2006

A study of machine learning performance in the prediction of juvenile diabetes from clinical test results

Shibendra Pobi
University of South Florida

A Study of Machine Learning Performance in the Prediction of Juvenile Diabetes from Clinical Test Results

by

Shibendra Pobi

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Lawrence O. Hall, Ph.D.
Dmitry Goldgof, Ph.D.
Rangachar Kasturi, Ph.D.

Date of Approval:
July 12, 2006

Keywords: ensembles, decision trees, neural networks, diabetes prediction, over-sampling

DEDICATION

To my fiance

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Lawrence Hall for his constant support and guidance in successfully accomplishing this project. I would also like to thank Dr. Jeffrey Krischer of the Moffitt Cancer Research Center for his expert guidance during various phases of the research. My sincere gratitude to Dr. Dmitry Goldgof and Dr. Rangachar Kasturi for providing their expertise and thoughtful suggestions.

This study has been partly supported by National Institutes of Health grant 1U54RR019259-01.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A STUDY OF MACHINE LEARNING PERFORMANCE IN THE PREDICTION OF JUVENILE DIABETES FROM CLINICAL TEST RESULTS

Shibendra Pobi

ABSTRACT

Two approaches to building models for prediction of the onset of Type 1 diabetes mellitus in juvenile subjects were examined. A set of tests performed immediately before diagnosis was used to build classifiers to predict whether the subject would be diagnosed with juvenile diabetes. A modified training set consisting of differences between test results taken at different times was also used to build classifiers to predict whether a subject would be diagnosed with juvenile diabetes. Neural networks were compared with decision trees and ensembles of both types of classifiers. Support Vector Machines were also tested on this dataset. The highest known predictive accuracy was obtained when the data was encoded to explicitly indicate missing attributes in both cases. In the latter case, high accuracy was achieved without test results which, by themselves, could indicate diabetes.

The effects of oversampling of minority class samples in the training set by generating synthetic examples were tested with ensemble techniques like bagging and random forests. It was observed, that oversampling of diabetic examples, lead to an increased accuracy in diabetic prediction demonstrated by a significantly better F-measure value. ROC curves and the statistical F-measure were used to compare the performance of the different machine learning algorithms.

CHAPTER 1

INTRODUCTION

1.1 Type-1 Diabetes

Type 1 diabetes occurs when the body's immune system attacks and destroys certain cells in the pancreas, an organ about the size of a hand that is located behind the lower part of the stomach. These cells - called beta cells - are contained, along with other types of cells, within small islands of endocrine cells called the pancreatic islets. Beta cells normally produce insulin, a hormone that helps the body move the glucose contained in food into cells throughout the body, which use it for energy. But when the beta cells are destroyed, no insulin can be produced, and the glucose stays in the blood instead, where it can cause serious damage to all the organ systems of the body.

Type 1 diabetes is usually diagnosed in children and young adults, and was previously known as juvenile diabetes. Scientists do not yet know exactly what causes Type 1 diabetes, but they believe that autoimmune, genetic, and environmental factors are involved.

Incidence of Type-1 diabetes in the US is estimated at about 30,000 cases annually and about 40 per 10,000 children. Type 1 diabetes accounts for 5% to 10% of all diagnosed cases of diabetes. According to the National Institute of Allergy and Infectious Disease (NIAID) the prevalence rate of Type 1 diabetes is approximately 1 in 800 or around 340,000 people in the US.

The risk of developing type 1 diabetes is higher than virtually all other severe chronic diseases of childhood. Peak incidence occurs during puberty, around 10 to 12 years of age in girls, and 12 to 14 years of age in boys. According to the Juvenile Diabetes Research Foundation as many as 3 million Americans may have type-1 diabetes. Each year over 13,000 children are diagnosed with diabetes in the U.S. That's 35 children each and every day.

As is evident from the above discussion, Type 1 diabetes is a serious medical concern not only in the United States but across the world. There has been quite a lot of research in the medical community on how to treat this disease. Our objective has been to analyze a Type 1 diabetes dataset from the Diabetes Prevention Trial -1 and study whether it is possible to predict the onset of Type-1 diabetes from medical test results using learned classifiers.

1.2 Project Overview

There has been a lot of study in the area of machine learning on data from the domain of diabetes, especially on the Pima Indian Diabetes dataset [25], [2], [11] from the University of California, at Irvine (UCI) repository. There has also been tremendous interest in using machine learning algorithms for post diagnosis care, like prediction of blood glucose levels to control the dosage of insulin [21] and the use of association rules to predict the occurrence of certain diseases in diabetic patients [28], [16]. However our study differs from both these approaches in the sense that we use a dataset that is not restricted to a particular ethnicity but to a specific type of diabetes, namely Type 1 diabetes in the juvenile population and our objective is not to monitor diabetic patients but learn a model to predict the occurrence of this type of diabetes by taking the patient's past medical records into consideration.

We primarily used two types of classifiers: C4.5 based decision trees [23] and Cascade Correlation [24] based neural networks, to predict diabetic cases from non diabetic ones by using subject test results. This is not known to be a difficult problem for Physicians, but the reader will see building a good predictive model was not trivial. Next, we used the same base classifiers to predict diabetes, but this time the attributes were the differences in test results, between consecutive tests of the same type for a subject. This approach has the promise of allowing a prediction that someone is susceptible to diabetes before any test results indicate they may have it. Both random forests of decision trees [2] and bagged classifiers [1] for both neural networks and decision trees were used.

Surprisingly, it was necessary to explicitly encode missing attributes to achieve over 95% accuracy in diabetes prediction for both decision trees and neural networks.

The ensemble classifiers provided the best accuracy. Decision tree classifiers were comparable to cascade correlation neural network classifiers. Of interest was the fact that approximately 80% accuracy can be obtained in predicting diabetes from differences in test results over time without using data from the last time period before diagnosis as diabetic.

Another aspect that was tested was whether oversampling of the minority class examples (i.e. the diabetic examples) would improve the prediction accuracy of the classifiers. The oversampling technique that was used was the Synthetic Minority Oversampling Technique (SMOTE) [5], which uses a nearest neighbor approach to generate synthetic examples. Although the oversampling technique did not improve the overall accuracy of prediction (using Bagging and Random Forests), the number of True Positives did show a significant increase, with comparably higher F-measure values.

CHAPTER 2

PREVIOUS WORK

2.1 Research on the Pima Indian Diabetes Dataset

Most of the work related to machine learning in the domain of diabetes diagnosis has concentrated on the study of the Pima Indian Diabetes dataset in the UCI repository. In this context, Shanker [25] used neural networks to predict the onset of diabetes mellitus among the Pima Indian female population near Phoenix, Arizona. This particular dataset has been widely used in machine learning experiments and is currently available through the UCI repository of standard datasets. This population has been studied continuously by the National Institute of Diabetes, Digestive and Kidney Diseases owing to the high incidence of diabetes. The study chose 8 particular variables which were considered high risk factors for the occurrence of diabetes, like number of times pregnant, plasma glucose concentration at 2 h in an oral glucose tolerance test (OGTT), diastolic blood pressure, 2-h serum insulin, body mass index, diabetes pedigree, etc. All the 768 examples were randomly separated into a training set of 576 cases (378 subjects without diabetes and 198 subjects with diabetes) and a test set of 192 cases (122 non diabetic subjects and 70 diabetic cases). Using neural networks with one hidden layer, Shanker [25] obtained an overall accuracy of 81.25% which was higher than the prediction accuracy obtained using a logistic regression method (79.17%) and the ADAP model [27] (76%). Many other papers have reported results on this dataset.

2.2 Prediction of Blood Glucose Level and Other Health Risks of Diabetes Patients

Research on diabetes data (other than the Pima Indian dataset), related to the application of machine learning techniques, has mainly focused on trying to predict and monitor the Blood Glucose Levels (BGL) of diabetic patients [21] or possible health risks of such patients ([28]

and [16]). In [21], a combination of Artificial Neural Networks (ANN) and a Neuro-Fuzzy Optimizer was used to predict the BGL of a diabetic patient in the recent future and then a possible schedule of diet and exercise as well as the dosage of insulin for the patient was suggested. Although the BGL predictions were close to the actual readings, the dataset was restricted to only two Type 1 diabetic patients, which raises doubts about it's usability for large groups. In another study, by Karim Al Jabali [10], artificial neural networks were used to model and simulate the progression of Type 1 diabetes in patients as well as to predict the optimum (or adequate) dosage of insulin that should be delivered to maintain the blood glucose level (BGL). The study dataset was comprised of 70 patients with 30,000 training instances and the attributes considered were Previous Glucose Level, Short Term, Mid Term and Long Term Insulin release as well as some other features like exercise, meal, etc. A back propagation neural network with four layers was used to simulate the diabetic patient's metabolism and also simulate the controllers delivering insulin. The results showed that the use of complex neural network architectures could effectively emulate the working of controllers that deliver insulin to Type 1 diabetic patients.

Neuro-Fuzzy systems have also been used by Dazzi et al. [8], for the control of BGL in critical diabetic patients, with the main objective of being able to predict the exact dosage of insulin with the least number of invasive blood tests. A combination of back propagation (BEP) neural networks and fuzzy logic were used to predict the variation in insulin dosage. The neural networks were employed to discover the relationships between variables and find the right rules and adjust membership functions. For training the neural networks, a set of 1000 randomly simulated BG values were used and the corresponding insulin infusion rates noted. The trained neural nets were then tested with a set of 400 unseen BG values and the predicted insulin infusion rates were monitored and used to build a nomogram. The Neuro-Fuzzy system was able to provide fine tuned variations in insulin infusion in response to small glycemic variations and maintain BGL better than conventional control systems.

Another area of research in Type 1 diabetes, using machine learning techniques has been in the study of the genetic data associated with the occurrence of Type-1 diabetes (T1DM). A number of recent studies have aimed at unraveling the genetic basis of T1DM with a focus on whole genome screenings of families with affected sibling pairs (ASPs). Pociot et al., [22],

studied the application of data mining techniques to detect complex interactions of genes underlying the onset of Type 1 diabetes (i.e. non linear interactions between multiple-trait loci). The dataset they studied, had the genetic data from the analyses of 318 micro-satellite markers in 331 multiplex families. The subjects included 375 ASPs, 188 unaffected sib pairs, 564 discordant sib pairs making up a total of 1586 individuals. Decision trees and neural network approaches were used to analyze the data. Both these techniques were not only able to identify all the major linkage peaks that were identified by other non parametric linkage (NPL) analyses, but also found evidence of some new regions of interest that affect the onset of diabetes on certain specific chromosomes. The data mining techniques proved robust to missing and erroneous data. Moreover, these approaches could predict the Type 1 diabetic patients from the non diabetics, with training using sets of combinations of fewer markers. This study also emphasized that inherited factors influence both susceptibility and resistance to the disease. Linkage analysis of ASPs could not identify protective gene variants, whereas data mining analysis with unaffected subjects were able to identify certain combination rules that occurred only in non diabetics. The rules on marker interaction were generated by decision trees which were validated using neural network analysis.

For experiments concentrating on predicting potential health risks of diabetic patients, the machine learning algorithm of choice for most researchers is association rule mining. In [28], the authors make a comparative study of association rules and decision trees to predict the occurrences of certain diseases prevalent in diabetic patients. In [16], they deal solely with association rule mining on diabetes patient data, to come up with new rules for prediction of specific diseases in such patients. A Local Causal Discovery (LCD [6]) algorithm [26] is used to study how causal structures can be determined from association rules and generate rules to map symptoms to diseases. Moreover, exception rule mining leads to more useful rules from a medical point of view.

2.3 Diabetes Prediction Using Health Risk Assessment (HRA) Questionnaires

In [20] the prediction of diabetes from repeated Health Risk Appraisal (HRA) questionnaires of the participants using a neural network model was studied. It used sequential

multilayered perceptron (SMLP) with back propagation and captured the time-sensitiveness of the risk factors as a tool for prediction of diabetes among the participants. A hierarchy of neural networks was used, where each network outputs the probability of a subject getting diabetes in the following year. This probability value is then fed forward to the next neural network along with the HRA records for the next year.

Results show improvement in accuracy over time, i.e. the study of the risk factors over time rather than at any particular instant, yields better results. With the SMLP approach, the maximum accuracy of prediction obtained was 99.3% for non-diabetics and 83.6% for diabetics at a threshold (of output probability from each neural network in the hierarchy) of 20%. While [20] focuses on the importance of time-sensitiveness of the risk factors in diabetes predictions using only neural networks, our study compares decision tree learning methods (including random forests [2]) and an ensemble of neural networks applied to a specific juvenile diabetes dataset. Our study also differs from [20] in the following aspects:

1. [20] used HRA records of employees from a manufacturing firm with ages of the subjects ranging from 45 to 64, while our subjects are all juveniles.

2. The attributes in the dataset in [20] are general health parameters like Body Mass Index (BMI), Alcohol Consumption, Back pain, Cholesterol, etc. which are completely different from the attributes that we deal here with, like Intravenous Glucose Tolerance, C-Peptides and other medical tests that are specific to Type 1 diabetes.

In the current study we have used data from the Diabetes Prevention Trial - Type 1 [15], which was the first large-scale trial in North America designed to test whether intervention with antigen-based therapies, parenteral insulin and oral insulin would prevent or delay the onset of diabetes. In [4] it was shown that a strong correlation between first-phase (1 minute + 3 minute) insulin (FPIR) production during intravenous glucose tolerance tests (IV-GTT) and risk factors for developing type 1 diabetes existed using the DPT-1 data. In [14] the asymptotic group of cases in the DPT-1 trial whose diabetes could be directly diagnosed by the 2-h criteria on Oral Glucose Tolerance Test (OGTT) was studied. Both these studies [14] [15] helped identify the tests we used for our training data.

CHAPTER 3

DIABETES DATASET

In this chapter, the diabetes dataset that has been studied is discussed in detail. We start with a discussion of the Diabetes Prevention Trial, from which the current data has been extracted and then move on to how the datasets were built and formatted to train the classifiers.

3.1 Diabetes Prevention Trial - 1

The Diabetes Prevention Trial-Type 1 (DPT-1) was the first large scale trial in North America designed to test whether intervention during the prodromal period can prevent or delay the onset of Type 1 diabetes. The major objective of DPT-1 was to determine whether intervention with antigen-based therapies, parenteral insulin for high-risk subjects (>50% likelihood of developing diabetes in the next 5 years) and oral insulin (or placebo) for intermediate-risk subjects (25%-50% risk in the next 5 years), would prevent or delay the onset of diabetes. The study was designed to identify relationships between insulin production during the IV-GTT and known risk factors for Type 1 diabetes in a large population at increased risk for development of diabetes. It was a multicenter randomized trial involving first-degree relatives (ages 3-45 years) or second-degree relatives (ages 3-20 years) of persons who began insulin therapy before the age of 40 years. As of December 1998, 59,600 individuals had been screened and 2199 had positive findings (>=10 Juvenile Diabetes Foundation units) on their ICA test (3.69%). The initial staging IV-GTT had been done for 1622 subjects. The mean age of the 1622 subjects was 11 years (range, 3-45 years); 56% of these subjects were female and 44% were male. Of the 1622 subjects, 81.6% were white, 9.7% were Hispanic, 2.6% were black, and 3.4% were members of unspecified ethnic background.

A total of 84,228 first degree and second degree relatives of patients with diabetes for islet-cell antibodies were screened; 3152 tested positive; 2103 of the 3152 underwent genetic, immunologic, and metabolic staging to quantify their risk; 372 of the 2103 had a projected five-year risk of more than 50 percent.

3.2 Dataset Description

The dataset was obtained as a set of three tables. The tables were imported into a MS Access database to facilitate the querying of test results for individual patients. The following tables describe the attributes in each of the three tables. Table 3.1 shows the attributes in the Subject table which had the details of the 711 patients whose medical records were considered.

Table 3.1 Details of the SUBJECT table

Attribute Name	Description	Comments
PATID	Patient ID	Each of the 711 subjects had a unique PATID.
DATERAND	Date of randomization	Date from which the medical test results were recorded.
RACE	Ethnicity of the subject	The possible values are: W-White B-Black H-Hispanic O-Other X-Unknown
SEX	Sex of the Subject	The possible values are: Male Female
DATEDIAG	Date of Diabetes Diagnosis	For non-diabetic patients this field had no data

Table 3.2 shows in detail the contents of the Test table, which listed the details of the clinical tests that were conducted on these 711 patients, the date on which they were conducted and the outcome of the tests

The TESTNAME is associated with each of the Test Categories mentioned in Table 3.2. A test category may be considered as a set of individual tests whose results in combination determine the outcome of that test category. A simple example would be for AB-IVGTT, which is constituted of the following individual tests: -

Table 3.2 Details of the TEST table

Attribute Name	Description	Comments
PATID	Patient ID	Foreign key from the SUBJECTS table.
TESTUID	Unique ID for the particular test	Primary key for the TESTRES table
TESTNAME	Name of the Test Category	Some of the test categories are ICA, AB-IVGTT, CO-IVGTT, IAA and so on.
DATEDRAW	Date on which the test was conducted	
OUTCOME	Result of the Test Category	The values in the OUTCOME field are low, normal, high, absent, pos and neg.

1. GLU-4
2. GLU1
3. INS1
4. INS3

The test results of these 4 tests combined, determine the outcome of AB-IVGTT which can either be low, normal or high. The OUTCOME field in the TEST table has the outcome of these test categories and not the results of the individual tests that constitute that test category.

Table 3.3 Details of the TESTRES table

Attribute Name	Description	Comments
TESTUID	Unique ID for a set of tests	Foreign key from table TEST
SUBTSTN	Name of the individual test	Some of the values were ICA, GLU-4, GLU1, INS1, PEP120 and so on.
RESULT	Test result value	Numeric value of the medical test

Table 3.3 lists the attributes of the third table, TESTRES, which had the actual clinical test result values for each of the tests listed in Table 3.2.

3.3 Attributes

This section gives a detailed listing of the attributes that constituted the diabetes dataset. The attributes of the dataset are comprised of clinical tests and two demographic parameters (Race and Sex). The clinical tests that were chosen were found relevant to the diagnosis and the onset of Type 1 diabetes as part of Diabetes Prevention Trial studies. Table 3.4 shows the list of attributes, along with their respective data types (e.g. Nominal and Continuous).

In all there were 42 attributes or features that were available. Of these, 5 nominal attributes, AB_IVGTT, CO_IVGTT, FPG, OGTT and SUBTSTN_R were left out, which either were rarely used or their outcomes were determined by the combination of some other test results that were already considered (i.e. whether AB_IVGTT is normal or not is determined by the values of GLU-4, GLU1, INS1 and INS3). Moreover, the test DQAB was excluded as it had the value "ABSENT" for all the 711 patients. So there are 36 attributes that have been taken into consideration, of which 34 are real valued (representing medical test results) and 2 are nominal (representing Race and Sex).

3.4 Test Results Dataset

The training data was extracted differently for diabetic and non diabetic patients. For diabetic patients, only the last set of medical tests before their diagnosis of diabetes was considered. Of the 256 subjects who had diagnosed diabetes, 201 of them had tests in the immediate vicinity of their diagnosis. The remaining 55 patients had their last set of tests before diagnosis, quite far back in time (approximately 3 months before the date of diagnosis) from the diagnosis date.

For non-diabetic subjects, all their test results throughout their recorded medical history were considered. For repetitive tests, only the results of the first test were used. There were two compelling reasons, as to why this approach was adopted:

1. For non diabetic patients there was no reference date (unlike diabetic subjects who had a date of diagnosis) to consider tests in that vicinity.

Table 3.4 List of attributes

Test Category	Attribute Name	Attribute Type
Demography	Race	Nominal
	Sex	Nominal
FPG	F	Continuous
GAD65 Antibody	GAD65	Continuous
Blood Glucose	GLU 0	Continuous
	GLU 1	Continuous
	GLU-10	Continuous
	GLU 120	Continuous
	GLU 30	Continuous
	GLU-4	Continuous
	GLU60	Continuous
	GLU90	Continuous
HBA1C Antibody	HBA1C	Continuous
Insulin Auto Antibody	IAA	Continuous
Islet Cell Antibody	ICA	Continuous
	ICA512	Continuous
Insulin Level	INS1	Continuous
	INS10	Continuous
	INS-10	Continuous
	INS3	Continuous
	INS-4	Continuous
	INS5	Continuous
	INS7	Continuous
Micro Insulin Auto Antibody	mIAA	Continuous
C-Peptides	PEP0	Continuous
	PEP1	Continuous
	PEP10	Continuous
	PEP-10	Continuous
	PEP120	Continuous
	PEP3	Continuous
	PEP30	Continuous
	PEP-4	Continuous
	PEP5	Continuous
	PEP60	Continuous
	PEP7	Continuous
	PEP90	Continuous

2. The consideration of the results of all distinct tests that were conducted on the subject reduced the number of missing data values in the dataset. On average, each subject had 30 data values from the 36 attributes.

However, for diabetic patients, since only their latest tests before diagnosis were considered, there were a considerable number of missing data values. One of the diabetic subjects had just one test result in the test set closest to the diagnosis date. So for this particular patient, the set of tests, before the final test (prior to diagnosis) was also taken into consideration.

3.4.1 Dataset with Nominal Attributes

This particular dataset had both nominal and real valued attributes. The two nominal features were Race and Sex, while all attributes corresponding to the different medical test results, were real valued. The class attribute had only two possible values, 0 and 1, representing the classes diabetic and non-diabetic respectively. Another important aspect was the representation of missing values. Missing attribute values in this dataset were represented by "?", irrespective of whether the attribute was nominal or real valued. The decision tree implementation tool that was used, i.e. usfc4.5, has the ability to handle missing data represented as "?". So no special encoding of such missing values was required.

This particular dataset, was used in the evaluation of prediction accuracy of learning techniques like decision trees (using usfc4.5) and ensembles that used decision trees like bagging and random forests. This dataset, will henceforth be referred to as diabetes dataset $\mathcal{D}$.

3.4.2 Dataset with Bit Encoded Nominal Attributes

For the neural network approach, the missing data values had to be handled differently. Moreover the nominal attributes had also to be translated into numeric attributes. The reason for this was that the usfcascor implementation (like most neural networks) could only handle numeric attributes. The changes that were made to the representation of the dataset were:

1. In the dataset, $\mathcal{D}$, for the decision tree approach, the missing data values were represented by "?". In the neural network dataset, we adopted a different approach to handle missing data. Each test attribute (which is a numeric field) had another attribute asso-

ciated with it. Let us call this new attribute, an indicator attribute for that test field. The indicator attributes could have only 2 possible values, -0.5 and 0.5. If a test field, had a valid data value, then the value of the corresponding indicator attribute was 0.5, otherwise if the test field value was missing, then the indicator attribute's value was -0.5. Moreover, the value assigned to the test field if the data was missing, was also -0.5, i.e. if the indicator attribute has a value -0.5 then the associated test attribute too has a value of -0.5.

2. The data values for each of the attributes were scaled to lie between -0.5 and 0.5. To scale the value of an attribute, the minimum and maximum values for that attribute were calculated by traversing through the dataset. The scaled value was then computed by the formula:

   Scaled Value = ((Actual Value - Minimum)/ (Maximum - Minimum)) - 0.5

   Where, Minimum: Minimum value of that attribute

   Maximum: Maximum value of the attribute

3. The two nominal attributes: Race and Sex, were split up into a total of 7 distinct attributes (5 for Race and 2 for Sex) each indicating one of the possible values of the nominal attribute. Each of these 7 attributes has a value 0.5 or -0.5. A simple example, would be, the attribute indicating the Race value "W", would be 0.5 if the race of the subject is "W", and all the other 4 Race attributes would have value -0.5. No separate indicator attributes were associated with these nominal fields. In case of a missing value, each of the attributes associated with that nominal feature (either Race or Sex) has a value of -0.5.

This dataset will be referred to as the modified diabetes dataset, $\mathcal{D}_{mod}$, in the latter sections.

3.5 Differential Dataset

The next phase of analysis, involved the comparison of the Decision tree and Neural Network approaches in predicting the onset of Type 1 diabetes by looking at the difference

in successive test results. The objective of such an experiment was to see if it was possible to predict with reasonable accuracy, whether a subject is potentially at risk of getting Type 1 diabetes by examining the change of different diagnostic test results over time. Ensembles of classifiers (both decision trees and neural networks) were used to improve the accuracy of prediction.

3.5.1 Differential Dataset with Nominal Attributes

This dataset was built by computing the difference in successive test results of the same test type for a particular subject. For non-diabetic subjects, all the tests throughout their recorded medical history (i.e. collected throughout the duration of the study) were considered. On the other hand, for diabetic patients, their medical tests from the date of randomization to the date of diagnosis of diabetes, except the last set of tests immediately before diagnosis, were considered. The reason for leaving out the last set of tests before diagnosis was because these results would have most likely differentiated the diabetic patients from the non-diabetic subjects, as they presumably enabled a diagnosis. For the 55 diabetic patients who had their last tests conducted a long time before their diagnosis, this last set of tests (which were conducted as far back as 3 months before diagnosis of diabetes) were considered. In the latter experiments, this dataset will be referred to as the differential diabetes dataset, $\mathcal{DD}$.

In the differential dataset, each example corresponds to a particular subject (either diabetic or non-diabetic). Each of the attributes in the dataset is a difference value between consecutive tests of the same test type. To build this dataset, the following steps were performed:

1. All the relevant test results for each of the 711 subjects were collected from the original database.

2. For each test type, the difference was computed between one test and the next and so on. For example, the test type F was repeated 4 times for a patient and so there were three attribute values corresponding to the 3 difference values (between the 1st F test value and the 2nd F test value, and between the 2nd test and 3rd test and so on).

3. Since each patient may have a test performed a different number of times, the number of difference values for each subject may be different. To calculate the number attributes in

the dataset, the maximum number of times a test is performed on a patient is computed for each of the test types. These maximum numbers minus 1, would give the maximum number of difference values for that test. Hence, the maximum number of differences for a test type determines the number of attributes associated to that test. For example, the test GAD65 was repeated a maximum of 15 times for a particular patient among all the 711 subjects, then the maximum number of difference values is 14 for GAD65. The attributes are named from GAD65_0 to GAD65_13. For subjects who have less than 14 GAD65 tests, the remaining values are treated as missing values. The total number of attributes in the dataset was calculated to be 473 including the 2 nominal attributes: Age and Sex.

Similar to the dataset $\mathcal{D}$ for the decision tree approaches, the differential data extracted was formatted into dataset $\mathcal{DD}$. The dataset has two nominal attributes, Race and Sex, and 471 real valued features. Each of these 471 attributes represent a difference value, between two consecutive tests of the same type. The Race attribute could have the value "W", "B", "H", "O" and "X". The Sex attribute could take the values "M" and "F". The class attribute had two possible values of 1 and 0, representing the two classes Diabetic and Non-Diabetic, respectively. The missing data values in the dataset, both for nominal and real valued features, were replaced by the character "?". The usfc4.5 implementation (i.e. the decision tree approach) can handle missing data represented in this manner, and so the dataset $\mathcal{DD}$ was used with usfc4.5 based decision trees and ensemble techniques like bagging and random forests that used decision trees for the individual predictors.

3.5.2 Differential Dataset with Bit Encoded Nominal Attributes

The differential dataset had to be reformatted to be used with the Cascade Correlation based learning technique. The usfcascor [11] tool was used in this case. The changes in the dataset were related to the representation of the missing values and the nominal attributes, Race and Sex. The following changes were made to the dataset: -

1. An indicator attribute was associated with each attribute in the differential dataset, and it indicates whether the associated difference attribute has a valid value or not. If the

associated difference attribute has a valid value then the indicator attribute value is 0.5 otherwise it is -0.5. If the indicator attribute is -0.5, then the value of the corresponding difference attribute is also -0.5.

2. The values of each of the 471 difference attributes were scaled between 0.5 and -0.5. In order to scale the attribute values, the maximum and minimum values of each of the difference attributes were calculated and then the scaled value was computed as follows:

   Scaled Value = ((Actual Data Value - Minimum)/(Maximum - Minimum)) - 0.5

   Where, Minimum: Minimum value of the difference attribute in the differential dataset.

   Maximum: Maximum value of the difference attribute in the differential dataset.

3. The nominal attributes Race and Sex were split up into separate attributes each representing individual values (i.e. 5 attributes for Race corresponding to the 5 different Race values, "W", "B", "H", "O" and "X" and 2 attributes for Sex corresponding to the values "F" and "M"). Indicator attributes are not required for any of these, as a missing nominal attribute value is represented by all the value attributes being -0.5 i.e. if a Race value is missing, then all the 5 attributes (representing the 5 different Race values) are -0.5.

This dataset will henceforth be referred to as the modified differential dataset, $\mathcal{DD}_{mod}$, in the latter sections.

CHAPTER 4

LEARNING APPROACHES USED

In this chapter, the different machine learning algorithms and their implementations that have been used in studying the DPT-1 data are discussed in some detail. Also discussed is an algoritm for oversampling of minority class examples, called SMOTE [5], which was used in combination of different learning predictors.

4.1 Cascade Correlation Neural Networks

The Cascade-Correlation neural networks [24] are based on two main concepts: the *Cascade Architecture* and the *learning method*. The *Cascade architecture* refers to the adding of hidden units one at a time and then freezing the input weights of the added units before adding the next new unit. The *learning algorithm* deals with creating and installing the hidden units. The addition of each hidden unit tries to maximize the *correlation* between the output of the new hidden unit and the residual *error* that the network is trying to eliminate.

The building of the cascor network starts initially with just the input nodes and the output nodes with no hidden units. The number of input and output units are determined by the problem and the nature of data representation. The input nodes are each connected to each of the output nodes, with each connection having an adjustable weight and the *bias* is permanently set to 1. The output values can just be a weighted sum of the inputs or the learning method may employ some non linear activation function. In cascor, the default activation function is a symmetric sigmoidal function (hyperbolic tangent) with an output range is -1.0 to +1.0. The hidden units are added one at a time, with each hidden unit receiving an input from each of the input nodes as well as the previous hidden nodes and having a output connection to each of the output nodes. The input weights are all frozen when adding a new hidden layer and only the output weights need to be adjusted.

As discussed earlier, there are no hidden units at the beginning except the input and the output nodes. These connection weights are trained as well as possible over the entire training set. Cascor does not use backpropagation, instead it uses either some well known single layer neural network learning scheme like Widrow-Hoff ("delta rule") or the Quickprop method. After some time, the training reaches an asymptote, whereby no significant reduction in error occurs over a certain number of epochs, then the error is noted. A new hidden unit is added with it's input connection weights frozen and the output weights are once again trained. The process continues until the error reaches an acceptable limit.

The *usfcascor* tool [11], developed at the University of South Florida, is a modification of Fahlman's original cascor implementation that has been modified to read the same data format as the USF implementation of C4.5 (the *usfc4.5* tool) as discussed in Section 4.2.

4.2 C4.5 Decision Trees

A decision tree is a learning mechanism, which uses a "divide and conquer" strategy to classify instances. It consists of leaf nodes labeled with a class and test nodes that connect two or more subtrees. Every test node computes a result based on some particular attribute, and this outcome decides along which subtree the example should go. Every instance starts from the root of the tree and traverses down to the leaf, depending on the attribute tests at each node until it reaches a leaf, whereby the class label of the leaf, determines the class prediction of the example. In order to build a decision tree, at every node an attribute needs to be chosen that would offer the best split among the training data. The decision is based on the value of information gain or gain ratio (i.e. the attribute that most effectively splits the training examples according to their class labels). The attribute that has the best information gain at any node, is used to split that node. The decision tree, could also be geometrically represented as partitioning the x-dimensional data space (defined by x attributes) by decision planes corresponding to the tests at each node.

Real data in most cases contain noisy data, and the divide and conquer approach to building trees tends to build such discrepancies into the classifier, which evidently leads to lower prediction accuracy on unseen test data. To overcome such *overfitting* of the training

data, decision trees are typically pruned. Smaller, pruned trees are not only easy to understand but also are generally more accurate on test data. Unpruned trees however are useful in ensembles like random forests, generally producing better accuracy percentage. Ensembles and random forests have been discussed in a latter section.

The C4.5 [23] algorithm for generating decision trees is based on the ID3 (Iterative Dichotmiser 3) algorithm. The splitting criteria used by C4.5 at each node of the decision tree is an information-based measure, the *gain ratio*, which takes into consideration different probabilities of test outcomes. Let D be a dataset and T be a particular test that has outcomes $T_1,\ T_2,\ T_3,\ ...,\ T_k$ which is used to partition D into subsets $D_1,\ D_2,\ D_3,\ ...,\ D_k$, such that D_i contains the set of examples having the outcome T_i for the test T. Now, if C denotes the number of classes and $p(D,j)$, the proportion of cases in D that belong to the j^{th} class, then the residual uncertainty about the class to which a case in D belongs is expressed as

$Info(D)$=-$\sum_{j=1}^{C} p(D,j)$ x $\log_2(p(D,j))$

and the corresponding information gain from a test T with k outcomes is mathematically represented as:

$Gain(D,\ T) = Info(D)$ - $\sum_{i=1}^{k} \frac{|D_i|}{|D|}$ x $Info(D_i)$.

Again, the potential information obtained by partitioning a set of examples is based on which D_i subset the example lies and is known as the *split information*, defined mathematically as:

$Split(D,\ T)$ = -$\sum_{i=1}^{k} \frac{|D_i|}{|D|}$ x $\log_2(\frac{|D_i|}{|D|})$

The *gain ratio* of a test is defined as the ratio of it's information gain to its split information. The C4.5 algorithm computes the gain ratio of every test at a node, and the one that yields the maximum value is used to split the node. Another alternative stopping criteria is to stop splitting a node when all the examples in the node belong to the same class (i.e. the gain ratios for all tests are zero). In the case of continuous attributes, it must be noted, that they can be used multiple times to split nodes. The C4.5 algorithm has some improvements over ID3 in terms of handling missing data and dealing with continuous attributes.

The *usfc4.5* tool is a tool based on C4.5 that has been developed at the Computer Science and Engineering department at University of South Florida. The dataset has two main files, a .data file and a .names file. The .names file contains the class labels and list of all the

attributes, with the attribute type (nominal, continuous, etc.) and for nominal attributes, the possible values. The .data file contains each example in a separate line and the attribute values for each attribute being separated by commas, with the last value being the class label.

4.3 Ensembles

Ensemble techniques [9] are learning algorithms that build a set of classifiers and then classify unseen examples by taking a (weighted or unweighted) vote of their predictions. Bagging, Boosting and Random Forests are some of the ensemble techniques. A necessary and sufficient condition for a classifier to be more accurate when used in ensembles is that the classifier be accurate (with accuracy of greater than random guessing) and diverse (the classifiers make different errors on new examples). There are three main reasons as to why most ensembles produce good results (better than a single classifier). The first is statistical, where a single classifier suffers from lack of sufficient training data i.e. too small compared to the size of the hypotheses space, $\mathcal{H}$. Building an ensemble of accurate classifiers and averaging the predictions, reduces the error of choosing the wrong classifier in the hypotheses space. The second reason is Computational, where a classifier can get stuck in a local optima e.g. for neural networks which use gradient descent or decision trees using a greedy approach. Finding the best hypotheses using neural networks or decision trees is an NP-hard problem and creating ensembles which calculate the local points from different starting points can give a better approximation. The third reason is representational, whereby the use of ensemble does not restrict the classifier to find the best hypotheses for a single training set, but uses different training sets and hence searches a greater space of possible best hypotheses within $\mathcal{H}$.

Following are some of the most popular methods of building ensembles:

1. Different sets of training examples are used for building each classifier in the ensemble, which works well for unstable classifiers like decision trees and neural networks.

2. Building different classifiers with different combinations of features.

3. Ensemble of classifiers can be built by partitioning the classes (if there are a large number of classes) into two groups, one having a single class and the other all the other classes

combined and then relabel the training and test sets. Repeating this procedure with each of the classes, builds an ensemble.

4. Introducing randomness into the method of building classifier e.g. randomly choosing the initial weights of the nodes in a back propagation neural network or randomly choosing an attribute from among the top ten attributes with the best information gain, to split a node on a decision tree.

4.3.1 Bagging

Bagging predictors [1] is an ensemble method for generating multiple versions of a classifier and aggregating the predictions. For numeric predictions, the results from individual predictors are averaged to give the final predicted value, while for classifiers, a plurality vote among the predictors decides the predicted class. One important requirement for bagging to yield better accuracy than a single predictor is for the underlying predictor to be unstable, where unstable is defined as when perturbing the training set causes significant changes in the predictor constructed.

To better illustrate how bagging works, let us consider a learning dataset $\mathcal{L}$ which consists of the data points $\{(y_n,\mathbf{x}_n)$, n=1, 2, ..., N$\}$, where the y's are either class labels or numeric results and the predictor's prediction is denoted by $\varphi(\mathbf{x}, \mathcal{L})$. If an ensemble of k predictors is built, then each of the $\{\mathcal{L}_k\}$ learning sets consist of N independently drawn observations from the same underlying dataset $\mathcal{L}$. For numeric predictions, i.e. if y is numeric, then $(\mathbf{x}, \mathcal{L})$ is given by the average of $(\mathbf{x}, \mathcal{L}_k)$ over all k i.e. by $\varphi_A(\mathbf{x}) = \mathcal{E}_{\mathcal{L}}\varphi(\mathbf{x}, \mathcal{L})$, where $\mathcal{E}_{\mathcal{L}}$ represents the expected value over $\mathcal{L}$, and $\varphi_A(\mathbf{x})$ denotes the aggregated value. In case y represents class labels with a value $j \in \{1,2, ..., J\}$, then the aggregation is done using voting. If N_j represents the number of votes for the j^{th} class from all the k classifiers, then the voted prediction of the ensemble is given by $\varphi_A(\mathbf{x}) = \text{argmax}_j N_j$. In bootstrap aggregation or bagging, different training sets are generated by repeatedly taking bootstrap samples $\{\mathcal{L}^{(B)}\}$ from $\mathcal{L}$. The aggregation techniques remain the same i.e. the average value for numeric predictions and the voted majority class for classification. The $\{\mathcal{L}^{(B)}\}$ each have N examples, drawn randomly *w*ith replacement from $\mathcal{L}$. The accuracy obtained using bagging depends on the stability of

the predictor φ. If changes in $\mathcal{L}$ produce substantial changes in the predictions, i.e. if the predictor is unstable, then the bagged results should give better accuracy. The implementation of bagged ensembles is pretty simple whereby individual predictors are iteratively created using different training sets and the class/numeric predictions are aggregated at the end. However, one tradeoff for better accuracy is that, the bagged predictions are less interpretable than the results obtained from a single classifier (like decision trees).

4.3.2 Random Forests

In Random Forests [2], multiple classification trees are grown and to classify each new example, the input vector is sent down each of these trees and the individual trees vote to decide on the final prediction. Whichever class gets the maximum votes, is predicted. To build a single tree in the forest, the following steps are followed: -

1. If the number of training examples are N, then N samples are randomly selected from the training set, *with replacement*, similar to bagging. This sampled dataset is used to build the tree

2. Of all the available features, say M, a subset m (such that m << M) of features are randomly selected at each node. The feature among the chosen m that produces the best split is used to split the node.

3. Each of the trees are grown to pure leaves, i.e. unpruned trees. This however does not lead to overfitting, as each tree is grown with a different set of training examples.

The best prediction accuracy is obtained when the subset of attributes, m, lies within a specific range. In the current study, we have experimented with different values of m to come up with the subset that generates the best results for the particular dataset used.

Some of the main features of random forests could be enumerated as:

- It runs efficiently on large databases, especially ones with huge number of attributes or features.

- It can give an estimate about the importance of a particular feature.

- It generates an internal unbiased estimate of the generalization error as the forest grows.
- It can effectively estimate missing data values and maintain accuracy when a large proportion of data is missing.
- It can balance error for unbalanced datasets (unequal class distribution).
- It provides for an experimental method to detect variable interactions.

4.4 Support Vector Machines

Support Vector Machine (SVM) [7] is a relatively new method of learning for two-class classification problems. The SVM maps the input vectors non-linearly to a high dimensional feature space and builds a linear decision boundary (i.e. a hyperplane) within this feature space. The main objective is to find an optimal separating decision plane that will give the best generalization among all the hyperplanes in the high dimensional feature space. The optimal hyperplane is defined as a linear decision function with the maximum distance between the vectors of the two classes as illustrated in Figure 4.1. To build the optimal hyperplane, only a small amount of training set examples need to be considered. These examples are called the *support vectors*.

Mathematically, given a training set of labeled instances (x_i,y_i), i = 1,2,..., n where $x_i \in R^d$ and $y \in \{1,-1\}^n$, the SVM involves the solution to the optimization problem:

$\min_{w,b,\xi}(1/2w^T w + C\sum_{i=1}^{n} \xi_i)$

subject to $y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i$,

$\xi_i \geq 0$

A function ϕ maps the training instances to a higher dimensional space and SVM finds the plane that separates them with the maximum margin in this high dimensional space. $C>0$ is the cost factor and $\phi(x_i)^T\phi(x_j)$ is called the *kernel function*. The common kernels are *linear*, *polynomial*, *radial basis function* and *sigmoid*.

The Support Vector Machine implementation that has been used here is the open source LIBSVM tool [3]. The basic steps required for testing a dataset involve:

- Transform the dataset to the software specific format.
- Try a few different kernels with different parameter values.
- Test the predictor with unseen examples.

SVM requires that each of the data attributes are real numbers. To achieve higher accuracy, it's generally good to scale the data values, so that larger numeric ranges do not dominate over features with smaller ranges. Linear scaling of data is commonly used. Also the most common model (or kernel) used is the *radial basis function* (RBF). This kernel non-linearly maps the samples to a high dimensional space and thus is able to handle non-linear relationships between the features and the classes. The parameters also need to be tuned to give the best results for choosing the optimal separating hyperplane. A grid search technique is generally employed to determine the best combination of cost factor, C and the kernel parameter γ.

4.5 Synthetic Minority Over-Sampling Technique

The problem of imbalanced datasets, where there is a wide disparity in the number of examples belonging to each class, has been studied by many like Japkowicz [17], Kubat and Matwin [18], Ling and Li [19], to mention a few. It had been shown that under-sampling of majority class examples enables better classifiers than over-sampling of majority

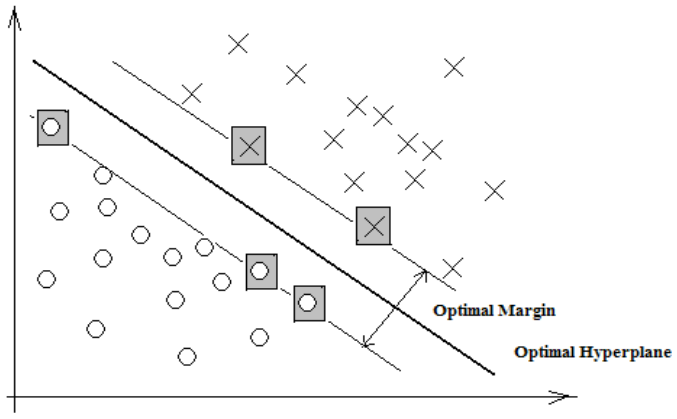

Figure 4.1 A representation of data in 2 dimensional space with support vectors marked in grey squares, with margins of separation

class examples. However, the most common approach to over-sampling has been to sample with replacement from the original data. Another over-sampling approach has been proposed by Chawla, et al. [5] called SMOTE (Synthetic Minority Over-sampling Technique). In this particular method, minority class examples are over-sampled by generating "synthetic" examples rather than over-sampling with replacement. The generation of synthetic examples operates in the feature space rather than in the data space. The minority class is over sampled by taking each minority class example and generating new synthetic examples along the line segments joining any or all of the k minority class nearest neighbors. Neighbors from the k nearest neighbors are randomly chosen depending on the amount of over sampling required. As an example, if the amount of over-sampling is 200%, two of the k nearest neighbors are randomly chosen and then one synthetic example is generated along each of these two nearest neighbor samples. To generate a synthetic example, the following steps need to be done:

1. The difference between the feature vector (sample) and the nearest neighbor is computed.

2. The difference is multiplied by a random number chosen from 0 to 1 and the product is added to the feature vector under consideration

This causes the selection of a random point in the line segment joining the two feature vectors and thus forces the decision region of the minority class to be more general. The synthetic examples create larger, less specific decision boundaries helping decision trees generalize better.

CHAPTER 5

EXPERIMENTS AND RESULTS

This chapter discusses in detail, the different experiments that were conducted on the clinical test results dataset and the differential datasets.

5.1 Predicting Type 1 Juvenile Diabetes Using Test Results

In this set of experiments, the datasets $\mathcal{D}$ and $\mathcal{D}_{mod}$ were used to test the performance of the different learning methods in predicting juvenile diabetes by looking at actual test results. The details of how these datasets were compiled have been discussed in sections 3.4 and 3.5.

5.1.1 Experiments with Decision Trees and Ensembles

For the decision tree approach, the dataset $\mathcal{D}$, with nominal attributes was used. The dataset $\mathcal{D}$ represented missing values with "?". The decision tree implementation that was used, was the *usfc4.5* tool.

A 10-fold cross validation, using usfc4.5 as the classifier, was done and the resultant overall accuracy obtained was 88.8%. The confusion matrix generated from the 10-fold cross validation is shown in Table 5.1. This result was a bit disappointing, given the fact that a physician almost always diagnoses diabetes accurately.

We suspected that for the 55 diabetic patients who had their last tests conducted almost three months before diagnosis, it would be hard to predict. So their prediction results were examined separately. The predictions for these 55 patients were isolated from the test folds of the 10-fold cross validation results and checked. 5 of the 55 subjects were classified as diabetics and the rest were classified as non-diabetics. The reason, as to why almost all of these 55 patients were wrongly classified might be attributed to the absence of test results in the immediate vicinity of their date of diabetes diagnosis.

Table 5.1 Confusion matrix after 10-fold cross validation using usfc4.5

Classified as →	Diabetics	Non Diabetics
Diabetics	188	68
Non Diabetics	11	444

5.1.2 Experiments with Neural Networks and Ensembles

In the following experiments, the dataset $\mathcal{D}_{mod}$, as described in Section 3.4.2, was used. A 10-fold cross validation using the usfcascor [11] neural network classifier yielded an overall accuracy of 99.72%, which was quite an astonishing figure. To better understand the reason for such a high accuracy rate, a decision tree was generated using this particular dataset. The decision tree, shown in Fig 5.1, reveals that the absence of two particular tests in a subjects records, improved the prediction accuracy from the previously obtained 88.8% (using dataset $\mathcal{D}$) to 99.85% (using dataset $\mathcal{D}_{mod}$).

From the decision tree in Figure:5.1, it is evident that the absence of the tests PEP 4 and GLU0 (as shown by their respective indicator attributes SUBTSTN_PEP_4_IND and SUBTSTN_GLU0_IND having value -0.5), plays a vital role in achieving such high accuracy. If these tests are not done, then the patient is likely to be diagnosed with diabetes.

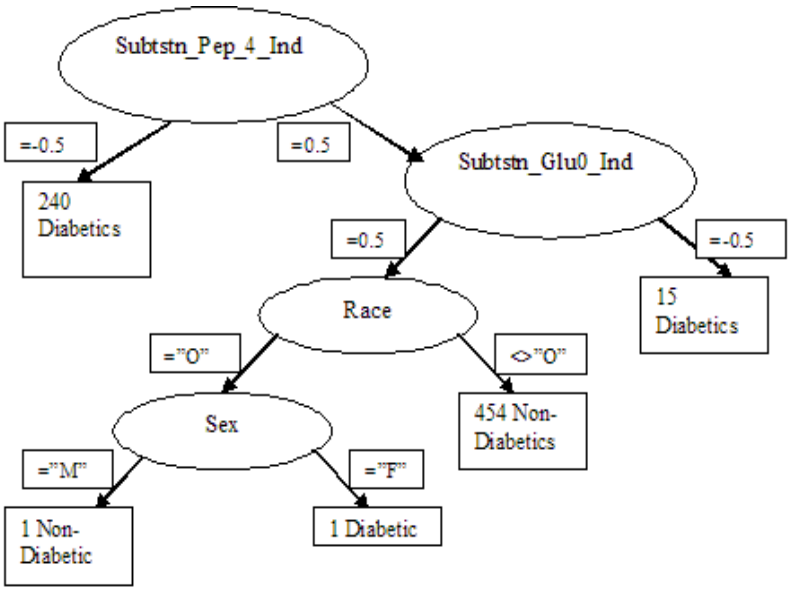

Figure 5.1 Decision tree generated using the modified data set

5.2 Type 1 Juvenile Diabetes Prediction from Differences in Test Results

The next phase of analysis, involved the comparison of the Decision tree and Neural Network approaches in predicting the onset of Type 1 diabetes by looking at the difference in successive test results. The objective of this experiment was to see if it was possible to predict with reasonable accuracy, whether a subject is potentially at risk of getting Type 1 diabetes by examining the change of different diagnostic test results over time. Ensembles of classifiers (both decision trees and neural networks) were used to improve the accuracy of prediction.

5.2.1 Experiments with Decision Trees and Ensembles

For the decision tree approach, the dataset $\mathcal{DD}$ was used in which the missing values were represented by "?" and the Age and Sex attributes were treated as nominal attributes. The 10-fold cross validation using a single decision tree yielded an overall accuracy of 66.8%. As can be seen, the average accuracy obtained from the 10 fold cross validation using the usfc4.5 classifier was quite low and was close to the accuracy obtained (64%) if all the examples were predicted to be of the majority (i.e. non-diabetic) class. Table 5.2 shows the confusion matrix that was the result of the predictions over a 10-fold cross validation using a single decision tree on the modified differential dataset, $\mathcal{DD}_{mod}$.

Table 5.2 Confusion matrix after 10-fold cross validation: using a single decision tree on the differential dataset

Classified as →	Diabetics	Non Diabetics
Diabetics	129	127
Non Diabetics	109	346

Next, random forests with usfc4.5 decision tree classifiers were used to see if ensembles of decision trees would yield better prediction accuracy. Different subsets of randomly chosen attributes were tested for the random forest approach, to come up with the best number of attributes which would yield maximum prediction accuracy. It was seen, that a subset of 50 attributes using an ensemble of 100 decision trees yielded maximum prediction accuracy. A 10 fold cross validation with random forests using 50 attributes and 100 trees yielded an average

accuracy of 71.31%. The confusion matrix of the random forests approach is shown in Table 5.3.

Table 5.3 Confusion matrix after 10-fold cross validation: using random forests with 50 attributes and 100 trees on the differential dataset

Classified as →	Diabetics	Non Diabetics
Diabetics	99	157
Non Diabetics	47	408

Although the overall accuracy of the random forest approach was quite an improvement over a single decision tree, the number of correct diabetic predictions was still low. The number of diabetic subjects who were incorrectly classified as non-diabetics far out numbered the ones correctly classified as diabetics. This low number of True Positives was a case for concern.

5.2.2 Experiments with Neural Networks and Ensembles

The 10-fold cross validation results using usfcascor on the modified differential dataset $\mathcal{DD}_{mod}$, yielded an overall accuracy of 72.71%, which is a marginal improvement over the random forests approach discussed in Section 4.3.2. The confusion matrix obtained from the 10-fold cross validation is shown in Table 5.4

Table 5.4 Confusion matrix after 10-fold cross validation: using a single neural network on the modified differential dataset

Classified as →	Diabetics	Non Diabetics
Diabetics	151	105
Non Diabetics	89	366

Although, the overall accuracy obtained using Cascor based neural networks did not improve much from the random forests results, a significant increase in correct diabetic predictions was noted. The True Positives increased from 99 to 151, which is a 52% increase over the random forest results.

Due to the increase in the number of true Positives using Neural Networks and the increase in accuracy with ensembles (random forests), the next set of experiments used ensembles of Neural Networks.

The *usfcascor* tool was modified in order to implement the ensemble of neural networks. To build an ensemble of bagged neural networks, the first step was to build the training sets of each of the neural networks that form part of the ensemble. The training examples for each of the neural networks, are randomly chosen with replacement (the replacement is determined by the percentage value specified and by default it is taken as 100%), from the original training set. A random seed value is chosen which determines the random samples that make up the training data of each individual neural network. Having built the training set, the neural network is built and the predictions of the unknown test examples are noted. The process is repeated for each of the individual predictors. To get the final prediction for each of the test examples, a majority vote of all the predictions from each predictor for that example is considered.

A bagged ensemble of Cascor based Neural Networks was used with different percentages for replacement while drawing training samples. When bags with 100% replacement of drawn samples (i.e. 100% of the training samples were drawn with replacement) were used, with an ensemble of 100 Neural Networks, the average accuracy obtained from a 10-fold crossvalidation was 77.21%. The confusion matrix obtained is shown in Table 5.5. Using an ensemble of neural networks also shows improvement with respect to some other statistical measures like the F-value, which has been discussed in Section 5.3.1. Experiments with bagged ensembles of neural nets with 90% replacement (to have greater variability in the training data) were also conducted on the modified differential dataset $\mathcal{DD}_{mod}$. The accuracy from 10-fold cross-validation with 100 neural nets was 77.07% which did not increase from the accuracy with 100% bags.

Table 5.5 Confusion matrix after 10-fold cross validation: using bagged ensemble of 100 neural networks on the modified differential dataset

Classified as →	Diabetics	Non Diabetics
Diabetics	162	94
Non Diabetics	68	387

The modified differential dataset $\mathcal{DD}_{mod}$ was also used with the decision trees and random forests to test whether these learning approaches would yield better results. The accuracy re-

sults obtained after a 10-fold cross validation using the decision tree approach on the modified differential dataset, $\mathcal{DD}_{mod}$ are:

- A single decision tree yielded an overall accuracy of 72.29% which is a significant increase from 66.8% which was obtained using the differential dataset $\mathcal{DD}$.
- Using bagging with 100% bags and 100 decision trees gave an accuracy of 79.04%.
- On using 100% bags with 100 trees but each tree in the ensemble are unpruned to make them more unstable, the average accuracy marginally increases to 79.76%
- Random forests with lg(n) (where n is the number of attributes) random attributes at each node and with 100 trees had an accuracy of 77.78%.
- Using random forests with subsets of 100 attributes and 100 trees yielded a higher accuracy of 80.31%.

As can be seen from the discussion above, random forests with a subset of 100 random attributes at each node and 100 trees, yielded a maximum average accuracy over a 10-fold cross validation of 80.31%. The confusion matrix is shown in table 5.6.

Table 5.6 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes and 100 trees on the modified differential dataset

Classified as →	Diabetics	Non Diabetics
Diabetics	156	100
Non Diabetics	40	415

A different set of experiments were conducted to test the effect of varying the random seed value that is used to split the 10 folds on using cross validation. A series of 5 experiments were done each with a different seed value and the minimum accuracy obtained was 78.91% and a maximum of 80.59% (which is slightly better than the best accuracy of 80.31% obtained using the default seed). The average accuracy over these five tests was 79.75%.

A summary of some of the prediction accuracies obtained from 10-fold cross validation using different learning algorithms on the differential dataset is shown in Table 5.7.

Table 5.7 Summary of prediction accuracies obtained from 10-fold cross validation using different learning methods on the modified differential dataset

Classifier	Average Accuracy Percentage
Single Neural Network	72.71
100% bags and 100 Neural Networks	77.21
90% bags and 100 Neural Networks	77.07
Single Decision Tree	72.29
100% bags and 100 Pruned Decision Trees	79.04
100% bags and 100 Unpruned Decision Trees	79.76
Random Forests with lg(n) attributes and 100 Trees	77.78
Random Forests with 100 attributes and 100 trees	80.31

5.2.3 Experiments with Support Vector Machines

Support Vector Machine (SVM) based predictors were also used to classify the differential data. As mentioned in Section 4.4, the tool that has been used is the LIBSVM [3]. Testing the prediction accuracy of the SVM using this tool requires that the following be done:

1. Transform the dataset into the format required by the LIBSVM tool.
2. Scale the data linearly.
3. The available kernels are:
    - Linear
    - Polynomial
    - Radial Basis Function
    - Sigmoid
    - Pre-computed Kernel (The kernel values are specified in a special file)
4. Cross validation is used to determine the best combination of cost, C and kernel parameters.
5. These parameter values are used to train the SVM.
6. An unknown test set is used to calculate the accuracy.

The LIBSVM software [3], [12], comes with additional tools, written as Python scripts, to scale the dataset as well as to perform a grid search to determine the optimum values of C and kernel parameters that would give the best prediction accuracy. A 10-fold cross validation is used with these optimized values, to obtain the highest overall accuracy.

The modified differential datatset, $\mathcal{DD}_{mod}$, was reformatted for LIBSVM [12]. LIBSVM [3], requires that all attribute values in the dataset be real valued, i.e. all nominal attributes need to be converted to numeric attributes. A m-valued attribute is split into m different numeric attributes and only one attribute has a value of 1 while the other (m-1) attributes are 0. In the case of dataset, $\mathcal{DD}_{mod}$, nominal attributes like Race and Sex are split up into their values but the attribute that is turned "on" is represented by the value 0.5 (instead of 1) and the remaining attributes have the value -0.5 (instead of 0). The dataset $\mathcal{DD}_{mod}$, also has the attribute values were scaled linearly as discussed in section 3.5.2. So the dataset did not require any further scaling. However, the $\mathcal{DD}_{mod}$ had to be modified as LIBSVM, needed the following changes:

- The class labels had to be changed to +1 (representing diabetic patients) and -1 (representing non-diabetic subjects)
- Each attribute value is preceded by the attribute serial number and they are separated by a colon. The .names file is not required for LIBSVM, as the only attribute data type is numeric.

The Radial Basis Function was used as the kernel and a grid search was conducted to come up with the best combination of C and the RBF parameter γ value. The grid search technique optimized the parameters C and γ for 10-fold cross validation by taking into account the entire training and test data in the dataset, $\mathcal{DD}_{mod}$. Hence the accuracy obtained is an overly optimistic accuracy for the SVM approach. The highest accuracy with 10-fold cross validation was obtained with a cost value of C=70 and γ=0.00048828125. The accuracy percent was 79.747%. As can be seen the best accuracy figures with SVM did not improve over the best random forests accuracy, however it performs better than bagged ensemble of neural networks (77.21%) as discussed in Section 5.2.2. The confusion matrix obtained after

a 10 fold crossvalidation using the LIBSVM tool with the best combination of parameters is shown in Table 5.8.

Table 5.8 Confusion matrix after 10-fold cross validation: using support vector machine on the modified differential dataset

Classified as →	Diabetics	Non Diabetics
Diabetics	178	78
Non Diabetics	66	389

If the confusion matrix obtained from LIBSVM in Table 5.8 is compared to the confusion matrix from Random Forests using 100 attributes and 100 trees in Table 5.6, which yields the best overall accuracy on the modified differential dataset, $\mathcal{DD}_{mod}$, then it can be seen that SVM yields significantly greater number of True Positives (i.e. predicts greater number of diabetic examples correctly). As will be shown later in Section 5.3.1, the F-measure value for the SVM results is higher than the F-value from Random Forests approach.

5.2.4 Experiments and Results Using SMOTE

The smoting implementation that has been used, is the smote tool developed at USF [5]. The implementation however, does not have a method to generate synthetic examples belonging to the minority class, in the presence of missing data values. In order to generate a synthetic example, the algorithm performs the following tasks:

- The distance between the features of the sample example and each of the remaining minority class examples is computed
- The nearest neighbors are ranked according to the distance.
- Of the k-nearest neighbors, the example in whose direction the synthetic example is to be generated is chosen randomly.
- The offset that is introduced in the synthetic example is the product of the difference between the features and a random number between 0 and 1. The direction of the offset is determined by the nearest neighbor i.e. the synthetic example lies somewhere in between the candidate and the nearest neighbor in the feature space.

However, for missing values, it is not possible to calculate the difference between the features, unless a value is assigned to the missing attribute (in either the candidate or the nearest neighbor). The strategy that we adopted was:

- If one of the attribute values were missing (either in the sample or the nearest neighbor) then the difference between examples for that feature is calculated as 10% of the product of the non-missing value and a random number between 0 and 1. The direction of the difference value (positive or negative) does not matter in this case, as the sum of squares of the differences are computed to rank the neighbors.

- If both the values were missing, then the difference would be zero.

These differences are computed for determining which minority class examples are the nearest neighbors of the candidate example. Once the nearest neighbor has been identified the synthetic example needs to be generated. The attribute values of the synthetic example, as mentioned earlier, are computed by adding an offset to the example attribute value in the direction of the nearest neighbor. The offset value is the product of a random number (between 0 and 1) and the difference between the attribute values. If either the candidate or nearest neighbor example has a missing value, then the offset is computed as 10% of the product of the non-missing value and a random number between -1 and 1. Finally, if both the candidate attribute value and the nearest neighbor attribute are missing, then the offset is zero and the synthetic attribute has the same value as that of the candidate.

In all the experiments that were performed, 10-fold cross validation was used to estimate the prediction accuracy of each of the learning techniques. In 10-fold cross validation, the entire dataset is divided into 10 disjoint folds, where the examples in each fold are drawn independently and randomly from the original dataset. The experiments are run 10 times, each time with a different set of 9 training folds and the remaining fold as a test set. If the whole data set is oversampled using SMOTE, then the resulting accuracy would be an overestimate. This is owing to the fact that the test examples (belonging to the minority class) would have also been oversampled and some of the synthetic examples generated from them, could have been in the training set used to build the classifier. To prevent such a bias from overestimating the accuracy, the original dataset was first split up into the 10 disjoint

folds (for cross validation) and then the training folds were oversampled while the test fold was kept as is. The same process was repeated for each of the 10 rounds of the 10-fold cross validation.

Another aspect that was tested was whether to oversample or smote, the difference dataset or to smote the medical test results and then compile the difference dataset for each round of the 10 fold cross validation process. In the first round of experiments that were conducted using oversampling, the differential dataset (used in neural networks) were used and the 10 disjoint folds for 10-fold cross validation were built. A number of experiments were performed on the oversampled dataset to come up with the best prediction accuracy. Random forests with subset of 100 attributes and 100 trees gives the best accuracy results with the modified differential dataset $\mathcal{DD}_{mod}$. So ensembles with random forests were tested using the oversampled dataset.

As discussed in section 4.3.2, the prediction accuracy using random forests, differs with the number of attributes that are randomly selected at each node for consideration as a possible test at the node and that the best accuracy occurs within a specific range of attribute subsets. In order to determine, the subset of attributes which yields the maximum accuracy, the random forests experiments were performed with different attribute subsets. The percentage of oversampling (or smoting) is also varied to observe how the accuracy figures vary with the number of synthetic examples in the training data. The sampling percentage determines the number of synthetic examples per minority class example that will be generated in the smoted dataset (e.g. a smoting percentage of 200% implies that for every minority class example that is smoted, two synthetic examples would be generated). The sampling percentage also determines how many nearest neighbor examples would be considered, i.e. for 200% sampling, for every minority class example, two nearest neighbors would be randomly selected and one synthetic example would be built along each of the two nearest neighbors' direction.

The average accuracy obtained after 10-fold cross validation was 78.2% when random forests with lg(Total Number of attributes) (i.e. lg(473)), an ensemble of 100 trees and a smoting sampling percentage of 100%. When the number of random attributes were increased to 50 in the previous combination (i.e. 100% oversampling and 100 trees) the accuracy percentage makes a slight jump 78.45%. The best overall accuracy of 79.89%, was obtained with

a combination of 50 random forests attributes and an ensemble of 200 trees and 100% oversampling (using SMOTE [5]) of the differential dataset. Table 5.9 illustrates the confusion matrix obtained from 10-fold crossvalidation using random forests with 100 attributes, 200 trees and 100% oversampling.

Table 5.9 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes, 200 trees and 100% smoting of difference values

Classified as →	Diabetics	Non Diabetics
Diabetics	150	106
Non Diabetics	37	418

The accuracy figures did not show any marked improvements over non-smoted results (as discussed in subsections 5.2.1 and 5.2.2). Another interesting thing to note is that the number of True Positives does not show any increase in numbers when the training data is oversampled, which should have been the case, since the presence of synthetic diabetic examples should have shifted the bias of the individual predictors towards the minority class. Therefore, a different approach to oversampling of the minority data in the training set was adopted to test if the prediction accuracy on diabetic examples could be improved.

In the previous set of experiments, the training examples that were over-sampled using SMOTE [5], were the test difference examples in the dataset $\mathcal{DD}_{mod}$. In the following experiments, however, a different approach to smoting was adopted. Instead of smoting the difference in test results, the medical test results of the diabetic (minority class) patients were smoted. For this purpose a different dataset was built. In order, to build the difference dataset, all the medical tests for each patient had been collected depending on whether the subject was diagnosed with diabetes or not, i.e. for the former case, the tests are till the date of diagnosis excepting the last tests before diagnosis and for the latter over the entire span of the medical data collected. This collection of medical tests data, a dataset was compiled for use with machine learning techniques. The number of times a single test was repeated in a patient was computed, and the maximum determined the number of attributes that would be associated with that test, i.e. of all the 711 patients, a particular patient had the PEP90 test repeated for a maximum of 20 times and as a result the dataset had 20 attributes associated with PEP90 named from PEP90_0 to PEP90_19. For patients having less than the

maximum number of tests, the remaining attributes data would be considered as missing, i.e. if a patient has 10 PEP90 tests then the values of the attributes from PEP90_10 to PEP90_19 would be missing for this patient. The nominal attributes are not bit encoded, as the SMOTE implementation can handle nominal attributes for oversampling. A 10-fold cross validation involved the following steps:

1. The dataset $\mathcal{DD}_{tests}$, was divided into 10 disjoint folds, and the examples in each fold, were randomly selected.

2. For each of the 10 rounds in the cross-validation, the data comprising the 9 training folds were collected into a single temporary dataset.

3. This temporary training data was then smoted (i.e. the minority class or diabetic examples were oversampled) depending on the sampling percentage.

4. The synthetic examples were then combined with the actual examples and a combined training set $\mathcal{T}_{temp}$ was built.

5. The differences in consecutive tests of the same test type for each patient were computed for both the dataset $\mathcal{T}_{temp}$ and the test fold (i.e. the remaining 10th fold extracted in step 1 from dataset $\mathcal{DD}_{tests}$) and the maximum number of differences for each test type was also noted. The values of the nominal attributes, Race and Sex, were kept unchanged.

6. The differential training dataset $\mathcal{DD}^{train}_{tests}$ was built using the difference values from $\mathcal{T}_{temp}$. The test set $\mathcal{DD}^{test}_{tests}$ was built from the differences from the test fold. The names file was created using the names file of the dataset $\mathcal{DD}_{tests}$ with the number of attributes associated with each test being reduced by 1, owing to the obvious fact that a set of n tests would yield (n-1) differences between consecutive values.

7. This differential dataset, $\mathcal{DD}^{train}_{tests}$, was then used to train the ensemble of predictors and the ensemble was tested on the test set $\mathcal{DD}^{test}_{tests}$.

8. For the next round of cross validation the steps were repeated from step 2.

In this set of experiments with oversampling, the medical test results dataset, $\mathcal{DD}_{tests}$, used in each of the 10 rounds of the 10-fold cross validation, was not normalized. Since the SMOTE

implementation, cannot handle missing values represented by "?", the missing values were replaced by the number -100000. The reason for choosing such a number was that no difference value had such a large value (i.e. no test result differed from the next result of the same test by a margin of 100000). Like before, random forests with different subsets of attributes and different oversampling percentages were tested to come up with the combination that would produce the best accuracy figures. As is evident, increasing the sampling percentage of SMOTE, increases the number of diabetic examples in the training set (actual and synthetic combined) and this shifts the bias of the learned classifier towards the diabetic examples. So the classifiers give better accuracy on diabetic test examples. However, the prediction accuracy on non-diabetic examples, go down significantly too, which results in the overall accuracy remaining almost unchanged.

Table 5.10, shows the confusion matrix from a 10-fold cross validation with random forests with the number of random attributes chosen to split each node being determined the value of lg(n) (where n is the number of attributes). An ensemble of 100 trees was used with the smote sampling percentage being 100%. The overall prediction accuracy obtained was 76.65%.

Table 5.10 Confusion matrix after 10-fold cross validation: using random forests with lg(n) attributes, 100 trees and 100% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	193	63
Non Diabetics	103	352

In the next experiment, random forest with a subset of 50 attributes was used. As can be seen from the confusion matrix obtained after a 10 fold cross validation and using 100 trees with 100% smoting, as show in Table 5.11, not only does the number of true positives i.e. correct diabetic predictions increase, but overall accuracy increases too, from 76.65% to 79.04%.

Table 5.11 Confusion matrix after 10-fold cross validation: using random forests with 50 attributes, 100 trees and 100% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	200	56
Non Diabetics	93	362

In view of the results of the increase in prediction accuracy with increasing the subset of attributes to 50, the next experiment used a subset of 100 attributes. The confusion matrix from a 10-fold cross validation with random forests of 100 attributes and 100 trees with 100% smoting (i.e. one synthetic example per minority class example in the training set is generated) has been shown in Table 5.12. Although the average accuracy showed a marginal increase to 79.18%, the number of true positives showed a significant decrease.

Table 5.12 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes, 100 trees and 100% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	189	67
Non Diabetics	81	374

In order to study if increasing the number of decision trees in the ensemble would increase the prediction accuracy, experiments with random forests using 50 attributes and 100% over-sampling, were conducted with different ensemble sizes. Table 5.13 shows the results of 10-fold cross validation with 150 trees which gives an average accuracy of 79.32% and Table 5.14 shows the results with 200 trees, with accuracy of 79.75%. With 150 trees, both the overall accuracy as well as the number of correct diabetic predictions increases, while with 200 trees although the overall accuracy increased marginally, the number of true positives decreased.

Table 5.13 Confusion matrix after 10-fold cross validation: using random forests with 50 attributes, 150 trees and 100% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	200	56
Non Diabetics	91	364

Table 5.14 Confusion matrix after 10-fold cross validation: using random forests with 50 attributes, 200 trees and 100% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	178	78
Non Diabetics	69	389

Another interesting aspect was to test the effect of the variation of sampling percent i.e. varying the number of synthetic examples that are used in the training set. From the

previous experiments, it became apparent, that increasing the minority class examples, by oversampling (i.e. generating new synthetic training data) shifts the classifier bias towards the minority class, thus resulting in greater accuracy in prediction of diabetic (minority class) patients. However, it was also noticed that the overall accuracy does not improve, owing to the increase in the number of false positives, i.e. non-diabetic examples being wrongly classified as diabetics. This prompted the study of whether, lowering the sampling percentage, would result in an optimal oversampling of minority class examples, such that the overall accuracy in prediction would reach a maximum without generating too many false positives. In Table 5.15, the 10-fold cross validation results with 50% oversampling, and random forests using 50 attributes and 150 trees, have been shown. The overall accuracy in this case was 79.04%.

Table 5.15 Confusion matrix after 10-fold cross validation: using random forests with 50 attributes, 150 trees and 50% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	176	80
Non Diabetics	69	386

Table 5.16 shows the 10-fold cross validation results using 50% smoting, 100 attributes and 150 trees, with the average accuracy being 80.59%.

Table 5.16 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes, 150 trees and 50% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	181	75
Non Diabetics	63	392

This increase in overall accuracy, was further investigated by increasing the sampling rate to 75% to see if it would lead to further improvements in the overall accuracy figures. As can be seen from the confusion matrix after 10-fold cross validation using 75% smoting, 100 random forests attributes and 100 trees, in Table 5.17 gives an accuracy value of 79.04%.

Random forests with 75 attributes and 75% smoting, using 150 trees, yielded the best overall accuracy of 80.73% which is marginally better than the best accuracy figures (of 80.31%) obtained using random forests without oversampling. The confusion matrix of this experiment has been shown in Table 5.18.

Table 5.17 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes, 100 trees and 75% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	188	68
Non Diabetics	81	374

Table 5.18 Confusion matrix after 10-fold cross validation: using random forests with 75 attributes, 150 trees and 75% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	196	60
Non Diabetics	77	378

The sampling percentage was also increased beyond 100% to 125% and 200% to study the best accuracy that can be obtained on just the diabetic examples. The best accuracy with 125% oversampling that was obtained was 80.03% when random forests with 100 attributes and 150 trees was used. Table 5.19 shows the confusion matrix from a 10-fold cross validation of the same experiment.

Table 5.19 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes, 150 trees and 125% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	202	54
Non Diabetics	88	367

With 200% oversampling rate, the number of diabetic examples (including the synthetic examples) in the training set outnumbers the non-diabetic examples and so the resultant predictors would be biased towards the diabetic class and hence should result in greater number of True Positives. The best overall accuracy with 200% oversampled data, obtained was 75.95% using random forests with subsets of 100 attributes and 100 trees. The confusion matrix from a 10-fold cross validation has been illustrated in Table 5.20.

As can be seen, the overall accuracy dips when the oversampling rate is increased to 200%, however the number of True Positives increases significantly from 196 (obtained with 75% smoting, that produced the best average accuracy) to 213.

Table 5.20 Confusion matrix after 10-fold cross validation: using random forests with 100 attributes, 100 trees and 200% smoting

Classified as →	Diabetics	Non Diabetics
Diabetics	213	43
Non Diabetics	128	327

Increasing the oversampling rate definitely improves the performance of the predictors on the minority class (i.e. diabetic) examples, however, the presence of synthetic data, also increases the number of False Positives too and as a result the overall accuracy figure suffers. The more the oversampling rate is increased the effect of False Positives become more pronounced and the average accuracy dips significantly.

5.3 Analysis of Different Learning Techniques

Two different statistical measures, the *F-measure* and *ROC curve*, have been used in this study to compare the performances of the different learning methods in predicting diabetes. The following two sections discuss the results of this analysis.

5.3.1 F-Measure

Before we go into the results and F-values, let us discuss what the F-measure means and how it is computed from the prediction results of a classifier.

The computation of the F-value for a particular predictor, two values need to be computed first. These are:

1. Precision: Percentage or fraction of the relevant test examples that were correctly predicted. Mathematically, Precision = True Positives/(True Positives + False Positives)

2. Recall: Percentage or fraction of the test examples that were predicted as relevant were actually relevant. Mathematically, Recall = True Positives/(True Positives + False Negatives)

Once the *precision* and *recall* values have been computed, then the F-value is calculated by the following formula:

F-value = 2 x *precision* x *recall* / (*precision* + *recall*)

The F-measure values were calculated from the confusion matrices that were generated using 10-fold cross validation for all the different learning techniques, in order to compare their prediction performance.

The F-values for the different learning techniques have been listed in the Table 5.21.

Table 5.21 Summary of F-measure values for different learning techniques

Datasets and Experiments	Learning Technique	F-value
Differential Dataset	Single Decision Tree	0.5222
	Random Forest=50, 100 trees	0.4925
Modified Differential Dataset	Single Neural Network	0.6089
	100% Bagging 100-NN	0.6532
	Random Forest = 100, 100 trees	0.68
	SVM with optimized parameters	0.712
Oversampled Dataset using Smote	100% Smote lg(n) random forests 100 trees	0.6993
	100% Smote random forests=50 100 trees	0.6966
	100% Smote random forest=100 100 trees	0.7186
	100% Smote, Random Forests=50, 150 trees	*0.789*
	100% Smote, Random Forests=50, 200 trees	0.7077
	50% Smote, Random Forests=100, 150 trees	0.724
	50% Smote, Random Forests=50, 150 trees	0.7063
	75% Smote, Random Forests=75, 150 trees	0.741
	125% Smote, Random Forests=100, 150 trees	0.74
	200% Smote, Random Forests=100, 100 trees	0.713

As can be seen from Table 5.21, oversampling using smote with 100% sampling percentage and random forests with subset of 50 attributes and 150 trees, yields the best F-value, although this method did not yield the best overall accuracy as shown in the confusion matrix in Table 5.13. Moreover, for the differential dataset, the best F-value was obtained was 0.789 with the random forests approach on the 100% oversampled data using 50 attributes and 150 trees.

Another very interesting fact that can be observed is that the oversampling of the training data using SMOTE, does not produce any significant improvement in the overall accuracy of prediction as can be seen from comparing the confusion matrices. However, the F-values for all learning experiments using oversampled data are significantly higher compared to the non-oversampled experiments. The reason being, that oversampling of minority class examples in the training set, shifts the bias of the classifier towards the minority class (in this case the diabetic class) and thus has a higher True Positives as well as higher number of False Positives, which brings down the overall accuracy but the high True Positive value improves the F-measure.

5.3.2 ROC Curves

Receiver Operating Characteristic (ROC) or the receiver operating curve [13], is a graphical method of comparing binary classification accuracy. It plots the values of two statistical measures, *sensitivity* and (1-*specificity*) computed from the results of a binary classifier system as it's discrimination threshold is varied. Sensitivity of a binary classifier, is defined as the proportion of of all positive predictions that were actually positive examples, i.e. mathematically:

Sensitivity = True Positives/(True Positives + False Negatives)
Specificity on the other hand is mathematically represented as:

Specificity = True Negatives/(True Negatives + False Positives)

In other words, in the context of this study, it is defines as the proportion of people who actually were non-diabetic among all the predicted non-diabetic examples. A classifier that has a high specificity, has low Type 1 error. The ROC can also be generated by plotting the fraction of True Positives against True Negatives. ROC curves are not only used in comparing classifiers but also in medical research, epidemiology or psychophysics.

A completely random predictor would generate a straight line at an angle of 45 degrees with the horizontal, from left bottom to top right. The reason being, as the threshold is raised, the True Positives and False Positives increase equally. Classifiers with ROC curves higher than this straight line are better than a random classifier. The statistic that is most

commonly calculated from a ROC for comparing classifier performance is the Area Under Curve (AUC).

The overall accuracy figures (from 10-fold cross validation) and the F-measure values computed from the different learning techniques clearly demonstrate, that the use of ensembles (with neural networks and decision trees) perform significantly better over single classifier performance. Moreover, the representation of differential data in the dataset $\mathcal{DD}_{mod}$ gives better prediction accuracy than the representation in dataset $\mathcal{DD}$. However, the difference in the performance of ensembles like random forests on oversampled and non-oversampled data is not that evident from the overall accuracy figures or the F-measure values. So the ROC curves were plotted for the random forests method using smoted and non-smoted data. Different combinations of random attributes and number of trees in the ensemble were tested to compare their ROC curves.

To build a ROC curve for an ensemble method like random forests or bagging, the technique that was used, was to plot the True Positive percent against the False Positive percent by varying the decision threshold. In this case the decision threshold was the number (or the percentage) of votes of the individual predictors comprising the ensemble that would decide the final classification (i.e. class label) of the particular example. For this purpose, the *usfc4.5* was modified, such that for every test example:

- The class prediction of each of the individual ensemble predictors is noted
- If the number of votes for the diabetic class exceed the chosen threshold then the particular example is classified as diabetic, e.g. if the threshold is 10% out of 100 trees in the ensemble, then if the number of votes for diabetic is 10 or more then the example is classified as diabetic.

The ROC curve shown in Figure 5.2, is a graphical comparison of the diabetic prediction accuracy (i.e. how the True Positive rate varies with respect to the False Positive rate) as the threshold values is varied. The threshold is varied in increments of 10% of the total number of trees in the ensemble, starting from 0 to 100 (in order to get more points on the curve, increments of 5% were made when increasing the decision threshold from 70% to 100%). As

the threshold is increased, the number of True Positives (TP) starts decreasing, while the False Positives (FP) increase.

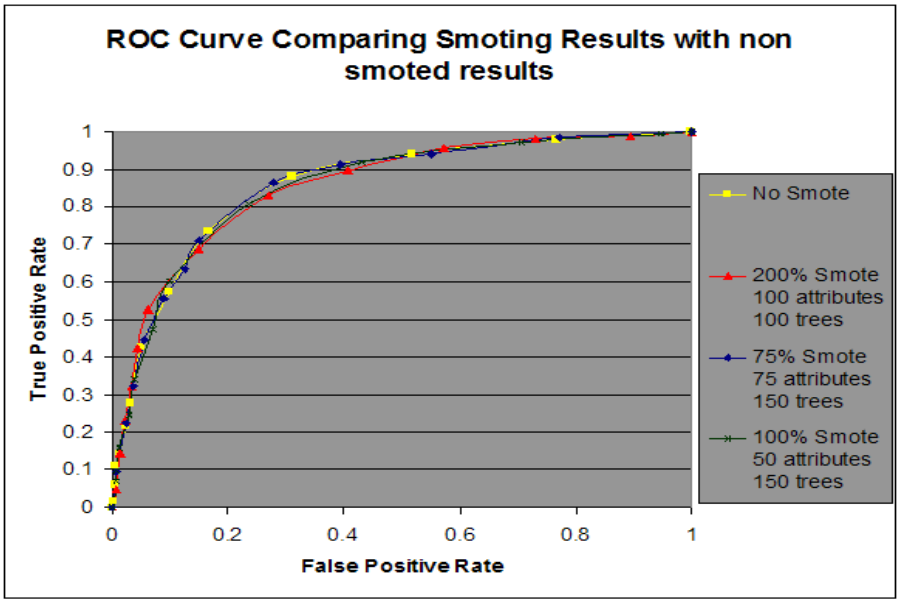

Figure 5.2 ROC curve for comparing the performance of random forests with and without oversampling

Since random forests with 100 attributes and 100 trees gave the best overall accuracy on the non-smoted dataset $\mathcal{DD}_{mod}$, this ensemble was tested against the results of the random forests method on smoted datasets with 75% oversampling, 75 attributes and 150 trees (which yields the best overall accuracy among the smoted experiments) and 200% smoting with 100 random attributes and 100 trees (which yields the best True Positive value). Also shown in Figure 5.2 is the ROC curve for random forests approach with 50 attributes and 150 trees trained on 100% oversampled data as this particular method produced the best F-value.

As can be seen from the ROC curve in Figure 5.2, no particular ensemble technique used with oversampling performs significantly better than the non-smoted results. Furthermore, no convex hull exists that would clearly distinguish whether oversampling would perform better under certain conditions or not.

CHAPTER 6

SUMMARY AND DISCUSSION

From the results of the experiments conducted, it can be seen that ensemble approaches with decision trees (random forest [2] and bagging [1]) give comparable and sometimes even better diabetes prediction accuracy for juvenile subjects (in the context of our dataset) than cascade correlation based neural networks. Another area where decision tree approaches have potential advantages over neural networks is that they are fast and easy to build and understand. However, it may also be noted that a bagged [1] ensemble of cascor networks are able to predict more diabetic subjects correctly than decision tree based approaches (as can be seen from the confusion matrices, discussed earlier). Further, a well tuned neural network of another type may well provide a higher ensemble accuracy.

Another interesting observation that can be made from the results is that the representation of data in the current context has important implications and affects the accuracy of prediction to a great extent. In the first series of experiments, where the medical test records were directly used for diabetes prediction, the accuracy figures improved by over 10% (from 88.8% to 99.85%) in the case of decision trees, when the learning set was modified (i.e. using associated indicator attributes for test existence and normalizing the data values between -0.5 to 0.5). Similarly, when the differential dataset was modified for the neural network approach (with indicator attributes and scaled data values) the accuracy of prediction showed a significant increase even with ensemble of decision trees (random forests [2] and bagging [1]) from 66.8% to a maximum of 80.31% (using random forests [2] with 100 attributes).

Decision trees have built-in methods to deal with missing attributes (for example allowing examples to go down multiple branches with weights as in our implementation). However, in this case a more explicit representation of missing values was very useful.

Moreover, it can also be inferred from the results, that the absence of some medical tests reveal important information which helps to predict the occurrence of Type 1 diabetes in juvenile subjects with better accuracy. Such information needs to be embedded, as done here with indicator attributes associated with each test result attribute, in the learning data to improve prediction accuracy.

REFERENCES

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] H. P. Chase, D. Cuthbertson, and L. M. D. et. al. First-phase insulin release during the intravenous glucose tolerance test as a risk factor for type 1 diabetes. *Journal of Pediatrics*, 138:244–249, 2001.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:341–378, 2002.

[6] G. F. Cooper. A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. *Machine Learning*, 1(2):203–224, 1997.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[8] D. Dazzi, F. Taddei, A. Gavarini, E. Uggeri, R. Negro, and A. Pezzarossa. The control of blood glucose in the critical diabetic patient: a neuro-fuzzy method. *Journal of Diabetes Complications*, 15(2):80–87, Mar-Apr 2001.

[9] T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[10] A. K. El-Jabali. Neural network modeling and control of type 1 diabetes mellitus. *Bioprocess and Biosystems Engineering*, 27(2):75–79, April 2005.

[11] S. Eschrich. *Learning from Less: A Distributed Method for Machine Learning*. PhD thesis, Dept. of CSE, Univ. of South Florida, 2003.

[12] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[13] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[14] C. J. Greenbaum, D. Cuthbertson, and J. P. Krischer. The diabetes prevention trial of type 1 diabetes study group, type 1 diabetes manifested solely by 2-h oral glucose tolerance test criteria. *Diabetes 2001*, 50:470–476.

[15] D. S. Group. The diabetes prevention trial of type 1 diabetes [abstract]. *Diabetes*, 43 (suppl 1):159A, 1994.

[16] W. Hsu, M. L. Lee, B. Liu, and T. W. Ling. Exploration mining in diabetic subjects databases: Findings and conclusion. In *6th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000.

[17] N. Japkowicz. The class imbalance problem: Significance and strategies. In *International Conference on Artificial Intelligence: Special Track on Inductive Learning, Las Vegas, Nevada*, 2000.

[18] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One sided selection. In *Fourteenth International Conference on Machine Learning, Nashville, Tennesse*, pages 179–186, 1997.

[19] C. Ling and C. Li. Data mining for direct marketing problems and solutions. In *Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, New York, NY, 1998. AAAI Press.

[20] J. Park and D. W. Edington. A sequential neural network model for diabetes prediction. *Artificial Intelligence in Medicine*, 23(3):277–293, November 2001.

[21] K. Patterson and W. Sandham. Neural network and neuro-fuzzy systems for improving diabetes therapy. In *20th Annual International Conference of the IEEE in Engineering in Medicine and Biology Society*, Hong Kong, China, 1998.

[22] F. Pociot, A. E. Karlsen, C. B. Pedersen, M. Aalund, and J. Nerup. Novel analytical methods applied to type 1 diabetes genome-scan data. *The American Society of Human Genetics*, 74(4):647–660, April 2004.

[23] J. Quinlan. C4.5: Programs for machine learning. *Morgan Kaufmann Publishers Inc.*, 302, 1993.

[24] S.E.Fahlman and C. Lebiere. The cascade-correlation architecture. *Advances in Neural Information Processing Structures*, 2:524–532, 1990.

[25] M. Shanker. Using neural networks to predict the onset of diabetes mellitus. *J Chem Inform Computer Science*, 36:35–41, 1996.

[26] C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 4(2-3):162–192, 2000.

[27] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *12th Annual Symposium on Computer Applications in Medical Care*, pages 261–265, November 1988.

[28] M. Zorman, G. Masuda, P. Kokol, R. Yamamoto, and B. Stiglic. Mining diabetes database with decision trees and association rules. In *15th IEEE Symposium on Computer-Based Medical Systems*, pages 134–139, 2002.