Support Vector Machine Classifier

Problem Statement: Spam email classification using Support Vector Machine: In this assignment you will use a SVM to classify emails into spam or non-spam categories. And report the classification accuracy for various SVM parameters and kernel functions. You have to submit the report file in pdf format. No programs need to be submitted. Data Set Description: 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 are available in this link (data folder):

https://archive.ics.uci.edu/ml/datasets/Spambase

The data format is also described in the above link. You have to randomly pick 70% of the data set as training data and the remaining as test data. You have to study performance of the SVM algorithms. You have to submit a report in pdf format. The report should contain the following sections:

1. **Methodology: Details of the SVM package used.**
2. **Experimental Results:**
   - **A. You have to use each of the following two kernel functions (a) Linear, (b) Quadratic,**
   - **B. For each of the kernels, you have to report training and test set classification accuracy for the best value of generalization constant C. The best C value is the one which provides the best test set accuracy that you have found out by trial of different values of C. Report accuracies in the form of a comparison table, along with the values of C.**

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data"

colnames=list(range(57))
colnames.append("spam")
# print(colnames)

dataset = pd.read_csv(url, names=colnames)
dataset.head()
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	2
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.64	0.00	0.00	0.00	0.32	0.00	1.29	1.93	0.00	0.96	0.0	0.0
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	0.21	0.79	0.65	0.21	0.14	0.14	0.07	0.28	3.47	0.00	1.59	0.0	0.4
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	0.38	0.45	0.12	0.00	1.75	0.06	0.06	1.03	1.36	0.32	0.51	0.0	1.1
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0.00	0.00	0.31	0.00	0.00	3.18	0.00	0.31	0.0	0.0
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0.00	0.00	0.31	0.00	0.00	3.18	0.00	0.31	0.0	0.0

In [5]:

```
X = dataset.drop('spam', axis=1)
y = dataset['spam']
```

In [6]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

In [ ]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[815  31]
 [ 64 471]]
              precision    recall  f1-score   support

           0       0.93      0.96      0.94       846
           1       0.94      0.88      0.91       535

    accuracy                           0.93      1381
   macro avg       0.93      0.92      0.93      1381
weighted avg       0.93      0.93      0.93      1381
```

In [ ]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=2)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[822  24]
 [436  99]]
              precision    recall  f1-score   support

           0       0.65      0.97      0.78       846
           1       0.80      0.19      0.30       535

    accuracy                           0.67      1381
   macro avg       0.73      0.58      0.54      1381
weighted avg       0.71      0.67      0.60      1381
```

In [13]:

```
from sklearn.svm import SVC
C=3.0
svclassifier = SVC(kernel='linear' , C=C)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[806  40]
 [ 62 473]]
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       846
           1       0.92      0.88      0.90       535

    accuracy                           0.93      1381
   macro avg       0.93      0.92      0.92      1381
weighted avg       0.93      0.93      0.93      1381
```

In [14]:

```
from sklearn.svm import SVC
C=10000.0
svclassifier = SVC(kernel='linear' , C=C)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[774  72]
 [ 50 485]]
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       846
           1       0.87      0.91      0.89       535

    accuracy                           0.91      1381
   macro avg       0.91      0.91      0.91      1381
weighted avg       0.91      0.91      0.91      1381
```

In [7]:

```
from sklearn.svm import SVC
C=3.0
svclassifier = SVC(kernel='poly', degree=2, C=C)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[831  17]
 [436  97]]
              precision    recall  f1-score   support

           0       0.66      0.98      0.79       848
           1       0.85      0.18      0.30       533

    accuracy                           0.67      1381
   macro avg       0.75      0.58      0.54      1381
weighted avg       0.73      0.67      0.60      1381
```

In [8]:

```
from sklearn.svm import SVC
C=10000.0
svclassifier = SVC(kernel='poly', degree=2, C=C)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[818  30]
 [195 338]]
              precision    recall  f1-score   support

           0       0.81      0.96      0.88       848
           1       0.92      0.63      0.75       533

    accuracy                           0.84      1381
   macro avg       0.86      0.80      0.81      1381
weighted avg       0.85      0.84      0.83      1381
```

In [9]:

```
from sklearn.svm import SVC
C=100000.0
svclassifier = SVC(kernel='poly', degree=2, C=C)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[814  34]
 [118 415]]
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       848
           1       0.92      0.78      0.85       533

    accuracy                           0.89      1381
   macro avg       0.90      0.87      0.88      1381
weighted avg       0.89      0.89      0.89      1381
```

Observations

1. **Using a linear kernel gave good results even without using any regularization**
2. **Using a polynomial(quadratic) kernel, we get good results only for the negative results. Using a large regularization term gives good results overall**
3. **It is observed that as we increase the regularization term(C), the results on negative side become less and less accurate, and the positive side accuracy increases, at least for this particular case, in both the kernels.**

Model used

The model used here is from sklearn.svm package, from which SVC is used. It is the C-support vector classification. This is a good model for a few thousand data samples, but not for very large datasets, like tens of thousands or lakhs. However, for our purposes it suits well enough. Through this, we can define the regularization parameter, type of kernel, degree(if polynomial), along with other plethora of options.