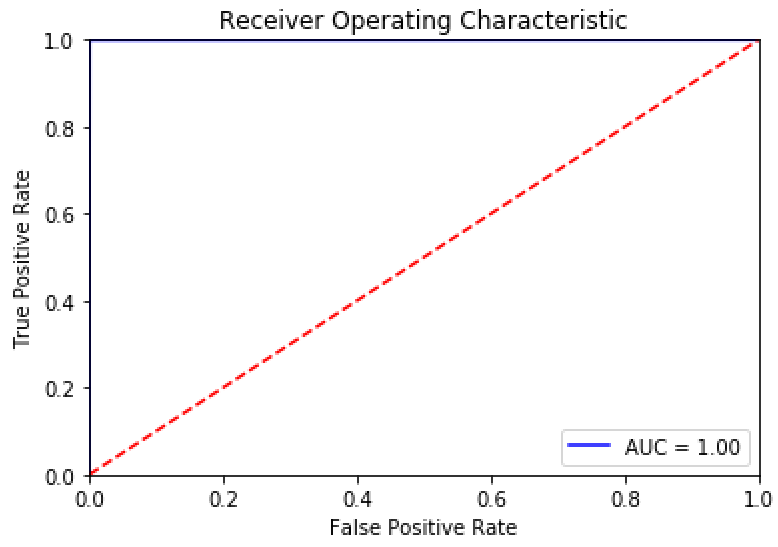
AML Assignment 1 Report

Question 0:

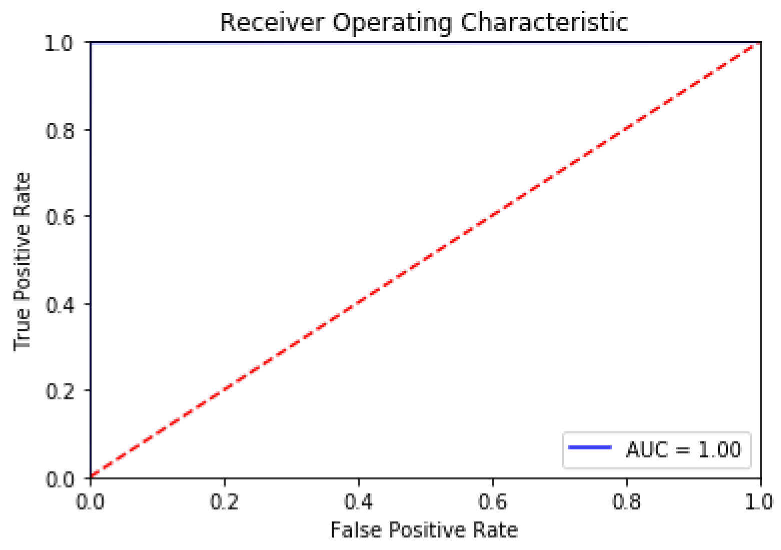
ACC on 5 fold cross validation:
[0.9986472776462632, 0.9983090970578289, 0.9976319350473613, 0.9986463620981387, 0.9993231810490694]

ROC:



ACC_SV: [0.9972945552925262, 0.9915454852891444, 0.9925575101488497, 0.9986463620981387, 0.9993231810490694]

ROC_SV:



Mean and Standard deviation of Accuracy for Both cases:

Mean (trained on whole Dataset) : 0.9985115705797323
Std (trained on whole dataset) : 0.0005495545308993313

Mean (trained on Support Vectors) : 0.9958734187755457
Std (trained on Support Vectors) : 0.0032042459861048736

Analysis:
Both were Almost Same as standard deviation was also very less in 5-fold cross validation. The reason can be that support vectors which were selected by classifier trained on whole training data, are also selected by classifier when it gets trained only on classifier which is getting trained on support vectors. The newly selected Support vectors are subset of bigger subset of support vectors. These are helping in deciding class for a sample and if both classifier are having same vectors then they are naturally going to have same decision for that sample. It can also be seen using accuracy as both of them are having nearly same acc on 5 fold cross validation. It can also be seen experimentally that in last iteration, both classifier were choosing same number of support vectors. It seems reasonable as these vectors which are mutual in both classifier are those which are lying on the boundary.

Q1.

Fig_1) → Accuracy: 0.65

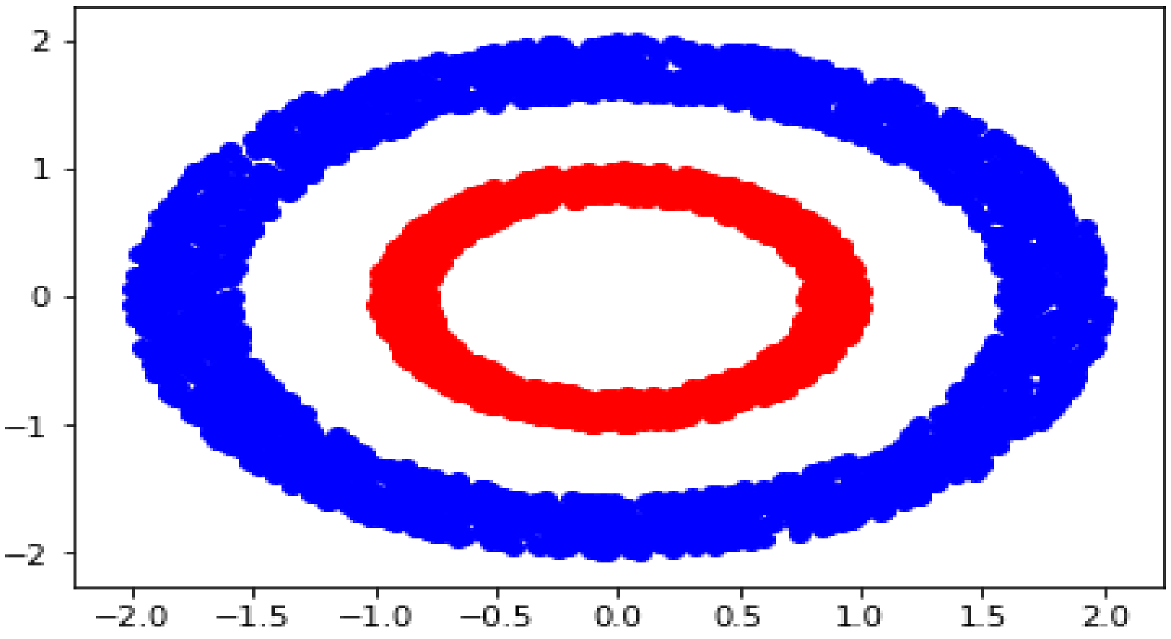


Fig2) Accuracy: 1.0

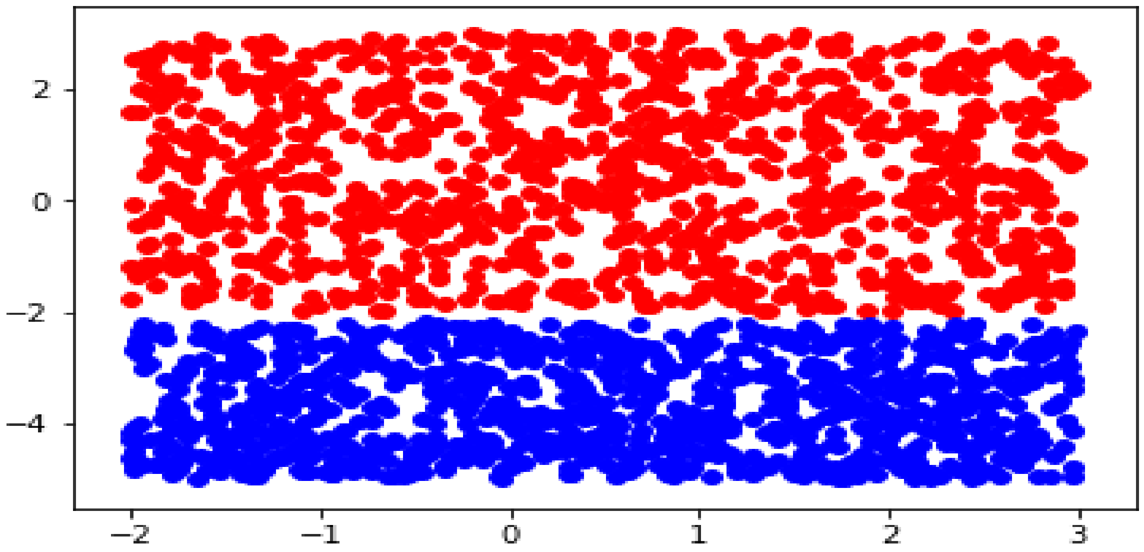
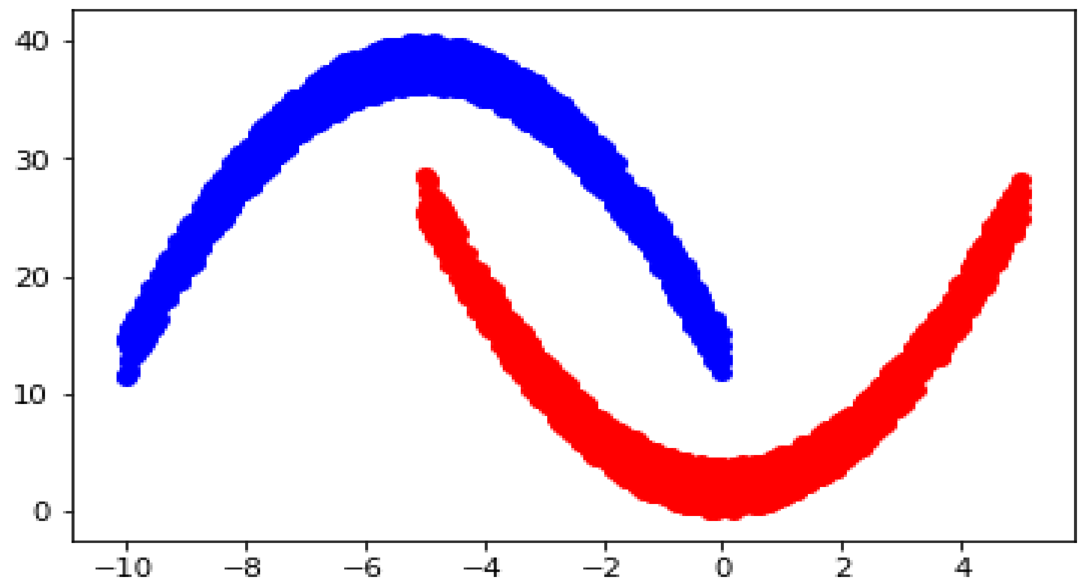


Fig3) Accuracy: 0.8971



After Grid Search : 1st Dataset:

Best Model:

```
SVC(C=0.01, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

Accuracy: 1.0

2nd dataset:

Best Model:

```
SVC(C=0.1, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma=10, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

Accuracy: 1.0

3rd dataset

Best Model:

```
SVC(C=0.01, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma=100, kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

Accuracy: 1.0

d) With (squared_Hinge) Loss:

Accu_A : 0.6599131693198264
Accu_B : 1.0
Accu_C : 0.8946875

e)
With rbf kernel:
A - 1.0
B - 1.0
C - 1.0

With poly kernel:
A- 0.6599131693198264
B - 1.0
C - 0.9978125

With sigmoid kernel:
A - 0.4390738060781476
B - 0.614
C - 0.153125

f)

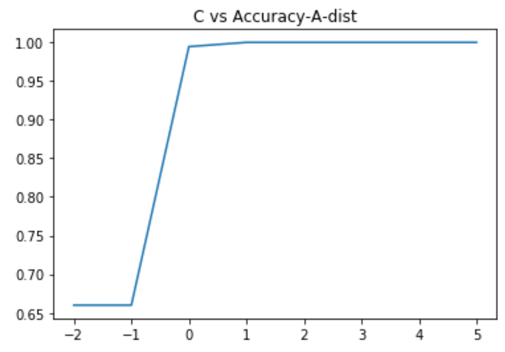
1. Ques : Why is a linear kernel able to classify/misclassify on the 3 distributions.
   Ans: As can be seen from decision boundary in 1st distribution below which was drawn on the distribution with linear kernel, it is not able to segregate the two classes as they are not linearly separable.
   For 2nd distribution, as it is linearly separable, it is able to give accurate decision boundary.
   For 3rd distribution, again it is not linearly separable though, the decision boundary drawn by linear svm here is giving 89% accuracy as it is partially separable with some cases classified as other class.

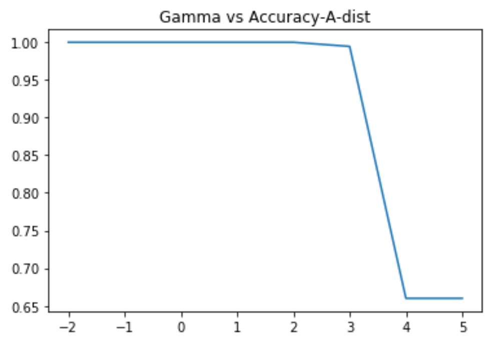
2. Ques:Why and how does the accuracy change on varying C and gamma parameter in part(C)
   Ans: C parameter is trade-off between mis-classification of training sample and smoothing of decision function. Low value of C helps in smoothing decision function Whereas high(C) means that classifier is trying to classify each sample correctly by Considering more number of points as Support Vectors.
   “Gamma” parameter regulates the radius of influence of a support vector. If large, no Effect of ‘C’ but if small then model won’t be able to understand shape of distribution.

Accuracy change

Y-Axis - Accuracy:
X-Axis = log(C/Gamma)

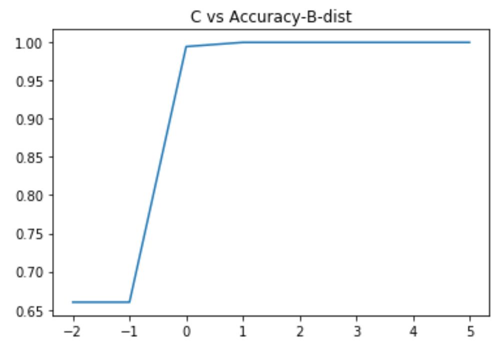
For distribution A:

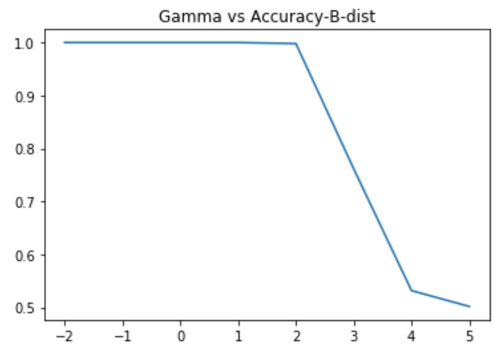




Here increasing C value is increasing accuracy as it is trying to become hard margin classifier , with low value of gamma as low gamma means less radius of influence for Support vectors. When C is very large, and with increase in gamma value and decrease in C value, accuracy is decreasing, as it means number of Support vectors has been increased which taking part in decision and thus accuracy is again decreasing.

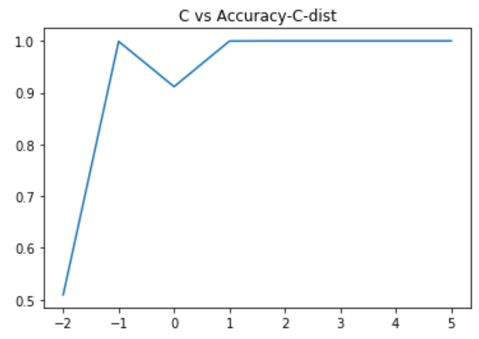
For Distribution B:

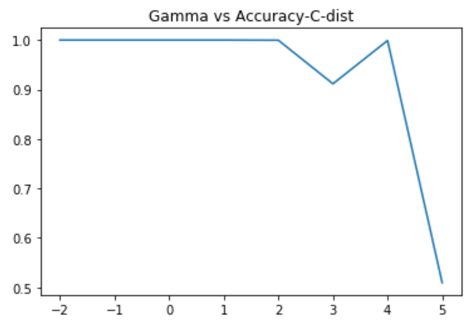




Here Same pattern as previous distribution but gamma value is having its effect early than previous case as in this distribution as when C value is low than we are allowing classifier to have misclassification and at the same time gamma value is large which means there are more number of support vectors taking part in decision so accuracy drops. This distribution is linearly separable so increasing value C will force to become hard margin classifier and as it is linearly separable so it is having 100% accuracy.

For Distribution C:





Here there is drop in between at C=1 and gamma 1000 and then again increase with c value increasing and gamma value decreasing.

3) Ques:Why and how does the accuracy change on changing the loss in part?
Ans: Accuracy is same in both the cases, reason can be that the loss function has Less effect in case of classification problem than regression problem in case of SVM. In some case there might be some difference in the accuracy using these loss function But in the experiment performed, both gave the same accuracy with linear kernel.

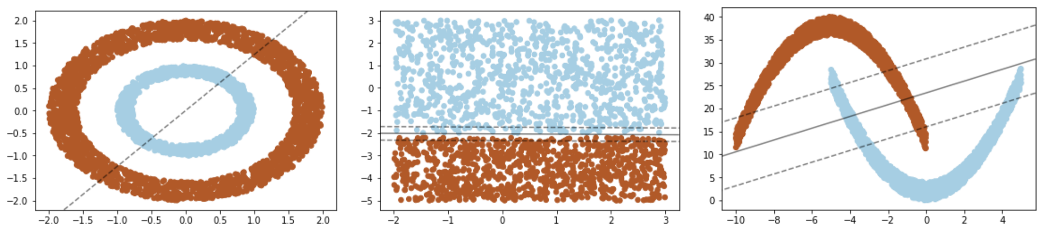
4) Ques: Why and how does the accuracy change on changing the kernels in above part
Ans: with different kernels , the result of accuracy differs. For first distribution, rbf and polynomial are able to perform well as data is not linearly separable and both kernel can have non-linear boundary so they are able to segregate classes. For B, C same pattern is observed. Kernel is able to separate them because they are linearly separable and can be separated by curves such as “rbf” and “poly”.
Whereas in case of C again they are able to perform well because non-linear boundary Able to segregate the data into class.

Accuracy with different kernel:

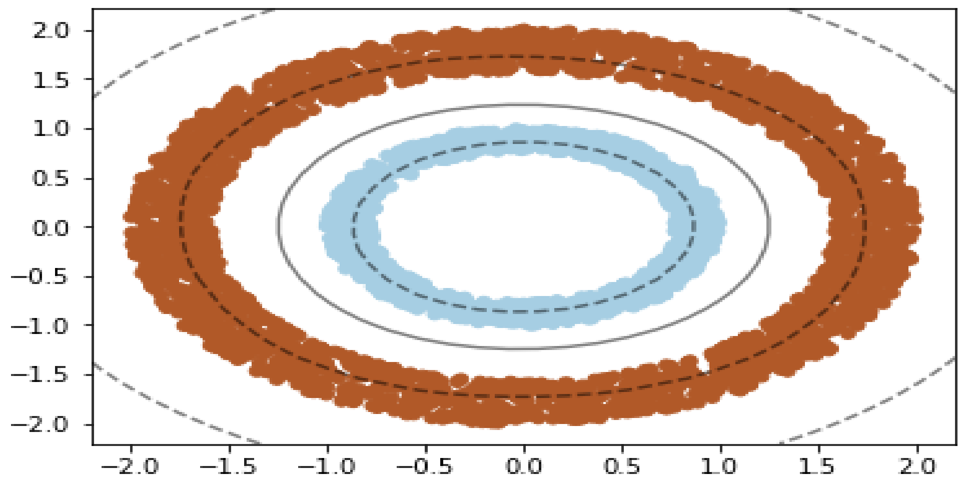
With rbf kernel:	With poly kernel:	With sigmoid kernel:
A - 1.0	A - 0.6599131693198264	A - 0.43907380607
B - 1.0	B - 1.0	B - 0.614
C - 1.0	C - 0.9978125	C - 0.153125

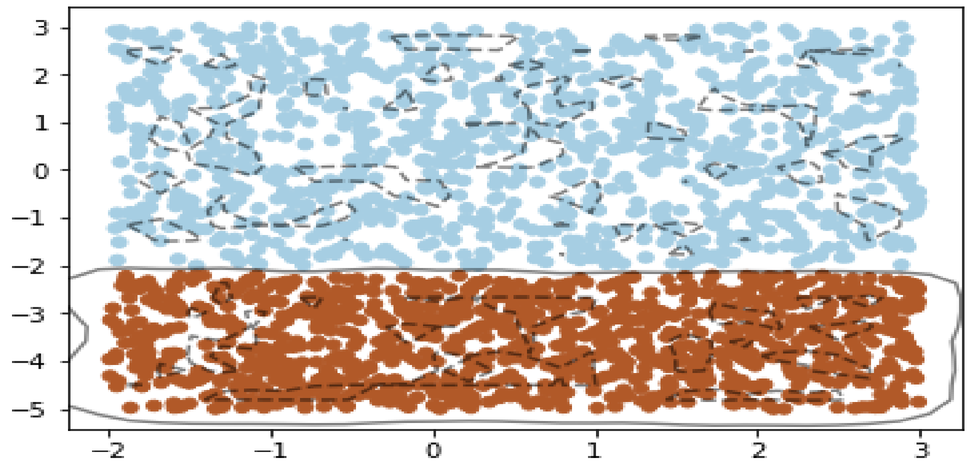
g)

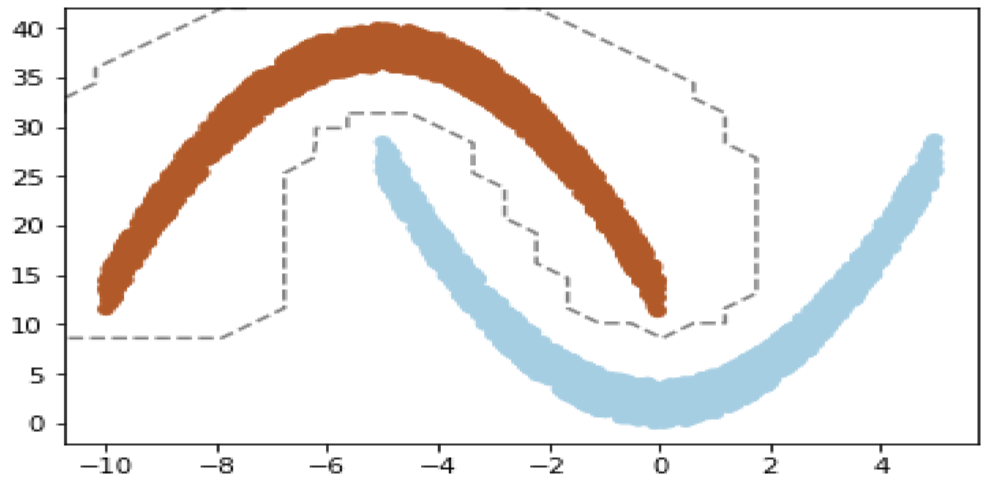
1) Plotting Decision Boundary in part b→



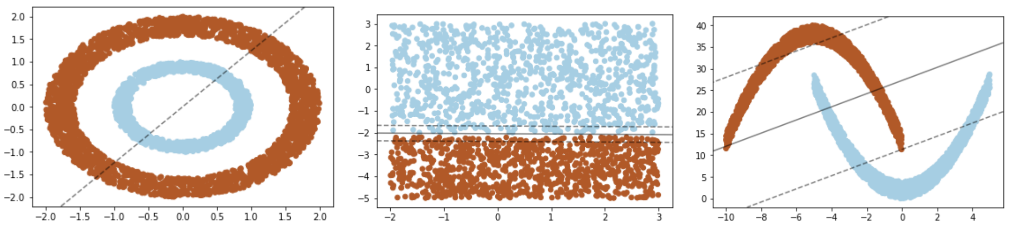
2) Plotting Decision Boundary for the best results obtained from the grid search in ©





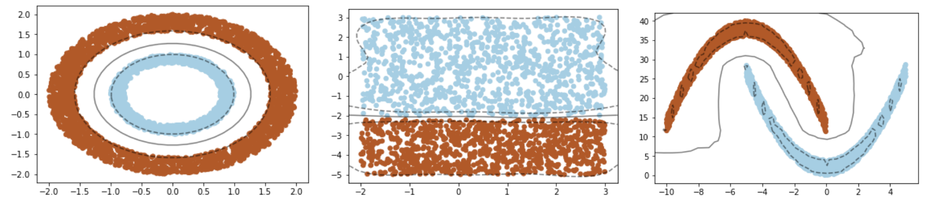


3) For loss = “Squared Hinge”

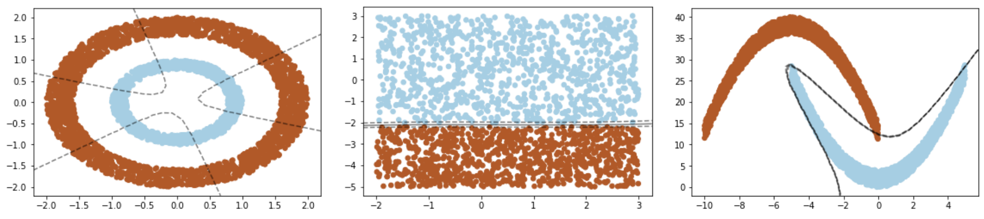


4)

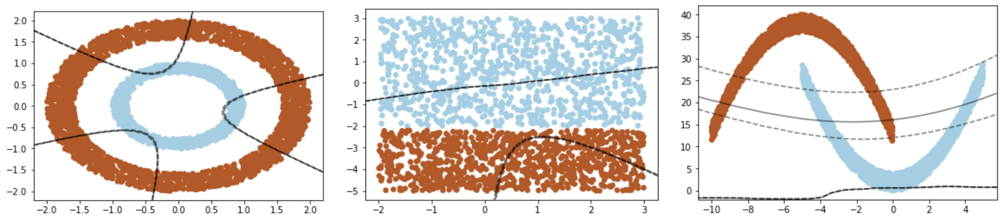
With “RBF kernel”:



With “Poly Kernel”:

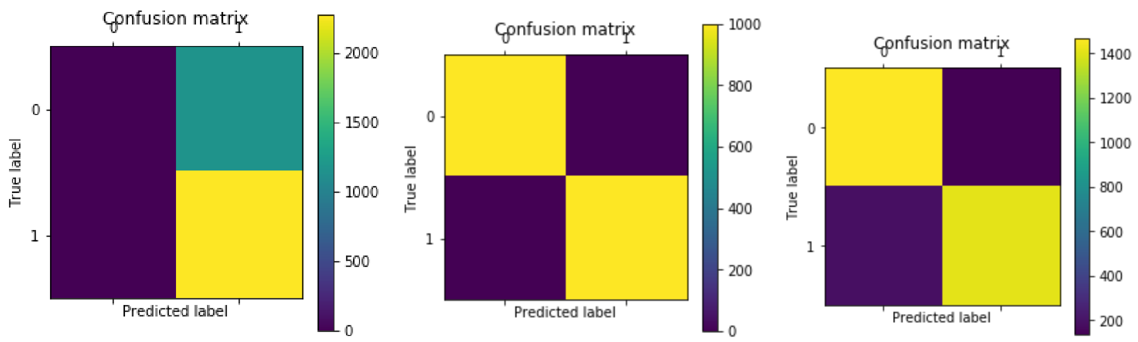


With Sigmoid:



Confusion Matrix:

b) with Linear kernel:

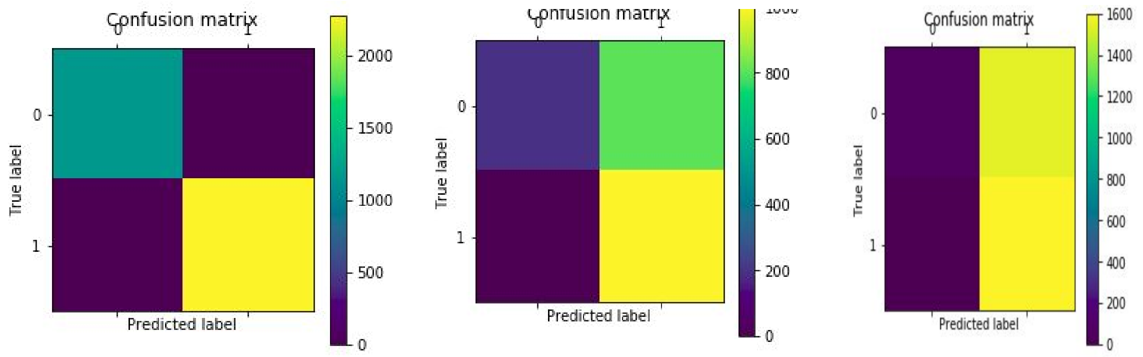


A-dist → TP,FP,TN,FN are 2280 1175 0 0

B-dist → TP,FP,TN,FN are 1000 0 1000 0

C-dist → TP,FP,TN,FN are 1411 134 1466 189

c) With Grid Search: on 3 dataset:

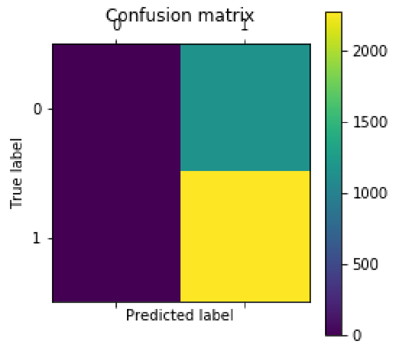


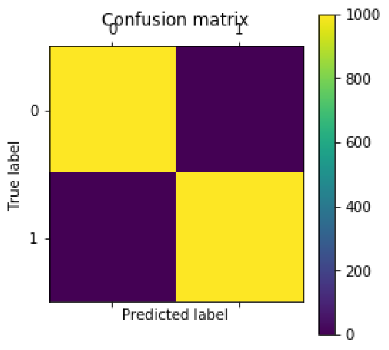
```
TP,FP,TN,FN are 2280 0 1175 0
```

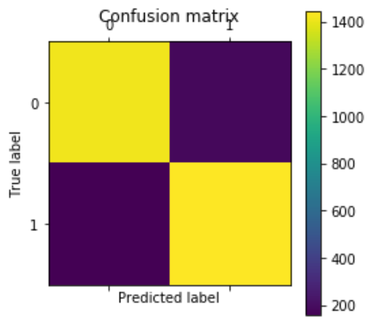
```
TP,FP,TN,FN are 1000 805 195 0
```

```
TP,FP,TN,FN are 1600 1544 56 0
```

d) for “squared hinge loss” :

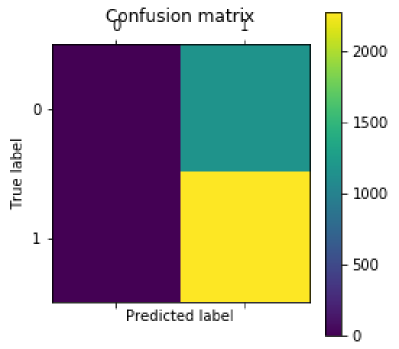


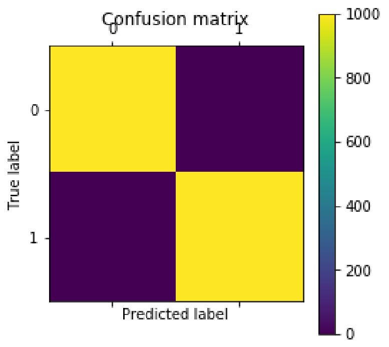


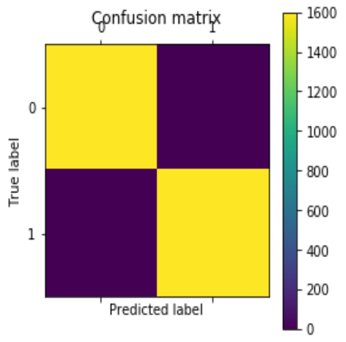


- TP,FP,TN,FN are 2280 1175 0 0
- TP,FP,TN,FN are 1000 0 1000 0
- TP,FP,TN,FN are 1494 132 1043 531

e) For “rbf” kernel:

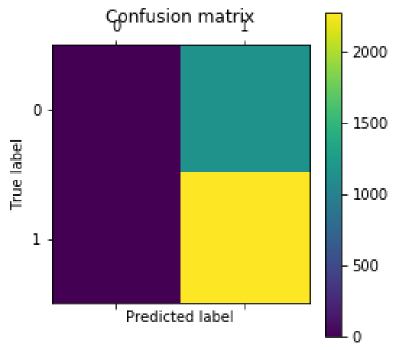


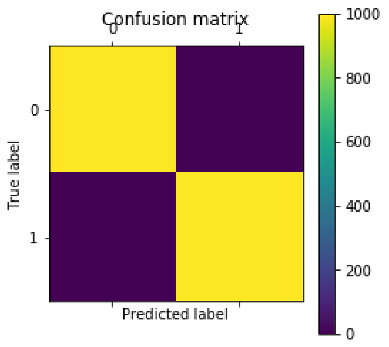


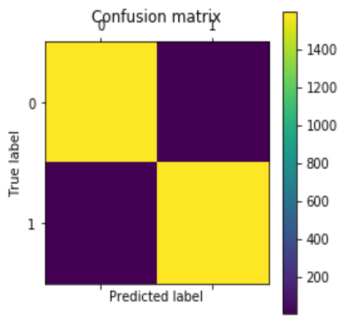


- TP,FP,TN,FN are 2280 1175 0 0
- TP,FP,TN,FN are 1000 0 1000 0
- TP,FP,TN,FN are 1600 0 1175 425

For “poly” kernel:

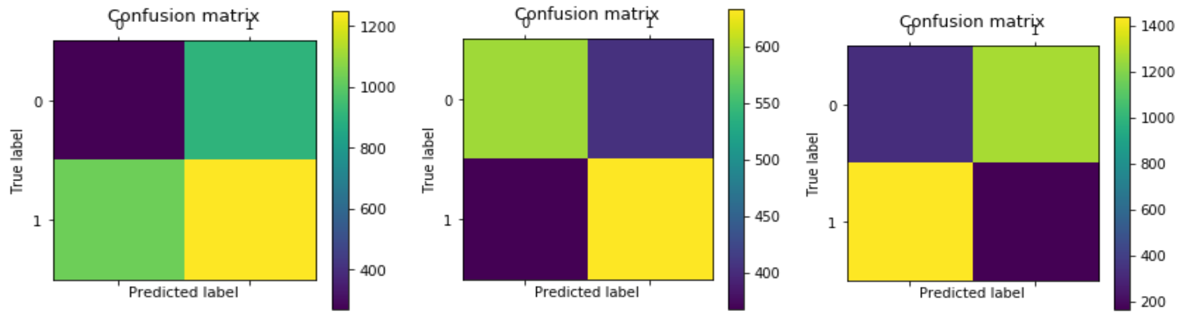






- TP,FP,TN,FN are 2280 1175 0 0
- TP,FP,TN,FN are 1000 0 1000 0
- TP,FP,TN,FN are 1597 2 1173 428

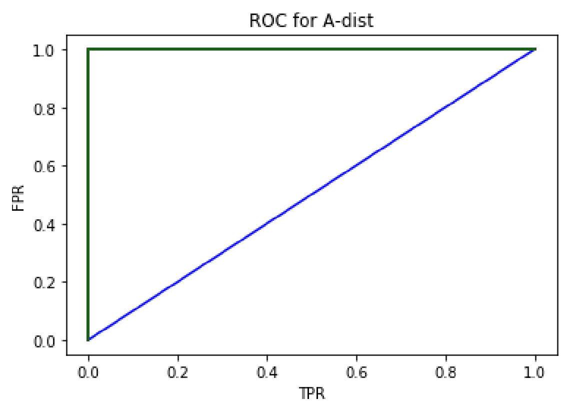
For “Sigmoid Kernel”:

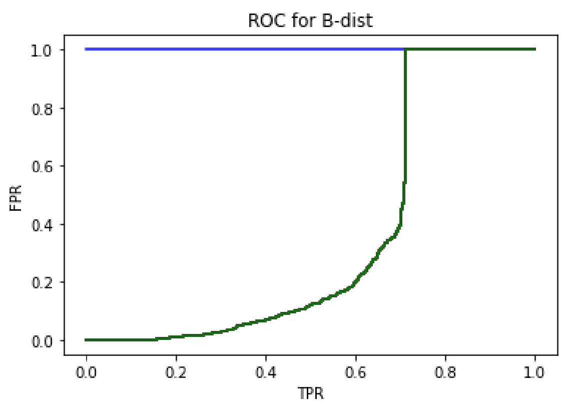


- TP,FP,TN,FN are 1249 907 268 1031
- TP,FP,TN,FN are 633 405 595 367
- TP,FP,TN,FN are 508 926 249 1517

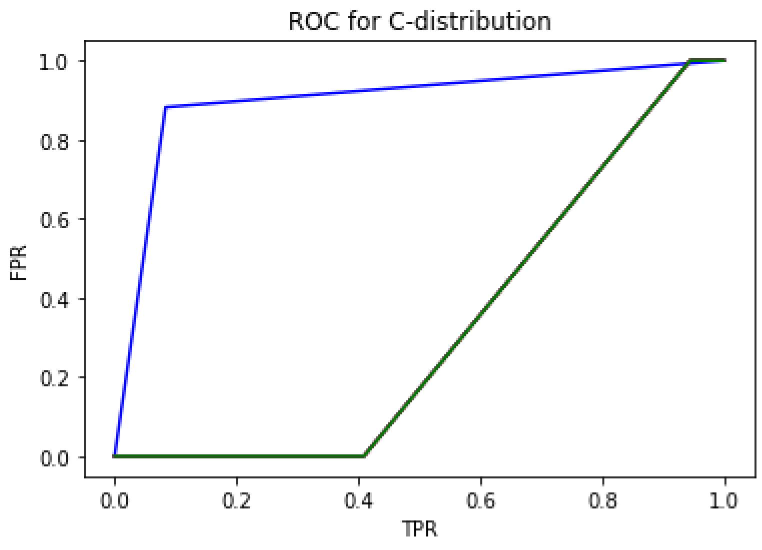
For data distribution a)

For data distribution b)



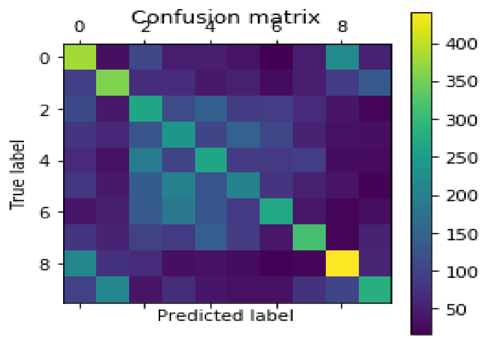


For data distribution c)

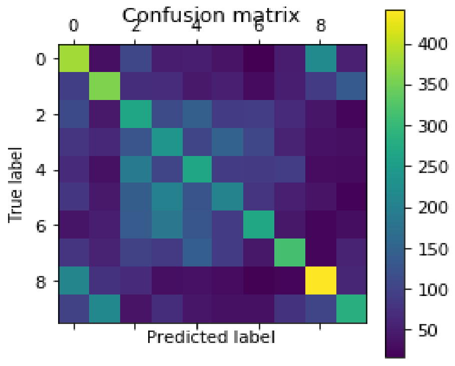


Q2)
Accuracy: one vs one: 0.3035
Accuracy: one vs all: 0.3035

Confusion Matrix: one vs one



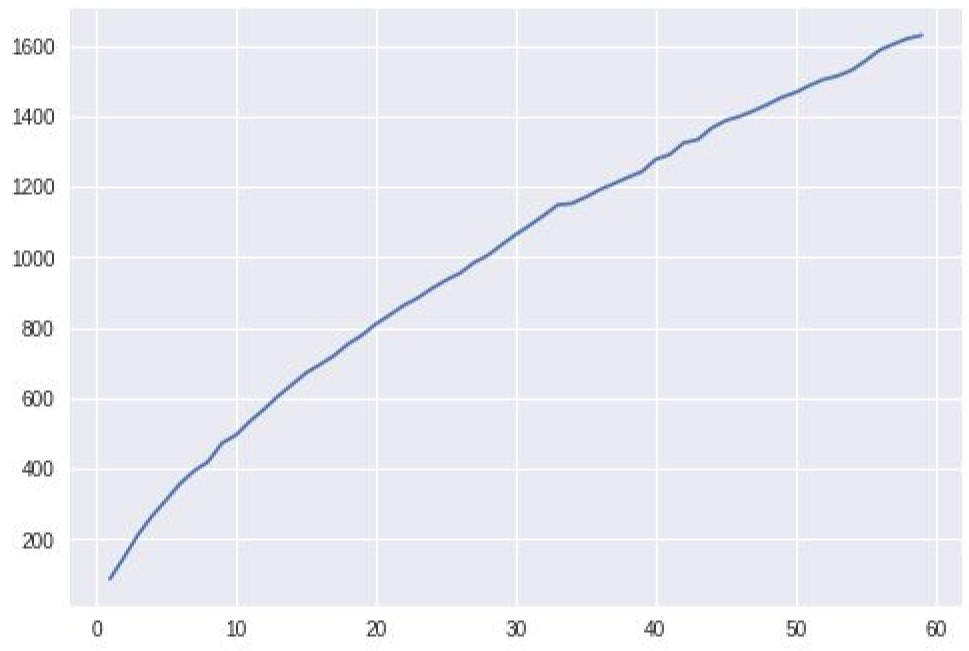
Confusion Matrix : one vs rest:



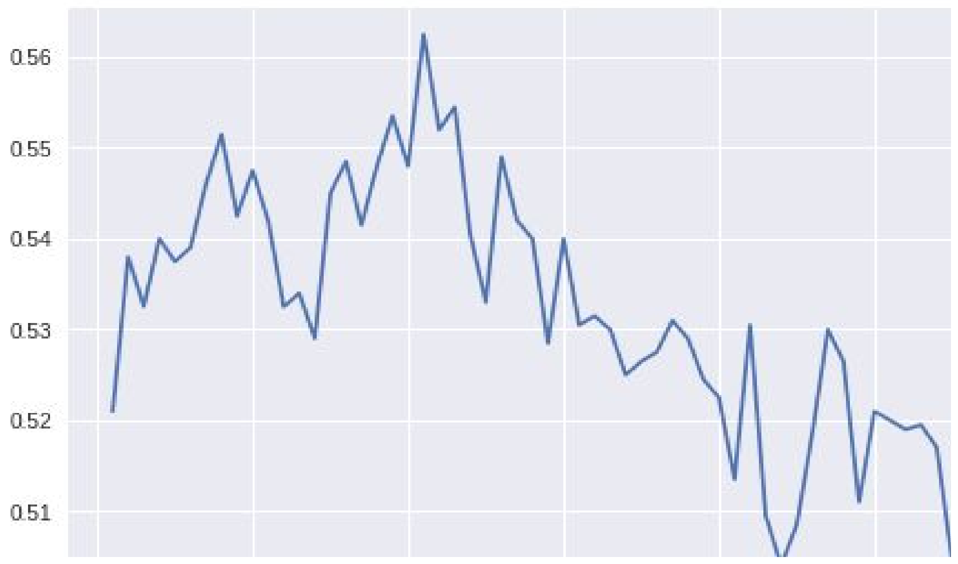
Reasoning:
The difference is the number of classifiers you have to learn, which strongly correlates with the decision boundary they create.
One vs all will train one classifier per class in total N classifiers.This often leads to imbalanced datasets meaning generic SVM might not work, but still there are some workarounds.
In one vs one you have to train a separate classifier for each different pair of labels. This leads to N*(N+1)/2 classifiers. This is much less sensitive to the problems of imbalanced datasets but is much more computationally expensive.
Class Imbalance can be a reason for this too. There can be case that both types of classifier are having same pool of support vectors per each class because of which both are giving same accuracy.

Q3)

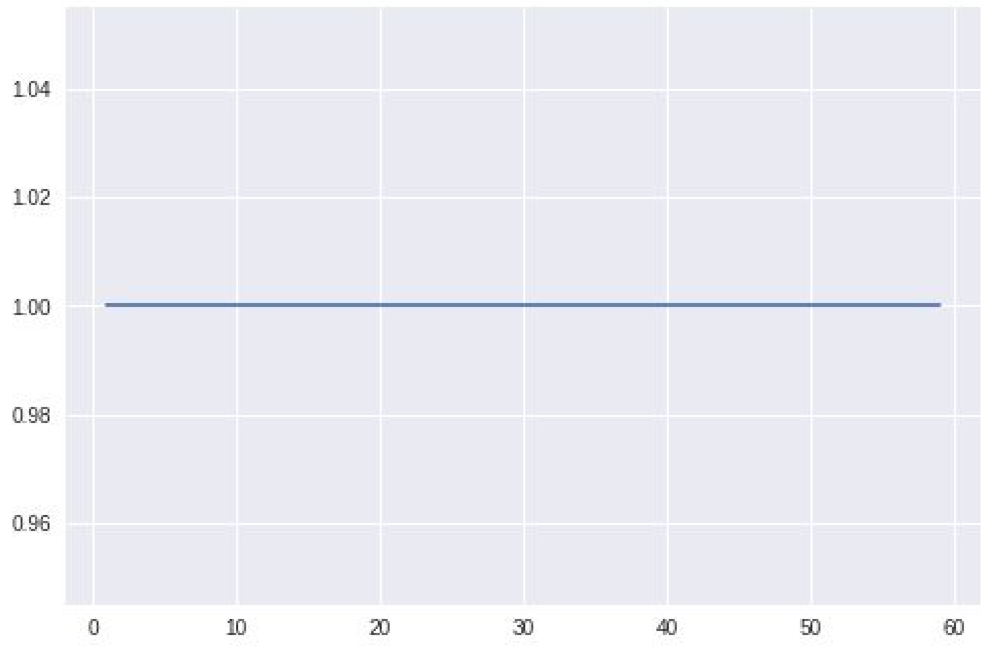
Iteration and Number of SV



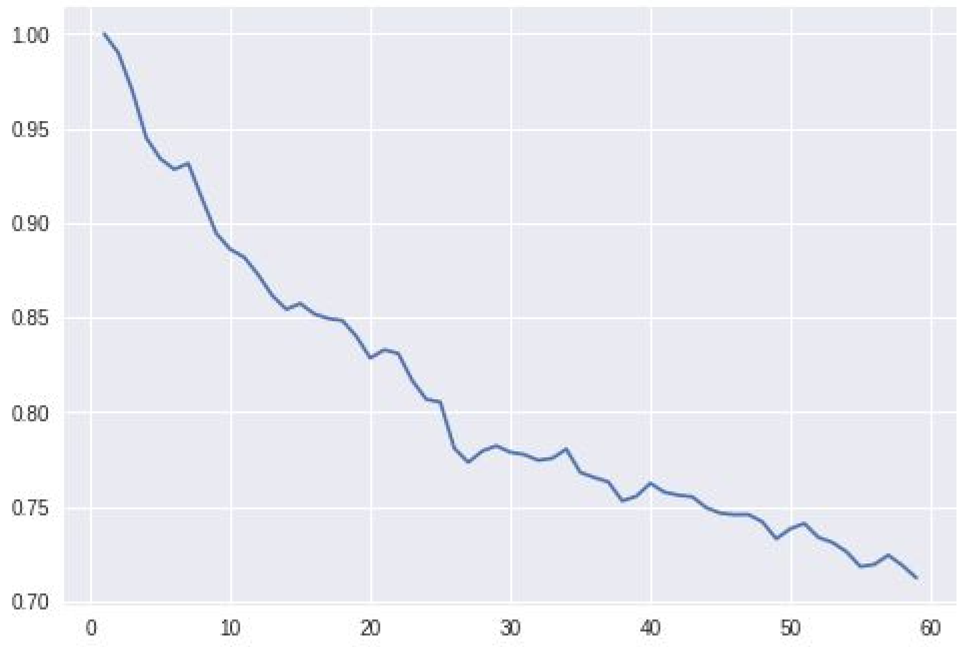
Iteration and Testing Set Accuracy:



Iteration and Current Train Set Accuracy:



Iteration and Cumulative Set accuracy:



Model Used: SVM with linear kernel and default parameters.
Explanation about result:

Algorithm for Linear SVM: For implementing incremental SVM , Batch size of 100 samples from total training sample of Cat and Dogs and classifier is learned on these samples. The support vectors of this classifier is extracted and added to next batch where new classifier is trained and this process goes on to give classifier.

The accuracy is in between (50-55) for whole test batch which do not have any pattern.
After adding support vectors of previous batch is not having effect on accuracy here.
While calculating Cumulative accuracy for the training part that has been processed, the accuracy start falling down which shows it can converge to what accuracy was there for test Set.
Strangely for current batch , accuracy was always 100% which shows the support vectors for every batch are able to differentiate the current batch but not the entire batch.
Number of support vectors increased on every iteration which shows that at every batch , some global support vectors are picked up by classifier. The number off final support vectors were also around 40% of total samples.

Q4)

NuSVM:

1. Time taken to train the model:	0.14580798149
2. Model performance:	0.9498997995991983

3. The number of FP and FN in the testing and training set : (491, 8, 457, 42) → 2nd and 4th value

C_SVM:

1. Time taken to train the model:	0.0397169589996
2. Model performance:	0.9438877755511023

3. The number of FP and FN in the testing and training set : (482, 17, 460, 39) → 2nd and 4th value

4. Comparison with simple SVM with a linear kernel:
   Accuracy:
      CSVM : 0.9479479479479479
      NuSVM: 0.9469469469469469
      LSVM: 0.9424424424424425

5. The number of support vectors of nu-SVM, C-SVM and simple SVM with a linear kernel

SV in NuSVM: 1009	SV in LSVM: 250
SV in CSVM: 281	