INTRODUCTION TO MACHINE LEARNING

Assignment 2-2021AMA2090(Rahman Alam)

Contents

`

NON-AIP701-PART 1_1

1. LIBSVM library

LIBSVM library is developed by Chih-Chung Chang and Chih-Jen Lin which provide various modules for SVM. We can provide hard margin to soft margin, choose type of degree, choose type of kernels type also can be chosen. Few advantages are here as follows (Taken directly)

- *Different SVM formulations*
- *Efficient multi-class classification*
- *Cross validation for model selection*
- *Probability estimates*
- *Various kernels (including precomputed kernel matrix)*
- *Weighted SVM for unbalanced data*

Options available in library are.

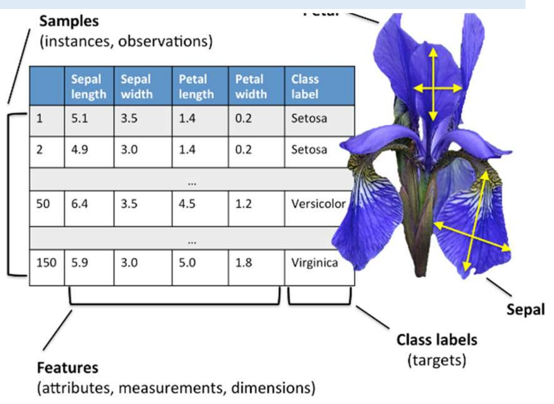
```
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
```

Methodology-
We will basically be using or changing the svm type, kernel type, degree, Kernel parameter(gamma),and the margins on the svm .keeping other constants.

The Toy Data set used will be the well-known IRIS dataset which is having 4 input features and one out having 3 classes.

This dataset is having features as sepal length, sepal width Petal length, and petal width. The features can be scaled also for Faster conversions. But this data set is very small just 150 data We don't need the scaling. I will be using LIBSVM to train on 120 data points and test on 30 datapoints.

https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html


Samples
(instances, observations)
Sepal
Class labels
(targets)
Features
(attributes, measurements, dimensions)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

2. Other Library

There is SVM library in scikit learn also where almost all the features which are available in LIBSVM can be found.

But Lib SVM is faster than library of SVM in scikit learn.

3. SVM training on Toy data set(IRIS)

Training on default parameter of LIBSVM

param=svm_parameter('-s 0 -c 1 -t 2 -g 1 -d 3 ')

Report Card

True Positive= 0

False Positive= 0

True Negative= 25

False Negative= 5

Confusion Matrix-->

```
                  T   F(Actual)

Predicted   T  0   0

            F  5  25
```

Precision= nan

Recall= 0.0

Accuracy= 83.3

F Score= 0.0

Parameters-

Time to Train(ms)= 3.50

svm_type c_svc

kernel_type rbf

gamma 1

nr_class 3

total_sv 36

label 0 1 2

nr_sv 8 14 14

Here the type of SVM used is C-SVC(s=0)

Type of kernel used is RBF(t=2)

The kernel parameter used is 1(g=1)

Margin used is 1(c=1)

And degree used is 3(d=3)

We get on this setting accuracy around 83.3% which we don't know whether we can achieve more than that or not.

For that we will iterate and check how high and how low we are getting the accuracy.

Precision and recall came to be NAN and 0 respectively as TP and FP are zero.

Labels are encoded as

0-Iris-Setosa

1-Iris-versicolor

2-Iris-Verginica

Gamma

It is the kernel coefficient for kernels 'rbf', 'poly' and 'sigmoid'.

4. Varying the parameters

Varying the kernel type t=[0,1,2,3]

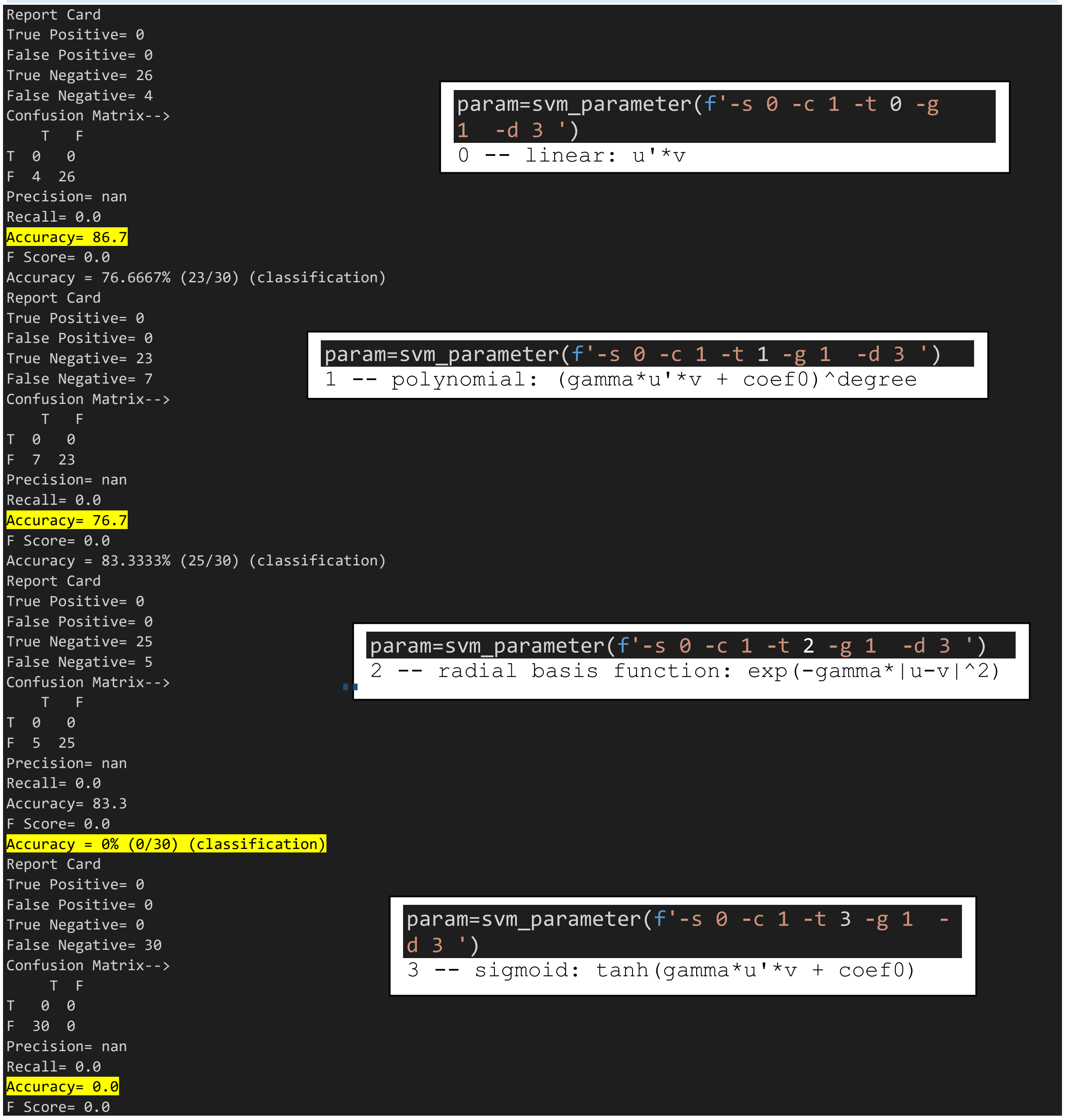
same kernel parameter

margin same

degree same=3

Maximum Accuracy achieved is=86.7%

Changing the SVM type results are changing(showing results below for few values only)



```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
```

```
param=svm_parameter(f'-s 0 -c 1 -t 0 -g
1  -d 3 ')
0 -- linear: u'*v
```

```
Accuracy = 76.6667% (23/30) (classification)
Report Card
True Positive= 0
False Positive= 0
True Negative= 23
False Negative= 7
Confusion Matrix-->
    T   F
T  0   0
F  7  23
Precision= nan
Recall= 0.0
Accuracy= 76.7
F Score= 0.0
```

```
param=svm_parameter(f'-s 0 -c 1 -t 1 -g 1  -d 3 ')
1 -- polynomial: (gamma*u'*v + coef0)^degree
```

```
Accuracy = 83.3333% (25/30) (classification)
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
```

```
param=svm_parameter(f'-s 0 -c 1 -t 2 -g 1  -d 3 ')
2 -- radial basis function: exp(-gamma*|u-v|^2)
```

```
Accuracy = 0% (0/30) (classification)
Report Card
True Positive= 0
False Positive= 0
True Negative= 0
False Negative= 30
Confusion Matrix-->
     T  F
T   0  0
F  30  0
Precision= nan
Recall= 0.0
Accuracy= 0.0
F Score= 0.0
```

```
param=svm_parameter(f'-s 0 -c 1 -t 3 -g 1  -
d 3 ')
3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

Same kernel type

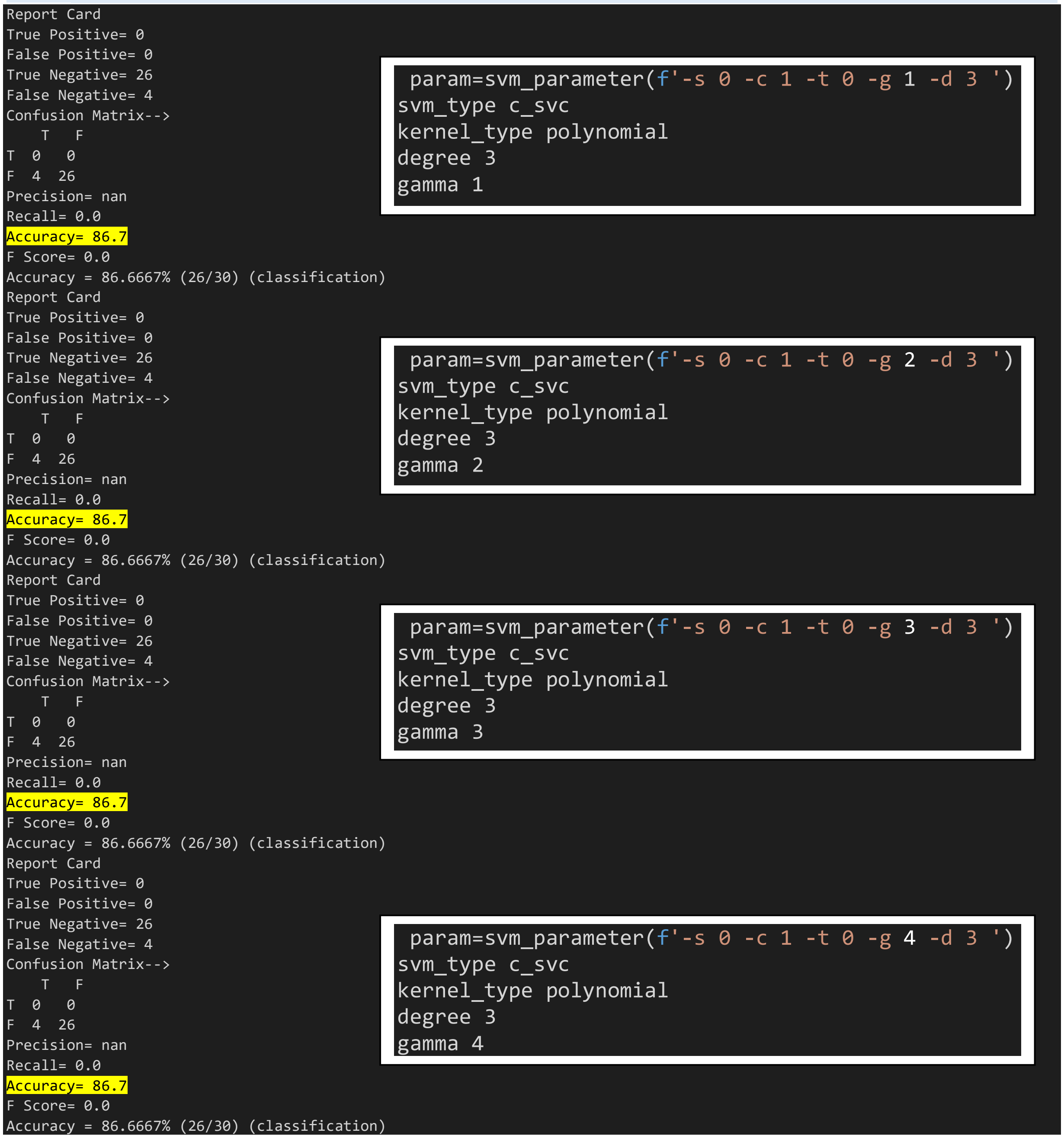
Varying the kernel parameter ***(g=np.linspace (1,15,num=15))***

margin same

degree same=3

Maximum Accuracy achieved is=86.7%

Changing parameter is not changing the results. (showing results below for few values only)



```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
Accuracy = 86.6667% (26/30) (classification)
```

```
 param=svm_parameter(f'-s 0 -c 1 -t 0 -g 1 -d 3 ')
svm_type c_svc
kernel_type polynomial
degree 3
gamma 1
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
Accuracy = 86.6667% (26/30) (classification)
```

```
 param=svm_parameter(f'-s 0 -c 1 -t 0 -g 2 -d 3 ')
svm_type c_svc
kernel_type polynomial
degree 3
gamma 2
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
Accuracy = 86.6667% (26/30) (classification)
```

```
 param=svm_parameter(f'-s 0 -c 1 -t 0 -g 3 -d 3 ')
svm_type c_svc
kernel_type polynomial
degree 3
gamma 3
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
Accuracy = 86.6667% (26/30) (classification)
```

```
 param=svm_parameter(f'-s 0 -c 1 -t 0 -g 4 -d 3 ')
svm_type c_svc
kernel_type polynomial
degree 3
gamma 4
```

Same kernel type

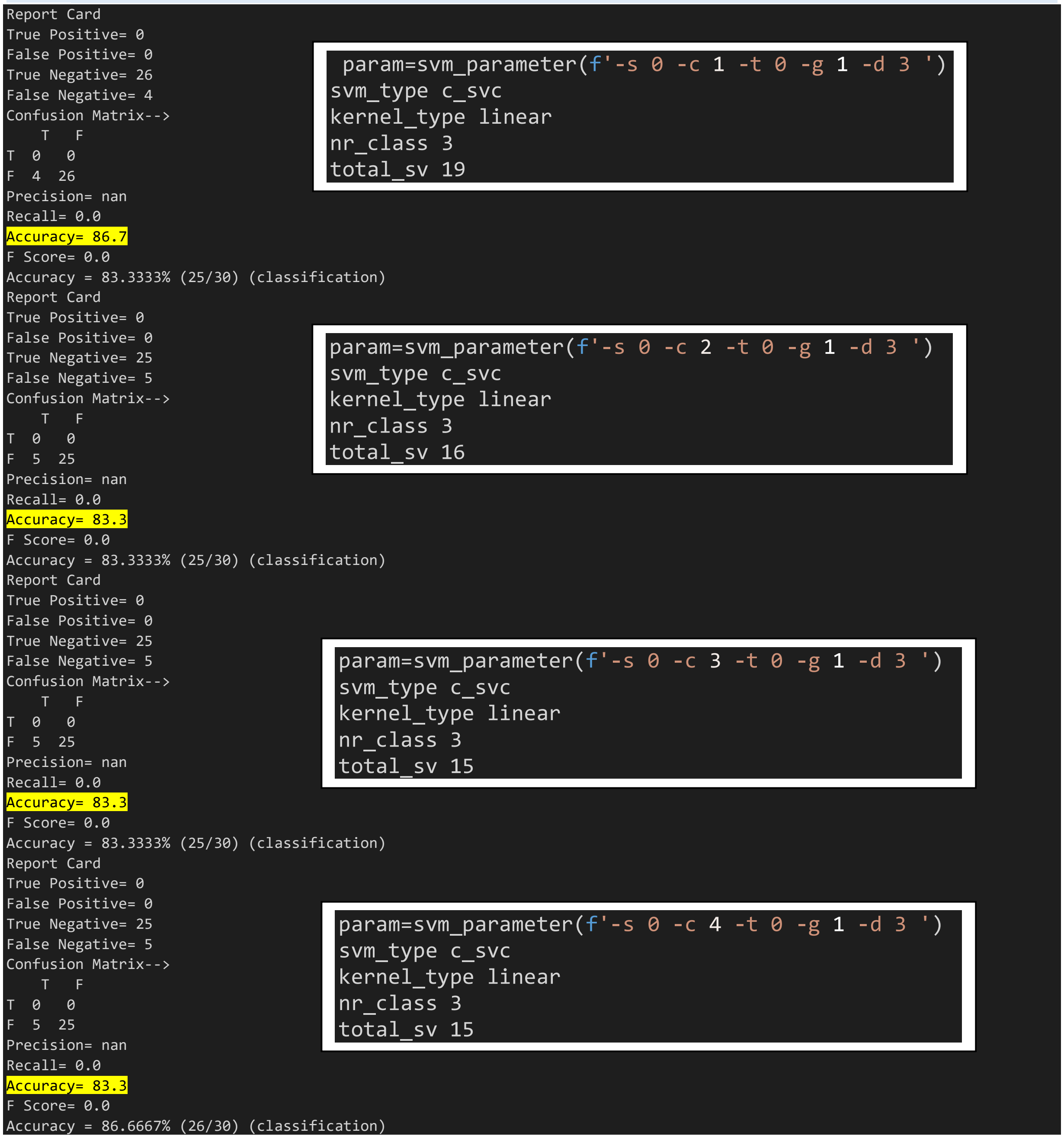
Same kernel parameter

*Varying the margin (**c=np.linspace(1,15,num=15)**)*

degree same=3

Maximum Accuracy achieved is=86.7%

Changing parameter is changing the results. (showing results below for few values only)



```
param=svm_parameter(f'-s 0 -c 1 -t 0 -g 1 -d 3 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 19
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
Accuracy = 83.3333% (25/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 2 -t 0 -g 1 -d 3 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 16
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
Accuracy = 83.3333% (25/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 3 -t 0 -g 1 -d 3 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 15
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
Accuracy = 83.3333% (25/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 4 -t 0 -g 1 -d 3 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 15
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
Accuracy = 86.6667% (26/30) (classification)
```

Same kernel type

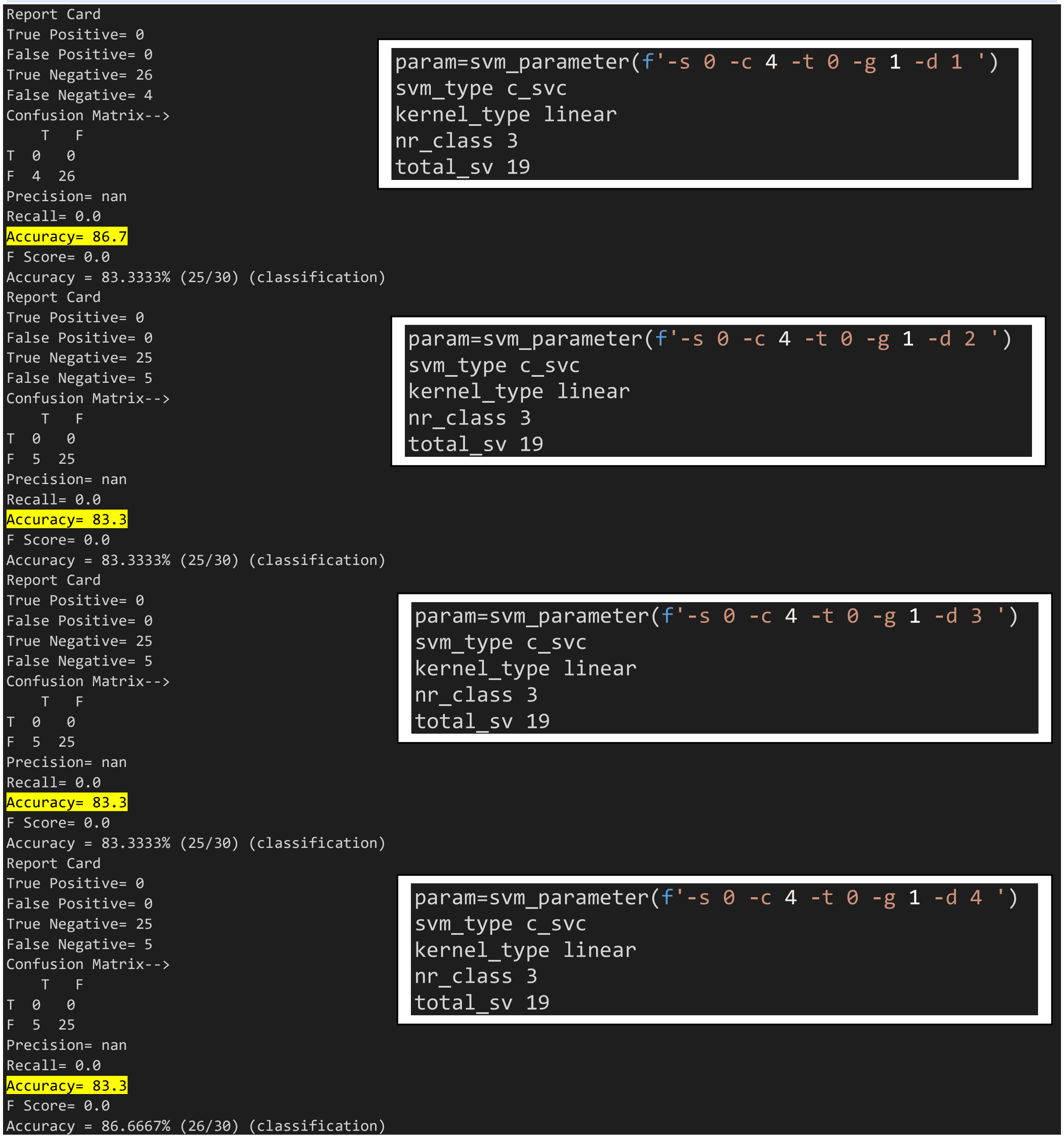
Same kernel parameter

Same margin

degree changing from 1 to 10

Maximum Accuracy achieved is=86.7%

Changing parameter is changing the results. (Showing results below for few values only)



```
Report Card
True Positive= 0
False Positive= 0
True Negative= 26
False Negative= 4
Confusion Matrix-->
    T   F
T  0   0
F  4  26
Precision= nan
Recall= 0.0
Accuracy= 86.7
F Score= 0.0
Accuracy = 83.3333% (25/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 4 -t 0 -g 1 -d 1 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 19
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
Accuracy = 83.3333% (25/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 4 -t 0 -g 1 -d 2 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 19
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
Accuracy = 83.3333% (25/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 4 -t 0 -g 1 -d 3 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 19
```

```
Report Card
True Positive= 0
False Positive= 0
True Negative= 25
False Negative= 5
Confusion Matrix-->
    T   F
T  0   0
F  5  25
Precision= nan
Recall= 0.0
Accuracy= 83.3
F Score= 0.0
Accuracy = 86.6667% (26/30) (classification)
```

```
param=svm_parameter(f'-s 0 -c 4 -t 0 -g 1 -d 4 ')
svm_type c_svc
kernel_type linear
nr_class 3
total_sv 19
```

5. Conclusion-

- *Changing degree from 1 to 10 does not changing the prediction accuracy.*
- *Changing the margin also not affecting the accuracy.*
- *Changing the kernel parameter for polynomial is not affecting the accuracy of the prediction.*
- *Varying the Kernel type changing the accuracy even from range of 0 to 86.7 which is interesting to see. Which is obvious also as it will affect the boundary type.*

NON-AIP 701 PART 1_2(BINARY CLASSIFICATION)

6. Problem statements-

A. **Choose just 2 out of the 10 classes in your data, and train an SVM**
B. **Leave some data aside for validation, or ideally, use cross-validation.**
C. **Study the effects of changing the different hyperparameter values, including the type of kernel function being used.**
D. **How do they affect the accuracy**
E. **Can you distinguish cases of overfitting, underfitting, and good fitting?**

7. 2 classes(0 and 1)) and 25 features

Chosen only class 0 and class 1 for the training.

`[1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,…….]`
`600X1 array of classes`

Shuffling is used as the data is in sorted manner with respect to classes.

Cross validation used for the best model parameter selection.

```
for c in range(-3,10):
        for g in range(-8,10):
                for t in range(0,4):
                        for d in range(1,5):
```

where 4 parameter such as the C the penalty on the datapoints is varied in ranges between -3 to 10 gamma which is parameter for the kernel type which have varied from -8 to 10, t which is the **kernel type** which vary from 0 to 3 and d which is **degree** in the kernel is used as the parameters for the **cross validation**.

```
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

Results from the cross validation comes as shown below.

There is total **3744** searches done via **CV** which then gives the best and worst model as shown below.

Parameters[c,g,t,d]

best parameter [0.125, 0.00390625, 0, 1]

worst parameter [0.125, 0.00390625, 1, 2]

Testing on best parameter

Train accuracy for best parameter

Accuracy = 100% (480/480) (classification)

Test accuracy for best parameter

Accuracy = 99.1667% (119/120) (classification)

Testing on worst parameter

Train accuracy for worst parameter

Accuracy = 50.2083% (241/480) (classification)

Test accuracy for worst parameter

Accuracy = 49.1667% (59/120) (classification)

Time to Train(sec)= 142.6

We can clear see that the best parameter not even the best they are generalized well also.

Kernel type comes to be linear for the best and **generalized model,** which uses **degree 1.**

```
[1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0,……………]
```

The worst model is not even worse it also underfitting the model and only predicts 0 whatever the input is 1 this is using **polynomial** as the kernel type and **degree 2.**

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,..0]
```

There are cases of overfitting also are there which are not totally overfitting but towards it. Accuracy for these models is as below.

Train accuracy

Accuracy = 95.2083% (457/480) (classification)

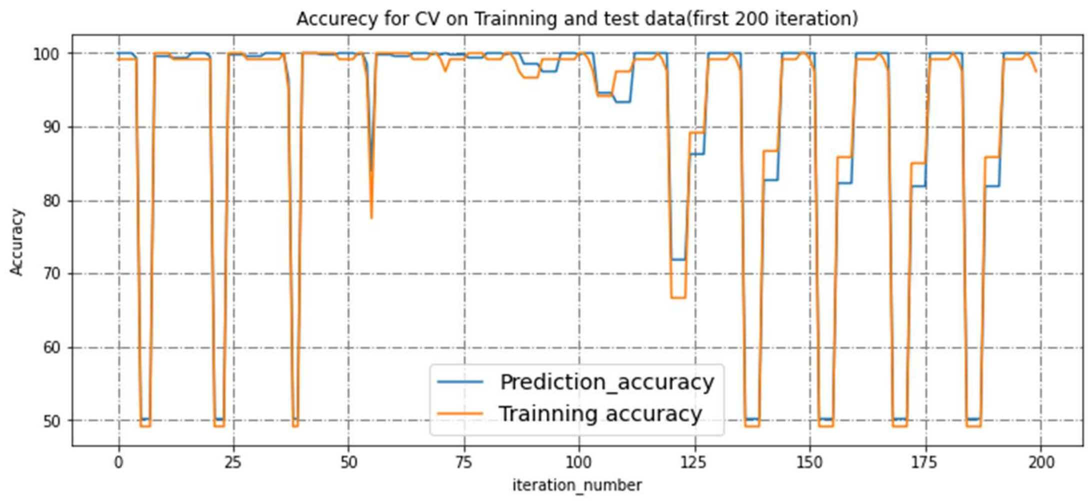
Test accuracy

Accuracy = 90% (108/120) (classification)

8. Conclusion

Changing the parameters are affecting the accuracy a lot. Few iterations are as below.

We can say that higher order Kernels are not working well, this dataset can be classified only using the **linear SVM with degree 1**. Kernel type is affecting the accuracy a lot. There are a number of combinations are available where accuracy can be achieved maximum.



```
C=-3,gamma=-8,kernaltype=0,degree=1
Accuracy = 100% (480/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=0,degree=2
Accuracy = 100% (480/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=0,degree=3
Accuracy = 100% (480/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=0,degree=4
Accuracy = 100% (480/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=1,degree=1
Accuracy = 99.375% (477/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=1,degree=2
Accuracy = 50.2083% (241/480) (classification)
Accuracy = 49.1667% (59/120) (classification)
Test accuracy= (49.166666666666664, 0.5083333333333333, nan)
C=-3,gamma=-8,kernaltype=1,degree=3
```

9. Selecting 2 classes (0 and 1) and 10 features

F. Also try using only the first 10 features, instead of all 25, and compare the results in the two cases

Chosen only class 0 and class 1 for the training as before but this time we will be having only 10 features for training the SVM.

`[1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,…….]`

`600X1 array of classes`

`Input matrix will be a 600X10 matrix.`

Shuffling is used as the data is in sorted manner with respect to classes.

Cross validation used for the best model parameter selection.

```
for c in range(-3,10):
        for g in range(-8,10):
                for t in range(0,4):
                        for d in range(1,5):
```

where 4 parameter such as the C the penalty on the datapoints is varied in ranges between -3 to 10 gamma which is parameter for the kernel type which have varied from -8 to 10, t which is the **kernel type** which vary from 0 to 3 and d which is **degree** in the kernel is used as the parameters for the **cross validation**.

```
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

Results from the cross validation comes as shown below.

There is total **3744** searches done via **CV** which then gives the best and worst model as shown below.

Parameters[c,g,t,d]

best parameter [0.25, 0.125, 2, 1]

worst parameter [0.125, 0.00390625, 1, 2]

Testing on best parameter

Train accuracy for best parameter

Accuracy = 100% (480/480) (classification)

Test accuracy for best parameter

Accuracy = 100% (120/120) (classification)

Testing on worst parameter

Train accuracy for worst parameter

Accuracy = 49.5833% (238/480) (classification)

Test accuracy for worst parameter

Accuracy = 51.6667% (62/120) (classification)

Time to Train(sec)= 99.31

We can clear see that the best parameter not even the best they are generalized well also.

Kernel type comes to be radial basis function for the best and **generalized model,** which uses degree 1.

Radial basis function is not giving good result with more features in hand but working well with lower features.

The worst model is remain the same with less feature also, it also underfitting the model and only predicts 0 whatever the input is .this is using **polynomial** as the kernel type and degree 2.

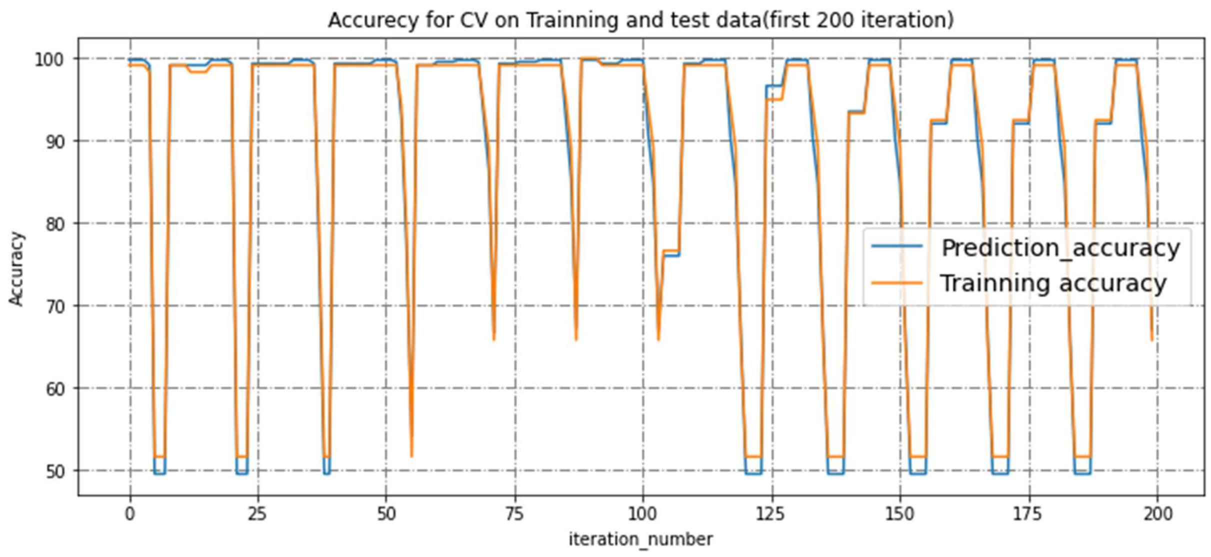
```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,..0]
```

There are cases of overfitting also are there which are not totally overfitting but towards it. Accuracy for these models is as below.

10. Conclusion

Changing the parameters are affecting the accuracy a lot. Few iterations are as below.

We can see that more features working well with linear SVM rather than polynomial or RBF but less features are working well with RBF. So, these are purely hyperparameter which we should be using CV to get the optimal accuracy.



After seeing above two graph we can say more features and less features can give comparable results with cross validation.

Although we see a **fall** in accuracy after 75 iterations in less feature model.

The time taken by less feature is less compared to more feature model which is obviously intuitive.

```
C=-3,gamma=-8,kernaltype=0,degree=1
Accuracy = 99.7917% (479/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=0,degree=2
Accuracy = 99.7917% (479/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=0,degree=3
Accuracy = 99.7917% (479/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)
C=-3,gamma=-8,kernaltype=0,degree=4
Accuracy = 99.7917% (479/480) (classification)
Accuracy = 99.1667% (119/120) (classification)
```

Test accuracy= (99.16666666666667, 0.008333333333333333, 0.9671951886276654)

C=-3,gamma=-8,kernaltype=1,degree=1

Accuracy = 99.1667% (476/480) (classification)

Accuracy = 98.3333% (118/120) (classification)

Test accuracy= (98.33333333333333, 0.016666666666666666, 0.9354838709677419)

C=-3,gamma=-8,kernaltype=1,degree=2

Accuracy = 49.5833% (238/480) (classification)

G. Also try using only the first 10 features, instead of all 25, and compare the results in the two cases. Repeat this exercise for at least two more pairs of classes out of the 10 given to you. Do you consistently get the best results for the same hyperparameter settings, or does it vary a lot depending on which pair of classes you're looking at?

11. Selecting 25 features and 2 classes (2 and 3))

The computation time for this class was very high around 13 min which was 2 min for the case of 25 features and 0 and 1 class. So I have reduced the parameter span of c from -3 to 10 to -3 to 5 and gamma span from -8 to 10 to -5 to 5 as shown below.

```
for c in range(-3,5):
        for g in range(-5,5):
                for t in range(0,4):
                        for d in range(1,5):
```

best parameter [0.125, 0.125, 1, 2] Different best parameter from 0 and 1 class with 25 features.

worst parameter [0.125, 0.25, 2, 1] Different best parameter from 0 and 1 class with 25 features.

Testing on best parameter

Train accuracy for best parameter

Accuracy = 100% (480/480) (classification)

Test accuracy for best parameter

Accuracy = 99.1667% (119/120) (classification)

Testing on worst parameter

Train accuracy for worst parameter

Accuracy = 51.4583% (247/480) (classification)

Test accuracy for worst parameter

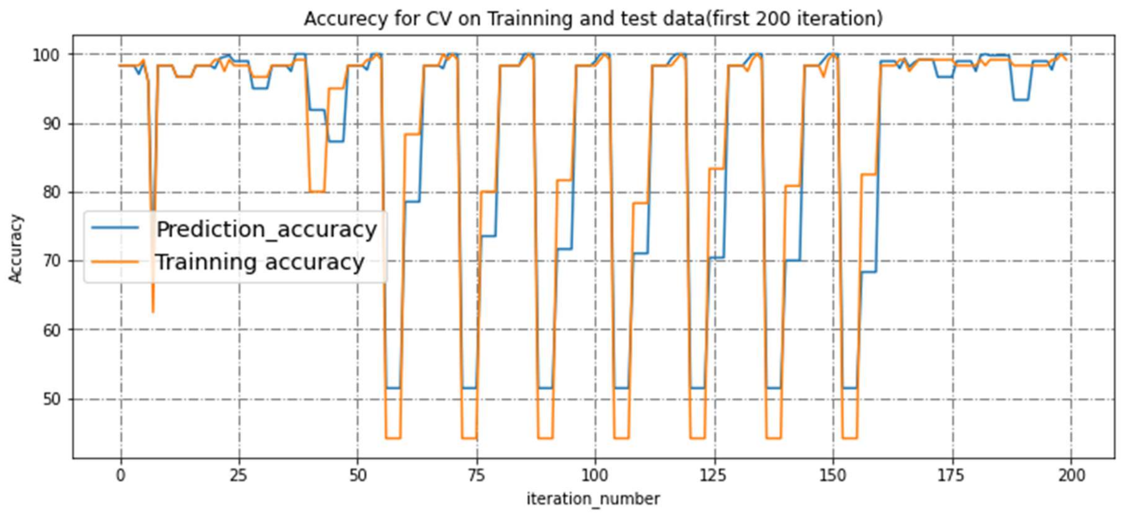
Accuracy = 44.1667% (53/120) (classification)

Time to Train(sec)= 54.27 After reducing the parameters span. Else if we train on same parameter of O and 1 class it took 13 min to train.

12. Conclusion

So, we conclude that same hyperparameter setting is giving **different result for the classes**.

So yes, it is depending on the pair of classes we are looking at.



13. Selecting 10 features and 2 classes (4 and 5)

This class got less time to train which is 3 min even on more parameters.

```
for c in range(-3,10):
        for g in range(-8,10):
                for t in range(0,4):
                        for d in range(1,5):
```

best parameter [0.125, 1.0, 1, 3] Different best parameter from above cases.

worst parameter [0.125, 0.00390625, 1, 2] Different best parameter from above cases

Testing on best parameter

Train accuracy for best parameter

Accuracy = 100% (480/480) (classification)

Test accuracy for best parameter

Accuracy = 96.6667% (116/120) (classification)

Testing on worst parameter

Train accuracy for worst parameter

Accuracy = 51.0417% (245/480) (classification)

Test accuracy for worst parameter

Accuracy = 45.8333% (55/120) (classification)

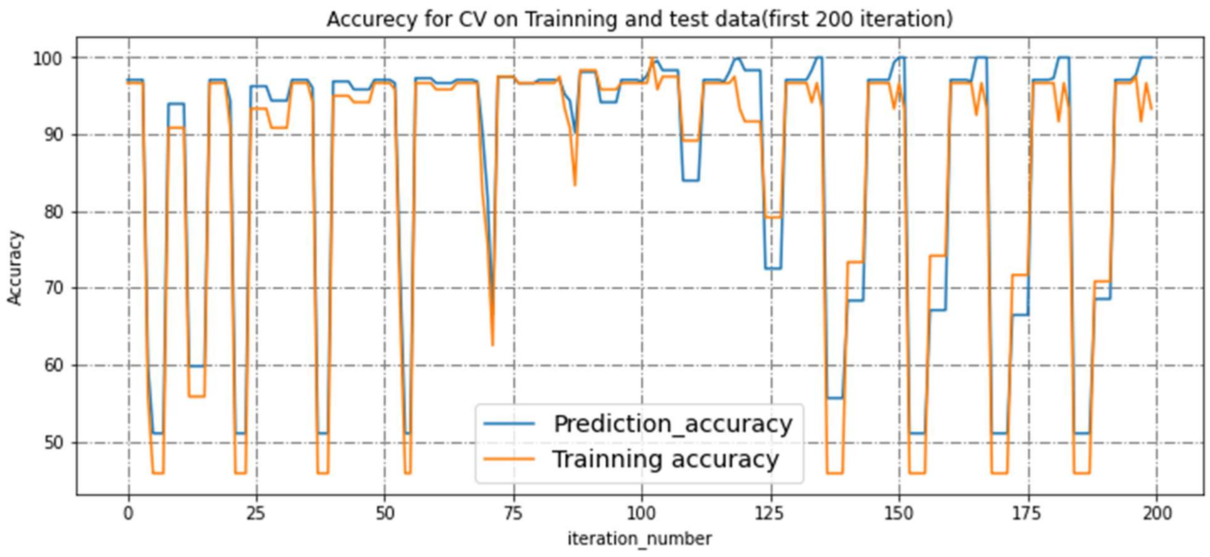
Time to Train(sec)= 202

14. Conclusion

Here also again the training time is less but we can see best parameters are giving sort of overfitting.

Worst parameters are almost same. But it is interesting to note that we have more parameter setting having very low accuracy. So, we conclude that same hyperparameter setting is giving **different result for the classes**.

So yes, it is depending on the pair of classes we are looking at. After 125 iteration we can find many overfitting cases.



H. Try visualising your data using just the first 2 features. Does this help you get some sense of which pairs of classes are harder to distinguish between, or which specific data points tend to be misclassified?

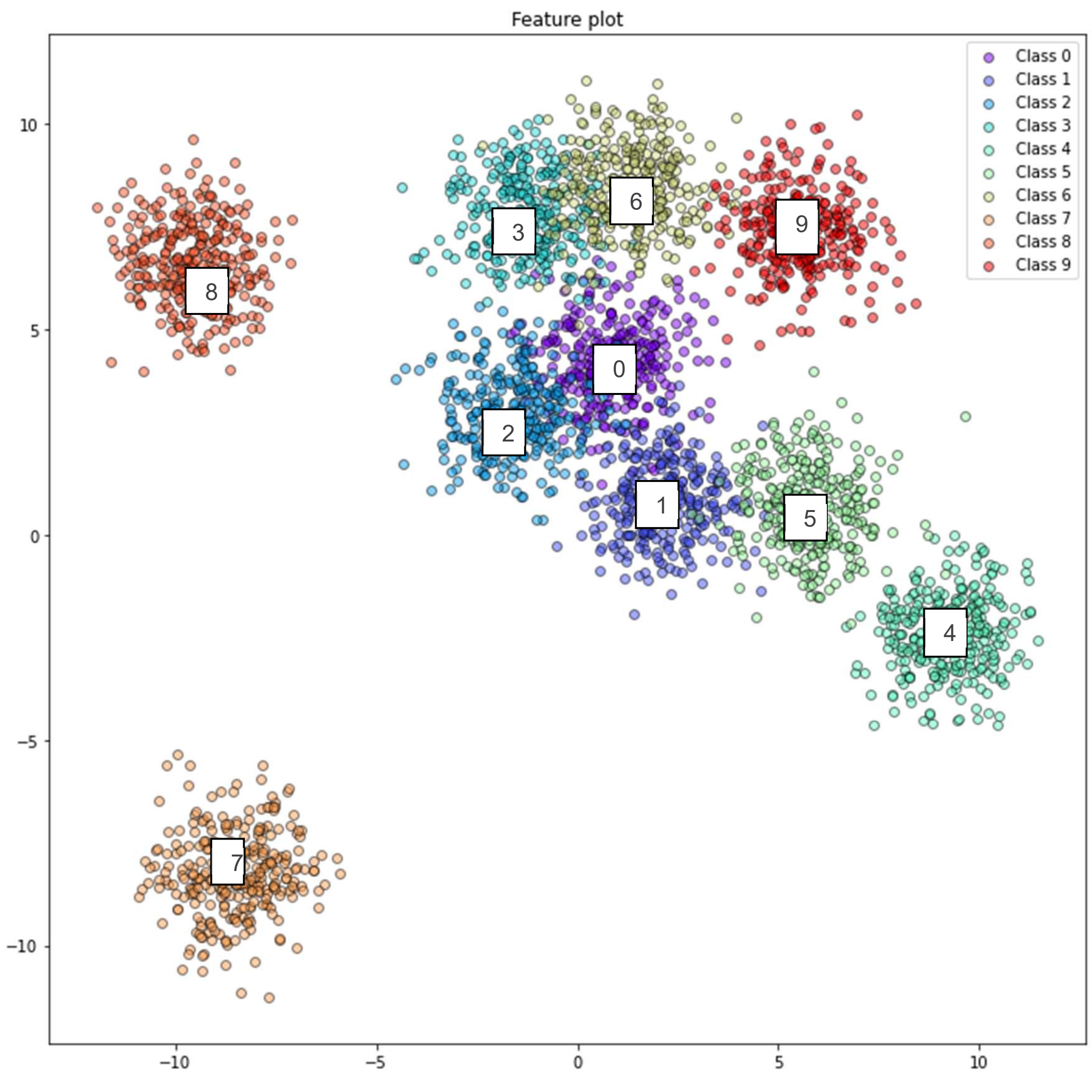
15. Visualizing the data(first 2 features and all classes)

By seeing the 3Dimension visualisation into 2D we can see why our results are now seems intuitive.

as 0 and 1 ,2 and 3 ,4 and 5 among these pair 0 and 1 pair is showing minimum distinguishment as shown here also they are very close to each other.

The it is harder to distinguish among **0,1,2,3,5,9,5** as they are having very close boundaries.

So these points are very prone to misclassification.



NON-AIP 701 PART 1_3(MULTICLASS CLASSIFICATION)

A. Now we would like to train a classifier for all 10 classes Figure out what method(s) your chosen library uses for multiclass classification.

16. Multiclass classification model(25 features and 10 classes)

Model trained using LIBSVM library. Which initial parameter chosen as follows param=svm_parameter('-s 0 -c 1 -t 1 -g 3 -d 2 ') which means as

```
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

-d is degree which taken to be 2.

-c are the penalty which taken to be 1.

-g is gamma which is parameter for the polynomial which chosen as 3

After training on this multiclass classification model accuracy came as 90.83 which can be improved using the cross-validation training. Accuracy = 90.8333% (545/600) (classification)

Confusion matrix are as below. Which shows that 8 is also having mis classification which should not be happen if we go with two feature conclusions above.

Predicted

Actual

	0	1	2	3	4	5	6	7	8	9
0	60	0	0	0	0	0	2	0	0	0
1	0	67	0	0	0	0	0	0	2	0
2	1	0	49	1	1	0	0	1	3	0
3	1	1	0	57	1	0	0	0	2	4
4	0	0	1	0	54	1	0	1	3	1
5	0	1	0	0	1	56	1	0	4	1
6	0	0	0	0	1	1	53	0	0	0
7	0	0	0	0	1	0	0	60	1	3
8	1	2	1	1	1	2	1	0	37	1
9	0	0	1	0	1	1	0	0	0	52

Classification report for above model is as below.

```
 '0.0':
 'precision': 0.9523809523809523,
 'recall': 0.967741935483871,
 'f1-score': 0.96,
 'support': 62},

'1.0':
'precision': 0.9436619718309859,
 'recall': 0.9710144927536232,
 'f1-score': 0.9571428571428571,
 'support': 69},

'2.0':
'precision': 0.9423076923076923,
 'recall': 0.875,
 'f1-score': 0.9074074074074073,
 'support': 56},

'3.0':
'precision': 0.9661016949152542,
 'recall': 0.8636363636363636,
 'f1-score': 0.912,
 'support': 66
'4.0':
 'precision': 0.8852459016393442,
 'recall': 0.8852459016393442,
 'f1-score': 0.8852459016393442,
 'support': 61},

'5.0':
'precision': 0.9180327868852459,
 'recall': 0.875,
 'f1-score': 0.8959999999999999,
 'support': 64
'6.0':
'precision': 0.9298245614035088,
 'recall': 0.9636363636363636,
 'f1-score': 0.9464285714285715,
 'support': 55
'7.0':
'precision': 0.967741935483871,
 'recall': 0.9230769230769231,
 'f1-score': 0.9448818897637796,
 'support': 65
'8.0':
'precision': 0.7115384615384616,
 'recall': 0.7872340425531915,
 'f1-score': 0.7474747474747475,
 'support': 47
'9.0':
{'precision': 0.8387096774193549,
 'recall': 0.9454545454545454,
 'f1-score': 0.8888888888888888,
 'support': 55
```

Only two features are not enough to distinguish the whole classes. We need more to get good accuracy.

B. build a classifier for all the classes and evaluate using validation or cross-validation. Again, study the effects of changing the various hyperparameters and kernel function. Try and finetune them to obtain the best possible performance.

17. Cross validation on multi class classification(25 features and 10 classes)

Validation parameter grid has taken same as before. This grid however was very fast for less feature and few classes(i.e., 2). But for more classes and more feature into account this takes almost 230 min time. So, we can decrease the window based on prior knowledge.

```
for c in range(-3,10):
        for g in range(-8,10):
                for t in range(0,4):
                        for d in range(1,5):
```

CV Results

best parameter [0.125, 0.25, 1, 3]

worst parameter [0.125, 0.00390625, 1, 2]

Testing on best parameter

Train accuracy for best parameter(Overfitting case)

Accuracy = 100% (2400/2400) (classification)

Test accuracy for best parameter

Accuracy = 93.5% (561/600) (classification)

Testing on worst parameter

Train accuracy for worst parameter

Accuracy = 10.5417% (253/2400) (classification)

Test accuracy for worst parameter

Accuracy = 7.83333% (47/600) (classification)

Time to Train(min)= 230

Binary Classification setting with 25 features and 2 classes are

best parameter [0.125, 0.00390625, 0, 1]   (Linear with degree 1)

worst parameter [0.125, 0.00390625, 1, 2]   (Polynomial with degree 2)

Multiclass Classification setting with 25 features and 10 classes are

best parameter [0.125, 0.25, 1, 3]   (Polynomial with degree 3)

worst parameter [0.125, 0.00390625, 1, 2] (Polynomial with degree 2)

Few Iterations for multiclass classification

C=-3,gamma=-8,kernaltype=0,degree=1

Accuracy = 92.5833% (2222/2400) (classification)

Accuracy = 90% (540/600) (classification)

Test accuracy= (90.0, 1.4083333333333334, 0.8355743254221609)

C=-3,gamma=-8,kernaltype=0,degree=2

Accuracy = 92.5833% (2222/2400) (classification)

Accuracy = 90% (540/600) (classification)

Test accuracy= (90.0, 1.4083333333333334, 0.8355743254221609)

C=-3,gamma=-8,kernaltype=0,degree=3

Accuracy = 92.5833% (2222/2400) (classification)

Accuracy = 90% (540/600) (classification)

Test accuracy= (90.0, 1.4083333333333334, 0.8355743254221609)

C=-3,gamma=-8,kernaltype=0,degree=4

Accuracy = 92.5833% (2222/2400) (classification) (Generalised)

Accuracy = 90% (540/600) (classification)

Test accuracy= (90.0, 1.4083333333333334, 0.8355743254221609)

C=-3,gamma=-8,kernaltype=1,degree=1

Accuracy = 70.7083% (1697/2400) (classification)

Accuracy = 70.1667% (421/600) (classification)

Test accuracy= (70.16666666666667, 5.243333333333333, 0.5893366684394232)

C=-3,gamma=-8,kernaltype=1,degree=2

Accuracy = 10.5417% (253/2400) (classification)

Accuracy = 7.83333% (47/600) (classification)

Test accuracy= (7.833333333333334, 21.511666666666667, nan)

C=-3,gamma=-8,kernaltype=1,degree=3 (Under fitting)

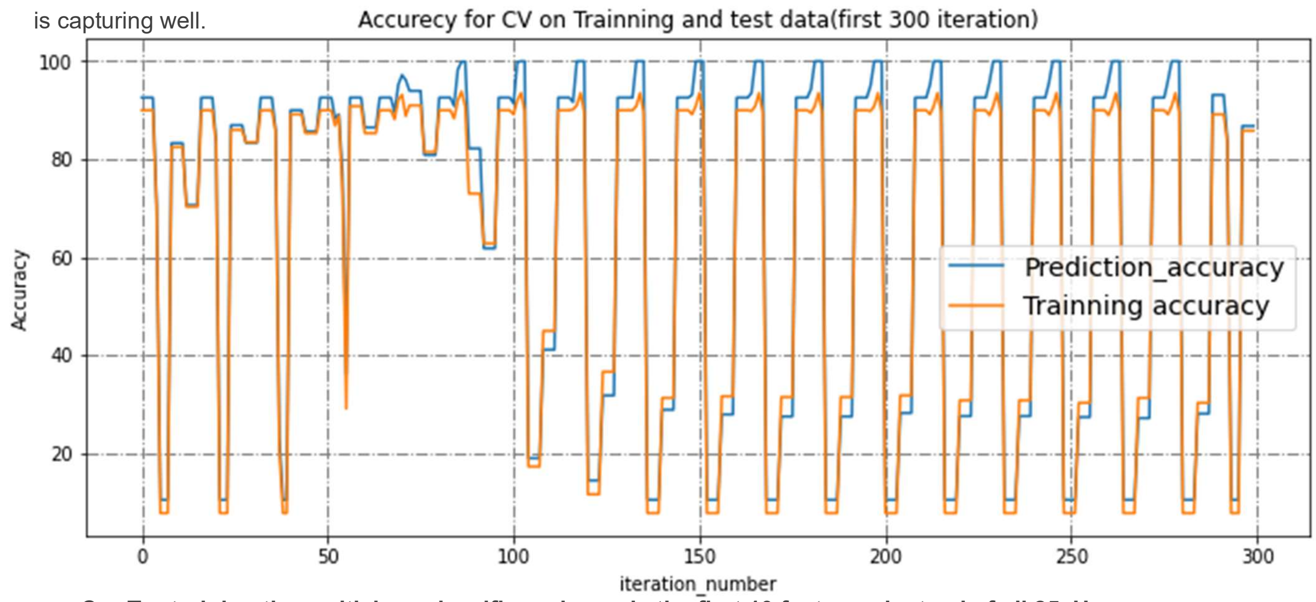
Accuracy = 10.5417% (253/2400) (classification)

Accuracy = 7.83333% (47/600) (classification)

Test accuracy= (7.833333333333334, 21.511666666666667, nan)

18. Conclusion-

There is major difference we can observe as best model is having linear model for binary classification which is very intuitive because of linear boundary may occur between two classes, but as the class goes to 10 it is not very likely a linear SVM will be able to distinguish the classes. Also, below graph for iteration shows that it is not possible to get 100 percent accuracy till the 300 iteration of CV which was possible for the binary classification even under 200 iterations. One interesting observation is that training accuracy is less as compared to testing accuracy in few cases that is due to the simple data set with less number of data points also our model is complex for those iterations which is capturing well.



C. **Try training the multiclass classifier using only the first 10 features, instead of all 25. How does this affect your results? What does this tell you about the usefulness of the different features?**

19. First 10 features and 10 classes(Multiclass svm)

Validation parameter grid has taken same as before. This grid however was very fast for less feature and few classes(i.e., 2). But for more classes and more feature into account this takes almost 260 min time. So, we can decrease the window based on prior knowledge. The accuracy for the initial parameter in case of full feature was 90 percent which in this model which is on lesser features(10) we got around 81 percent which seems to be very intuitive as it is harder with the lesser feature ,as we have already seen with two feature classification.

Accuracy = 81.3333% (488/600) (classification)

The confusion matrix for this model is having more misclassification than the 25-feature model .

Miss classification of 9 is found to be highest which if we see the 2-feature graph seems pretty surprising as 9 class is very well segregated in that.

Actual \ Predict	0	1	2	3	4	5	6	7	8	9
0	62	0	1	2	0	2	1	0	0	0
1	0	55	1	0	0	0	0	0	1	0
2	0	0	47	3	1	7	1	1	1	3
3	1	0	2	35	4	3	1	1	5	2
4	0	0	0	1	47	1	0	2	2	2
5	0	2	3	1	3	54	1	0	5	1
6	0	0	1	2	0	1	49	0	0	0
7	2	0	1	0	1	1	0	54	1	2
8	1	2	3	2	1	3	0	0	44	4
9	0	0	2	0	11	2	0	0	1	41

20. Cross validation on multi class classification(25 features and 10 classes)

Parameter grid used is same, but it took less time as 120 min due to less features.

```
for c in range(-3,10):
        for g in range(-8,10):
                for t in range(0,4):
                        for d in range(1,5):
```

CV Results

best parameter [0.125, 1.0, 1, 4] (polynomial with degree 1)

worst parameter [0.125, 0.00390625, 1, 2] (polynomial with degree 2)

Testing on best parameter

Train accuracy for best parameter

Accuracy = 100% (2400/2400) (classification) (Overfitting)

Test accuracy for best parameter

Accuracy = 79.5% (477/600) (classification)

Testing on worst parameter

Train accuracy for worst parameter

Accuracy = 10.2917% (247/2400) (classification)(Underfitting)

Test accuracy for worst parameter

Accuracy = 8.83333% (53/600) (classification)

Time to Train(min)= 260

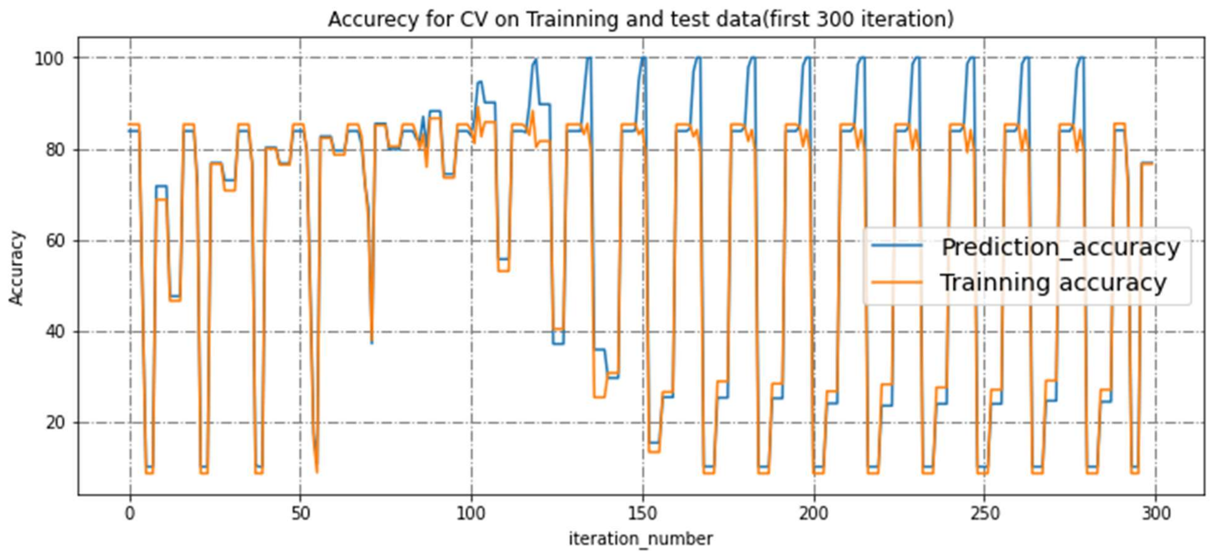
Few Iterations for multiclass classification(10 features and 10 classes)

```
C=-3,gamma=-8,kernaltype=0,degree=1

Accuracy = 83.875% (2013/2400) (classification)

Accuracy = 85.3333% (512/600) (classification)

Test accuracy= (85.33333333333334, 2.4833333333333334, 0.7185757253066388)

C=-3,gamma=-8,kernaltype=0,degree=2

Accuracy = 83.875% (2013/2400) (classification)

Accuracy = 85.3333% (512/600) (classification)

Test accuracy= (85.33333333333334, 2.4833333333333334, 0.7185757253066388)

C=-3,gamma=-8,kernaltype=0,degree=3

Accuracy = 83.875% (2013/2400) (classification)

Accuracy = 85.3333% (512/600) (classification)

Test accuracy= (85.33333333333334, 2.4833333333333334, 0.7185757253066388)

C=-3,gamma=-8,kernaltype=0,degree=4

Accuracy = 83.875% (2013/2400) (classification)

Accuracy = 85.3333% (512/600) (classification)

Test accuracy= (85.33333333333334, 2.4833333333333334, 0.7185757253066388)

C=-3,gamma=-8,kernaltype=1,degree=1

Accuracy = 47.6667% (1144/2400) (classification)

Accuracy = 46.6667% (280/600) (classification)

Test accuracy= (46.666666666666664, 10.25, 0.14888104150491555)

C=-3,gamma=-8,kernaltype=1,degree=2

Accuracy = 10.2917% (247/2400) (classification)

Accuracy = 8.83333% (53/600) (classification)

Test accuracy= (8.833333333333334, 10.813333333333333, nan)

C=-3,gamma=-8,kernaltype=1,degree=3

Accuracy = 10.2917% (247/2400) (classification)

Accuracy = 8.83333% (53/600) (classification)

Test accuracy= (8.833333333333334, 10.813333333333333, nan)

C=-3,gamma=-8,kernaltype=1,degree=4
```

21. Conclusion-

So, we can see that other features are how much important as we can see we have increased misclassification from our initial models from 25 to 122.Cross validation will take lesser time but that is very intuitive as we need less computation due to only 10 features. So, less feature increases the chances for the misclassification however we can achieve more accuracy by cross validation. So, we can see hyperparameter are different than what we get in 25 feature model also one more interesting this it got overfit for the best model.

The graph shows that till 300 iteration accuracy is till 80% only which shows less data affecting to get high accuracy.



Other features are helping us in not overfitting and other feature help us computationally effective which is surprising.

NON-AIP 701 PART 2(LEADERBOARD ASSIGNMENT)

The data is extended version of previous data . Above we have used 2400 data for training and 600 data for testing. Here as we have 8000 data points so I will take 5600 data for training and 2400 data for testing.

Initial parameter setting was giving the test accuracy to be 93 percent, with help of cross validation we can increase his accuracy. Accuracy is also depending upon the training and testing fraction,0.9 fraction working very well.

```
param=svm_parameter('-s 0 -c 1 -t 1 -g 3  -d 2 ')
```

Accuracy = 93.9583% (2255/2400) (classification)

The confusion matrix for initial model shows it is harder to classify the 9,3 and 8 classes

					Predict						
		0	1	2	3	4	5	6	7	8	9
	0	236	1	2	2	0	0	0	0	1	1
	1	0	249	1	0	0	0	0	2	0	0
	2	1	0	235	2	0	1	0	5	2	3
	3	0	0	5	223	2	3	2	3	3	3
Actual	4	0	1	2	0	224	0	0	0	2	8
	5	2	1	2	3	1	189	3	1	5	2
	6	4	0	0	1	3	3	215	0	0	0
	7	0	2	4	0	3	2	0	231	0	4
	8	2	1	4	8	0	2	1	1	216	1
	9	0	0	7	4	8	1	0	0	1	237

Classification report shows now the vector needed is maximum goes to 269 which was in previous case limited to 69.

```
 '0':
'precision': 0.963265306122449,
  'recall': 0.9711934156378601,
  'f1-score': 0.9672131147540984,
  'support': 243},

'1.0':
{'precision': 0.9764705882352941,
  'recall': 0.9880952380952381,
  'f1-score': 0.9822485207100591,
  'support': 252},
'2.0'
'precision': 0.8969465648854962,
  'recall': 0.9437751004016064,
  'f1-score': 0.9197651663405088,
  'support': 249
 '3.0'
'precision': 0.9176954732510288,
  'recall': 0.9139344262295082,
  'f1-score': 0.9158110882956878,
  'support': 244
 '4.0'
 precision': 0.9294605809128631,
  'recall': 0.9451476793248945,
  'f1-score': 0.9372384937238494,
  'support': 237,
 '5.0':
'precision': 0.9402985074626866,
  'recall': 0.9043062200956937,
  'f1-score': 0.9219512195121952,
  'support': 209},
'6.0
'precision': 0.9728506787330317,
  'recall': 0.9513274336283186,
  'f1-score': 0.9619686800894854,
  'support': 226},
 '7.0'
 'precision': 0.9506172839506173,
  'recall': 0.9390243902439024,
  'f1-score': 0.9447852760736196,
  'support': 246},
'8.0'
'precision': 0.9391304347826087,
  'recall': 0.9152542372881356,
  'f1-score': 0.9270386266094419,
  'support': 236},
 '9.0'
precision': 0.915057915057915,
  'recall': 0.9186046511627907,
  'f1-score': 0.9168278529980656,
  'support': 258},

 'accuracy': 0.9395833333333333,
 'Macro avg':
'precision': 0.9401793333393991,
  'recall': 0.9390662792107948,
  'f1-score': 0.9394848039107011,
```

We will test for these parameters setting and best and worst parameter setting got after the cross validation.

22. Cross validation

Cross validation parameter window we have changes to shorter range as it takes a lot of time also as above, we have seen negative value of c and g are not playing any role for better result, so we have converted the window for those 1 to 5 now.

best parameter [0.5, 0.5, 1, 2, 100.0]

worst parameter [0.5, 0.5, 1, 0, 11.25]

Testing on best parameter

Train accurecy for best parameter

Accuracy = 100% (6400/6400) (classification)

Test accurecy for best parameter

Accuracy = 95.0625% (1521/1600) (classification)

Testing on worst parameter

Train accurecy for worst parameter

Accuracy = 11.25% (720/6400) (classification)

Test accurecy for worst parameter

Accuracy = 10.75% (172/1600) (classification)

Time to Train(min)= 30

We are showing some combination for overfitting as below.

C=1,gamma=2,kernaltype=2,degree=2

Accuracy = 100% (5600/5600) (classification)

Accuracy = 17.4583% (419/2400) (classification)

Test accuracy= (17.458333333333336, 12.22625, 0.11105010414008007)

C=1,gamma=2,kernaltype=2,degree=3

Accuracy = 100% (5600/5600) (classification)

Accuracy = 17.4583% (419/2400) (classification)

Test accuracy= (17.458333333333336, 12.22625, 0.11105010414008007)

C=1,gamma=2,kernaltype=2,degree=4

Accuracy = 100% (5600/5600) (classification)

Accuracy = 17.4583% (419/2400) (classification)

Test accuracy= (17.458333333333336, 12.22625, 0.11105010414008007)

C=1,gamma=2,kernaltype=2,degree=5

Accuracy = 100% (5600/5600) (classification)

Accuracy = 17.4583% (419/2400) (classification)

Test accuracy= (17.458333333333336, 12.22625, 0.11105010414008007)

C=1,gamma=2,kernaltype=2,degree=6

Accuracy = 100% (5600/5600) (classification)

Accuracy = 17.4583% (419/2400) (classification)

23. Best parameter results

Parameters[c,g,t,d]

best parameter [0.5, 0.5, 1, 2]

Confusion matrix

						Predict					
		0	1	2	3	4	5	6	7	8	9
	0	236	1	2	2	0	0	0	0	1	1
	1	0	249	1	0	0	0	0	2	0	0
	2	1	0	235	2	0	1	0	5	2	3
	3	0	0	5	223	2	3	2	3	3	3
Actual	4	0	1	2	0	224	0	0	0	2	8
	5	2	1	2	3	1	189	3	1	5	2
	6	4	0	0	1	3	3	215	0	0	0
	7	0	2	4	0	3	2	0	231	0	4
	8	2	1	4	8	0	2	1	1	216	1
	9	0	0	7	4	8	1	0	0	2	236

After 8 submissions on the KAGGLE competition my scores are as below.

Without CV = 96.0

With CV =97.833

24. References

- python - How to extract only datein ddmmyy format from a timestamp - Stack Overflow
- How to Concatenate Two Pandas DataFrames (With Examples) - Statology
- sklearn.metrics.r2_score — scikit-learn 1.1.2 documentation
- https://www.youtube.com/watch?v=eXjWSU1G14A   ## CV
  https://www.youtube.com/watch?v=3F0ePcwm_YU   ## LIBSVM
- https://docs.w3cub.com/scikit_learn/auto_examples/calibration/plot_calibration#sphx-glr-auto-examples-calibration-plot-calibration-py
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/