University of New Orleans
Department of Computer Science

FALL 2019: CSCI 6522
Programming Assignment # 3

Machine Learning - II

Submitted to:
Professor Dr. Tamjidul Hoque

By
Name: **Astha Sharma**
Student ID: **2564173**

Describe Each Classifier in 5-10 sentences

1. **Support Vector Classifier (SMO):** SVM supports binary class classification and can be extended to support multi-class classification and regression problems. The algorithm works by finding a line that best separates the data into the two groups. The classification is done by finding a line that ideally separates data into identical groups. It is achieved by using an optimization process that considers only those data instances (support vectors) in the training dataset which are closest to the line that best separates the given classes. Ideally, a line cannot be drawn to exactly separate the classes, and therefore, a margin is added around the line to relax the constraint, allowing some instances to be misclassified but allowing a considerably better overall result.

2. **Logistic Regression Classifier:** This classifier assumes that the input variables have a Gaussian distribution. However, it will still give good results even if the data is not Gaussian. The algorithm for this classifier learns a coefficient for each of the input values, which are linearly combined into a regression function and then transformed using a logistic function. Logistic regression is fast and simple and also can be very effective on some problems.

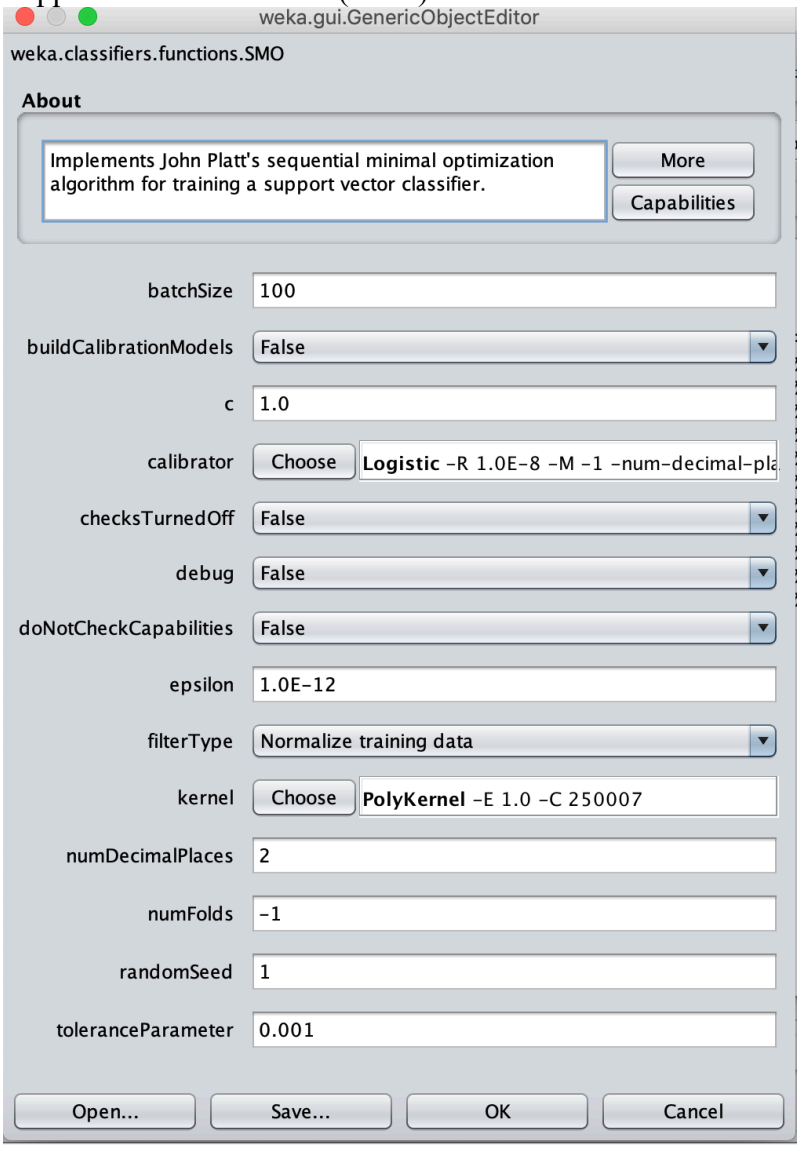
3. **KNN Classifier:** The k-nearest neighbor algorithm can be used for both classification and regression problems. The algorithm works by storing the entire training dataset and then finding the **k** most-similar training patterns as the prediction is made. The similarity can be measured in terms of distance (typically, the Euclidean distance) between the data instances. Overall, it is a very simple algorithm and often achieves very good performance. In classification problem, as it is used to make prediction, KNN takes the mode of the k most similar instances in the training dataset.

4. **Bagging Classifier:** Bagging, also known as Bootstrap Aggregation, is an ensemble method that creates different samples of the training dataset and creates a unique classifier for each of those samples. The result from these multiple classifiers are then combined based on say, average value or voting. The thing to remember here is that each sample of the training dataset is different which in-turn gives different (trained) classifier and definitely a different focus and perspective on the problem.

5. **Random Forest Classifier:** Random Forest is an extension of bagging. In case of a bagging, the decision trees are created using a greedy algorithm that selects the best split point at each step of the tree building process. Having said that, the resulting trees end up looking very similar to each other and therefore, reduces the variance of the predictions from all the bags. This, in turn, hurts the robustness of the predictions made. Random Forest is an improvement of bagged decision trees that disrupts the greedy splitting algorithm for tree creation. Here, the split points can only be selected from a random subset of the input attributes. This is indeed a simple change but makes a great deal in decreasing the similarity between the bagged trees and the associated final predictions.

Describe the parameters/hyper-parameters chosen to train each classifier:

1. Support Vector Classifier (SVM)

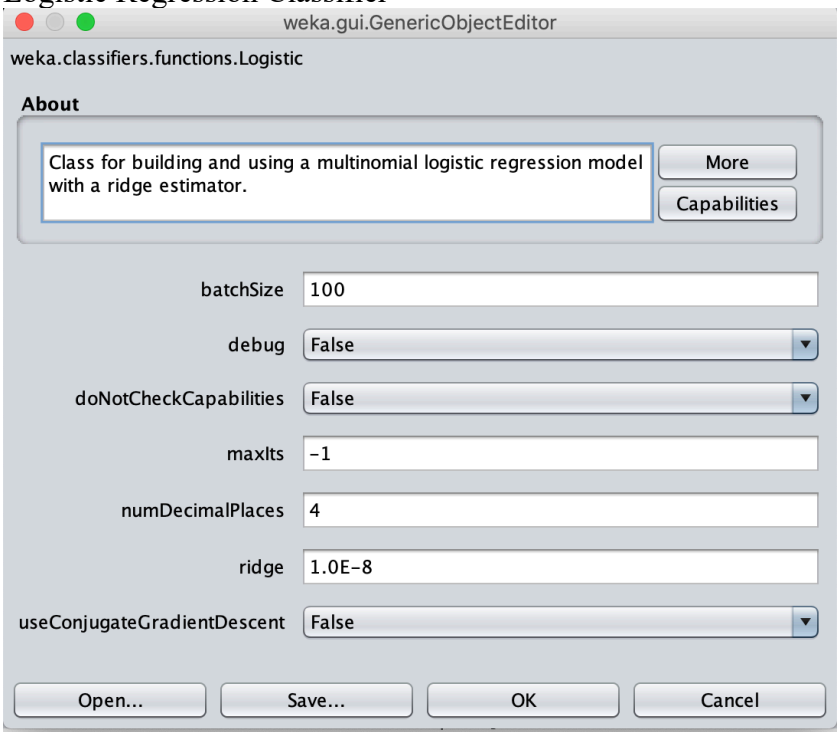


Parameters Involved:

buildCalibrationModels: Takes True or False value. Determines whether to fit calibration models to the SVM's output.
numFolds: Indicates the number of folds for cross-validation that is used to generate training data for calibration models (-1 means use training data).
randomSeed: Indicates a random number used as a seed for the cross-validation.
C: Denotes the complexity parameter C.

numDecimalPlaces: The number of decimal places to display in the output.
batchSize: The preferred number of instances to process if batch prediction is being performed. The number of instances may vary depending on the requirement.
kernel: Denotes the kernel to use.
checksTurnedOff: Turns time-consuming checks off.
debug: If set to true, classifier may provide additional info via console.
filterType: Determines how/if the data will be transformed.
toleranceParameter: The tolerance parameter (should not be altered).
calibrator: Indicates the calibration method to use. Using, Logistic Calibrator.
doNotCheckCapabilities: If set, classifier capabilities are not checked before classifier is built
epsilon: The epsilon for round-off error (should not be altered).

2. Logistic Regression Classifier

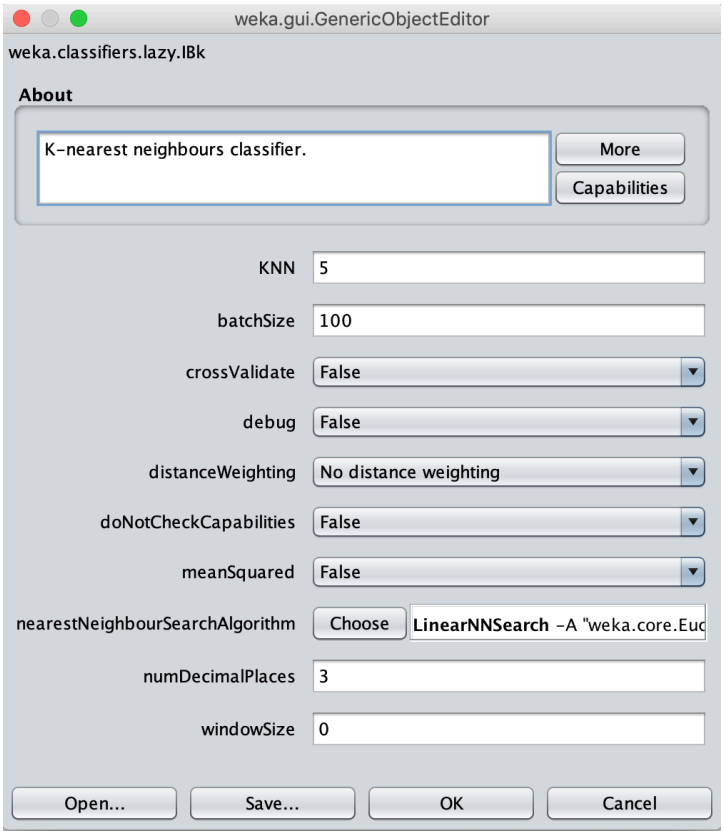


Parameters:

Ridge: Defines how much pressure to put on the algorithm to reduce the size of the coefficients. Setting a 0 to it will turn off the regularization.
numDecimalPlaces: Defines the number of decimal values for output
batchSize: Preferred number of instances to process, if batch prediction is being performed
debug: If set to true, classifier may output additional info to the console.
doNotCheckCapabilities: If set, classifier capabilities are not checked before classifier is built (Use with caution to reduce runtime).
maxIts: Maximum number of iterations to perform.

useConjugateGradientDescent: Give possibility to switch between the default BFGS and Conjugate Gradient Descent

3. KNN Classifier

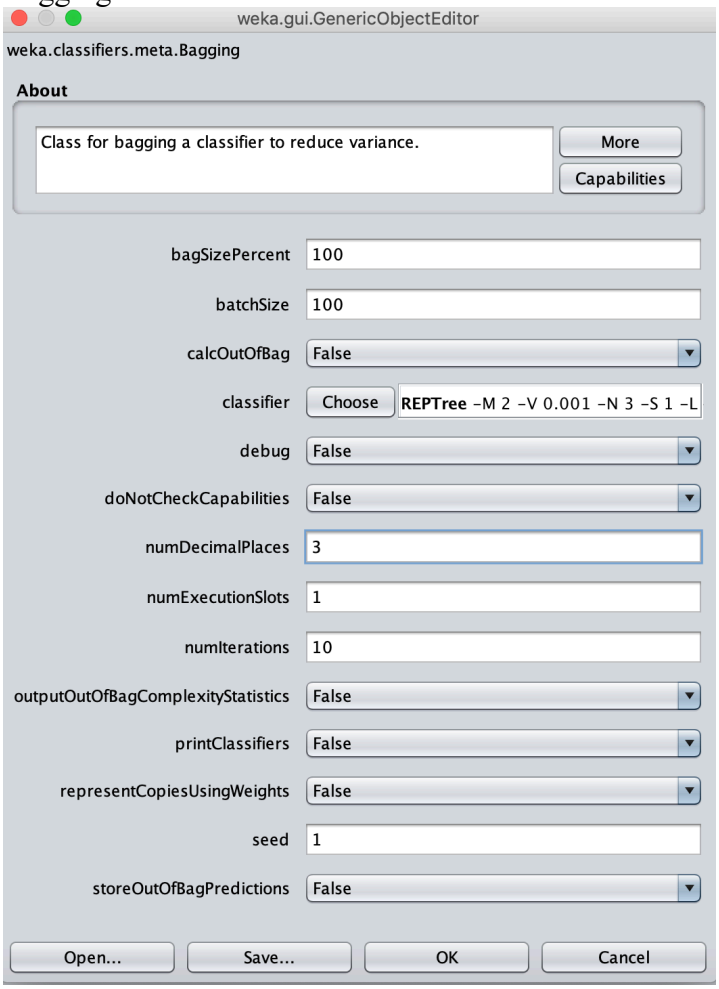


Parameters

numDecimalPlaces: The number of decimal places to be used for the output of numbers in the model.
batchSize: The preferred number of instances to process if batch prediction is being performed. More or fewer instances may be provided, but this gives implementations a chance to specify a preferred batch size.
KNN: The number of neighbours to use.
distanceWeighting: Gets the distance weighting method used.
nearestNeighbourSearchAlgorithm: The nearest neighbour search algorithm to use (Default: weka.core.neighboursearch.LinearNNSearch).
debug: If set to true, classifier may output additional info to the console.
windowSize: Gets the maximum number of instances allowed in the training pool. The addition of new instances above this value will result in old instances being removed. A value of 0 signifies no limit to the number of training instances.
doNotCheckCapabilities: If set, classifier capabilities are not checked before classifier is built (Use with caution to reduce runtime).

meanSquared: Whether the mean squared error is used rather than mean absolute error when doing cross-validation for regression problems.
crossValidate: Whether hold-one-out cross-validation will be used to select the best k value between 1 and the value specified as the KNN parameter.

4. Bagging Classifier

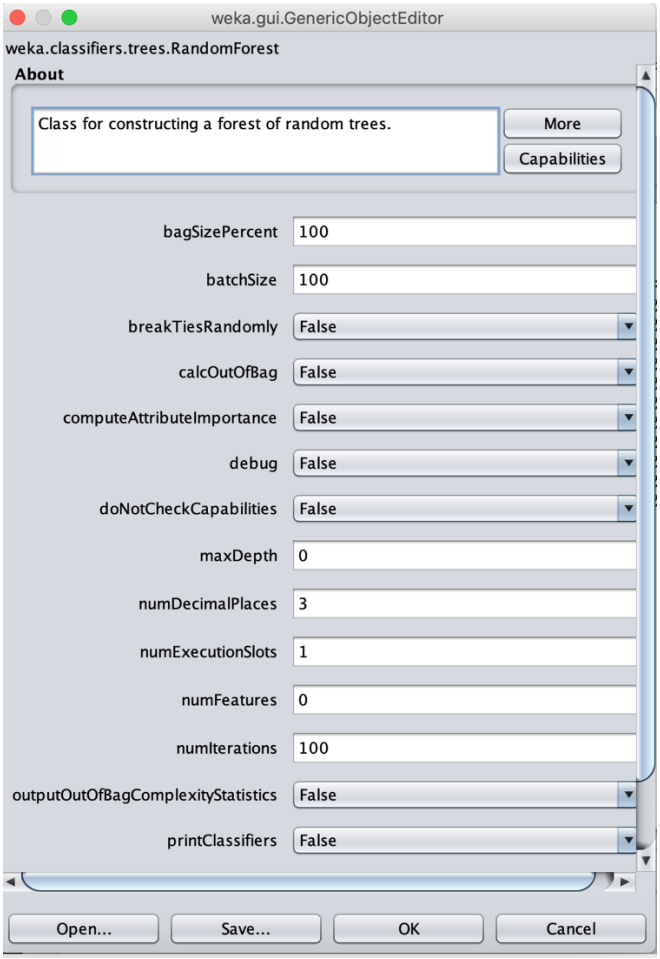


Parameters

bagSizePercentage: Size of each bag as a percentage of the training set size
batchSize: Preferred number of instances to process, if batch prediction is being performed
calacOutOfBag: Whether to enable the calculation of out-of-bag error
classifier: The base classifier to be used – using: Fast Decision Tree Learner
debug: If set to true, classifier may output additional info to the console.
doNotCheckCapabilities: If set, classifier capabilities are not checked before classifier is built (Use with caution to reduce runtime).
numDecimalPlaces: The number of decimal places to be used for the output of numbers in the model.
numIterations: Number of iterations to be performed

outputOutOfBagComplexityStatistics: Whether to output complexity-based statistics when out-of-bag evaluation is performed
printClassifiers: Whether to print the individual classifiers in the output
representCopiesUsingWeights: Whether to represent copies of instances using weights
Seed: The random number seed to be used.
storageOutOfBagPrediction: Whether to store the out-of-bag prediction

5. Random Forest Classifier



Parameter

Seed: The random number seed to be used.
representCopiesUsingWeights: Whether to represent copies of instances using weights rather than explicitly.
storeOutOfBagPredictions: Whether to store the out-of-bag predictions.
numExecutionSlots: The number of execution slots (threads) to use for constructing the ensemble. **bagSizePercent**: Size of each bag, as a percentage of the training set size.
numDecimalPlaces: The number of decimal places to be used for the output of numbers in the model.

batchSize: The preferred number of instances to process if batch prediction is being performed. More or fewer instances may be provided, but this gives implementations a chance to specify a preferred batch size.
numIterations: The number of iterations to be performed.
Debug: If set to true, classifier may output additional info to the console.
outputOutOfBagComplexityStatistics: Whether to output complexity-based statistics when out- of-bag evaluation is performed.
Classifier: The base classifier to be used.
breakTiesRandomly: Break ties randomly when several attributes look equally good.
doNotCheckCapabilities: If set, classifier capabilities are not checked before classifier is built (Use with caution to reduce runtime).
maxDepth: The maximum depth of the tree, 0 for unlimited.
computeAttributeImportance: Compute attribute importance via mean impurity decrease **calcOutOfBag:** Whether the out-of-bag error is calculated.
numFeatures: Sets the number of randomly chosen attributes. If 0, int(log_2(#predictors) + 1) is used.

Define and describe following terms for measuring performance of a classifier:

1. True Positive (TP)
   True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)

2. False Positive (FP)
   False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one. (1)

3. Precision
   Precision is also called Positive Predictive Value (PPV). It gives how many classifications did we get right based upon the number of attempts we made. P = TP / (TP + FP)

4. Recall
   Recall is also called True Positive Rate (TPR) or Sensitivity. It gives how many classifications did we get right compared to the total number of correct classifications we could have made. R = TP / (TP + FN)

5. F-Measure
   F-Measure combines Precision (P) and Recall (R) into one measure that can be used to evaluate the overall performance of the classifier, rather than having to evaluate the trade-offs between precision and recall. It is the harmonic mean of P and R. It tests accuracy. F-

Measure = (2 x P x R) / (P + R)

6. Receiver Operating Characteristic (ROC) Area
   Receiver Operating Characteristics (ROC) Area is also called ROC curve. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The ROC curve is a fundamental tool for diagnostic test evaluation. It is widely used as a evaluation measure for binary classification models. The main purpose of ROC curve analysis is to provide tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution.

7. Confusion Matrix
   A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. In confusion matrix, the number of correct and incorrect predictions are summarized with count values and broken down by each class.

Performance of each classifier in terms of:

1. Correctly Classified Instances:

a. SVM:**95.8168 %**
b. Logistic: **89.0687 %**
c. KNN: **96.3791 %**
d. Bagging: **91.0575 %**
e. Random Forest: **96.7357 %**

2. TP Rate, FP Rate, Precision, Recall, F-Measure, ROC Area:

a. SVM:

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.993    0.008    0.961      0.993    0.977      0.973    0.998     0.992     0
                0.998    0.005    0.967      0.998    0.982      0.980    0.999     0.988     1
                0.948    0.002    0.979      0.948    0.963      0.959    0.993     0.983     2
                0.965    0.005    0.955      0.965    0.960      0.956    0.994     0.980     3
                0.929    0.004    0.957      0.929    0.943      0.938    0.991     0.978     4
                0.933    0.002    0.976      0.933    0.954      0.951    0.991     0.978     5
                0.974    0.002    0.976      0.974    0.975      0.973    0.996     0.988     6
                0.969    0.004    0.956      0.969    0.962      0.959    0.996     0.976     7
                0.917    0.002    0.976      0.917    0.946      0.942    0.988     0.970     8
                0.957    0.006    0.939      0.957    0.948      0.943    0.996     0.979     9
Weighted Avg.   0.964    0.004    0.964      0.964    0.964      0.960    0.995     0.983
```

b. Logistic:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.941    0.009    0.953      0.941    0.947      0.937    0.992     0.974     0
                 0.987    0.008    0.951      0.987    0.969      0.964    0.998     0.983     1
                 0.807    0.011    0.890      0.807    0.846      0.832    0.964     0.884     2
                 0.871    0.013    0.869      0.871    0.870      0.857    0.980     0.904     3
                 0.842    0.016    0.839      0.842    0.841      0.825    0.974     0.851     4
                 0.833    0.013    0.843      0.833    0.838      0.825    0.976     0.858     5
                 0.920    0.010    0.900      0.920    0.910      0.901    0.991     0.952     6
                 0.927    0.015    0.857      0.927    0.891      0.880    0.985     0.919     7
                 0.784    0.015    0.810      0.784    0.797      0.781    0.965     0.830     8
                 0.885    0.011    0.886      0.885    0.886      0.875    0.986     0.899     9
Weighted Avg.    0.891    0.012    0.891      0.891    0.890      0.879    0.983     0.916
```

c. KNN:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.993    0.008    0.961      0.993    0.977      0.973    0.998     0.992     0
                 0.998    0.005    0.967      0.998    0.982      0.980    0.999     0.988     1
                 0.948    0.002    0.979      0.948    0.963      0.959    0.993     0.983     2
                 0.965    0.005    0.955      0.965    0.960      0.956    0.994     0.980     3
                 0.929    0.004    0.957      0.929    0.943      0.938    0.991     0.978     4
                 0.933    0.002    0.976      0.933    0.954      0.951    0.991     0.978     5
                 0.974    0.002    0.976      0.974    0.975      0.973    0.996     0.988     6
                 0.969    0.004    0.956      0.969    0.962      0.959    0.996     0.976     7
                 0.917    0.002    0.976      0.917    0.946      0.942    0.988     0.970     8
                 0.957    0.006    0.939      0.957    0.948      0.943    0.996     0.979     9
Weighted Avg.    0.964    0.004    0.964      0.964    0.964      0.960    0.995     0.983
```

d. Bagging:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.958    0.013    0.935      0.958    0.947      0.936    0.996     0.987     0
                 0.987    0.006    0.963      0.987    0.975      0.971    0.999     0.995     1
                 0.854    0.014    0.873      0.854    0.863      0.848    0.988     0.948     2
                 0.859    0.011    0.887      0.859    0.873      0.860    0.988     0.945     3
                 0.919    0.011    0.891      0.919    0.905      0.895    0.991     0.961     4
                 0.797    0.009    0.882      0.797    0.837      0.826    0.983     0.914     5
                 0.901    0.008    0.923      0.901    0.912      0.903    0.994     0.967     6
                 0.935    0.004    0.954      0.935    0.944      0.939    0.996     0.982     7
                 0.856    0.013    0.845      0.856    0.851      0.839    0.986     0.927     8
                 0.943    0.011    0.889      0.943    0.915      0.907    0.996     0.976     9
Weighted Avg.    0.911    0.010    0.910      0.911    0.910      0.901    0.992     0.965
```

e. Random Forest:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.989    0.004    0.978      0.989    0.983      0.980    1.000     0.998     0
                 0.997    0.001    0.991      0.997    0.994      0.993    1.000     0.998     1
                 0.973    0.006    0.947      0.973    0.960      0.955    0.999     0.993     2
                 0.951    0.004    0.959      0.951    0.955      0.951    0.998     0.986     3
                 0.965    0.006    0.937      0.965    0.951      0.946    0.997     0.988     4
                 0.919    0.002    0.970      0.919    0.944      0.940    0.996     0.981     5
                 0.971    0.002    0.979      0.971    0.975      0.973    0.999     0.993     6
                 0.960    0.002    0.979      0.960    0.969      0.967    0.998     0.992     7
                 0.934    0.003    0.964      0.934    0.948      0.945    0.998     0.981     8
                 0.967    0.005    0.953      0.967    0.960      0.956    0.999     0.990     9
Weighted Avg.    0.967    0.004    0.967      0.967    0.967      0.964    0.999     0.991
```

3. Confusion Matrix:

a. SVM

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j   <-- classified as
 1176    0    4    4    2    2    5    0    1    0 |    a = 0
    0 1001    1    0    0    0    0    1    2    0 |    b = 1
    5    1  694    6   12    2    2    3    6    0 |    c = 2
    5    0   10  616    1   14    0    3    9    0 |    d = 3
    3    3   11    0  623    2    4    1    1    4 |    e = 4
   17    1    6   17    3  500    7    0    2    3 |    f = 5
    9    2    8    0    1    4  640    0    0    0 |    g = 6
    0    0    3    1    6    1    0  627    2    5 |    h = 7
    6    4   14    8    6   10    0    3  491    0 |    i = 8
    0    0    0    2    9    2    0   12    1  618 |    j = 9
```

b. Logistic:

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j   <-- classified as
 1123    2    9   10    6    9   12   11    9    3 |    a = 0
    0  992    1    2    2    0    2    1    4    1 |    b = 1
   13    7  590   22   23    4   15   23   24   10 |    c = 2
    7    4   12  573    3   21    6    8   17    7 |    d = 3
    3   12    8    3  549   18   13   14   15   17 |    e = 4
   14    6    9   16   13  463   13    6    9    7 |    f = 5
   11    4   11    0   10    8  611    2    7    0 |    g = 6
    0    1    4    7   11    1    0  598    8   15 |    h = 7
    7   12   17   19   15   20    4   10  425   13 |    i = 8
    0    3    2    7   22    5    3   25    7  570 |    j = 9
```

c. KNN:

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j   <-- classified as
 1186    0    3    1    0    1    3    0    0    0 |    a = 0
    0 1003    1    0    0    0    0    1    0    0 |    b = 1
   10    6  693    4    3    0    1    9    4    1 |    c = 2
    4    1    2  635    1    6    0    1    7    1 |    d = 3
    2   15    3    0  606    0    4    1    0   21 |    e = 4
   11    0    3   11    2  519    7    0    0    3 |    f = 5
   13    1    1    0    2    0  647    0    0    0 |    g = 6
    0    4    1    0    5    0    0  625    0   10 |    h = 7
    6    6    1   13    4    6    1    4  497    4 |    i = 8
    2    1    0    1   10    0    0   13    1  616 |    j = 9
```

d. Bagging:

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j   <-- classified as
 1144    3   10    2    2    5   11    1   15    1 |    a = 0
    0  992    1    0    0    1    2    1    5    3 |    b = 1
   19    3  624   19   18    4   10    5   19   10 |    c = 2
   10    0   13  565    2   33    1    5   17   12 |    d = 3
    2   10   13    0  599    2    8    0    4   14 |    e = 4
   25    1   10   25   12  443   15    7    9    9 |    f = 5
   15    7   15    0   11    4  598    2   10    2 |    g = 6
    0    2    9    4    6    1    0  603    3   17 |    h = 7
    8    7   15   15    9    9    3    4  464    8 |    i = 8
    0    5    5    7   13    0    0    4    3  607 |    j = 9
```

e. Random Forest:

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j   <-- classified as
 1181    0    3    2    2    2    4    0    0    0 |    a = 0
    0 1002    1    0    0    0    0    1    1    0 |    b = 1
    2    1  711    5    7    0    1    2    2    0 |    c = 2
    0    1    7  626    1    8    0    2   10    3 |    d = 3
    2    2    8    0  629    0    3    0    0    8 |    e = 4
   10    0    5   11    5  511    6    0    3    5 |    f = 5
    9    1    5    0    2    1  645    0    1    0 |    g = 6
    0    2    4    0    7    0    0  619    1   12 |    h = 7
    2    2    6    8    8    5    0    2  506    3 |    i = 8
    2    0    1    1   10    0    0    6    1  623 |    j = 9
```

References:

1. Witten, Ian & Hall, Mark & Frank, Eibe & Holmes, Geoffrey & Pfahringer, Bernhard & Reutemann, Peter. (2009). *The WEKA data mining software: An update*. SIGKDD Explorations. 11. 10-18. 10.1145/1656274.1656278
2. Altman, N. S. *An introduction to kernel and nearest-neighbor nonparametric regression*. The American Statistician. **46** (3): 175–185. 1992
3. Holmes, G., Donkin, A. & Witten, I.H. (1994). *WEKA: a machine learning workbench*. (Working paper 94/09). Hamilton, New Zealand: University of Waikato, Department of Computer Science.
4. Dietterich, T. (1998). *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization*, Machine Learning,1–22.
5. Sahoo, G., Kumar Y. (2012). *Analysis of parametric & nonparametric classifiers for classification technique using WEKA*. International Journal of Information Technology and Computer Science.
6. Altman, D. G.; Bland, J. M. (11 June 1994). "*Statistics Notes: Diagnostic tests 1: sensitivity and specificity*". BMJ. 308(6943): 1552.
7. Burke, Donald; Brundage, John; Redfield, Robert. *Measurement of the False Positive Rate in a Screening Program for Human Immunodeficiency Virus Infections*. The New England Journal of Medicine. 1988
8. Powers, David M W (2011). *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation* (PDF). Journal of Machine Learning Technologies. 2 (1): 37–63.
9. https://en.wikipedia.org/wiki/Support-vector_machine
10. https://en.wikipedia.org/wiki/Bootstrap_aggregating