Optimizing Performance Measures in Classification Using Ensemble Learning Methods

by

Neeraj Bahl

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

Ajay Bansal, Chair
Ashish Amresh
Srividya Bansal

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Ensemble learning methods like bagging, boosting, adaptive boosting, stacking have traditionally shown promising results in improving the predictive accuracy in classification. These techniques have recently been widely used in various domains and applications owing to the improvements in computational efficiency and distributed computing advances. However, with the advent of wide variety of applications of machine learning techniques to class imbalance problems, further focus is needed to evaluate, improve and optimize other performance measures such as sensitivity (true positive rate) and specificity (true negative rate) in classification. This thesis demonstrates a novel approach to evaluate and optimize the performance measures (specifically sensitivity and specificity) using ensemble learning methods for classification that can be especially useful in class imbalanced datasets. In this thesis, ensemble learning methods (specifically bagging and boosting) are used to optimize the performance measures (sensitivity and specificity) on a UC Irvine (UCI) 130 hospital diabetes dataset to predict if a patient will be readmitted to the hospital based on various feature vectors. From the experiments conducted, it can be empirically concluded that, by using ensemble learning methods, although accuracy does improve to some margin, both sensitivity and specificity are optimized significantly and consistently over different cross validation approaches. The implementation and evaluation has been done on a subset of the large UCI 130 hospital diabetes dataset. The performance measures of ensemble learners are compared to the base machine learning classification algorithms such as Naive Bayes, Logistic Regression, k Nearest Neighbor, Decision Trees and Support Vector Machines.

DEDICATION

Dedicated to my Mother, Father and my two elder Brothers

ACKNOWLEDGMENTS

I would like to first express my sincere gratitude and thanks to my thesis advisor Dr. Ajay Bansal, whose help, support and guidance was the cornerstone of this thesis. Without his guidance, this thesis would not have been possible. His guidance was invaluable to me at every stage of the thesis. Whenever I was stuck or I stumbled, Dr. Bansal helped me nudge and guide to the goal of completion of the thesis while learning the most from it.

Further, I would like to thank the R Foundation and the R community. The support of R community in providing the open source packages and software to implement and run the machine learning algorithms were integral to this research. Also, I would like to thank Arizona State University Libraries for the incredible resources provided to me for this thesis and in general for the facilities.

I would like to express my sincere gratitude and thanks to the thesis committee members, Dr. Srividya Bansal and Dr. Ashish Amresh for the constructive feedback, time and support they have given throughout my thesis research, defense and during the revisions of this thesis paper. Finally, I would like to thank my family and friends who gave me persistent encouragement and support while I was working on my thesis.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Motivation

There are wide variety of applications for machine learning algorithms. Also, there are various performance measures that needs to be evaluated when an algorithm is applied to a specific problem. We know that accuracy is one of the most common performance measure to evaluate any algorithm or model. However, we also know that higher accuracy does not necessarily means good performance of the algorithm. The other common performance measures used to evaluate the machine learning algorithms are specificity, sensitivity, precision, F1 score, ROC curve, etc. The type of dataset governs the need to evaluate these performance measures. For example, in the application of machine learning to medical diagnostics or medical domain in general, failing to predict positive individuals may result in fatal cost, however, failing to predict negative instances is also serious in some applications, for example, information retrieval, email spam filtering and facial recognition [Hsiao et.al 2014]. Hence, it is imperative to optimize sensitivity (which is a measure of the classification algorithm in avoiding false negative) and specificity (which is a measure of the classification algorithm in avoiding false positive). For example, in the case of email spam filtering, false positive can lead to an important email to land up in the spam mailbox.

There has been research done on optimizing the F1 measure or the F1 score to achieve a compromise between the performance measures of precision and recall by tuning the parameters in the algorithms especially by tuning the parameters in Support Vector Machines. However, in this thesis, a novel approach is used to evaluate and

optimize the sensitivity (recall) and specificity by using ensemble learning methods. It can be observed from the results that the optimization of the performance measures (sensitivity and specificity) with respect to each other is achieved by simply running the data over ensemble learners. The comparison of the performance measures is made with the base machine learning algorithms like the Logistic Regression, Naïve Bayes Classifier, k Nearest Neighbor, Decision Trees and Support Vector Machines.

The need for evaluation of these performance measures apart from accuracy and the need to optimize these performance measures with respect to each other becomes imperative especially when we have class imbalanced datasets. Class imbalance data leads to class imbalance problem. When the data has imbalanced amount of two class labels, it is called as class imbalanced data. For example, in Figure 1 below, we observe that the 80% of the actual target values are of class 0 (red color represents class 0) and 20% of the actual target values are of class 1 (green color represents class 1).

Figure 1: Class Imbalance Dataset Example 1

Similarly, in Figure 2 below, we observe that the 80% of the actual target values are of class 1 (green color represents class 1) and 20% of the actual target values are of class 0 (red color represents class 0). This is a classic example of a medical diagnostics data.

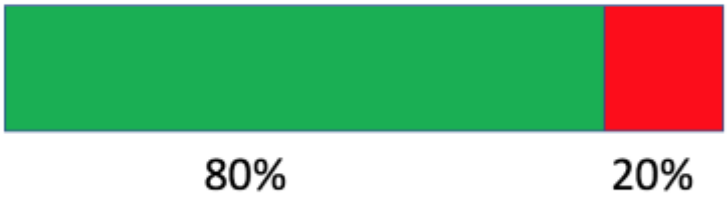

Figure 2: Class Imbalance Dataset Example 2

Most of the datasets these days are class imbalanced datasets and hence the need to evaluate and optimize the performance measures such as sensitivity, specificity, precision, F1 score, etc. are imperative for most of the datasets. Let us take an example of Figure 2 above, for instance, this is a medical diagnostics dataset and we are predicting whether unhealthy patients need to be readmitted to the hospital. In a scenario where only accuracy is considered for performance valuation of the algorithm/model applied to the dataset. Suppose, if we are training a model on this dataset to predict if a patient should be admitted to the hospital or discharged from the hospital based on diagnostic feature vectors, where the actual class 0 (in red) represents unhealthy patients and actual class 1 (in green) represents healthy patients, it is possible that with an algorithm or model that gives us 80% accuracy, we may still have all the 20% of unhealthy patients discharged from the hospital, which can be fatal. Hence, with this example, we underscore our claim that the performance measures such as sensitivity and specificity needs to be evaluated and optimized.

1.2 Introduction to Performance Measures

The emphasis of this thesis is on evaluating and optimizing the performance measures using different base machine learning algorithms and the ensemble learning algorithms. Hence, let us discuss the performance measures in detail.

Confusion Matrix: Confusion matrix is a table of 2 * 2 which gives us the number of true positives, true negatives, false positives and false negatives. This table helps us to find the performance measures required to evaluate the machine learning algorithms in question [Confusion Matrix Wikipedia Page].

	Predicted as 1	Predicated as 0
Actual 1	True Positive	False Negative
Actual 0	False Positive	True Negative

Table 1: Confusion Matrix

Sensitivity (or Recall) is the ratio of number of True Positive instances divided by the sum of number of True Positive and False Negative instances. Hence, higher the sensitivity, higher is the amount of class label predicted as 1 correctly. This performance measure is important in applications where we cannot afford to have any positive class label misclassified [Sensitivity and Specificity Wikipedia Page].

$$Sensitivity = \frac{Number\ of\ True\ Positives}{Number\ of\ True\ Positives + Number\ of\ False\ Negatives}$$

Specificity is the ratio of number of True Negative instances divided by the sum of number of True Negative and False Positive instances. Hence, higher the specificity, higher is the amount of class label predicted as 0 correctly. This performance measure is important in applications where we cannot afford to have any negative class label classified as 1 [Sensitivity and Specificity Wikipedia Page]. For example, for information retrieval, in a search engine, having a false positive can be costly to the functionality and business of the search engine.

$$Specificity = \frac{Number\ of\ True\ Negatives}{Number\ of\ True\ Negatives + Number\ of\ False\ Positives}$$

Precision (or Positive Predictive value) is the ratio of number of True Positive instances divided by the sum of number of True positive and False positive instances. Here, higher the precision, higher is the amount of fraction of positive class labels predicted from the total positive class labels predicted [Sensitivity and Specificity Wikipedia Page].

$$Precision = \frac{Number\ of\ True\ Positives}{Number\ of\ True\ Positives + Number\ of\ False\ Positives}$$

F1 score is a measure of the classification test's accuracy. It considers both Sensitivity (Recall) and Precision. F1 score is the weighted harmonic mean of precision and recall (sensitivity) [Sensitivity and Specificity Wikipedia Page].

ROC Curve illustrates the performance of a binary classifier. ROC stands for Receiver Operating Characteristics. ROC curve is the most common method to select an optimal model. ROC curve is created by plotting the true positive rate (which is same as Sensitivity) against the false positive rate (which is same as 1-specificity) at different threshold settings. [ROC Wikipedia Page]

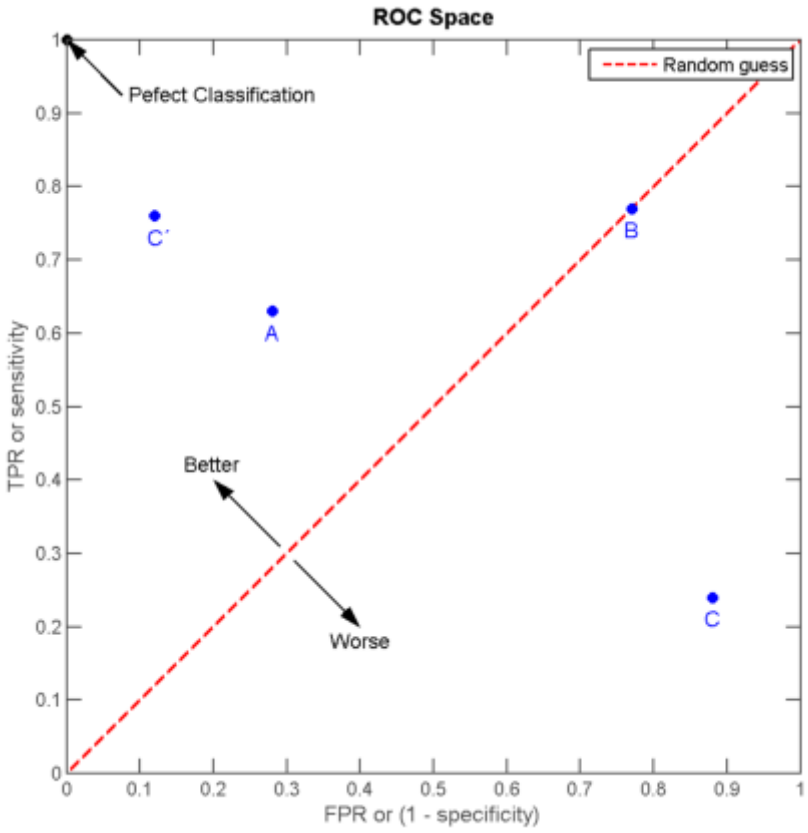

Figure 3: ROC Space Details [Kai walz ROC Curve Wikimedia]

By analyzing the ROC curve, we can find the optimal algorithm/model. As we can see in Figure 3, the best classifier would hug the whole ROC space at the upper left hand corner. The random guess is a diagonal line drawn at 45 degrees from the x-axis. The area above this line is better as it approaches the point However, in this thesis, ROC curve is not used to compare the ensemble learning models with the base learning models as this thesis does not evaluates and delves into the threshold settings of the various classifiers but only the absolute values. Also, this thesis specifically evaluates the specificity and sensitivity. Hence, the comparison is made using different graphs as seen in the Results section below using sensitivity and specificity.

1.3 Introduction to Ensemble Learning

In this thesis, ensemble learning algorithms were used to evaluate, compare and optimize the performance measures (specifically specificity and sensitivity). Here, let us discuss the ensemble learning methods. There are various types of ensemble learning methods which are considered a sub-category of hybrid intelligent systems [Woźniak et.al 2014]. Some of the types of ensemble learning methods are Bagging, Boosting, Adaptive Boosting, Stacked Generalization, Mixture of Experts, etc. [Robi Polikar, Ensemble Learning, 2009]. Ensemble learning is a method to combine multiple models such as classifiers to solve and predict a machine learning problem. It is typically used to improve predictive performance of a model, reduce the likelihood of selecting a poor learner and to assign to confidence to the decision made by the model. [Robi Polikar, Ensemble Learning, 2009]. In general, ensemble learning methods are also divided into the same category of topologies as multi-classifier system. The categories of multi-classifier systems are parallel and serial multi-classifier system as shown in the Figure 4 below.

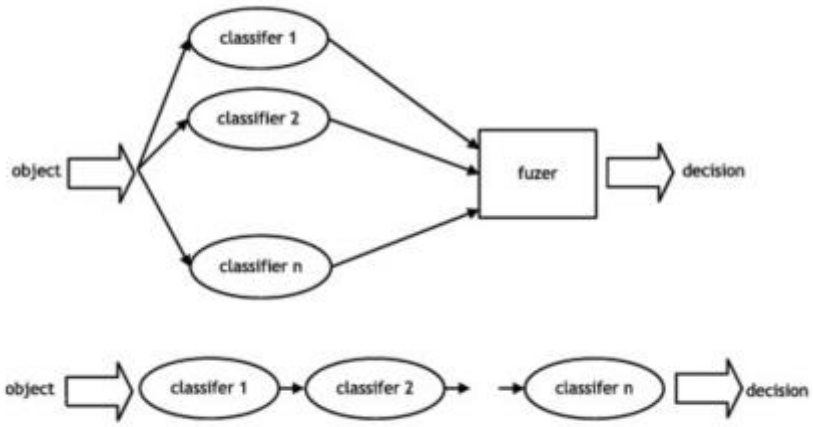

Figure 4: Parallel and Serial Topology of Multi-Classifier System [Woźniak et.al 2014]

As this thesis is limited to classification, let us see the various types of ensemble learning methods available for usage in classification. Figure 5 below shows a common architecture of the ensemble learning methods. Although the specific techniques to divide the data and the data fusion technique differs for bagging, boosting, adaptive boosting, however, Figure 5 shows the common components and the architecture of an ensemble learning method. Here, the input space (or sometimes input features) are divided into different subspaces and fed to the base classifier. After the classification is done by each weak classifier, the results are fused together using various fusion techniques like majority voting, weighted majority voting, etc. [Asmita, Shukla, 2014].

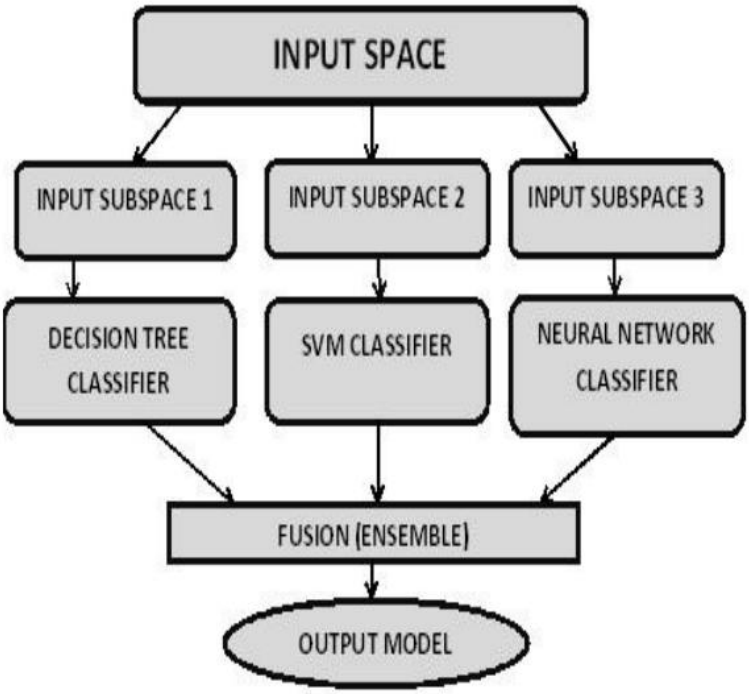

Figure 5: Common architecture of Ensemble Learning Methods [Asmita, Shukla, 2014]

Commonly used ensemble learning algorithms are [Robi Polikar, Ensemble Learning, 2009]:

1) Bagging: Bagging which stands for Bootstrap aggregation obtains diversity in classification by randomly sampling the data – with replacement – from the entire training dataset. Each training dataset is used to train a different classifier of the same type or different types. Simple majority vote is used to fuse the resultant models.
2) Boosting: Boosting is similar to the ensemble learning method of bagging in that it resamples the data, however, it strategically samples the subset of the training data to create several weak classifiers. The classifiers are combined through a n-way majority vote. For example, if there are three classifiers, the first subset of the training data is selected randomly, the second subset is selected in an informative way as per the boosting algorithm. The third subset is sampled based on the instances where both the first and second classifier disagrees with each other. Hence, during fusion of data, a strong classifier is created.
3) Adaptive boosting: This is considered as the best-known ensemble learning algorithm. In this method, the sampling of data is done in the same way as boosting, however, the samples are iteratively updated to focus on increasingly difficult instances. It essentially adapts to the difficult instances from the previous weak learners. Adaptive boosting in turn has many types.
4) Stacked Generalization: This method employs multi-tier architecture to train the bootstrap sample of training data creating Tier-1 models, whose output is then

used to train the Tier-2 models and so on. Thus, in the last Tier, we get the best model.

5) Mixture of experts: Mixture of experts method of ensemble learning generates several experts (classifiers) whose output are combined through a linear rule. Expectation maximization (EM) algorithm is used to assign weights to the combination of classifiers. This is typically used when heterogeneous set of feature space exists.

Apart from the above methods, various types of other ensemble learning algorithms are available which primarily focus on using the diversity of the classifiers and combining the classifiers to achieve stronger prediction in classification.

1.4 Scope

The scope of this thesis is to evaluate and optimize the performance measures of sensitivity and specificity for at least two of the ensemble learning methods (i.e. bagging and boosting) and compare them with the base machine learning algorithms. The optimization is achieved with respect to the values of sensitivity and specificity with each other. While doing so, this thesis achieves optimized sensitivity and specificity using ensemble learning methods as opposed to the previous methods that have focused on tuning the parameters of the base learners or creating framework. This thesis focuses on sensitivity and specificity for the analysis of the performance measures, using one dataset. However, this dataset is divided into various sub-sets of data and four types of cross validation methods are employed to divide the data into test and training. The hold-out cross-validation methods employed are 80-20, 50-50, 70-30, 90-10.

CHAPTER 2

LITERATURE REVIEW

As we know from Chapter 1, that, ensemble learning is the process by which multiple models, such as classifiers or experts, are "strategically generated and combined" to solve a machine learning problem. [Robi Polikar, Ensemble Learning, 2009]. We also need to understand why ensemble learners often perform better than any single classifier. The three reasons for which using an ensemble based system works better than a single classifier are statistical, computational and representational [Dietterich 2000]. First, the author notes that statistically, a learning algorithm can be considered searching a space of Hypothesis. For a single classifier, the *statistical* problem arises when the amount of training data is very less. If we look at the Figure 6 below, we see that we can find a good approximation to the true hypothesis *f* if the ensemble learning algorithm "can "average" their votes and reduce the risk of choosing the wrong classifier" [Dietterich, 2000].

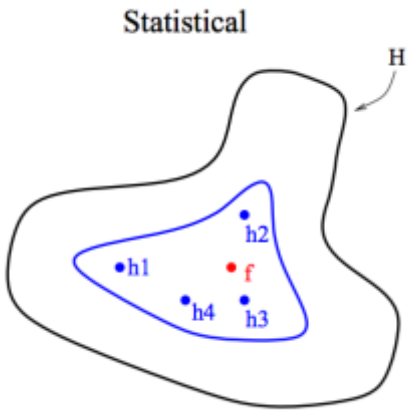

Figure 6: Statistical reason of better performance by ensemble learners [Dietterich, 2000]

Even if we have large amount of training data, the second reason why a single classifier can be weak in approximating to or determining the true hypothesis is *computational* reason. The greedy approach employed by the single classifiers in finding the local optima would result in finding the local optima as opposed to global optima [Dietterich, 2000]. As per this paper, Ensemble learners would do the "local search from many different starting points" and hence they would provide a better approximation to the true hypothesis as explained in the Figure 7 below.

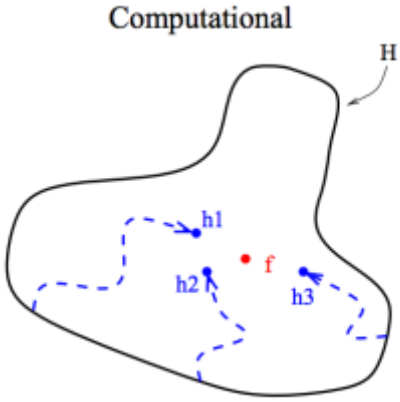

Figure 7: Computational reason of better performance by ensemble learners [Dietterich, 2000]

Further, as per the author, the third reason is *representational* which is "somewhat subtle". The hypothesis space for a single classifier is limited to the classifier, although decision tree and neural networks are both flexible algorithms. Given enough training data, they will "explore the space of all possible classifiers" [Dietterich, 2000]. Nevertheless, as the hypothesis space for these single classifiers is limited, using

"weighted sum of hypothesis", the hypothesis space can be expanded by the ensemble learners [Dietterich, 2000]. This is very accurately described in the Figure 8 below. The true hypothesis *f* can be explored outside the limited space of single classifiers.

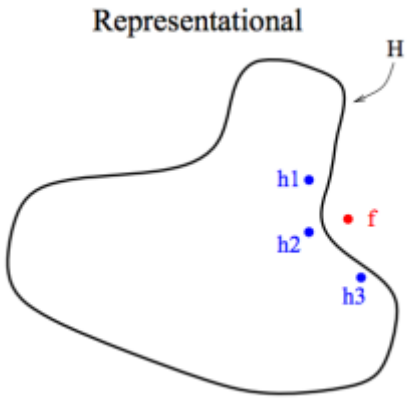

Figure 8: Representational reason of better performance by ensemble learners [Dietterich, 2000]

By studying the literature, we can observe that researchers have used different performance measures for optimization, depending on the application and domain. Typically, we can see that the various performance measures such as F1 score, Sensitivity, Specificity, Precision, Recall, ROC, Area Under the Curve (AUC) etc. are based on the confusion matrix which helps us better analyze and evaluate the performance of the classifiers. These measures are very important, which directly and indirectly puts emphasis on approximating to correct prediction for each instance in the dataset in an ideal case. In this thesis, however, the visualization and analysis of the results is done using sensitivity and specificity, by finding the relation with each other, and by visualizing the standard deviation for sensitivity, specificity and accuracy.

Now, we will study the literature for the research done attempting to optimize the performance measures using various methods that include base machine learning tuning, usage of ensemble learning and other methods to form a framework, etc. [Hsiao et.al, 2014] proposed a two-stage framework and a novel evaluation criterion, namely optimal specificity under perfect sensitivity (OSPS). They argued that for medical data classification, this criterion is more suitable than other conventional measures such as accuracy, f-score, or area-under-ROC curve. Here, they employ instance ranking strategy to optimize the specificity given perfect sensitivity using threshold for the perfect sensitivity. They used two toolkits to experiment on their dataset, one of which included Adaboost which is one of the ensemble learning methods. The Adaboost used Classification and Regression Trees (CART) as the base learner.

Further, Support Vector Machine (SVM) parameter tuning has also shown to optimize F1 score in unbalanced data. F measure, which is the harmonic mean of precision and recall, is considered the relevant performance measure to be achieved especially when we have unbalanced data. SVMs have been traditionally used for classification of unbalanced data by "weighting more heavily the error contribution from the rare class" [Musicant, et.al 2003]. In this paper, the authors provided significant and new theoretical results that support this popular heuristic. Specifically, they demonstrated that with the right parameter settings SVMs approximately optimize F-measure in the same way that SVMs have already been known to approximately optimize accuracy. They argue that this finding has numerous theoretical and practical implications for using SVMs in F-measure optimization.

Moreover, [Chai et. al 2012] investigated the connections and theoretical justification of the two methods commonly used to optimize F score. The two algorithms for learning discussed to maximize F-measures are: the empirical utility maximization (EUM) approach that learns a classifier having optimal performance on training data, and the decision-theoretic approach (DTA) approach that "learns a probabilistic model and then predicts labels with maximum expected F-measure." [Chai et. al 2012]. In the paper, the authors further studied the conditions under which one approach is preferable to the other using synthetic and real datasets. They claim that their results suggested that the two approaches are asymptotically equivalent given large training and test sets. Empirically they also prove that the EUM approach appeared to be more robust against model misspecification and the decision-theoretic approach appears to be better for handling rare classes and a common domain adaptation scenario.

Now, this thesis proposes to optimize performance measures using ensemble learning methods. We know that the diversity in the ensemble learning methods help achieve better accuracy and other performance measures. Hence, we also need to understand if there is any relation between accuracy and diversity in ensemble learning methods. [Zeng et. al, 2014] proposed a novel method to evaluate the quality of a classifier ensemble which is inspired by the F measure used in information retrieval. They proposed a Weight Accuracy Diversity (WAD) measure to assess the quality of the classifier ensemble. They claim that WAD would help find a balance between accuracy and diversity which is needed for enhancing the predictive ability of an ensemble learner over unknown data as higher diversity not necessarily means better performance by the ensemble learning algorithm. This research would not only help us understand the

relation between accuracy and diversity in classifying ensemble learning methods but would also help us qualitatively assess and compare the various ensemble learning methods using classification.

The literature helps us understand that past research is focused on tuning the existing base learning algorithms, developing frameworks based on the domain of the application, developing methods to optimize the F score, providing theoretical justification and empirical evidence for the methods already developed, providing reasons to use ensemble learning methods and to qualitatively assess and balance the ensemble learning methods with respect to accuracy and diversity to find a balance. The work done by these researchers have significantly helped discuss and achieve significant success in performance measures and ensemble learning methods. However, no apparent focus has been made in direct usage of ensemble learning methods for improving or optimizing the performance measures i.e. sensitivity (recall) and specificity. In this thesis, in the further chapters we will see how the ensemble learning methods are empirically proven to improve, optimize and balance the performance measures of sensitivity and specificity.

CHAPTER 3

METHODOLOGY

3.1 High Level Implementation

The Figure 9 below shows the high-level flowchart of the experimental implementation set up. These six high-level steps are consistent with the details provided below. These steps were implemented and were common to all the learners (base and ensemble learners). First for each learner, the libraries from the R Comprehensive R Archive Network (CRAN) Package were imported. Then, the data was pre-preprocessed. This step of preprocessing included various important and critical sub-steps. Once the data pre-processing was completed which includes factoring, normalization, making the feature vectors binary wherever necessary, the data was split using hold out cross validation methods into training and test datasets. The respective base machine learning or ensemble learning model were then trained using the training data over these cross validation methods. The test data is tested on the generated model using the predict methods. Finally, the predicted feature vector thus created is compared to the actual target feature vector with the help of confusion matrix.

The base learning classifiers and ensemble learning methods were implemented using the R CRAN packages library. To generate the confusion matrix or the cross table and the performance measures, the respective R library methods were used. The large dataset was divided into 10 subsets of 10000 instances.

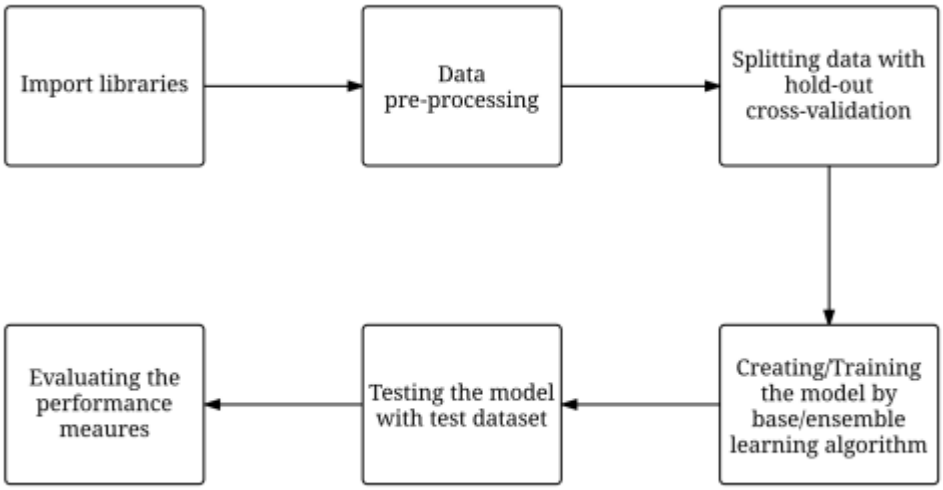

Figure 9: High Level Implementation Flowchart

3.2 Software Used

The following software have been used to implement and experiment this thesis:

- R version 3.2.3: R and the associated R CRAN packages [R CRAN Packages, 2017] was used for implementing the machine learning algorithms and evaluate the performance measures. R is free and open source.
- R Studio 1.0.136: R Studio which is a free and open source integrated development environment for R was used for the ease of implementation.
- Microsoft Excel 2016: Excel 2016 was used to generate the tables and plot the graphs. Excel 2016 was licensed through Arizona State University student license.
- Lucidchart: Lucidchart was used to create the flowchart. Lucidchart is free and available with Google account.

3.3 Dataset

In this thesis, the UC Irvine (UCI) Diabetes 130-US hospitals for the years 1999-2008 dataset [Strack et al. 2014] has been used. This dataset is multi-variate dataset, contains 100,000 instances and 55 features. "The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes. Information was extracted from the database for encounters that satisfied the following criteria:

(1) It is an inpatient encounter (a hospital admission).

(2) It is a diabetic encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis.

(3) The length of stay was at least 1 day and at most 14 days.

(4) Laboratory tests were performed during the encounter.

(5) Medications were administered during the encounter.

The data contains such attributes as patient number, race, gender, age, admission type, time in hospital, medical specialty of admitting physician, number of lab test performed, HbA1c test result, diagnosis, number of medication, diabetic medications, number of outpatient, inpatient, and emergency visits in the year before the hospitalization, etc." [Strack et al. 2014].

It is a three-class label dataset with the values being NO, <30, >30. Here, NO means that the patient was not readmitted after the discharge. <30 means that the patient was readmitted within 30 days of discharge. >30 means that the patient was readmitted after 30 days of discharge.

3.4 Data pre-processing

Before using this data in the implementation of machine learning algorithms to evaluate and compare the performance measures of ensemble learning methods to the base machine learning algorithms, the data is pre-processed. This was needed to ensure that all the classification algorithms and ensemble learners can be equally and fairly implemented on the dataset. First, the missing values in the dataset are converted into NA and in turn into 0. The following eight feature vectors were dropped from dataset as they do not actively represent the classification problem and do no contribute to the target class label values for classification: Encounter ID, Patient Number, Weight (this feature vector was dropped as 97% values are missing), Payer Code, Medical Specialty, Diagnosis 1, Diagnosis 2, Diagnosis 3. Then, the target feature vector "readmitted" was changed to a binary feature vector as 0 or 1 by converting the three class label dataset into two class label classification dataset as follows: If the target feature vector value is <30 or >30, it was changed to 1. If the target feature vector is NO, the target feature vector value was changed to 0.

To ensure consistency for the implementation, the nominal feature vectors were changed to binary feature vectors. After changing the feature vectors, the dataset now contained 98 feature vectors. Further, the numeric data which is now contained in the first eight feature vectors is normalized to ensure that the evaluation and comparison is fairly made when using these feature vectors for classification. After the normalization, the data in these 8 numeric feature vectors lied in the range of 0 to 100. The normalization range was chosen to be from 0 to 100 to ensure that the wide variety of the values in the dataset

are included without having any common values. For example, it would be statistically incorrect to bring 100000 instances with different nominal features in a range of 0 to 10 as most of the instances in that case would have common values. This would lose the consistency and integrity of data in the dataset. Now the dataset was made ready to be implemented for all the algorithms.

Now, the instances were separated in 10 blocks of 10000 instances of dataset to implement as individual separate datasets. This would give us wide range of results for effective comparison. The hold out cross-validation method used were 90-10, 80-20, 70-30 and 50-50 cross validation method. These different methods of cross validation helped to demonstrate whether any changes in the cross-validation affects the performance measures, especially, the measures that are critical to the application domain i.e. accuracy, sensitivity and specificity. In this thesis, only one subset of 10000 instances of the dataset was used.

3.5 Implementation & Experimental Setup

The base machine learning algorithms and the ensemble learning algorithms were implemented in the R version 3.2.3. As seen in the Table 2 below, the following algorithms were implemented and following are the corresponding R methods and packages used for the implementation. The training data was separated according to the hold out cross-validation methods described above and the model was trained on the training dataset. Once the model was trained, the test data was fed to the model using the respective prediction methods to determine the performance measures. In case of logistic regression, the threshold probabilities generated are converted into binary target values.

When the probability is greater than 0.5, the binary target value is changed to 1 else 0. In the case of k Nearest Neighbor (kNN), the k value is selected as the optimal value which is the square root of the number of training examples. This is dynamically calculated for each dataset and encoded in the code. The target value is converted into a factor wherever required. The Support Vector Machine (SVM) algorithm is implemented for both the radial basis function and the sigmoid basis function. For bagging, the treebag method (algorithm) is used which means the base learning algorithm for bagging was decision tree. Similarly, the boosting algorithm is implemented using Decision Trees and Gradient Boosting Model using Decision Trees.

Algorithm	R Methods	R Packages
Logistic Regression	Glm	Multiple
Naïve Bayes Classifier	naiveBayes	Multiple
K Nearest Neighbor (kNN)	knn	Multiple
Decision Trees	C5.0	C50/Multiple
Support Vector Machine (sigmoid)	svm	e1071/Multiple
Support Vector Machine (radial)	svm	e1071/Multiple
Bagging (CART)	treebag	caretEnsemble/Multiple
Boosting (Decision Trees)	C5.0	caretEnsemble/Multiple
Boosting (Gradient Boosting Machine)	gbm	caretEnsemble/Multiple

Table 2: Algorithms and corresponding R methods/packages

The bagging and both the boosting ensemble methods were resampled over 30 iterations. This value was referred in the article by [Brownlee, 2016]. As part of future work, more

experiments can be conducted by fine tuning the number of classifiers/iterations for each ensemble learning method that is implemented. For bagging, boosting and boosting using gradient boosting model, decision tree was used as a base machine learning algorithm. Decision tree was chosen as a base learning algorithm for the ensemble learners as it did not optimize both the performance measures as good as the other base learning algorithms on the same dataset, so to compare and evaluate if the ensemble learning algorithms perform better than the base learning algorithm, decision trees were used.

3.6 Evaluation Measures

The evaluation measures are generated and demonstrated by plotting the confusion matrix and by using the summary of the results using R packages. These results include various performance measures including the ones that are integral to this thesis i.e. accuracy, sensitivity and specificity. The two R methods that were used to generate the evaluation measures are CrossTable() and confusionMatrix(). CrossTable generates a 2*2 confusion matrix of the True Positives, True Negatives, False Positives and False Negatives. confusionMatrix() provides a detailed summary including the confusion matrix and the associated performance measure values.

CHAPTER 4

RESULTS

4.1 Analysis of Results

Here, the below tables give the summary of the results obtained for the first 10000 instances of the dataset. The below four tables are populated with the results of the four cross validation methods employed to test the model. After each table, the corresponding graphs are plotted to analyze the results. It can be clearly seen that the ensemble learning methods of Bagging using Decision Trees, Boosting using Decision Trees and Boosting using Gradient Boosting Model shows consistent improvement in the optimization of both sensitivity and specificity. It is also observed from the plots and the graphs that optimization is achieved without compromising the accuracy of the prediction. For example, the Sensitivity and Specificity are balanced and optimum consistently when bagging and both the boosting methods are used with 80-20 cross validation method. Further for the same dataset, in the column (bar) graph we can observe that both sensitivity and specificity almost have equal height. From the tables and figures followed below, we can see that the same results are consistently achieved across the different cross-validation methods that are experimented. Although, Logistic regression, kNN and SVM with radial basis function also demonstrates optimization as proved by the graphs below, we can see consistent optimized results are achieved with the help of ensemble learning algorithms such as Bagging, Boosting using CART (Decision Trees) and Boosting using Gradient Boosting Machine (GBM) using Decision Trees.

Now let us analyze the results in detail that are presented below based on the experimental set up discussed above. The results are arranged in the following order. First, the table of the results for each hold-out cross validation method is seen, Then, a graph of absolute values of sensitivity, specificity and accuracy are plotted as points that are connected with dotted lines. This graph is very helpful in visualizing whether a specific learner was able to achieve optimization/balancing of the performance measures. The x axis on this graph has the absolute names of the algorithms and y axis has the value of the specific learner. Further, a bar plot graph is plotted to compare only the height of sensitivity and specificity to better visualize the performance measures. Finally, in the end the standard deviation of each learner is calculated for accuracy, sensitivity and specificity. The lower the standard deviation, the more the algorithm is optimized with respect to the performance measure of sensitivity and specificity.

Algorithm	Sensitivity	Specificity	Accuracy
Logistic Regression	0.6683	0.5068	0.6125
Naive Bayes Classifier	0.9315	0.2632	0.584
K Nearest Neighbor (KNN)	0.6091	0.5631	0.591
Decision Trees	0.7759	0.4221	0.5975
SVM (kernel = sigmoid)	0.91692	0.04496	0.489
SVM (kernel = radial)	0.6076	0.5491	0.5775
Bagging Decision Tree	0.5873	0.5867	0.587
Boosting Decision Tree	0.6631	0.5338	0.601
Boosting GBM	0.6781	0.56	0.62

Table 3: Performance measures for 80-20 cross validation

First, for the 80-20 holdout cross validation method, we can observe from Table 3, that the performance measure values are diverse and lie in the range from 0.04 to 0.91. We have achieved wide variety of optimization of sensitivity and specificty with respect to each other. Hence, to visualize we observe from Figure 10, that the bagging, boosting and boosting using GBM have consistently performed better in optimizing both sensitvity and specificity. Although, we can observe that Logistic Regression, k Nearest Neighbor (kNN) and Support Vector Machine using Radial basis function has also performed better in optimizing both sensitvity and specificity. However, we also need to ascertain whether the results hold consistency and confidence across all the hold-out cross validation methods. The worst performance has been shown by Support Vector Machine (SVM) using Sigmoid basis function.

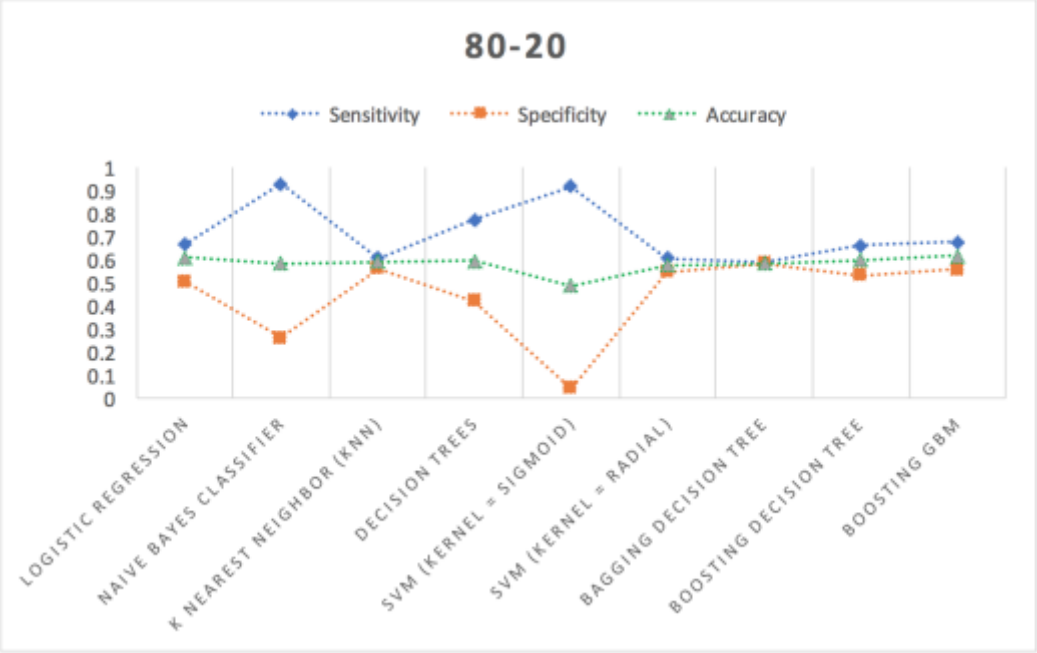

Figure 10: Plot to visualize optimization of 80-20 cross validation

In SVM using sigmoid, we see that Sensitvity is very high but specificity is very low. This performance may be accepted in medical domain but not in information retrieval where a higher specificity is accepted. This can be visualized from Figure 11 as well.

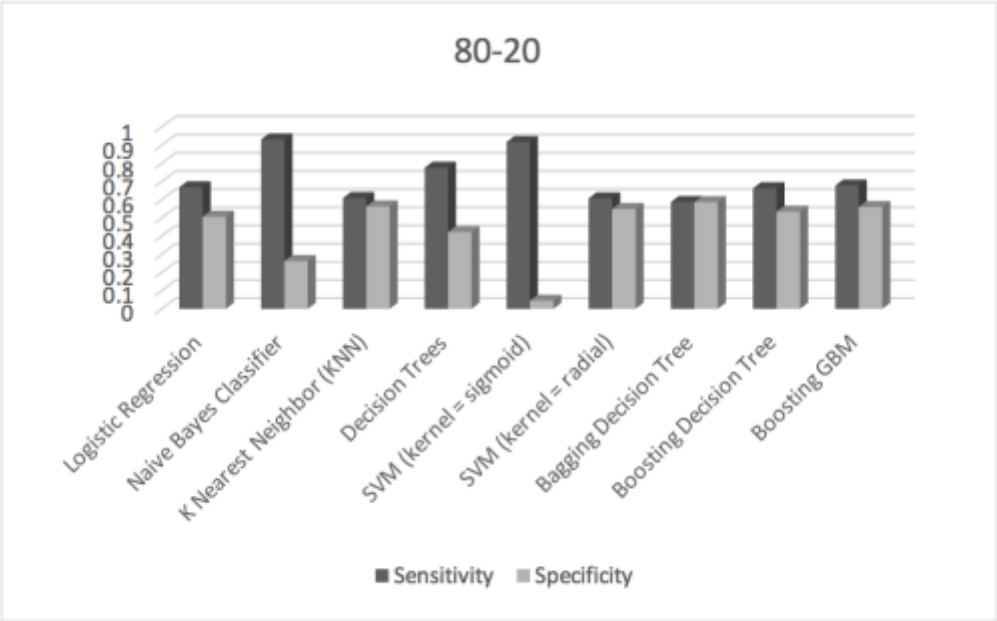

Figure 11: Bar Plot for 80-20 cross validation

Now, for the 50-50 holdout cross validation method, we can observe from Table 4, largely most of the values are consistent with 80-20 cross validation method. Here as well, we have achieved wide variety of optimization of sensitivity and specificty with respect to each other As per Figure 12, we can observe that again the bagging, boosting and boosting using GBM methods have consistently performed better in optimizing both sensitvity and specificity. Here too, we observe that Logistic Regression, k Nearest Neighbor (kNN) and Support Vector Machine using Radial basis function has performed better in optimizing both sensitvity and specificity.

Algorithm	Sensitivity	Specificity	Accuracy
Logistic Regression	0.6442	0.5515	0.5992
Naive Bayes Classifier	0.9473	0.1077	0.5312
K Nearest Neighbor (KNN)	0.6144	0.5628	0.589
Decision Trees	0.7382	0.4524	0.5978
SVM (kernel = sigmoid)	0.96855	0.01018	0.4978
SVM (kernel = radial)	0.5479	0.6277	0.5866
Bagging Decision Tree	0.606	0.5533	0.5798
Boosting Decision Tree	0.6022	0.7318	0.4645
Boosting GBM	0.6312	0.5928	0.6126

Table 4: Performance measures for 50-50 cross validation

There has been a subtle change though, in the observation of boosting using GBM, as per Figure 12 and Figure 13, the performance of Boosting using GBM has improved further.

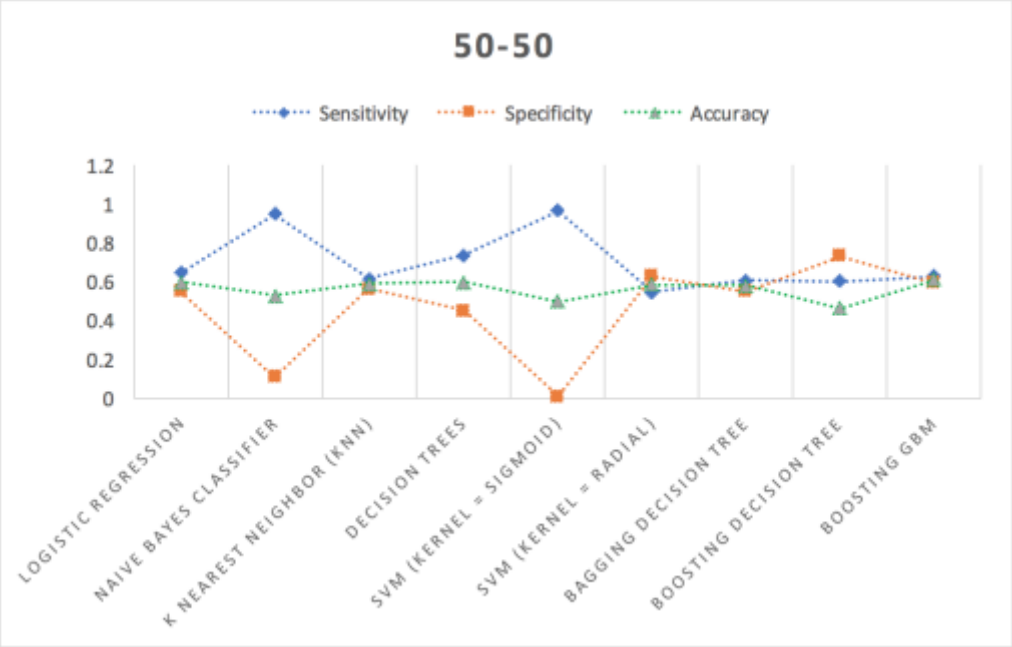

Figure 12: Plot to visualize optimization of 50-50 cross validation

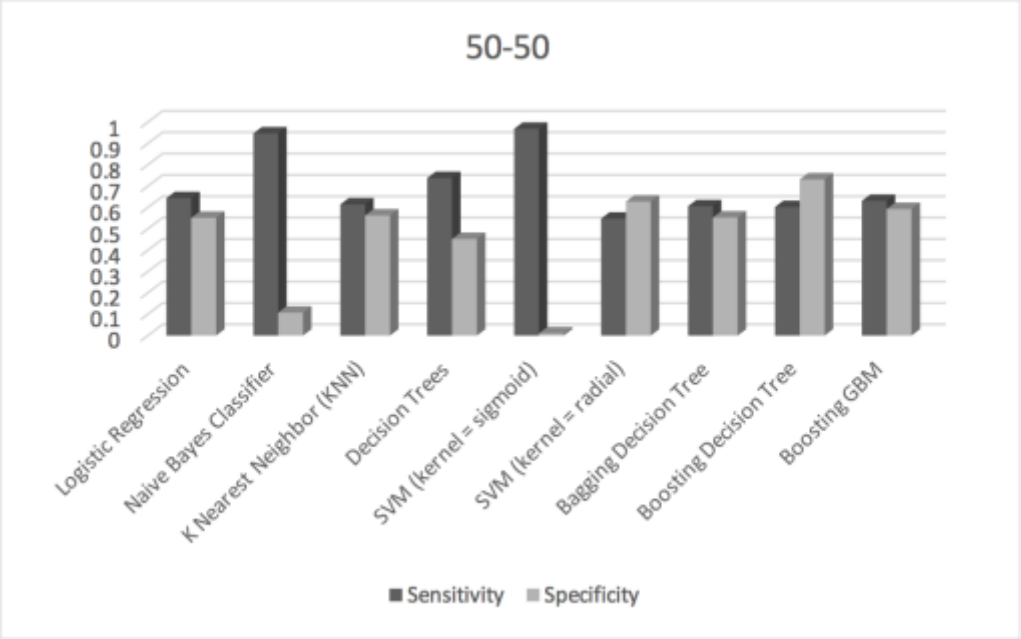

Figure 13: Bar Plot for 50-50 cross validation

Further, we can observe from Table 5, Figure 14 and Figure 15 for 70-30 hold out cross validation method, largely most of the values are consistent with 80-20 and 50-50, however, there has been a visible change in perfomrance of Logistic Regression, Naïve Bayes and Boosting using GBM. As per Figure 14, we can observe that again the bagging, boosting and boosting using GBM methods have consistently performed better in optimizing both sensitvity and specificity. In this case, we observe that only k Nearest Neighbor (kNN) and Support Vector Machine using Radial basis function has performed better in optimizing both sensitvity and specificity as compared to other base learners. The same observation is made through the bar plot of the 70-30 cross valudation method. The height of most of the bars for ensemble learning algorithms for both sensitvity and specificity are of mostly equal size.

Algorithm	Sensitivity	Specificity	Accuracy
Logistic Regression	0.7562	0.4156	0.5907
Naive Bayes Classifier	0.4514	0.6994	0.5743
K Nearest Neighbor (KNN)	0.6286	0.5425	0.585
Decision Trees	0.7886	0.4253	0.6087
SVM (kernel = sigmoid)	0.94184	0.02623	0.488
SVM (kernel = radial)	0.5435	0.5967	0.5693
Bagging Decision Tree	0.6077	0.5622	0.5847
Boosting Decision Tree	0.5878	0.5968	0.5922
Boosting GBM	0.6775	0.5331	0.6068

Table 5: Performance measures for 70-30 cross validation

In this case, for Logistic Regression, we observe that although sensitvity has increased, however, specificity has reduced as compared to the 80-20 and the 50-50 hold out cross validation method.

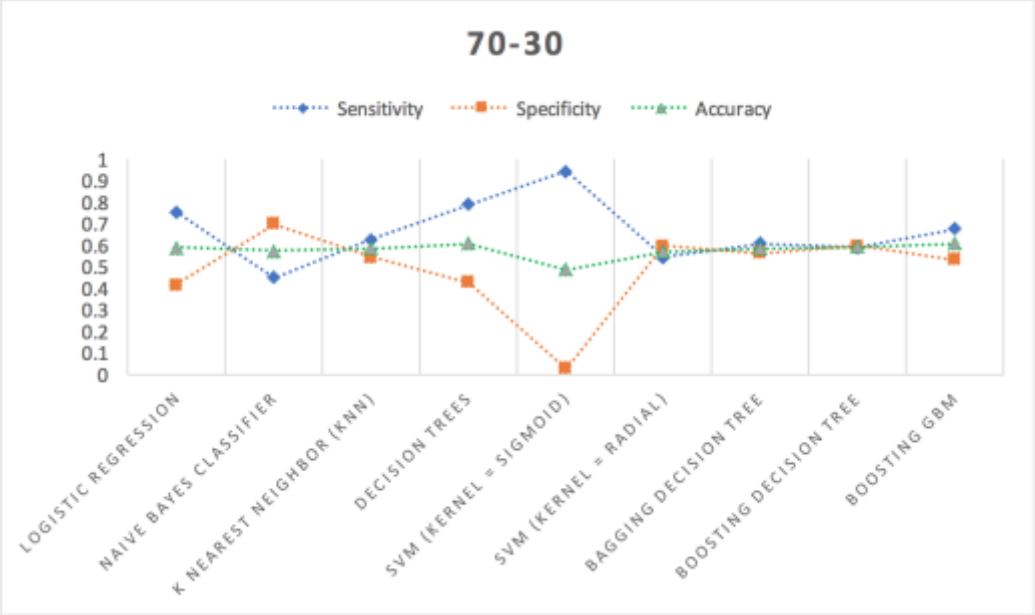

Figure 14: Plot to visualize optimization of 70-30 cross validation

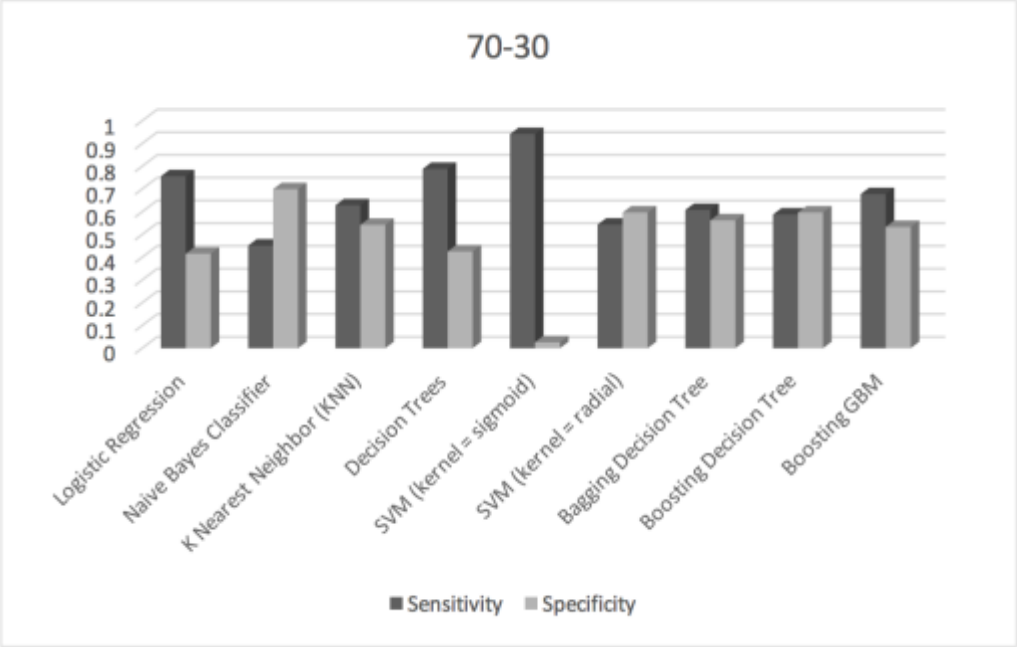

Figure 15: Bar Plot for 70-30 cross validation

Finally, for 90-10 hold out cross validation method, most of the results are consistent with 80-20 and 50-50. As per Table 6 and *Figure 16*, we can observe that again the bagging, boosting and boosting using GBM methods have consistently performed better in optimizing both sensitvity and specificity. In this case, the specificity for Logistic Regression has been regained and both the performance measures are similarly optimized as in 80-20 and 50-50. The height of most of the bars for ensemble learning algorithms for both sensitvity and specificity are of mostly equal size.

Algorithm	Sensitivity	Specificity	Accuracy
Logistic Regression	0.6941	0.5266	0.606
Naive Bayes Classifier	0.856	0.212	0.534
K Nearest Neighbor (KNN)	0.6085	0.5542	0.581
Decision Trees	0.7536	0.4617	0.605
SVM (kernel = sigmoid)	0.93023	0.03926	0.499
SVM (kernel = radial)	0.5578	0.6031	0.578
Bagging Decision Tree	0.6	0.534	0.568
Boosting Decision Tree	0.6965	0.4938	0.598
Boosting GBM	0.6498	0.5494	0.601

Table 6: Performance measures for 90-10 cross validation

From Figure 16 and Figure 17, we see that for all the hold out cross validation methods, sensitvity always have been inheriently high, which is a good observation as the number of false negatives can be reduced.

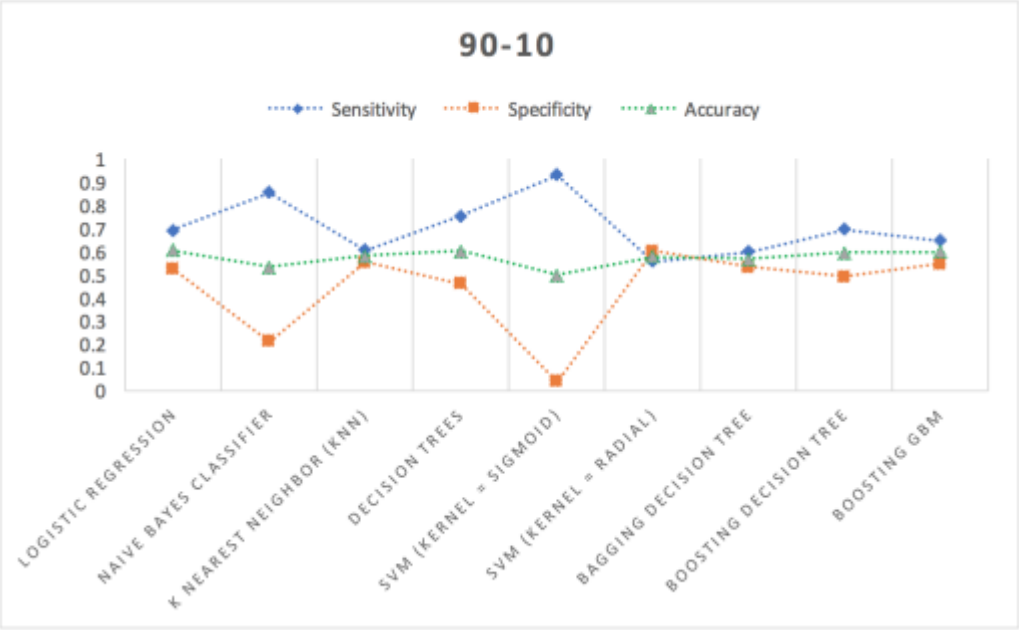

Figure 16: Plot to visualize optimization of 90-10 cross validation

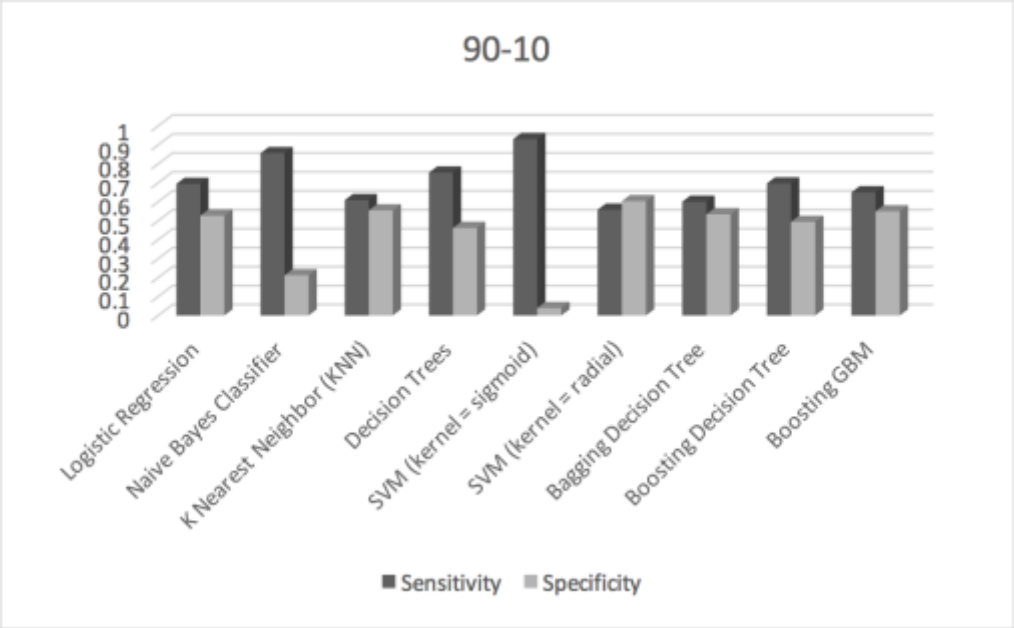

Figure 17: Bar Plot for 90-10 cross validation

4.2 Standard Deviation for Sensitivity and Specificity

Now, let us observe how spread out sensitivity and specificity across the four cross-validation approaches are for the ensemble learning and base machine learning algorithms. These standard deviations are calculated using values of accuracy, specificity and sensitivity. As accuracy is approximately same or better across all experiments conducted, higher standard deviation would mean, more the sensitivity and specificity are spread out and lower the standard deviation would mean, more the two measures are optimized and close to each other. This measure helps us understand the optimization process if the above graphs are hard to visualize.

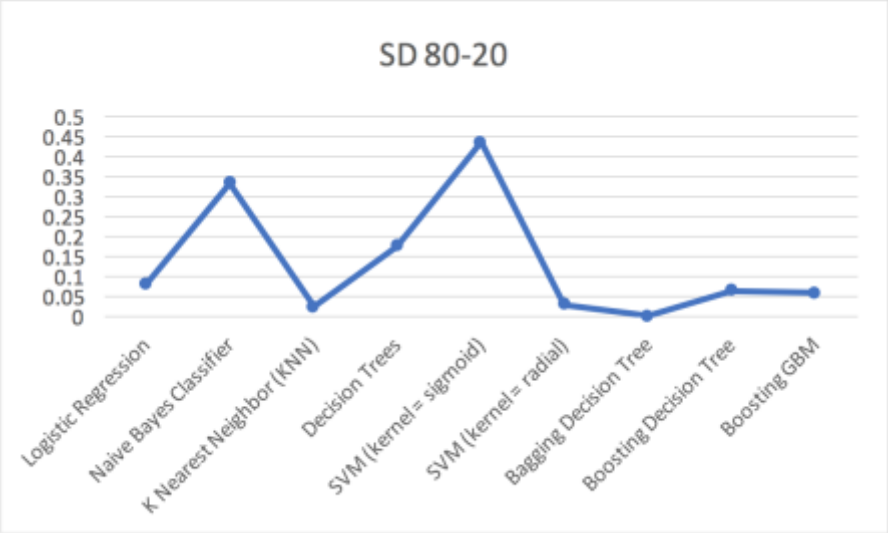

Figure 18: SD for 80-20

Here, in Figure 18 and Figure 19, as we have observed in earlier graphs, we can observe that the ensemble learning methods and three of the single classifier system viz., Logistic Regression, k Nearest Neighbor and SVM using Radial basis function has lower Standard Deviation.

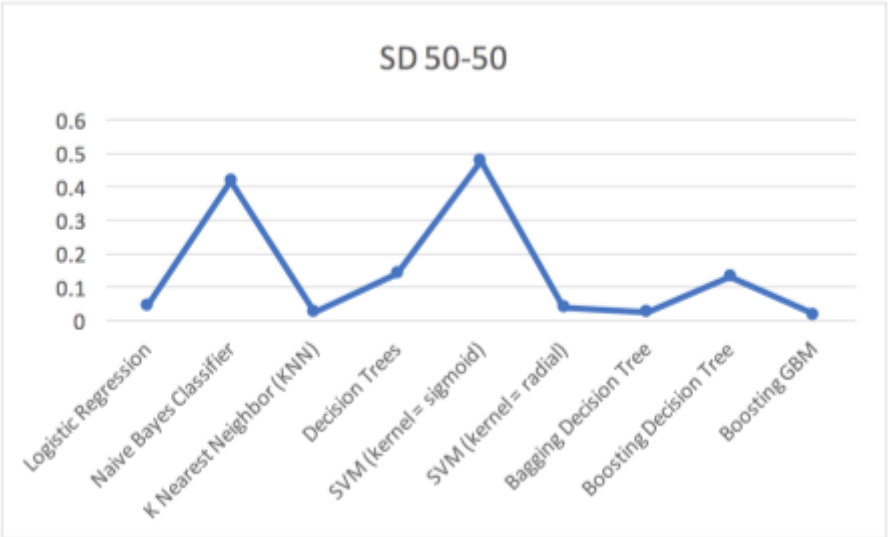

Figure 19: SD for 50-50

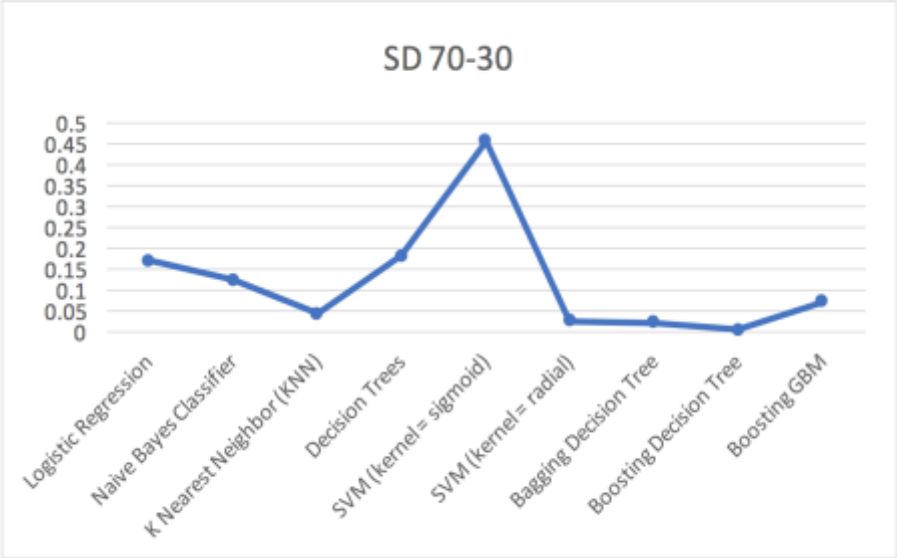

Figure 20: SD for 70-30

However, for 70-30 hold out cross validation method, as seen in Figure 20, the standard deviation for Logistic Regression is seen to be increased and hence the Sensitivity and Specificity are spread out from each other. Similarly, Figure 21 corresponds to the graphs and tables for 90-10.

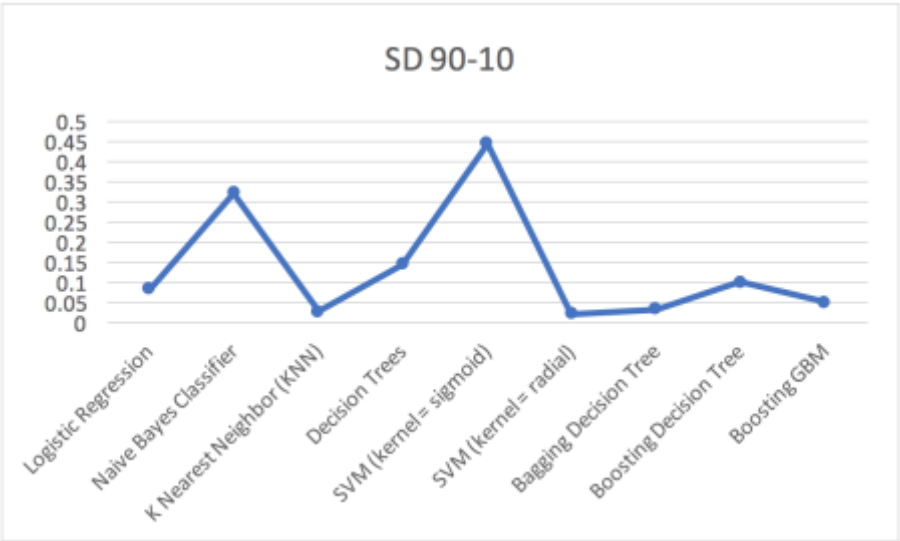

Figure 21: SD for 90-10

CHAPTER 5

CONCLUSION

Given the evaluation and analysis done on the results for the experiments conducted on this dataset over different cross validation approaches, in this thesis, it can be empirically concluded that by using ensemble learning methods such as bagging, boosting using decision trees and boosting -- Gradient Boosting Modeling (GBM) using decision tress, sensitivity and specificity can be optimized with respect to each other without compromising accuracy. This usage of ensemble learning methods for optimizing sensitivity and specificity would help apply this technique to various domain of applications. Although depending on the application and domain, the need for sensitivity and specificity differs, however, given this method, the avenues of balancing and nearly approximating the performance measures of sensitivity and specificity to 1 using ensemble learning methods are opened. The comparison made with the base machine learning algorithms helped underscore the claim.

Although, in some cases, it was observed that the base machine learning algorithms also optimized sensitivity and specificity with respect to each other, for example, Logistic Regression, Support Vector Machines using radial basis function and K Nearest Neighbor, however, over different cross validation methods, the optimization may not be consistent as can be seen with Logistic Regression which achieved a poor performance over 70-30 hold-out cross validation method. This work can be used in applications where equal values of sensitivity and specificity are expected to be achieved. With the help of future work and research on tuning of ensemble learning methods,

developing comprehensive frameworks, it is shown that indeed, we have possibility to achieve better performance for Sensitivity and Specificity.

Hence, as evident from the results, it is empirically concluded that the ensemble learning methods additionally provide the confidence of model selection with respect to the optimization of performance measures of sensitivity and specificity.

CHAPTER 6

FUTURE SCOPE

The significant and consistent improvement in optimization of the performance measures using ensemble learning demonstrated in this thesis provides a benchmark to further evaluate these performance measures for optimization by fine tuning the base machine learning algorithms and especially fine tuning the ensemble learners for detailed optimization research. For example, fine tuning the number of classifiers, type of classifiers and the configuration parameters. Further specific experiments can be conducted to assess and evaluate the results specifically for class imbalance classification problems. Moreover, apart from bagging, boosting and boosting -- Gradient Boosting Model (GBM), other ensemble learning methods like stacked generalization and mixture of experts can be experimented and studied in future. Further, the ensemble learning methods using different classifiers can be experimented as well.

As optimization in detail includes fine tuning the parameters and configuration of the ensemble learning algorithms as opposed to trying different ensemble learning algorithms, it would be beneficial to experiment and evaluate the ensemble learners at the configuration level. There is a scope to continue the experimentation over different datasets to underscore the claim that ensemble learners can optimize the sensitivity and specificity. Bagging and both the boosting ensemble methods were resampled over 30 iterations in this thesis. As part of future work, more experiments can be conducted by fine tuning the number of classifiers/iterations and experimenting the other subsets of the same dataset.

As the results showed that the base learning algorithms such as the kNN, Logistic Regression and SVM using radial basis function performed as better as the ensemble learning methods, further research can be conducted to investigate the cause of the same. Also, the ensemble of these base learners as opposed to only Decision Trees can be implemented to study if there are any further improvements in the performance measures. Further, theoretical research can be done to assess, evaluate and validate these improvements in performance measures by ensemble learning methods.

Finally, there is a scope to improve the computational efficiency of the ensemble learning methods as these take more computational resources as compared to the base learning algorithms.

REFERENCES

Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records” *BioMed Research International*, vol. 2014, Article ID 781670, 11 pages, 2014.

Cho-Yi Hsiao, Hung-Yi Lo, Tu-Chun Yin and Shou-De Lin, "Optimizing specificity under perfect sensitivity for medical data classification," *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, Shanghai, 2014, pp. 163-169.
doi: 10.1109/DSAA.2014.7058068
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7058068&isnumber=7058031

Michał Woźniak, Manuel Graña, Emilio Corchado, “A survey of multiple classifier systems as hybrid systems”, *Information Fusion*, Volume 16, March 2014, Pages 3-17, ISSN 1566-2535, http://dx.doi.org/10.1016/j.inffus.2013.04.006.

Robi Polikar (2009), “Ensemble learning”, *Scholarpedia*, 4(1):2776.

T.G. Dietterich, "Ensemble Methods in Machine Learning," *Int. Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, Vol. 1857, pp. 1-15, 2000, Springer-Verlag.

A.C. Tan, D. Gilbert, “Ensemble machine learning on gene expression data for cancer classification”, *Appl. Bioinform.*, 2 (2003), pp. S75–83

Musicant, D., Kumar, V., & Ozgur. A. (2003). Optimizing f-measure with support vector machines. Proc. FLAIRS.

Nan, Ye; Chai, Kian Ming; Lee, Wee Sun; Chieu, Hai Leong, “Optimizing F-measure: A Tale of Two Approaches”, 06/2012, ICML2012.

Diabetes 130-US hospitals for years 1999-2008 Data Set, *UC Irvine (UCI) Machine Learning Repository*, Last accessed: April 2, 2017, https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

R CRAN Packages, Last Accessed: April 2, 2017
https://cran.r-project.org/web/packages/available_packages_by_name.html

Confusion Matrix, *Wikipedia Page*, Last accessed: April 2, 2017, https://en.wikipedia.org/wiki/Confusion_matrix#Table_of_confusion

Indrajit Mandal. 2015. A novel approach for predicting DNA splice junctions using hybrid machine learning algorithms. *Soft Comput.* 19, 12 (December 2015), 3431-3444. DOI=http://dx.doi.org/10.1007/s00500-014-1550-z

Sensitivity and Specificity, *Wikipedia Page*, Last accessed: April 18, 2017, https://en.wikipedia.org/wiki/Sensitivity_and_specificity

ROC_space.png: Indon derivative work: Kai walz (talk) - ROC_space.png, CC BY-SA 3.0, Last Accessed: April 18, 2017, https://commons.wikimedia.org/w/index.php?curid=8326140

Receiver Operating Characteristics, *Wikipedia Page*, Last accessed: April 18, 2017, https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Jason Brownlee, 2016, How to Build an Ensemble of Machine Learning Algorithms in R, Last accessed: April 20, 2017, http://machinelearningmastery.com/machine-learning-ensembles-with-r/

Manuel Amunategui, 2015, Bagging / Bootstrap Aggregation with R, Last accessed: April 20, 2017, http://amunategui.github.io/bagging-in-R/index.html

Asmita, Shruti; Shukla, K.K., "Review on the Architecture, Algorithm and Fusion Strategies in Ensemble Learning", *International Journal of Computer Applications (0975 – 8887)*, Volume 108 – No. 8, December 2014

Xiaodong Zeng, Derek F. Wong, and Lidia S. Chao, "Constructing Better Classifier Ensemble Based on Weighted Accuracy and Diversity Measure," *The Scientific World Journal*, vol. 2014, Article ID 961747, 12 pages, 2014. doi:10.1155/2014/961747

APPENDIX A

IMPLEMENTATION CODE SAMPLE

The following R code provides an example of implementation of two of the learners (Boosting using Decision Trees and Stochastic Gradient Boosting Model) for 80-20 cross validation method of the first 10,000 instances of the total dataset for this thesis. The implementation for all the other learners is similar. The implementation includes data pre-processing, normalizing, training the model using training data, using the model to predict on the test data and generating/evaluating the performance measures:

```
##############################################################################
# Boosting using Decision Trees (C5.0) and GBM (Stochastic Gradient Boosting Models)
# Reference: http://machinelearningmastery.com/machine-learning-ensembles-with-r/
# Reference: http://amunategui.github.io/bagging-in-R/
##############################################################################
# importing the libraries
# Hold out Cross validation 80-20
library(caret)
library(e1071)
library(class)
# package to print the cross table
library(gmodels)
# this library for the ksvm method
library(kernlab)
# importing the dplyr library
library(dplyr)
# for ensemble learning
```

```
library (caretEnsemble)
##############################################################################
# Preprocessing the data

# We will be working on the first 10000 values from the top
diabDataBoosting <- head(diabData,10000)

# view the summary and the properties of the data
str(diabDataBoosting)
head(diabDataBoosting, 100)
View(diabDataBoosting)
names(diabDataBoosting)
dim(diabDataBoosting)
summary(diabDataBoosting)

# Removing useless variables
# Removing weight as it has 97% missing values
diabDataBoosting <- subset(diabDataBoosting,select=-c(encounter_id, patient_nbr,
weight))

# drop large factors
diabDataBoosting <- subset(diabDataBoosting, select=-c(diag_1, diag_2, diag_3))
```

```
# Removing "?"
diabDataBoosting[diabDataBoosting == "?"] <- NA

# convert all NAs into 0
diabDataBoosting[is.na(diabDataBoosting)] <- 0

# dropping payer_code and medical_specialty as 52% and 53% respectively of the values
are missing
diabDataBoosting <- subset(diabDataBoosting, select=-c(payer_code,
medical_specialty))

# dropping admission_type_id, discharge_disposition_id and admission_source_id
diabDataBoosting <- subset(diabDataBoosting,select=-c(admission_type_id,
discharge_disposition_id, admission_source_id))

# binarise target value -- data for those admitted <30 days
diabDataBoosting$readmitted <- ifelse(diabDataBoosting$readmitted == "NO", 0, 1)

# Determining factor values
#factor_values<-sapply(diabDataBoosting,function(x)is.factor(x))
#names(factor_values)
#non_factor_values<-diabDataBoosting[,names(which(factor_values=="TRUE"))]
```

```
#names(non_factor_values)


#############################################################################
# Reference: http://amunategui.github.io/bagging-in-R/
# binarize data
charcolumns <- names(diabDataBoosting[sapply(diabDataBoosting, is.character)])
for (colname in charcolumns) {
  print(paste(colname,length(unique(diabDataBoosting[,colname]))))
  for (newcol in unique(diabDataBoosting[,colname])) {
    if (!is.na(newcol))
      diabDataBoosting[,paste0(colname,"_",newcol)] <-
ifelse(diabDataBoosting[,colname]==newcol,1,0)
  }
  diabDataBoosting <- diabDataBoosting[,setdiff(names(diabDataBoosting),colname)]
}
#############################################################################
# As we have binarized the nominal data, we need not normalize the data
# as nominal data cannot be normalized. No further transformation is required for other
columns
# however, we require normalization for columns 1 to 8


# normalize function
normalize <- function(x) {
```

```
  return (((x - min(x)) / (max(x) - min(x)))*100)
}

# testing normalize function
normalize(c(1, 2, 3, 4, 5))

# testing normalize function
normalize(c(10, 20, 30, 40, 50))

# normalization on column 1 to column 8 -- TODO
diabDataBoosting[,1:8] <- data.frame(lapply(diabDataBoosting[,1:8], normalize))
#############################################################################
# incorporate cross validation to create the training and testing data subsets
# change this value to change the cross validation approach
split=0.80
trainIndex <- createDataPartition(diabDataBoosting$readmitted, p=split, list=FALSE)
trainBoosting <- diabDataBoosting[ trainIndex,]
testBoosting <- diabDataBoosting[-trainIndex,]
#############################################################################
library(plyr)
library(dplyr)
# Changing to as.factor because of the following error:
```

```
# http://stackoverflow.com/questions/23357855/wrong-model-type-for-regression-error-
in-10-fold-cross-validation-for-naive-baye

# checking if the target value is a factor
str(trainBoosting$readmitted)

# convert the target value into a factor
trainBoosting$readmitted<-as.factor(trainBoosting$readmitted)

# checking if the target value is a factor
str(trainBoosting$readmitted)
##############################################################################
# Implementing boosting algorithms

# this will create 30 models using trainControl method
control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 7
metric <- "Accuracy"

# C5.0 (Decision Tree)
set.seed(seed)
fit.c50 <- train(readmitted~., data=trainBoosting, method="C5.0", metric=metric,
trControl=control)
```

```
# Stochastic Gradient Boosting
set.seed(seed)
fit.gbm <- train(readmitted~., data=trainBoosting, method="gbm", metric=metric,
trControl=control, verbose=FALSE)

# summarize results
boosting_results <- resamples(list(c5.0=fit.c50, gbm=fit.gbm))
# accuracy min, max, meadian and mean
summary(boosting_results)

# plot results
dotplot(boosting_results)

# plots the summary of the model and the relevant features
summary(fit.gbm)

# Reference: Determining the predicted values for the test dataset
preds <- predict(object=fit.gbm, testBoosting)

# to calculate the accuracy
```

```
CrossTable(preds, testBoosting$readmitted,
        prop.chisq = FALSE, prop.t = FALSE,
        dnn = c('predicted', 'actual'))

# Accuracy using confusion matrix
confusionMatrix(preds, testBoosting$readmitted)

# plots the summary of the model and the relevant features
summary(fit.c50)

# Reference: Determining the predicted values for the test dataset
preds <- predict(object=fit.c50, testBoosting)

# to calculate the accuracy
CrossTable(preds, testBoosting$readmitted,
        prop.chisq = FALSE, prop.t = FALSE,
        dnn = c('predicted', 'actual'))

# Accuracy using confusion matrix
confusionMatrix(preds, testBoosting$readmitted)
#############################################################################
```