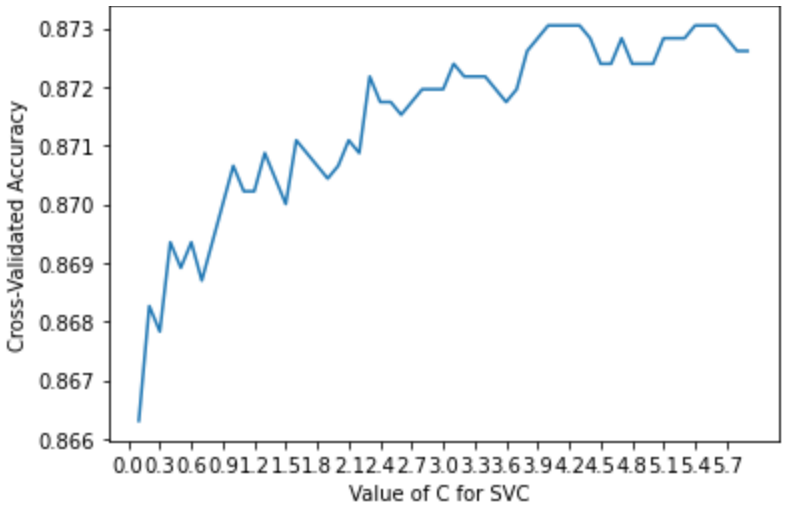
Performing SVM by taking hyperparameter C=0.3 and kernel as linear

```
Accuracy on testing data
0.8775362318840579
```

```
Accuracy on Training data
0.8728260869565219
```

FOR RBF KERNEL:





From above accuracy score for RBF kernel is highest for C = 4.

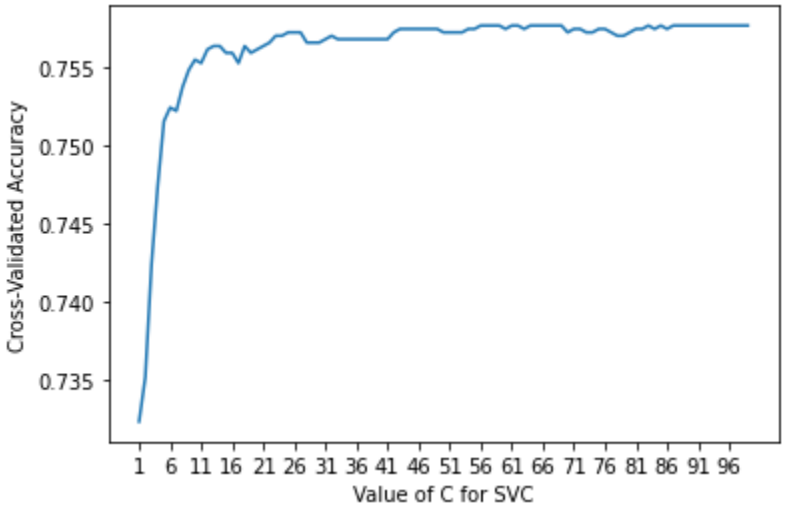
Performing SVM by taking hyperparameter C=4 and kernel as rbf

```
Accuracy on testing data:

0.8717391304347826
```

```
Accuracy on training data:
0.8726086956521739
```

FOR POLYNOMIAL KERNEL WITH DEGREE = 2:



for C around 51-60 the accuracy is highest.

Lets have more closer look

```
(acc_score[50:61])
✓ 0.6s
[0.8728260869565219,
 0.8728260869565219,
 0.8728260869565219,
 0.8730434782608697,
 0.8730434782608697,
 0.8730434782608697,
 0.8728260869565216,
 0.8726086956521739,
 0.8726086956521739]

Thus the highest accuracy is obtained for C = 53
```

Now performing SVM by taking hyperparameter degree=2 and kernel as poly and c= 53.

```
Accuracy on testing data:
0.7471014492753624
```

```
Accuracy on training data:
0.7571739130434783
```

KERNEL	Training Accuracy	Testing Accuracy	Value of C
Linear	0.8728260869565219	0.8775362318840579	0.3
Quadratic Kernel	0.7571739130434783	0.7471014492753624	53
RBF	0.8726086956521739	0.8717391304347826	4

Linear Kernel has outperformed both RBF and Quadratic Kernel in Test Accuracy