CS578 Project report: Loan Default Prediction

Priyank Jain: jain206@purdue.edu

December 5, 2016

Abstract

Can we predict whether or not it is safe to make a loan to a person based on past records of loans? In 2014, researchers at Imperial College London sponsored a competition challenging participants to predict the quantity of loss for a certain set of loans. In this project, I propose to simplify things by looking only at whether or not a loan will default.

1 Introduction

My goal in this project is to use the features of a loan to predict whether that loan would default or not. Such an application can be tremendously useful in banking, since banks make money by loaning money out to those in need. Identifying loans which might default can tremendously help banks to avoid loss and avoid unnecessary trouble/legal procedures.

2 Dataset and Technology

The training and test data files consist of 105471 and 210944 independent samples respectively, each with 771 anonymous features(one of them being the id and one of them being the 'loss' label, though the columns are numbered till f778). It has already been de-trended, randomly sorted, and standardized. The data set was suggested on the class website and can be found at: https://www.kaggle.com/c/loan-default-prediction

I use Python programming language and Git for Version Control. The code used for experiments could be found here: https://github.com/priyankjain/loan-default-prediction

Scikit python library is used for training and other machine learning related pipeline. Scikit is a popular machine learning library and has implementation of most machine learning algorithms. The code is written in object-oriented style. Charts are generated using matplotlib library in Python.

3 Preprocessing and running

I used the *train_v2.csv* from the Kaggle challenge in my experiments. The training set, cross-validation and testing set are all taken from this *train_v2.csv* file. I do not use *test_v2.csv* since the file does not have labels indicating whether the loan defaulted or not, leaving no way to report accuracy and other performance parameters. Before using the *train_v2.csv* file, the data is pre-processed as follows:
1. *train.csv* is generated from *train_v2.csv* file by dropping all rows with missing/NA values and dropping the first column corresponding to id. The loss function which has value from 0 to 100 in the raw data, is converted to -1 if it is greater than zero (indicating that the loan defaulted) or 1 otherwise (indicating that the loan was paid in full). This pre-processing leaves 51941 records in the *train.csv* file.
2. *test.csv* is generated so that is contains the last 5000 rows of the file *train.csv* (Command: "tail -n 5000 train.csv » test.csv" on Unix systems)

For experiments, I use a subset of these training examples to avoid long-wait times for experiments to complete and to not get involved in memory problems because of too much data.

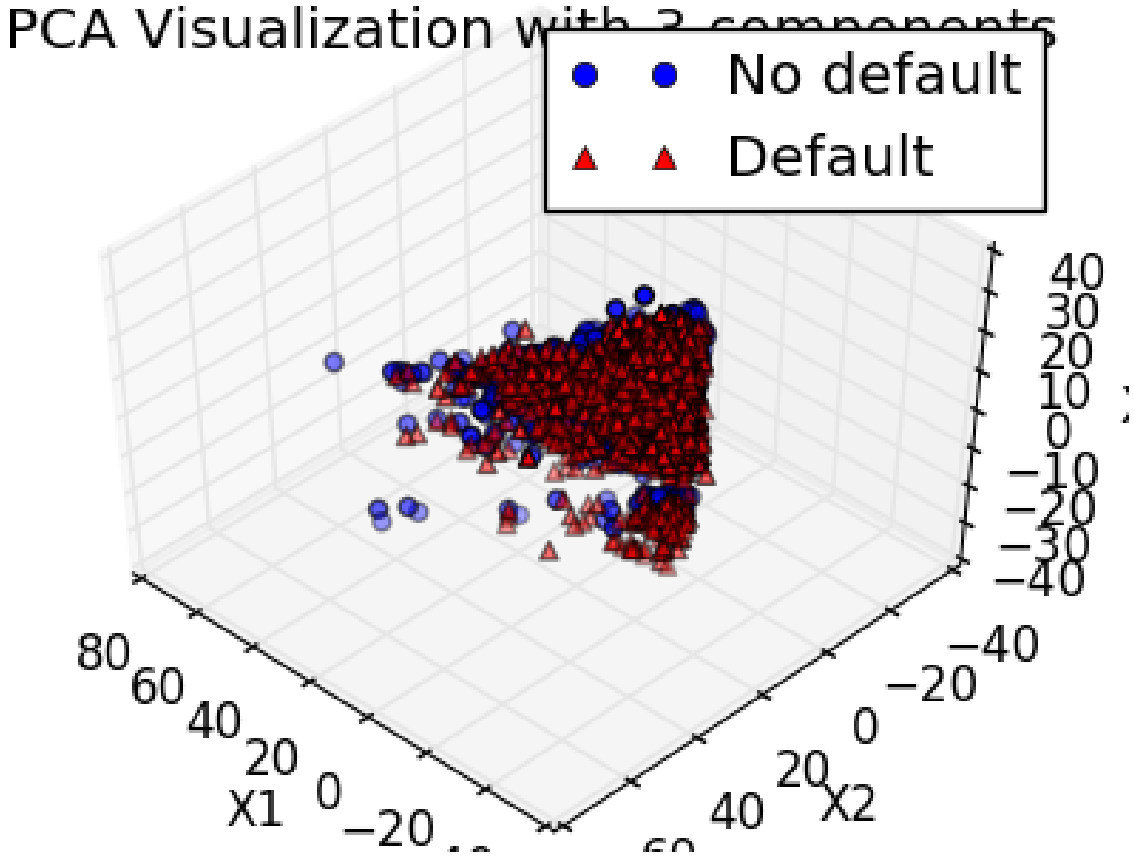

Figure 1: Visualization of dataset with dimensionality reduction from 769 to 3 using PCA

4 Experiment 1: Visualization using Principal Component Analysis

4.1 Dataset

This experiment makes use of the first 5000 training examples in the file train.csv. Out of this 5000 examples, 2500 are positive examples and 2500 negative examples.

4.2 Setup

In this experiment, I use Principal Component Analysis (PCA) and feature selection to understand the variation of features and their correlation to the loss label. I first use standard normalization to scale the features. Then I perform dimensionality reduction using PCA to reduce the dimensionality from 769 to 3 first. The results are plotted on a 3D Axes plot. Then I perform dimensionality reduction using PCA to reduce dimensionality from 769 to 2. The results are plotted on a 2D Axes plot.
Last part of this experiment involves visualizing features and their correlation with each other and the loss label. Feature selection is performed using mutual information criterion and the top 10 features are extracted. A scatter plot is then generated to visualize the correlation of these features and the loss label as a matrix.

4.3 Results

Figure 1 shows the visualization of data using PCA for dimensionality reduction to 3 dimensions. Figure 2 shows the visualization of data using PCA for dimensionality reduction to 2 dimensions. Figure 3 shows the scatter matrix of correlation between 10 highly correlated features to the label. The diagonal entries in the scatter matrix correspond to the distribution of values for the particular feature.

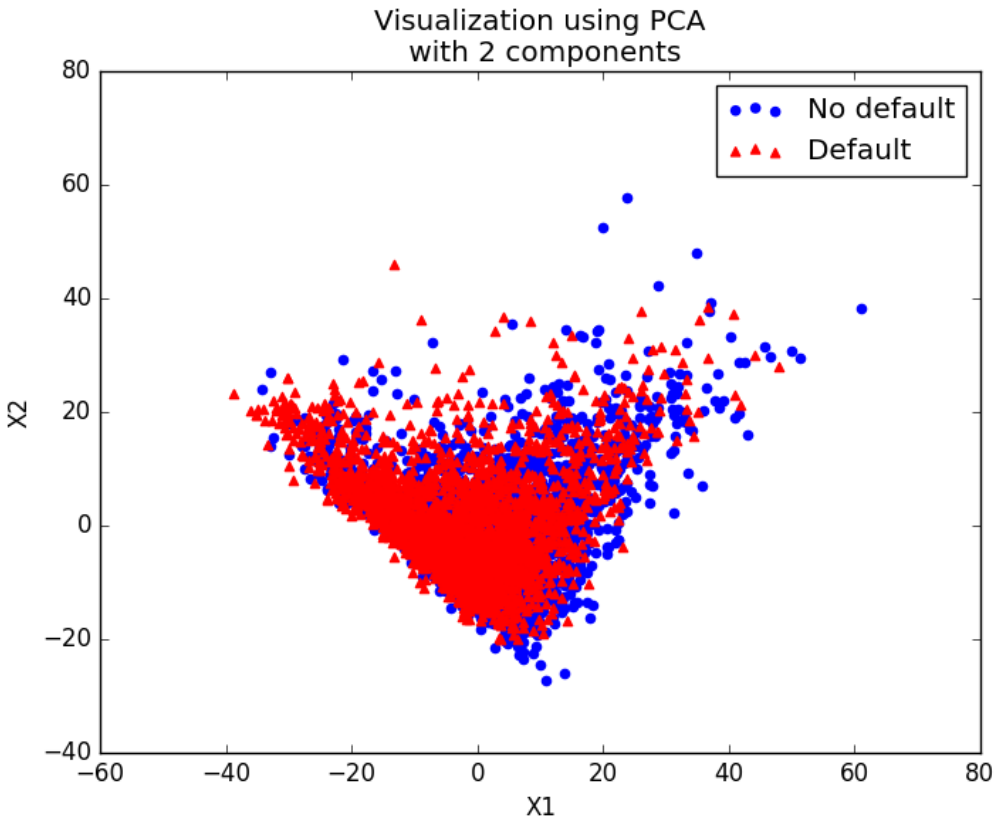

Figure 2: Visualization of dataset with dimensionality reduction from 769 to 2 using PCA

5 Experiment 2: Classification using Binary Support Vector Classifier

5.1 Dataset

This experiment makes use of the first 5000 training examples in the file train.csv. Out of this 5000 examples, 2500 are positive examples and 2500 negative examples. 5000 examples from test.csv file are used as the testing set, containing both positive and negative examples.

5.2 Setup and Hyper-parameter selection

K-fold cross validation is used to select the hyper-parameters. I use $K = 5$ since that is the standard value used. Before starting the training and cross-validation, each feature is normalized using standard score normalization. Then the Binary Support Vector Classifier is trained using the 5000 training samples, and K-fold cross validation is applied. I perform parameter tuning for the following parameters:
1. Percentile of features to be used (using mutual information criterion). The following percentiles are used, out of which the best one is selected: $(10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 100)$
2. Kernel to be used (Radial basis kernel, Polynomial kernel, Sigmoid kernel)
3. Kernel coefficient to be used.
4. Regularization parameter to be used (C). This corresponds to the number of points allowed to be classified by the classifier in the training set, to have a smooth seperating hyperplane and to generalize well.
The selection of these hyper-parameters is done using greedy search, in the sequence of the list given above. So the percentile of features is selected first, then the kernel and then kernel coefficient and so on. Accuracy of prediction is used to distinguish between two classifiers (rounded to 2 decimals). A better alternative would be run to Grid Search for hyper-parameter selection, trying out all possible combinations of the four hyper-parameters. Grid Search is not pursued due to

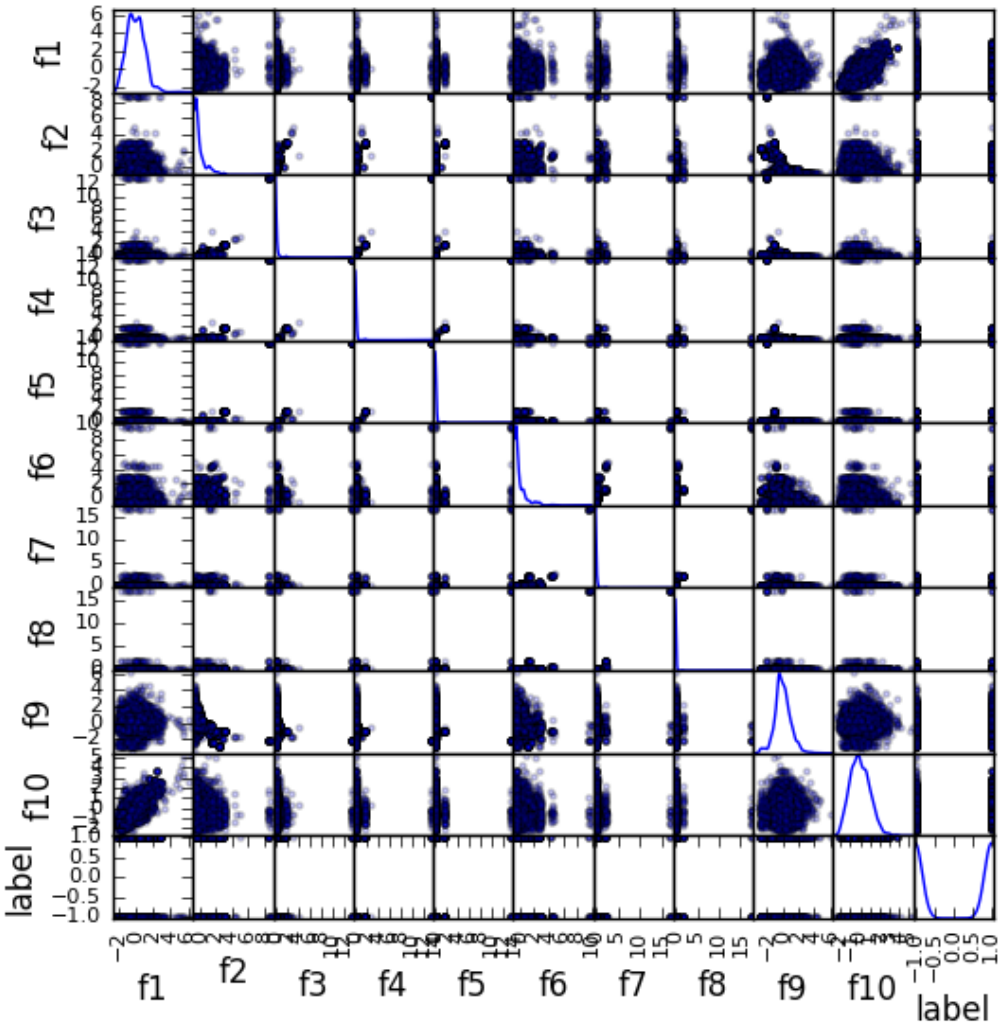

Figure 3: Scatter plot matrix of top 10 correlated features to the label along with the label. Diagonal elements corresponds to distribution of values.

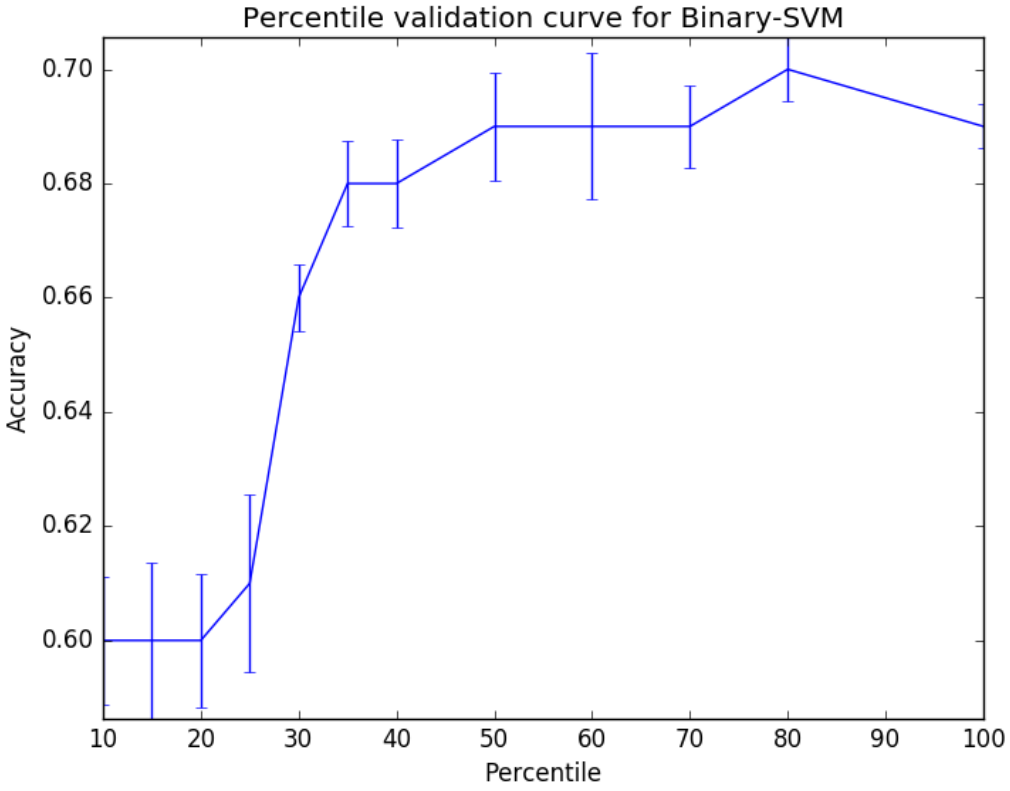

Figure 4: Percentile validation curve for Binary Support Vector Classifier

constraints of computation.

5.3 Cross Validation Results

Figure 4 below shows variation of accuracy as the percentile of features selected for classification task is varied. Figure 5 shows variation of accuracy of prediction with respect to different kernels (with percentile of features selected in the previous step). Figure 6 depicts variation of accuracy of prediction with respect to the kernel coefficient (using percentile of features and kernel selected in previous steps). Figure 7 depicts variation of accuracy of prediction with respect to the regularization parameter value (C), using the hyper-parameters from previous steps.

5.4 Results

Table 1 shows the performance of the classifier against an unseen test set.

6 Experiment 3: Classification using One Class Support Vector Classifier

6.1 Dataset

This experiment makes use of the first 5000 positive training examples in the file train.csv. This experiment frames the loan default prediction problem as an anomaly detection problem. 5000 examples from test.csv file are used as the testing set, containing both positive and negative examples.

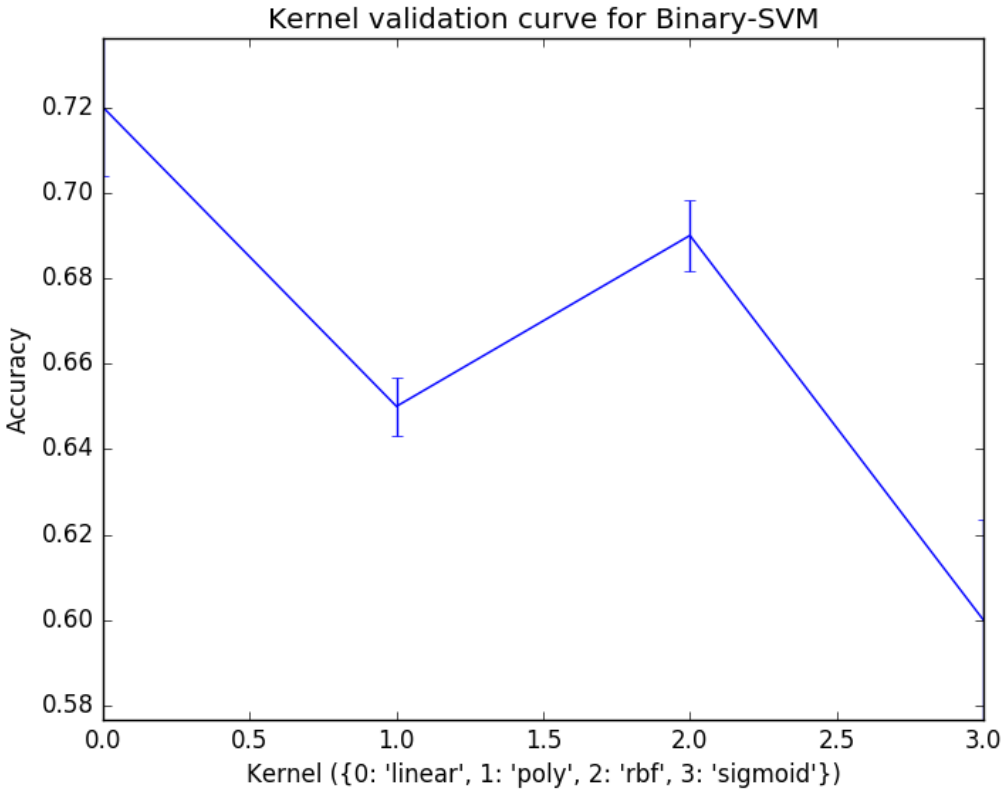

Figure 5: Kernel validation curve for Binary Support Vector Classifier

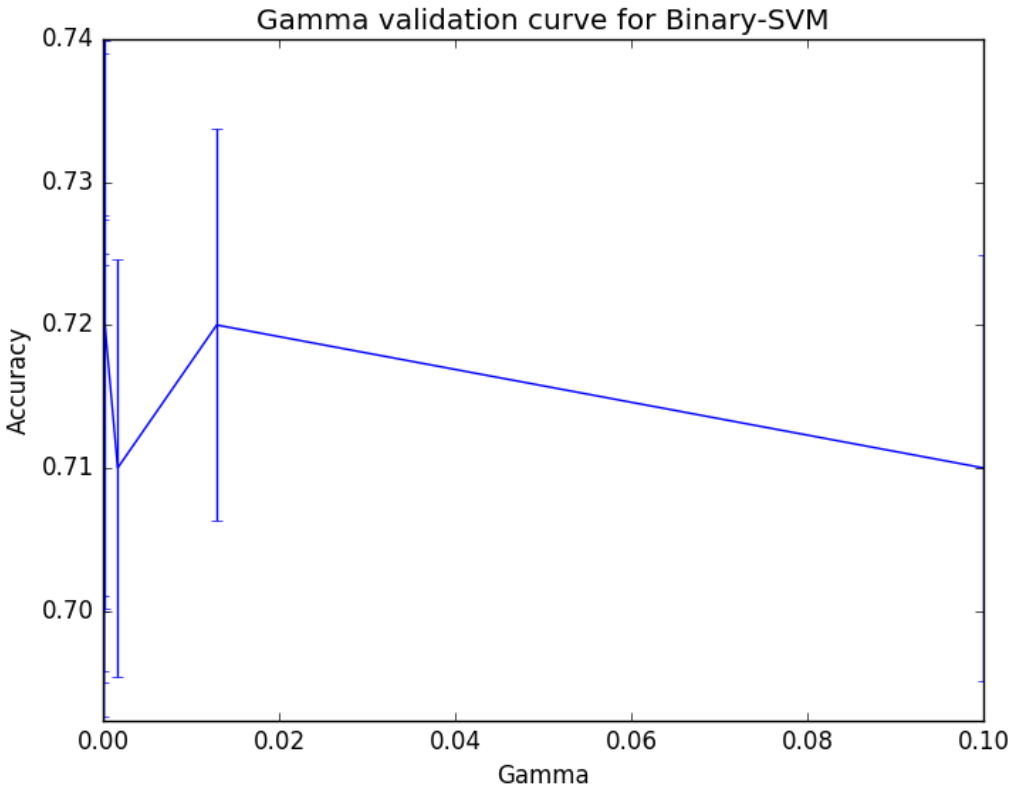

Figure 6: Kernel coefficient validation curve for Binary Support Vector Classifier

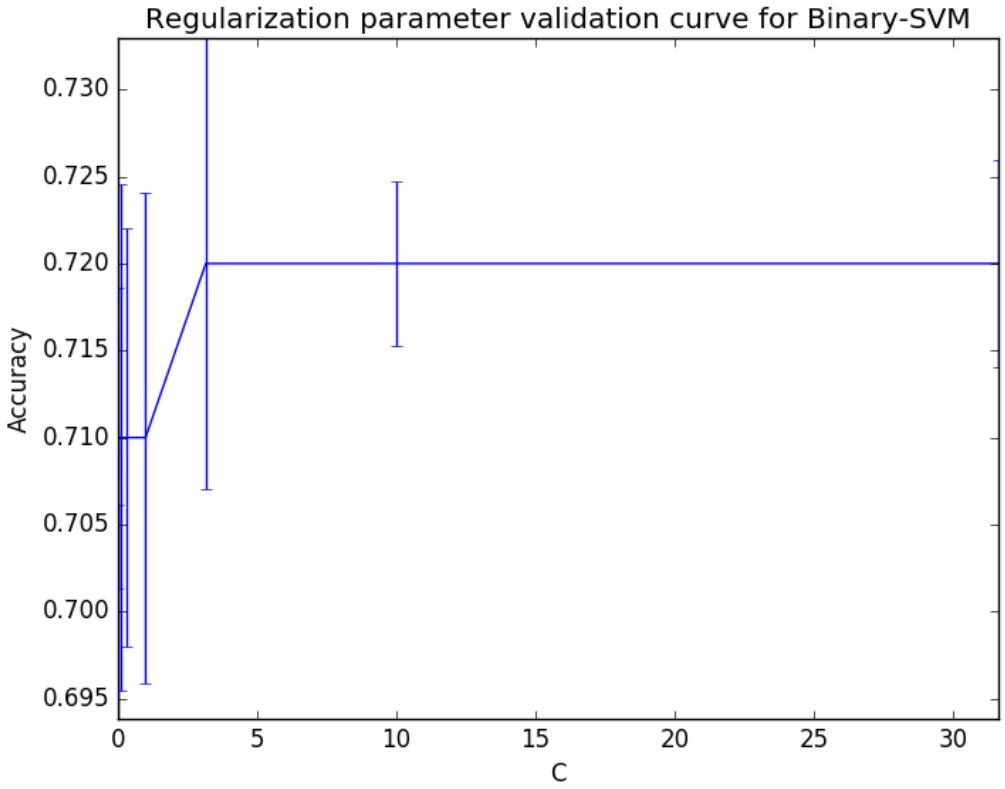

Figure 7: Regularization parameter validation curve for Binary Support Vector Classifier

Table 1: Classification Results for Binary Support Vector Classifier

Best Percentile of features	80
Best Kernel	linear
Best Gamma	0.000000001
Best Regularization Parameter	3.1622776602
Number of test examples	1000
True Positives	338
True Negatives	81
False Positives	24
False Negatives	557
Accuracy	0.42
Precision	0.93
Recall	0.38
F1 Score	0.54

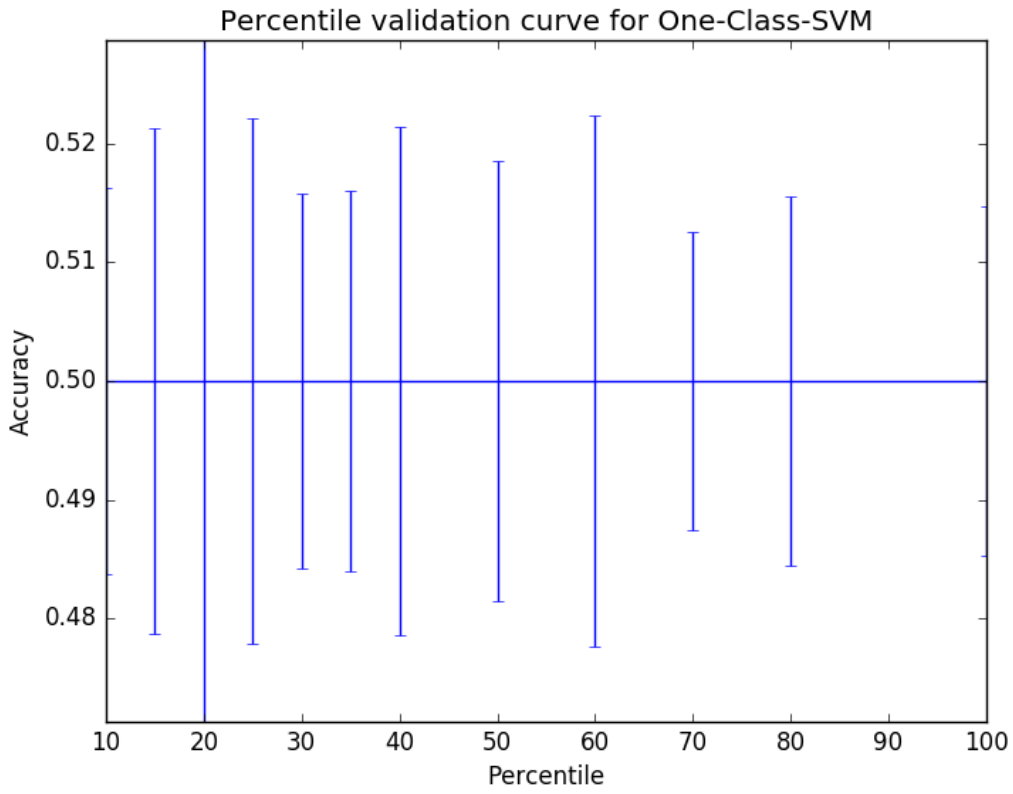

Figure 8: Percentile validation curve for One-Class Support Vector Classifier

6.2 Setup and Hyper-parameter selection

K-fold cross validation is used to select the hyper-parameters. I use $K = 5$ since that is the standard value used. Before starting the training and cross-validation, each feature is normalized using standard score normalization. Then the One-Class Support Vector Classifier is trained using the 5000 training samples, and K-fold cross validation is applied. I perform parameter tuning for the following parameters:
1. Percentile of features to be used (using mutual information criterion). The following percentiles are used, out of which the best one is selected: $(10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 100)$
2. Kernel to be used (Radial basis kernel, Polynomial kernel, Sigmoid kernel)
3. Kernel coefficient to be used.
4. Regularization parameter to be used (nu). This corresponds to fraction of training examples allowed to be mis-classified.
The selection of these hyper-parameters is done using greedy search, in the sequence of the list given above. Accuracy of prediction is used to distinguish between two classifiers (rounded to 2 decimals).

6.3 Cross Validation Results

Figure 8 below shows variation of accuracy as the percentile of features selected for classification task is varied. Figure 9 shows variation of accuracy of prediction with respect to different kernels (with percentile of features selected in the previous step). Figure 10 depicts variation of accuracy of prediction with respect to the kernel coefficient (using percentile of features and kernel selected in previous steps). Figure 11 depicts variation of accuracy of prediction with respect to the regularization parameter value (C), using the hyper-parameter values from previous steps.

6.4 Results

Table 1 shows the performance of the classifier against an unseen test set.

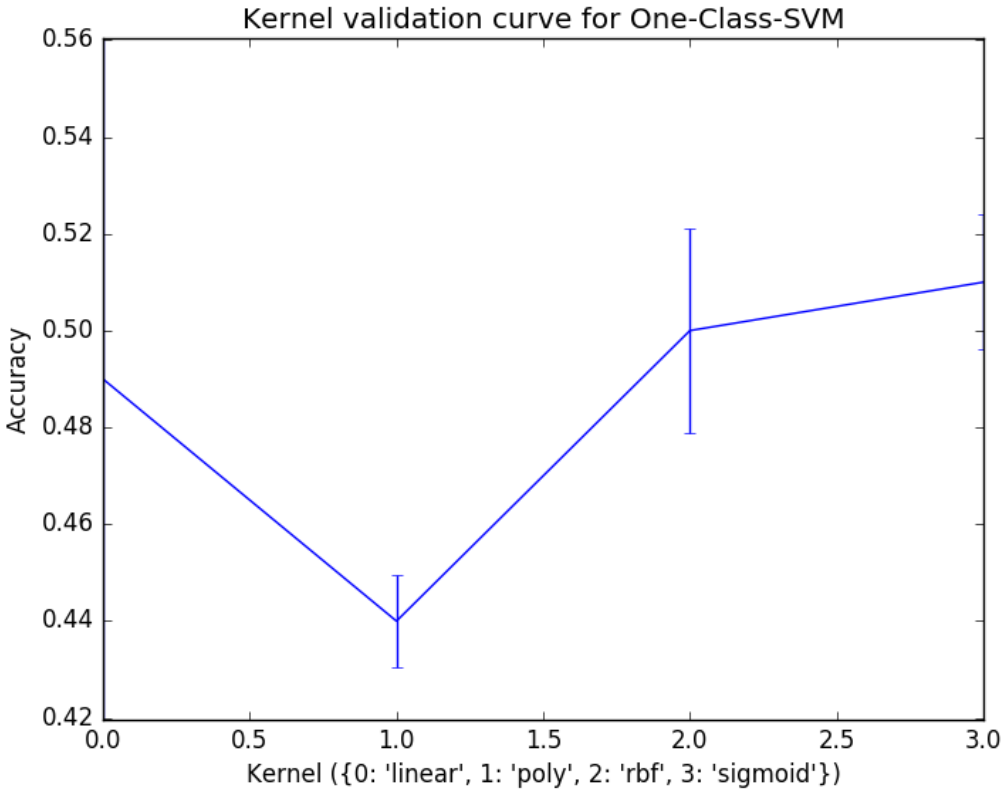

Figure 9: Kernel validation curve for One-Class Support Vector Classifier

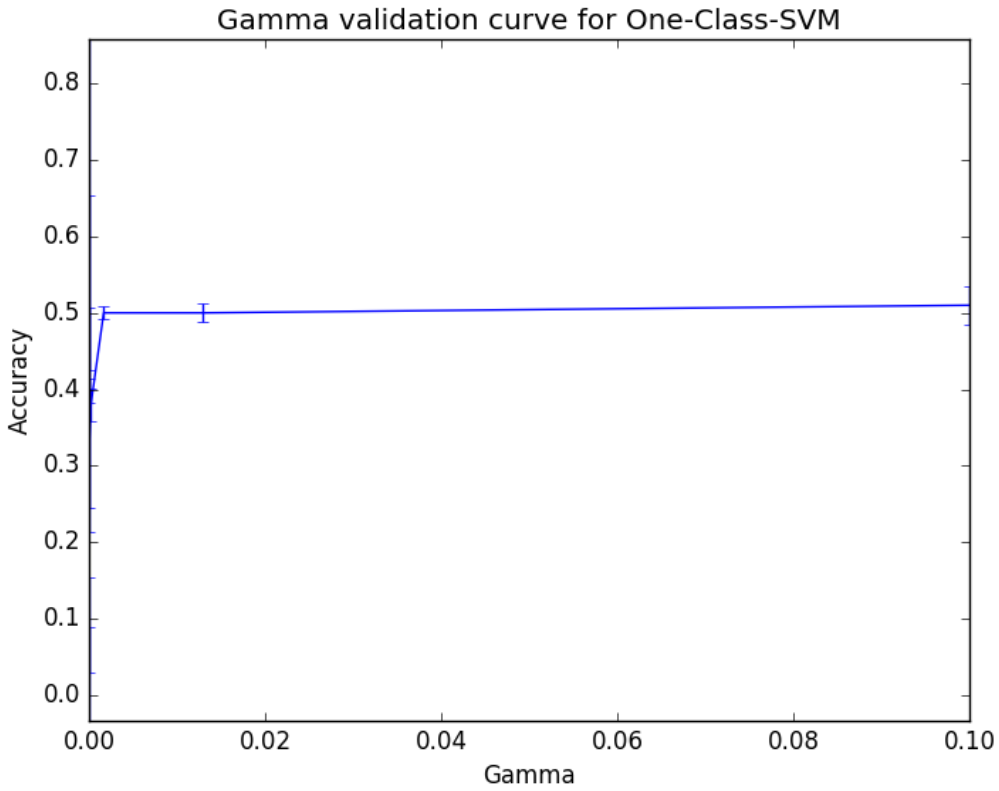

Figure 10: Kernel coefficient validation curve for One-Class Support Vector Classifier

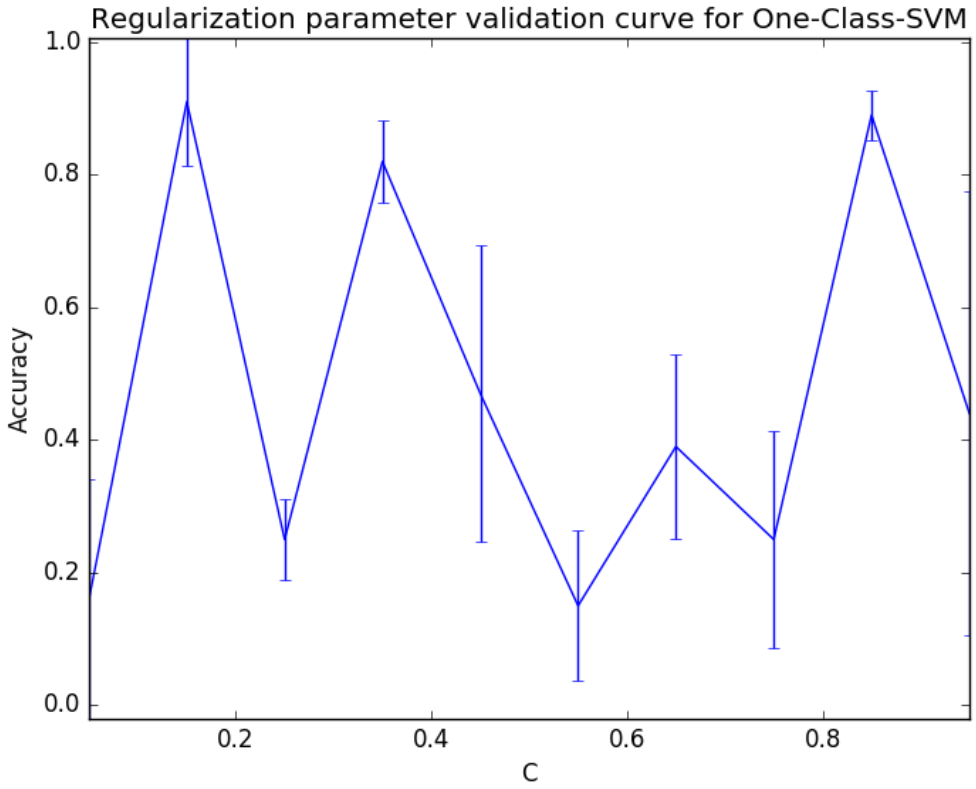

Figure 11: Regularization parameter validation curve for One-Class Support Vector Classifier

Table 2: Classification Results for One-Class Support Vector Classifier

Best Percentile of features	80
Best Kernel	linear
Best Gamma	0.000000001
Best Regularization Parameter	3.1622776602
Number of test examples	1000
True Positives	338
True Negatives	81
False Positives	24
False Negatives	557
Accuracy	0.42
Precision	0.93
Recall	0.38
F1 Score	0.54

Table 3: Classification Results for One-Class Support Vector Classifier

Best Percentile of features	80
Best Kernel	linear
Best Gamma	0.000000001
Best Regularization Parameter	3.1622776602
Number of test examples	1000
True Positives	338
True Negatives	81
False Positives	24
False Negatives	557
Accuracy	0.42
Precision	0.93
Recall	0.38
F1 Score	0.54

7 Experiment 4: Classification using Gradient Boosting Classifier

7.1 Dataset

This experiment makes use of first 5000 training examples in the file train.csv. 2500 of them are positive and rest negative. 5000 examples from test.csv file are used as the testing set, containing both positive and negative examples.

7.2 Setup and Hyperparameter selection

K-fold cross validation is used to select the hyperparameters. I use $K = 5$ since that is the standard value used. Before starting the training and cross-validation, each feature is normalized using standard score normalization. Then the Gradient Boosting Classifier is trained using the 5000 training samples, and K-fold cross validation is applied. I perform parameter tuning for the following parameters:
1. Percentile of features to be used (using mutual information criterion). The following percentiles are used, of which the best one is selected: $(10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 100)$
2. Number of estimators to be used (from 10 to 290 in steps of 10).
3. Maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. (Ranges from 1 to 6)
4. Learning rate (ranges from 0.1 to 1.1 in steps of 0.1).
The selection of these hyper-parameters is done using greedy search, in the sequence of list given above. Accuracy of prediction is used to distinguish between two classifiers (rounded to 2 decimals).

7.3 Cross Validation Results

Figure 12 below shows variation of accuracy as the percentile of features selected for classification task is varied. Figure 13 shows variation of accuracy of prediction with respect to different number of estimators (with percentile of features selected in the previous step). Figure 14 depicts variation of accuracy of prediction with respect to the depth of individual regression estimators (using percentile of features and number of estimators selected in previous steps). Figure 15 depicts variation of accuracy of prediction with respect to the learning rate, using the hyper-parameters from previous steps.

7.4 Results

Table 3 shows performance of the classifier against an unseen test set.

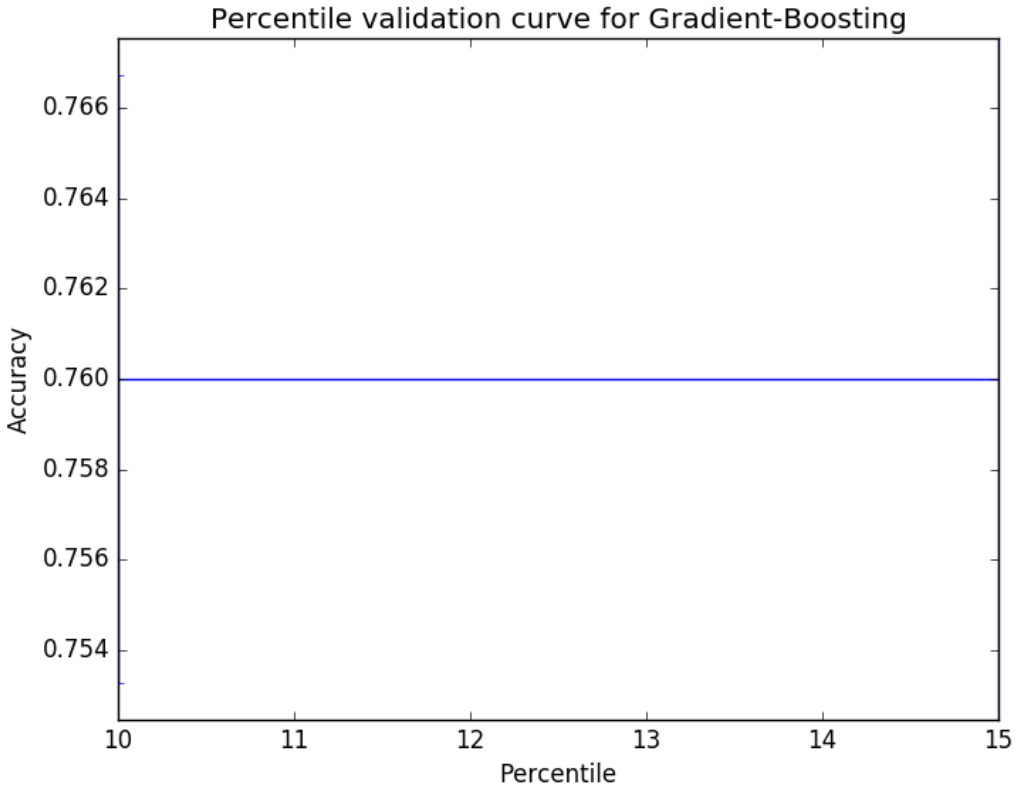

Figure 12: Percentile validation curve for Gradient Boosting Classifier

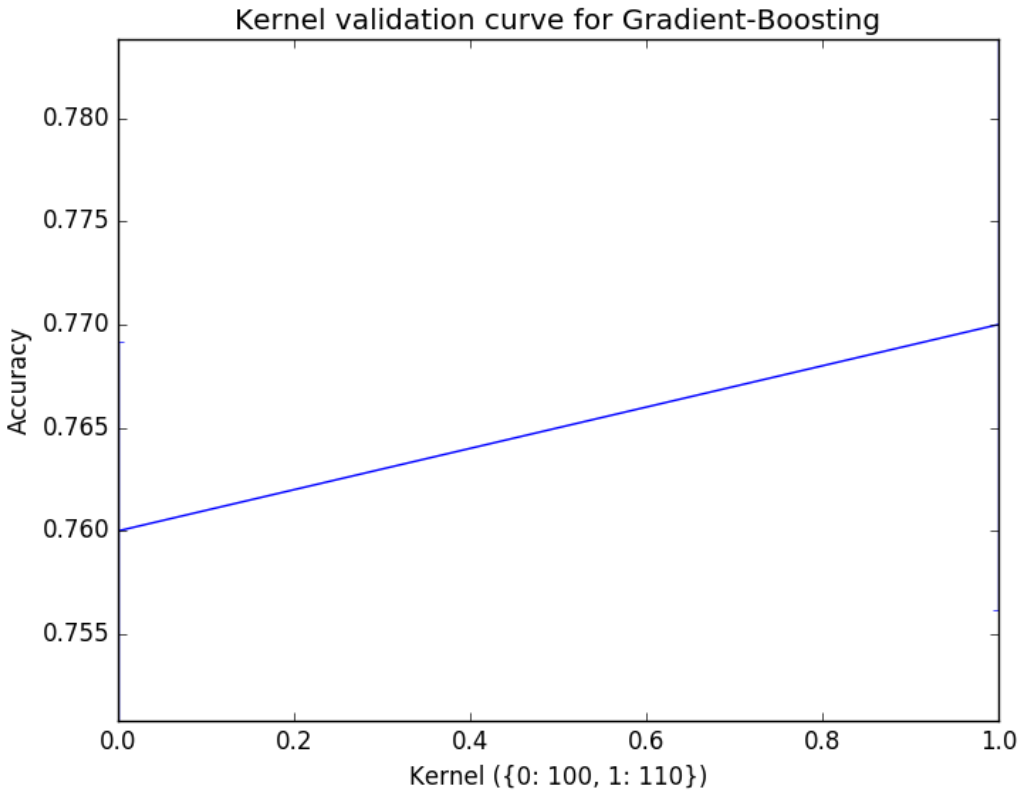

Figure 13: Number of estimators validation curve for Gradient Boosting Classifier

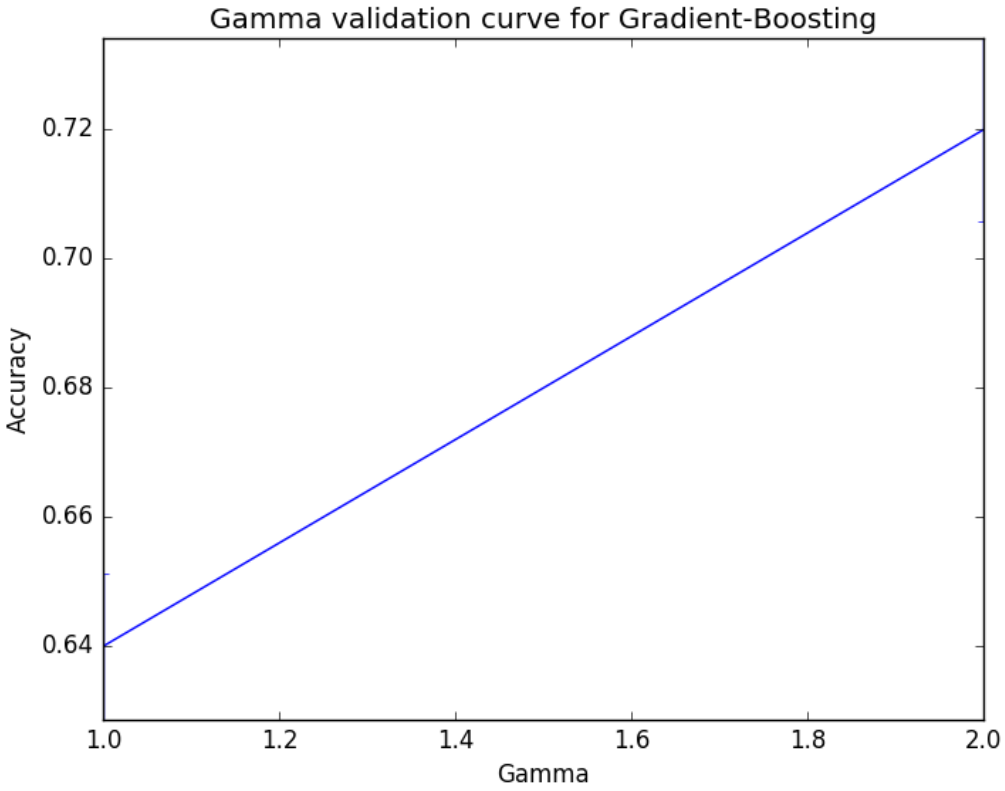

Figure 14: Max depth of individual estimators validation curve for Gradient Boosting Classifier

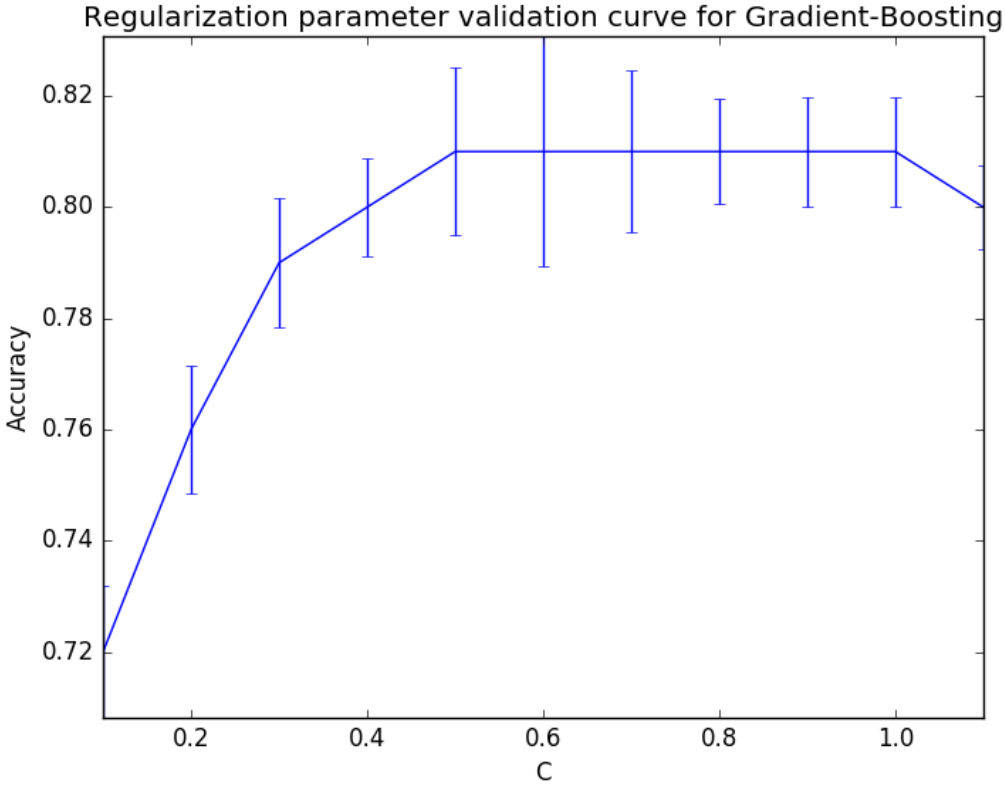

Figure 15: Learning rate validation curve for Gradient Boosting Classifier

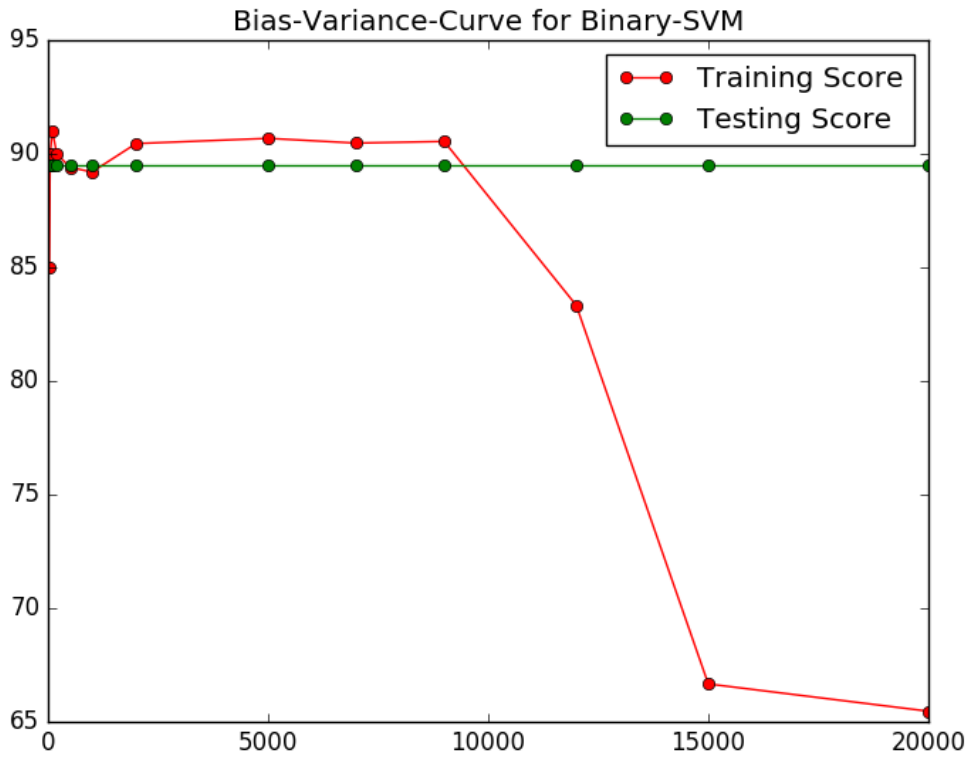

Figure 16: Learning curve for Binary Support Vector Classifier

8 Experiment 5: Bias-Variance/Learning Curves

In this experiment, I draw the bias variance curve for each model tried in the project (Binary SVC, One-Class SVC and Gradient Boosting classifier)

8.1 Dataset

This experiment makes use of the first 20000 training examples from the file train.csv. For One-Class SVM, all of them are positive. For the other two models, half of them are negative and half positive. 5000 testing examples are used from the file test.csv as the testing set.

8.2 Setup and Hyperparameter selection

The hyper-parameters for classification are fixed as obtained in the last 3 experiments. The classifier is trained for each of the following subsets of the training dataset: $[20, 50, 100, 200, 500, 1000, 2000, 5000, 7000, 9000, 1$
The testing accuracy is then calculated for the 5000 testing samples for each of these classifiers.

8.3 Results

Figure 16 shows the learning curve for Binary Support Vector Classifier. Figure 17 shows the learning curve for One-Class Support Vector Classifier. Figure 18 shows the learning curve for Gradient Boosting Classifier.

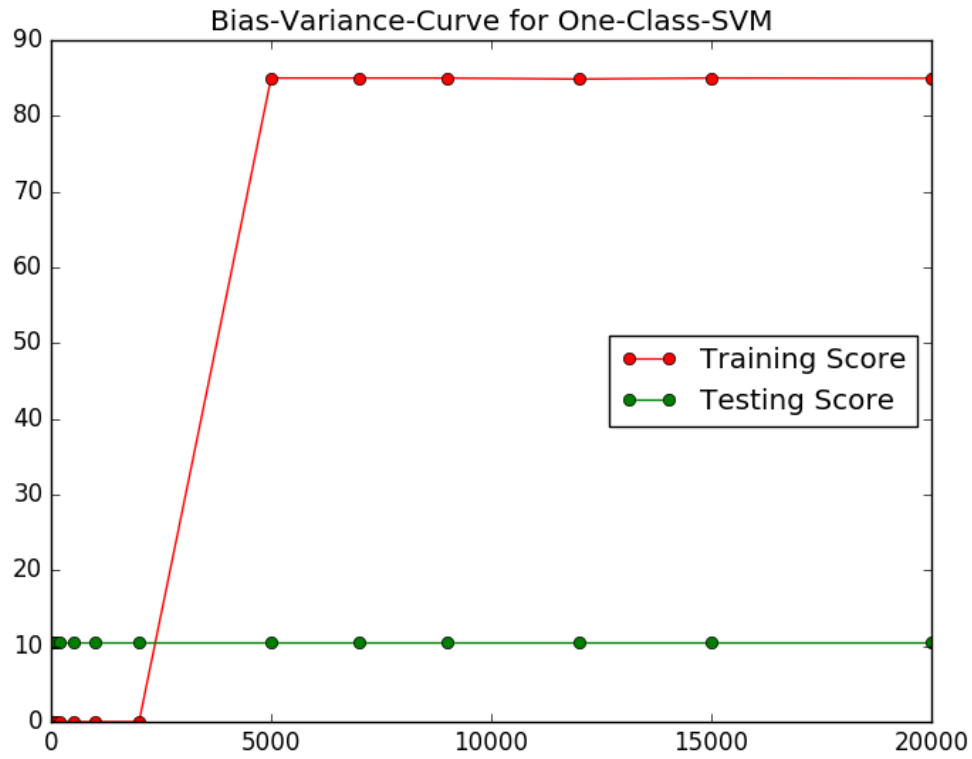

Figure 17: Learning curve for One-Class Support Vector Classifier

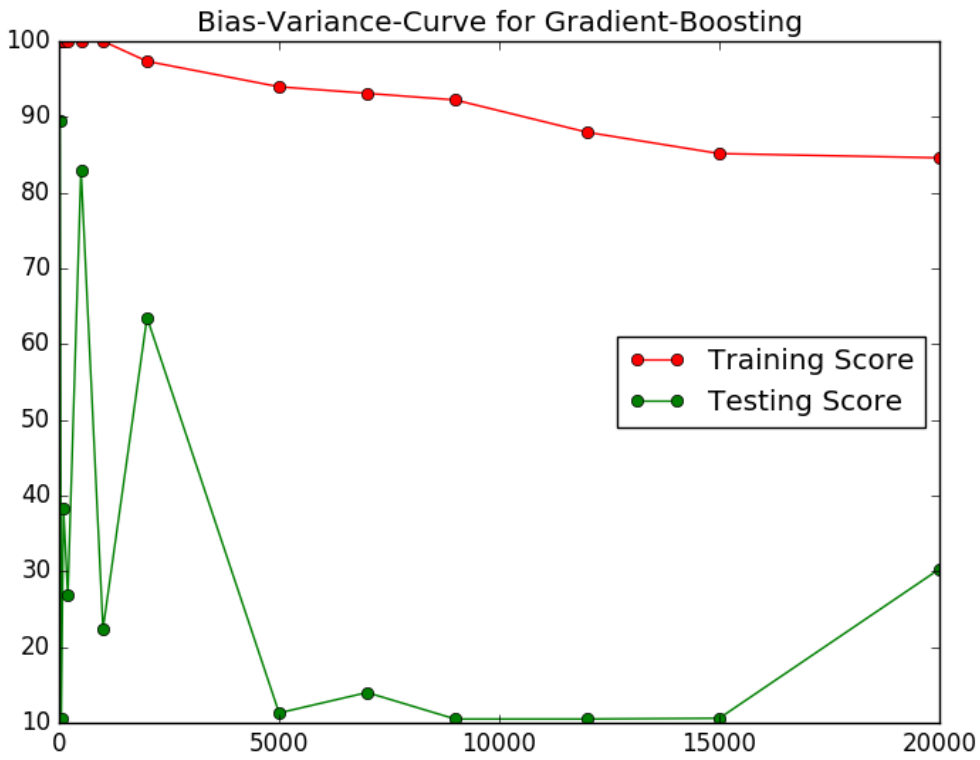

Figure 18: Learning curve for Gradient Boosting Classifier