


Pattern Recognition
Department of Electrical and Information Engineering
University of Cassino and Southern Latium, Second Semester 2018
Homework Assignment 3
Assigned 30 May 2018; due 11:59pm, 10 June 2018

In preparing my solutions, I did not look at any old home work, copy anybody's answers or let them copy mine.

Name: Md. Kamrul Hasan

Signature: 

Submitted By:

Md. Kamrul Hasan

E-mail: kamruleeekuet@gmail.com

MAIA (M2), UNICAS, Italy.

Submitted To:

Prof. Francesco Tortorella, Ph.D.

E-mail: francesco.tortorella@unicas.it

Universita degli Studi di Cassino e del
Lazio Meridionale, Italy.

Problem 3.1 [60 %] Consider the dataset contained in the file hw3data.csv available on the course site. It contains 8,000 samples coming from a two {class problem, each made of 10 numerical features and a binary label (± 1). Consider the following tasks:

(a) Split the data into a training set (40%), validation set (40%) and test set (20%).

(b) Use the training set for building a ν -SVM and the validation set for choosing the optimal value (i.e. the one maximizing the AUC) of ν .

(c) Evaluate the AUC on the test set.

Execute the steps above with 4 different SVM kernels (linear, polynomial of degree 2, RBF, sigmoid). For each test, consider at least 5 different splits and evaluate average AUC and standard deviation. Discuss the results obtained.

Hint: when you search the optimal value of ν , remember that it is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the total number of training examples. For example, if you set it to 0.05 you are guaranteed to find at most 5% of your training examples being misclassified (at the cost of a small margin, though) and at least 5% of your training examples being support vectors.

Solution of 3.1: Data given for this homework contains 8000 rows and 11 columns where 8000 rows indicate the measurements of the observations that also known as the instance of the class. There are 10 features (Column 1st to 10th) for each instances of the observations. Last column (11th) indicates the class label (+1 or -1) corresponding to each instance. Sample for each row there is a corresponding class label which are +1 and -1. The overall work flow used for the question 3.1 is shown in Figure 3.1.1.

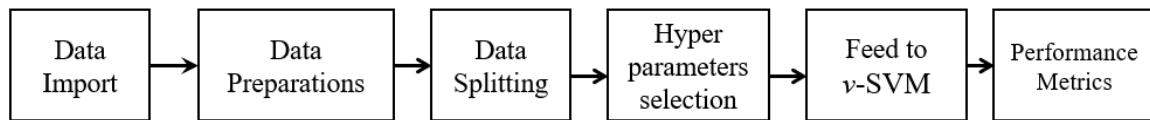


Figure 3.1.1: Overall pipeline of the home work for question 3.1

After importing the data, one preparation has been done to this data. Given data are not normalized which has greater inter feature variation that might produce unwanted biased to the classification. To solve features variance problem, standardizations has been done by using the equation Eq. 3.1.1.

$$Data(:, 1:10) = \frac{Data(:, 1:10) - \text{mean}}{\text{Standard Deviation}} \quad (3.1.1)$$

Where, both Mean and Standard Deviation are the 1×10 *vectors*. And we don't need to normalize last column because it's the class level only. To normalize the data, I have used "*sklearn*" preprocessing that are available in Python Library.

After normalizing, the next target is to split the data into train (40%), validation (40%) and test (20%). The block diagram used for the data splitting is shown in Figure 3.1.2. Firstly, data split into instances of the class and class labels. Then the data in each class (-1 or +1), are divided into 5 subsections (A, B, C, D and E). And then, train (40%), validation (40%) and test (20%) were selected randomly like Table 1.

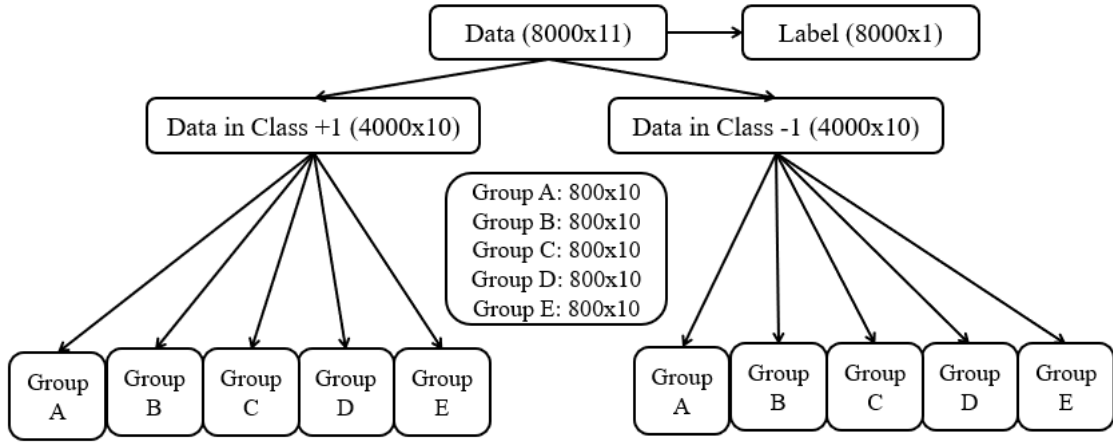


Figure 3.1.2: Data splitting block presentation

Table 1: Training (40 %), Validation (40 %) and Testing (20 %) data selections from the Split data

Case of Selections	Training Set		Validation Set		Testing Set	
	Data Class, +1	Data Class, -1	Data Class, +1	Data Class, -1	Data Class, +1	Data Class, -1
Case I	Group: A, B	Group: A, B	Group: C, D	Group: C, D	Group: E	Group: E
Case II	Group: B, C	Group: B, C	Group: D, E	Group: D, E	Group: A	Group: A
Case III	Group: C, D	Group: C, D	Group: E, A	Group: E, A	Group: B	Group: B
Case IV	Group: D, E	Group: D, E	Group: A, B	Group: A, B	Group: C	Group: C
Case V	Group: E, A	Group: E, A	Group: B, C	Group: B, C	Group: D	Group: D
Dimensions	1600x10	1600x10	1600x10	1600x10	800x10	800x10
Total Dimensions	Training: 3200x10		Validation: 3200x10		Testing: 1600x10	

From the Table 1, it is seen that, there are possible 5 cases of the Train, Validation and Test data selection. For each case, we need to find out the AUC for Test data after getting optimal ν from the Validation test. The overall work flow for this task is shown in Figure 3.1.3.

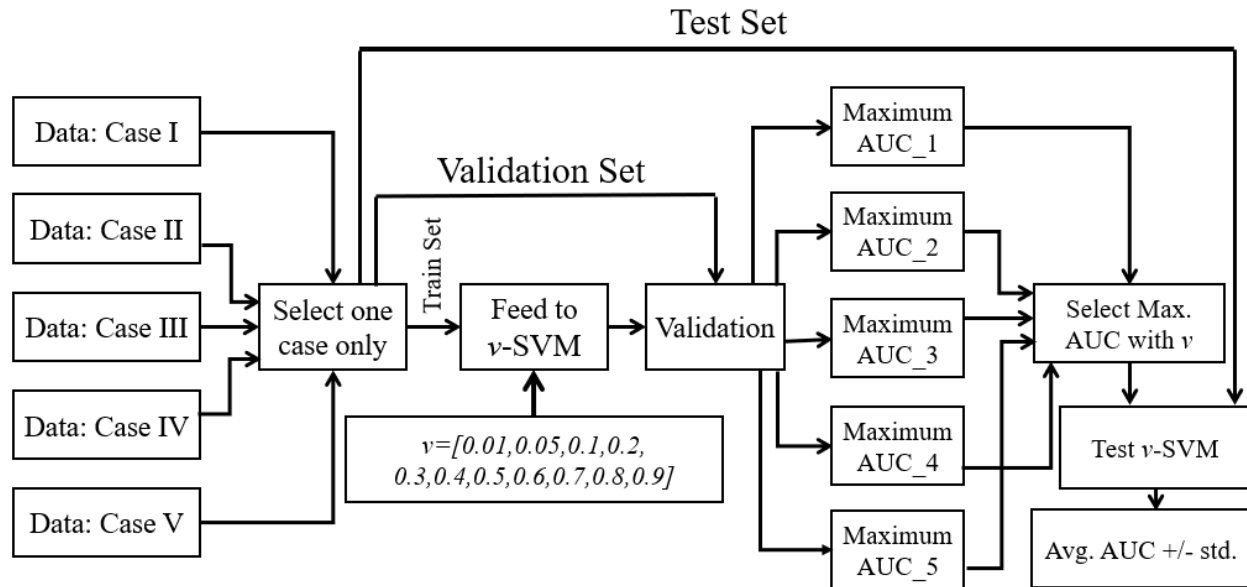


Figure 3.1.3: Flow diagram of nu-SVM train, validation and test

From the above Figure, it is seen that every time at a once only one case (that contain Test, Train and Validation data) should be select for input. For every value of **nu**, we need to train the nu-SVM and validation is used for getting maximum AUC. After selecting best **nu**, network is tested by the test data. That will give five different values of AUC for five different cases. Form those values of AUC, we need to calculate average AUC with standard deviation. For each case, performance metrics e.g. ROC curve also plot for all possible test data that will give 5 different ROC curves.

Results and Discussions

The overall results for this question is divided into 4 parts for four different kernels (RBF, Poly of degree 2, Sigmoid and Linear) used in the problem that are described below.

Results for RBF kernel: For this case, we are supposed to fix the kernel type as RBF. After fixing kernel, all steps that mention in Fig. 3.1.3 were followed. Results after validation to select the **nu** value are given in Table 2 and test results for RBF kernel are given in Table 3.

Table 2: Summery of the Validation results for “**rbf**” kernel

Test data	Max. Area Under ROC (AUC)	Best Nu value	Best AUC	Best Nu
Case I	0.914140625	0.3	0.914140625	0.3
Case II	0.912224121	0.4		
Case III	0.913650878	0.3		
Case IV	0.912174316	0.4		
Case V	0.912277832	0.4		

[N.B: Case1~Case5 described in Table 1]

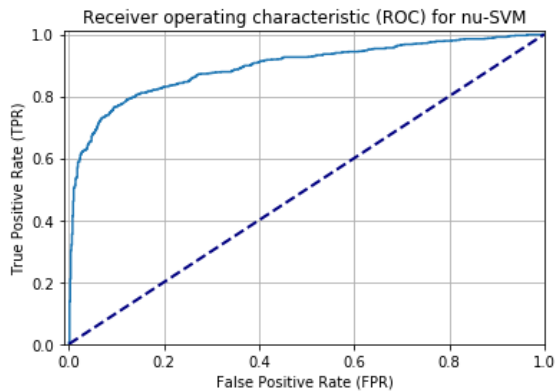
From the Table 2, we see that the maximum value of AUC is at case I validation sets of data having **nu** value is 0.3. So, the best value of nu for RBF kernel is 0.3 that maximize the AUC.

Table 3: Summery of the Test results for “**rbf**” kernel

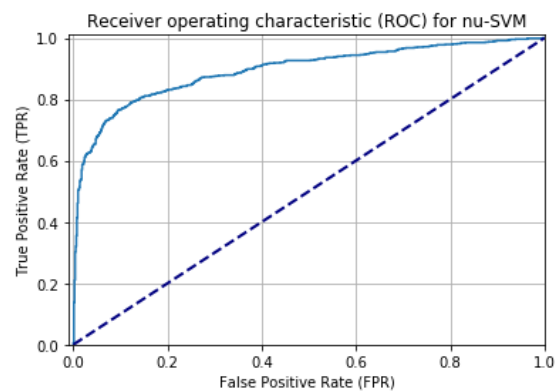
Test data	Area Under ROC (AUC)	Average AUC	ROC curve
Case I	0.8943382813	0.907 +/- 0.006	Figure 3.1.4 (a)
Case II	0.9080851563		Figure 3.1.4 (b)
Case III	0.9081640625		Figure 3.1.4 (c)
Case IV	0.9115546875		Figure 3.1.4 (d)
Case V	0.9116203125		Figure 3.1.4 (e)

[N.B: Case1~Case5 described in Table 1]

From the Table 3, it is noticeable that, for the different set of Test data from different cases, AUC is little bit different. Overall AUC is 0.907 +/- 0.006 i.e. [0.901, 0.913]. The ROC curves for each case of Test data are shown in Figure 3.1.4 (a)~ 3.1.4 (e).



(a)



(b)

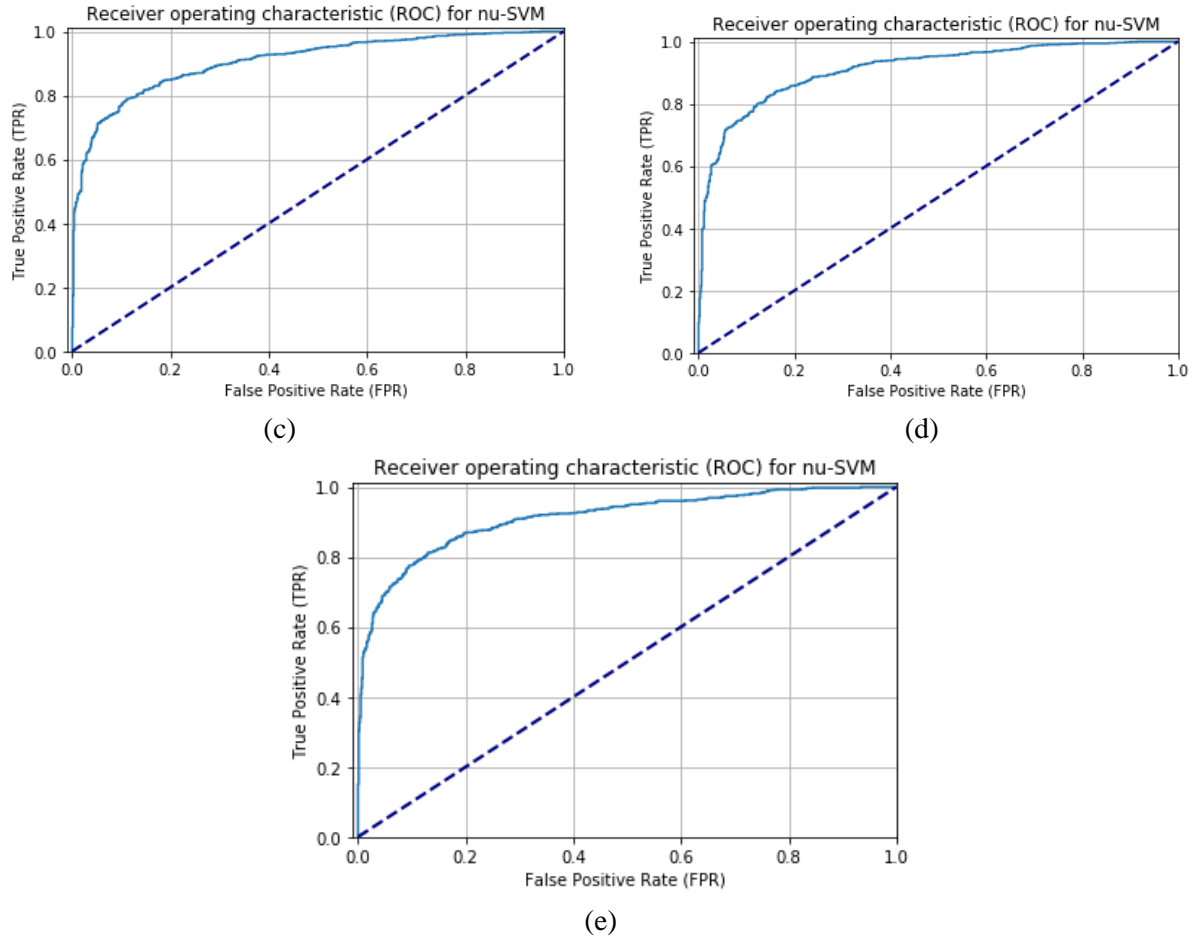


Figure 3.1.4: Test ROC curves for RBF kernel a) Case 1 Test, b) Case 2 Test, c) Case 3 Test, d) Case 4 Test, e) Case 5 Test

Results for Poly (Degree 2) kernel: As described earlier, for this case, we need to fix the kernel type as poly of degree 2. After that all steps that mention in Fig. 3.1.3 were followed. Results after validation to select the **nu** value are given in Table 4 and test results for poly kernel are given in Table 5.

Table 4: Summary of the Validation results for “poly” kernel

Test data	Max. Area Under ROC (AUC)	Best Nu value	Best AUC	Best Nu
Case I	0.878852050781	0.5	0.881087890625	0.5
Case II	0.877671386710	0.5		
Case III	0.877355468750	0.5		
Case IV	0.881087890625	0.5		
Case V	0.877156249999	0.5		

[N.B: Case1~Case5 described in Table 1]

From that Table 4, we see that the maximum value of AUC is at case IV validation sets of data having nu value is 0.5. So, the best value of nu for polynomial kernel is 0.5 that maximize the AUC for polynomial kernel. From the Table 5, it is noticeable that, for the different set of Test data from different cases, AUC is little bit different. Overall AUC is 0.873 ± 0.004 i.e. $[0.869, 0.877]$. The ROC curves for each case of Test data are shown in Figure 3.1.5 (a)~ 3.1.5 (e).

Table 5: Summery of the Test results for “poly” kernel

Test data	Area Under ROC (AUC)	Average AUC	ROC curve
Case I	0.8674609375	0.873 \pm 0.004	Figure 3.1.5 (a)
Case II	0.8680812499		Figure 3.1.5 (b)
Case III	0.8763078125		Figure 3.1.5 (c)
Case IV	0.8757523437		Figure 3.1.5 (d)
Case V	0.8763421874		Figure 3.1.5 (e)

[N.B: Case1~Case5 described in Table 1]

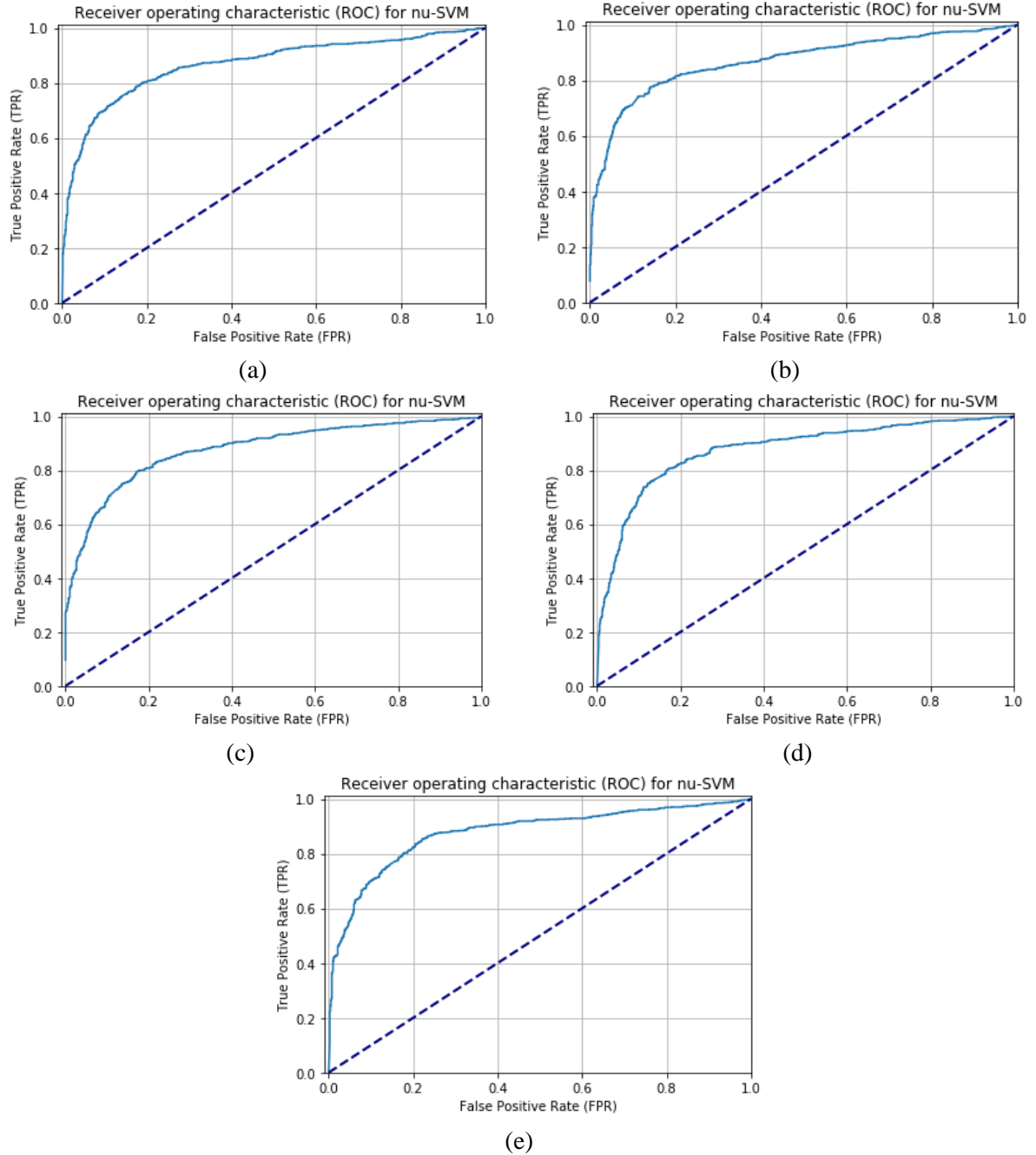


Figure 3.1.5: Test ROC curves for Poly of degree 2 kernel a) Case 1 Test, b) Case 2 Test, c) Case 3 Test, d) Case 4 Test, e) Case 5 Test

Results for Linear kernel: As described earlier, for this case, we need to fix the kernel type as linear. After that all steps that mention in Fig. 3.1.3 were followed. Results after validation to select the **nu** value are given in Table 6 and test results for linear are given in Table 7.

Table 6: Summery of the Validation results for “**linear**” kernel

Test data	Max. Area Under ROC (AUC)	Best Nu value	Best AUC	Best Nu
Case I	0.8395307617	0.6	0.8415498046	0.6
Case II	0.8379853515	0.6		
Case III	0.8371230468	0.6		
Case IV	0.8363706054	0.6		
Case V	0.8415498046	0.6		

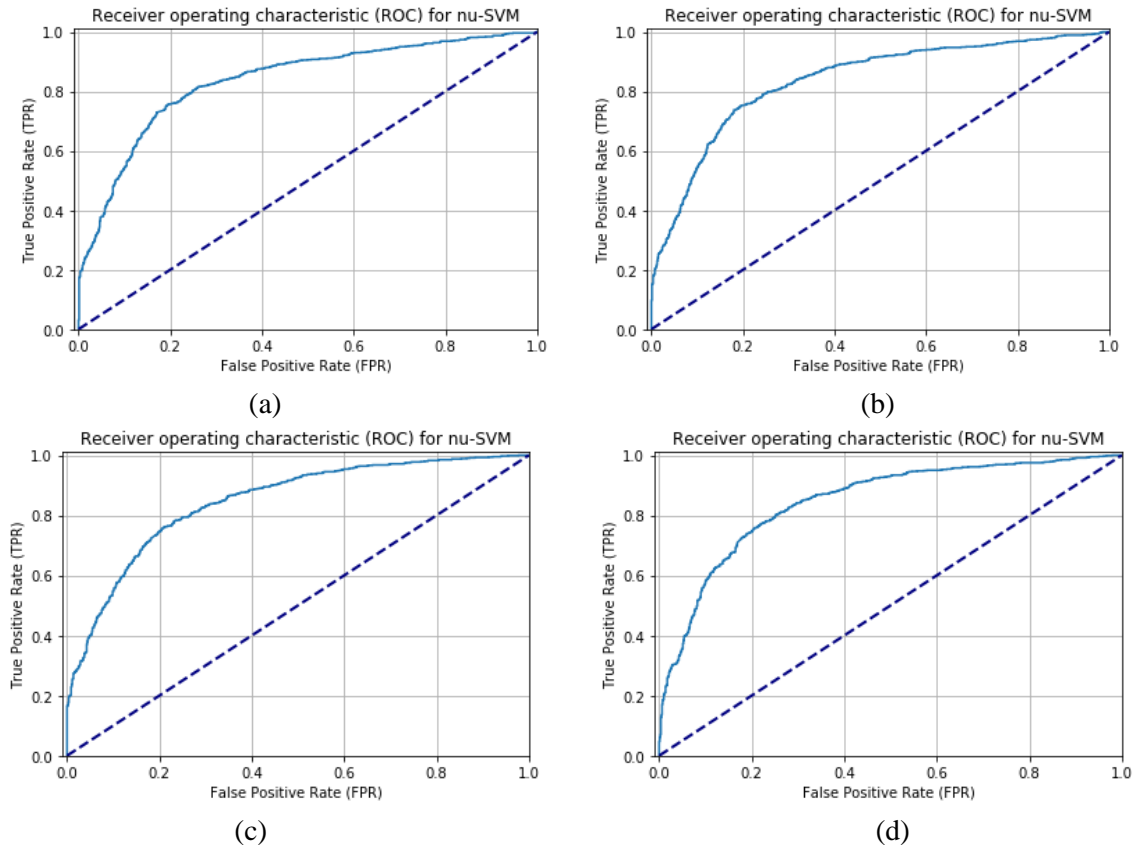
[N.B: Case1~Case5 described in Table 1]

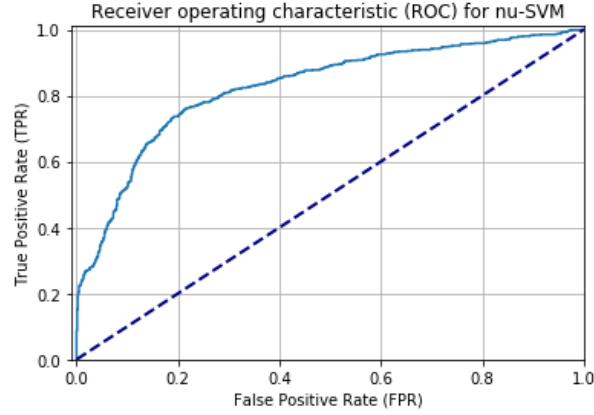
From that Table 6, we see that the maximum value of AUC is at case V validation sets of data having nu value is 0.6. So, the best value of nu for linear kernel is 0.6 that maximize the AUC for linear kernel. From the Table 7, it is noticeable that, for the different set of Test data from different cases, AUC is little bit different. Overall AUC is 0.838 +/-0.008 i.e. [0.83, 0.846]. The ROC curves for each case of Test data are shown in Figure 3.1.6 (a)~ 3.1.6 (e).

Table 7: Summery of the Test results for “**linear**” kernel

Test data	Area Under ROC (AUC)	Average AUC	ROC curve
Case I	0.83563203	0.838 +/-0.008	Figure 3.1.6 (a)
Case II	0.83671562		Figure 3.1.6 (b)
Case III	0.84590156		Figure 3.1.6 (c)
Case IV	0.84548593		Figure 3.1.6 (d)
Case V	0.82508125		Figure 3.1.6 (e)

[N.B: Case1~Case5 described in Table 1]





(e)

Figure 3.1.6: Test ROC curves for Linear kernel a) Case 1 Test, b) Case 2 Test, c) Case 3 Test, d) Case 4 Test, e) Case 5 Test

Results for sigmoid kernel: As described earlier, for this case, we need to fix the kernel type as sigmoid. After that all steps that mention in Fig. 3.1.3 were followed. Results after validation to select the nu value are given in Table 8 and test results for sigmoid are given in Table 9.

Table 8: Summery of the Validation results for “**sigmoid**” kernel

Test data	Max. Area Under ROC (AUC)	Best Nu value	Best AUC	Best Nu
Case I	0.8232861328124	0.9	0.8254848632812	0.9
Case II	0.8222333984375	0.8		
Case III	0.8212880859375	0.9		
Case IV	0.8205166015625	0.9		
Case V	0.8254848632812	0.9		

[N.B: Case1~Case5 described in Table 1]

From that Table 8, we see that the maximum values of AUC are at case one validation sets of data having nu value is 0.9. So, the best value of nu for sigmoid kernel is 0.9 that maximize the AUC for sigmoid kernel. From the Table 9, it is noticeable that, for the different set of Test data from different cases, AUC is little bit different. Overall AUC is 0.823 +/- 0.008 i.e. [0.815, 0.831]. The ROC curve for each case of Test data are shown in Figure 3.1.7 (a)~ 3.1.7 (e).

Table 9: Summery of the Test results for “**sigmoid**” kernel

Test data	Area Under ROC (AUC)	Average AUC	ROC curve
Case I	0.820465625	0.823 +/-0.008	Figure 3.1.7 (a)
Case II	0.82028515625		Figure 3.1.7 (b)
Case III	0.8324859375		Figure 3.1.7 (c)
Case IV	0.83026171875		Figure 3.1.7 (d)
Case V	0.80901640625		Figure 3.1.7 (e)

[N.B: Case1~Case5 described in Table 1]

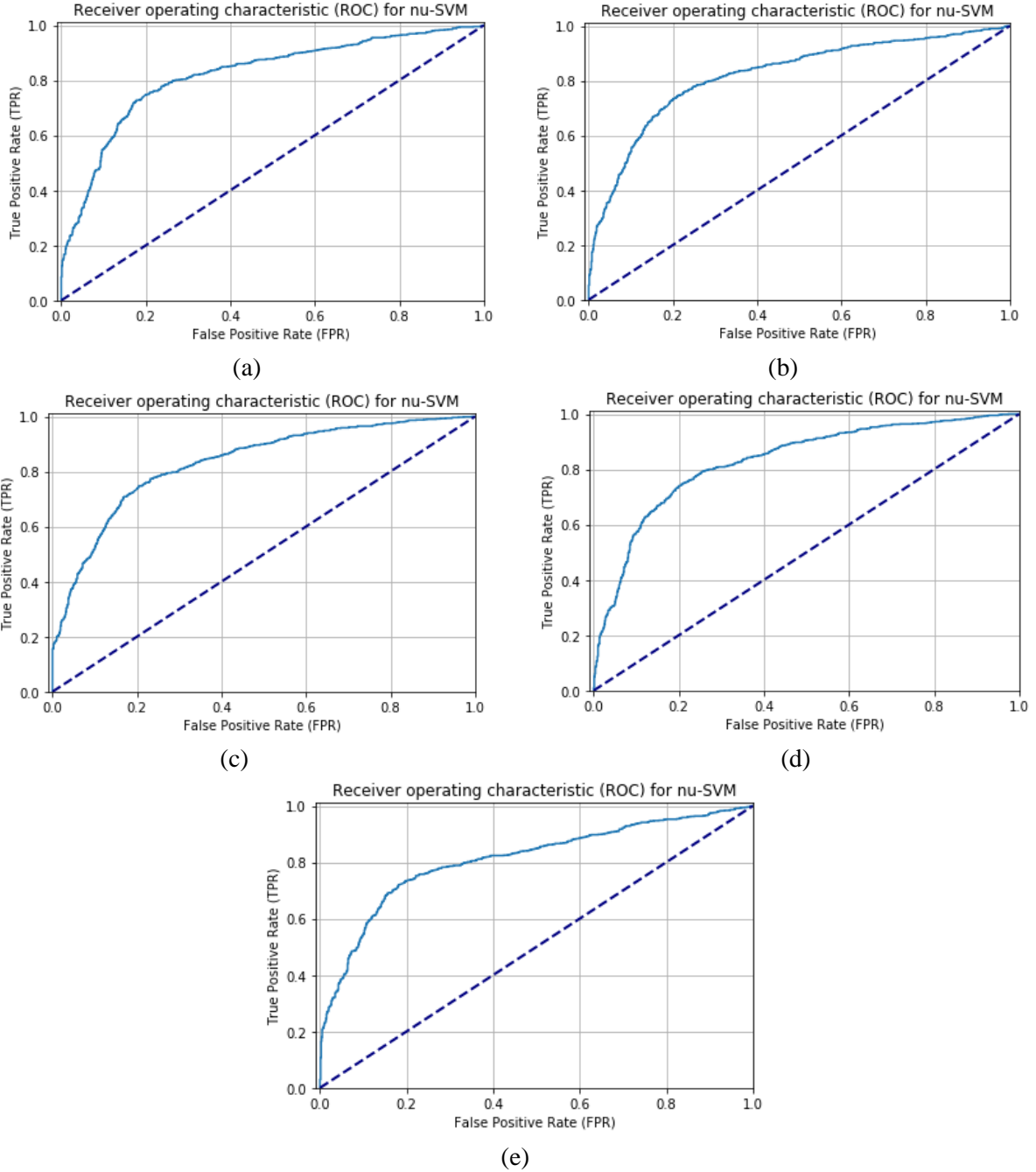


Figure 3.1.7: Test ROC curves for sigmoid kernel a) Case 1 Test, b) Case 2 Test, c) Case 3 Test, d) Case 4 Test, e) Case 5 Test

Conclusion of the Problem 3.1: After analyzing all those experiments on problem 3.1, we can conclude several outcomes. The AUC range for RBF, Poly (Degree 2), Linear and Sigmoid are $[0.901, 0.913]$, $[0.869, 0.877]$, $[0.83, 0.846]$ and $[0.815, 0.831]$ respectively. For the new sets of the testing data that are not present in train data, poly kernel has better stability then others kernel. Linear and Sigmoid kernel has less stability and performance metric e.g. AUC. So, considering performance metric i.e. AUC and stability, RBF kernel work better among all experimental kernels (Poly of degree 2, sigmoid and Linear). Final suggestion is that for the stability purposes poly of degree 2 with $\mathbf{nu}=0.5$ and for maximum AUC purposes RBF with $\mathbf{nu}=0.3$ will be best for this database. If you want to access this code, please see “**Problem_3_1.py**” in the folder that contain this report.

Problem 3.2 [40%] Use the dataset of Problem 3.1 and perform several splits into a training set and a test set (also with different sizes) to determine which combination of options among the ones you considered in Problem 3.1 ensures the highest AUC. Once you picked out the best model, save (see note below) and submit it together with your report. Your model will be run on a separate matrix containing new test data. Your grade will be based on the performance of your classifier on the new test data, which will contain a very large number of examples generated from the same distribution.

Solution of 3.2: There are two parts of this problem. First part is to select the best nu-SVM model and second part is to save that model for future prediction. The first part is shown in Block Diagram Figure 3.2.1.

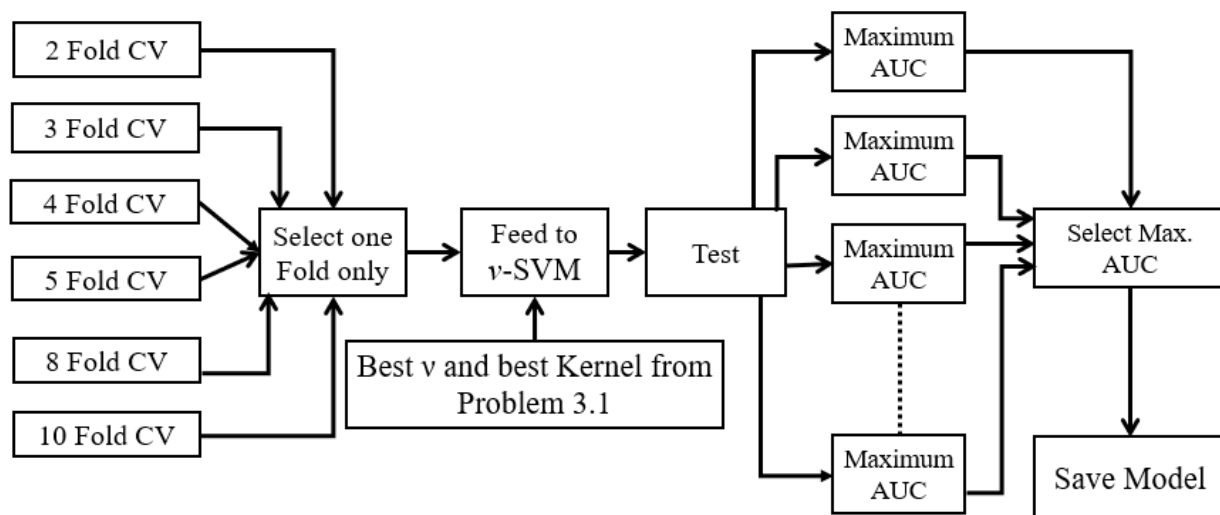


Figure 3.2.1: Block diagram of best model selection

Among all 6 number of folds of the data, we get different size (in percentage) of training and testing data. Using those train, test data and best kernel with nu (RBF with nu=0.3 and Poly of degree 2 with nu=0.5) from the problems 3.1, are used to get best model. Summary of first part to select best model is given in Table 10. To access the code for first part please see “***Problem_3_2_Part_1.py***” in the folder that contains this report.

Table 10: Summary of the best model selection

Test data	RBF with nu=0.3		Poly of degree 2 with nu=0.5	
	Max. Area Under ROC (AUC)	Accuracy	Max. Area Under ROC (AUC)	Accuracy
10-Fold CV	0.9135	84.500 %	0.8921	82.375 %
8-Fold CV	0.9144	84.583 %	0.8826	82.605 %
5-Fold CV	0.9168	84.438 %	0.8777	82.375 %
4-Fold CV	0.9146	84.200 %	0.8801	81.850 %
3-Fold CV	0.9102	83.485 %	0.8729	80.417 %
2-Fold CV	0.9123	83.775 %	0.8775	80.700 %

From that Table 10, it is seen that for all the folds, RBF kernel with nu=0.3 provides best AUC then poly kernel that I described in previous problem as well. Moreover, among all the Folds, 5-Fold CV provides

best AUC. So, from the Table 10, I conclude that to get best AUC from nu-SVM, kernel type RBF with nu=0.3 and 5-Fold CV (80 Training and 20 testing) is the best. So, in the next part, I will build the nu-SVM model with those selected parameters from Table 10 and save the model with *Joblib*. Using *Joblib*, I will save the model with named “*myModel.pkl*” that will be provided with this report to test future data.

To access the saved model named “*myModel.pkl*”, anyone need to follow the block diagram as shown in Figure 3.2.2.

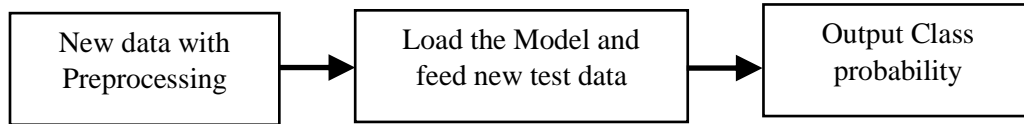


Figure 3.2.2: Block presentation of Load model and test new data

In the 1st block, you need to load new data for testing. Here, it is mentionable that the new data that will be classified with this model should be like in Eq. 3.2.1. And have the same distribution of the training data that means that same standard deviations and mean values. Suppose your new data matrix is A .

$$A = \begin{bmatrix} \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \end{bmatrix}_{row \times 10} \quad (3.2.1)$$

Which means that matrix A may have any numbers of rows but 10 columns only. After hitting the RUN button, this model will return the predicted binary class probability as like Eq. 3.2.2. If rows in A are in class label 0 then 1st column of y will have more probability and vice versa.

$$y = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}_{row \times 2} \quad (2.2.2)$$

N.B. The python code named “*mytest.py*” and the saved model (“*myModel.pkl*”) should be in the same working directory.

Attachments with this report:

1. Source data named as “*hw3data.csv*”.
2. Python Source code for 3.1 named as “*Problem_3_1.py*”.
3. Python Source code for 3.2 named as “*Problem_3_2_Part_1.py*”.
4. Python Source code for Load and access named as “*mytest.py*”.
5. Saved Item for the model named “*myModel.pkl*”