COMP3308 Assignment2 Report

Pima Indian Diabetes Study

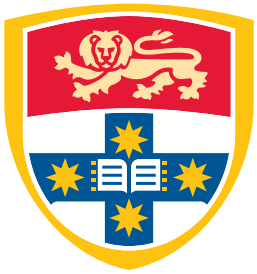
SID: 480472726

SID: 470527845

This report is submitted as partial fulfillment of
the requirements for the unit of
COMP3308 Introduction to Artificial Intelligence

School of Information Technology
The University of Sydney
Australia

12 March 2021



THE UNIVERSITY OF
SYDNEY

CONTENTS

CHAPTER 1

Project Overview

This report is based on the data analysis of Pima Indian Diabetes data set. The project group implements various different machine learning algorithms and respective classifiers to predict whether a person represented by some data observations is highly susceptible to or has suffered from type 2 diabetes. In conclusion, not only do we propose some feasible methods to improve the prediction accuracy but make suggestions to prevent type 2 diabetes as well.

1.1 Historical Overview of the Problem

Type 2 diabetes, constituent of more than 85 percent of all diabetes cases are a progressive condition,(1) whose cause is still uncertain. At present, genetic disposition and modifiable lifestyle factors are two possible explanations.(2) Pima Indians, historically resided in Arizona, USA are a group of people living in desert. After the European settlers invasion, the Pima people were forced to change their diet from low-fat, high-carbohydrate to one that ultimately derived more than forty percent of its energy from fat.(3) This tremendous change in lifestyle strikes the Pima people's health condition. Now more than half of the adult Pima Indians suffer from diabetes and 95 percent of whom are over-weighted. Though this event is a tragedy, the data about diabetes patients of Pima people become very valuable in understanding type 2 diabetes.

1.2 Project Aim

The purpose of this project is first to evaluate various machine learning classifiers' validity and accuracy on predicting the Type 2 Diabetes with the Pima Indian Diabetes dataset. The classifiers include both self-implemented classifiers and built-in classifiers of Waikato Environment for Knowledge Analysis(Weka), a data-analysis toolbox with various predictive modelling utilities. After analysing the performance of various classifiers, the project will delve deeper into the machine learning classifiers used. By detailed analysis and comparison before and after feature selection, this project will assess the correlation-based Feature Selection algorithm's effectiveness on different classifiers of Pima Indian Diabetes dataset.

1.3 Significance of the Research

The Pima Indian Diabetes data set is a classical data set with high significance in scientific understanding of type 2 diabetes. This project analyses existing classifiers delicately and implements some parameter-tuning methods on those classifiers, which will help other researchers choose the classifier effectively and shed light on further parameter tuning techniques on this data set.

Also, considering the fact that type 2 diabetes develops over a long period of time (years), it is particularly difficult for medical staff to make a exact diagnosis at the initial phase of the disease.(1) Therefore, type 2 diabetes patients are in bad need of a reliable preliminary diagnosis measure. In recent years, many clinical practices have demonstrated the feasibility of implementing machine learning method to assist Diabetes diagnosis.(4) Therefore, this project will further explore the effectiveness of machine learning methods in real medical practices.

CHAPTER 2

Data Set and Feature Selection

The data set used for this project is a modified version of the Pima Diabetes Data set. The original data set is from **National Institute of Diabetes and Digestive and Kidney Diseases**. The modified part is to substitute the missing value with the average of respective variable and refactor class to nominal values. However, it should be noted that the original database has certain limitations, especially that the observations in the data set are all from Pima Indian women patients aged 21 years or older.

2.1 Data Overview

The dataset includes 768 observations in total, with 9 different variables eight of which are numerical variable for predictive and one of which is a dependent variable to judge the correctness of the prediction. In regard of this class variable, there are 268 of 768 are yes(1), while the others are no(0). It should be noted that the class distribution of the target data set is noticeably unbalanced and therefore the classifiers trained may have a bias in favor of "no" class.

The detail of each variable is as follows:(5)

TABLE 2.1: Variable Details Table

#	Variable Name	Variable Type	Description
1	Pregnancies	Nominal	Number of times pregnant
2	Glucose	Nominal	Plasma glucose concentration
3	BloodPressure	Nominal	Diastolic blood pressure (mm Hg)
4	SkinThickness	Nominal	Triceps skin fold thickness (mm)
5	Insulin	Nominal	2-Hour serum insulin (mu U/ml)
6	BMI	Nominal	Body mass index (kg/(height in m)^2)
7	Pedigree	Nominal	Diabetes pedigree function
8	Age	Nominal	Age (years)
9	Outcome	Categorical	Class variable (0 or 1)

Apart from the initial null values data wrangling, all the nominal values of the data set are further filtered by Weka's built-in normalisation function to be within the range [0, 1]. This step makes the data fit into the machine learning model better. For example, without appropriate data normalisation, when testing k-NN classifiers, the results may be influenced unfavorably by distant discrete data.

2.2 Further Data Wrangling: Feature Selection

Before we import the data to the machine learning model, though we have implemented the nominal value normalisation, there is still more to consider. In respect of data modelling, some dimensions are less relevant than others. Therefore, though the data set in this project is rather small-scaled, only of nine dimensional, it is never too much to consider the feature selection. Feature selection is to measure the validity and relevance of features(variables) by a specific evaluation criterion and retrieve a subset of the original data set. Generally speaking, through

feature selection, redundant and irrelevant features in the original data set will be removed while the useful features are retained. This measure will reduce the dimensions and generalise the machine learning model, therefore reducing over-fitting. (6)

2.2.1 Correlation-based Feature Selection

There are many different algorithms to realise the feature selection like **Graph Structure Lasso**, **Forward Selection**, **Fish-Score** and etc. In this project, we choose to use **correlation-based** feature selection algorithm. The core of CFS method is to evaluate the value of feature subsets in a heuristic way.(7) A good subset often contains features that are highly related to the class, but the respective features are not related to each other.

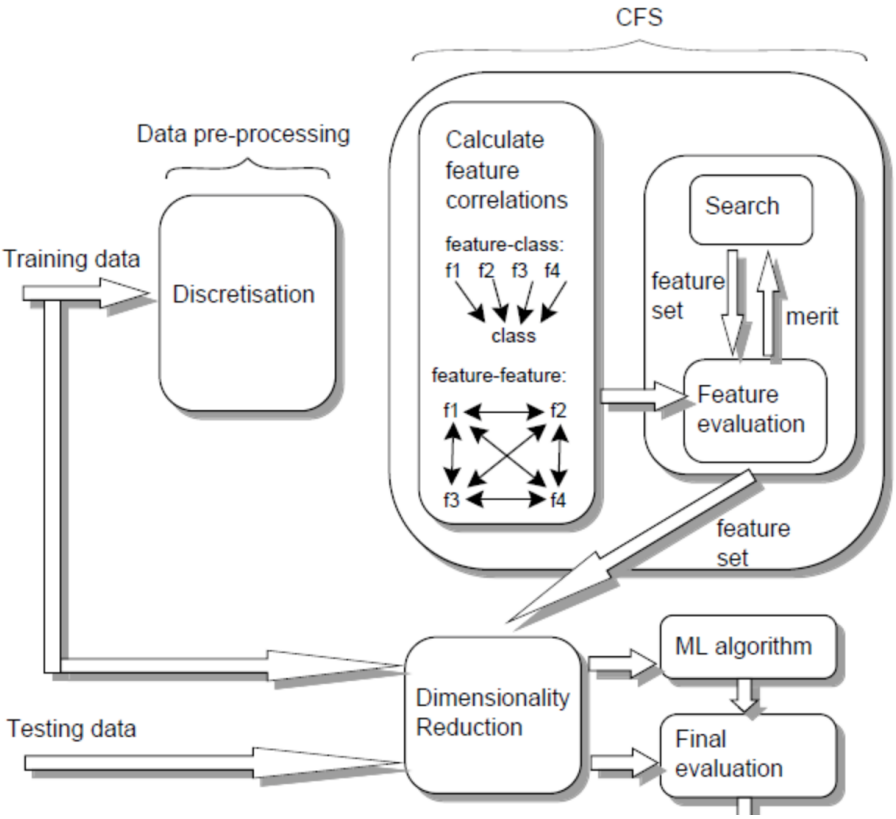


FIGURE 2.1: CFS Illustration(8)

In this project, we used Weka's built-in BestFirst Algorithm to do the feature selection. The features selected are as follows:

- Glucose
- Insulin
- BMI
- Pedigree
- Age

CHAPTER 3

Results Evaluation

After correlation-based feature selection, the data set is now prepared to be evaluated by classifiers. Similar to the situation in feature selection, there are a great many machine learning algorithms to implement the classifier. In this project, we evaluate many different classifier algorithms and some different implementations of the same algorithm. To find the optimal classifier for Pima diabetes data set, we use a systematic evaluation procedure. Firstly, we will consider the accuracy of the specific classifier; then we will analyze the best classifier and the classifier that performs most unsatisfactorily. Based on the accuracy analysis, we will take confusion matrix into consideration, trying to explore the distribution of the miss classified case to reconsider what the ideal classifier should be. At last, we will evaluate the time complexity to train and to make predication using different classifiers.After that, we had applied CFS and will observe how CFS influence the result of the different classifiers.

The details of classifiers evaluated are as follows:

- ZeroR
- 1-R
- k-Nearest Neighbor (Weka Implementation)
- k-Nearest Neighbor (Research Group Implementation)
- Naïve Bayes (Weka Implementation)
- Naïve Bayes (Research Group Implementation)
- Decision Tree (DT)
- Multi-Layer Perceptron (MLP)
- Support Vector Machine (SVM)
- Random Forest (RF)

3.0.1 Classifiers Introduction

In the field of machine learning, classification is a method to determine which classes the target variable belongs to given all the other independent variables. In short, classification is to predict the specific response of a data set. Therefore, as the name suggests, the classifier is a program to decide the classification and produce the dependent response.

ZeroR. To begin with, zeroR classifier in Weka is one of the simplest classifier. This method simply calculates statistical patterns of the input data set and output the result with largest possibility or algebraic average of the data. In most cases, the output of this classifier works as the baseline performance in control research.

1-R. OneR is a simple classification algorithm, which was invented in 1993. The core idea of this algorithm is to generate a single-layer decision tree. Specifically, this algorithm will calculate the results by every single feature and produce a different rule set respectively. The final features selected will be the one that achieved the highest performance during training.

K-Nearest Neighbor(K-NN). K-NN algorithm is another convenient and intuitive machine learning algorithm. In short, it simply tries to predict a test sample by looking at the k closest samples from the training data set. The closeness between different samples are defined through heuristic such as Manhattan distance.

Naïve Bayes. Naive Bayes classification is based on Bayes theorem. According to the theorem, if one wish to calculate the probability of a hypothesis H based on some given evidence E then he or she could follow this simple equation:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{3.1}$$

In simpler term, the probability is just the product of the probability of the evident occurring given that our hypothesis over the probability of the evident occurring. One important limitation of Naive Bayes comes from the "naive" hypothesis as suggested by the name. The hypothesis assumes that the importance of each of the feature are equal and each features are independent. The extent to which these hypothesis holds true in our data set will be further discussed in the following section.

Decision Tree. Decision Tree is an equally important supervised classifier in the field of machine learning. As the name implies, decision tree makes prediction by following a tree-like structure. The intermediate node will guide which of the branches that one sample should follow until it reaches the leaf node which represents a class.

The concept for constructing such tree using the training set is more complex than the concept of making a predicting with the tree. To construct a decision tree, one will use a heuristic function to determine the best feature to split the training data set. Often this heuristic function measures the purity of each path by observing how many different labels are in each of the divided branch. One example of this heuristic function is entropy. By comparing the information gained by each features for classification, one can make a more compact and effective decision tree.

Multi-Layer Perceptron. Multi-Layer Perceptron is a classification technique inspired by how our brains' neuron learns information. Each node in the hidden layers in a MLP is fully connected to all node in the next and previous layers. The signal of the previous layerx^{l-1} will be multiplied by some weight W, and added with some bias term b. This then goes through an activation function f to introduce non-linearity. In short we can express this using the equation:

$$x^l = f(w^l * x^{l-1} + b^l) \tag{3.2}$$

The input layer will goes through the hidden layer to the output layer in which the loss is calculated by comparing the true label. After which, back propagation will be taken place to update the weight in order to reach the local minimum of the loss function with respect to weight. This algorithm works as the activation function introduce non-linearity, making it able to approximate any continuous function based on the universal approximation theorem.

Support Vector Machine. Support Vector Machine(SVM) selects support vectors from each of the classes to determine the decision boundary that will linearly separate the classes. The training goal of SVM is to find support vectors that maximize the margin of the decision boundary in order to minimize the generalization error. One constraint that we set in SVM is that the trained decision boundary must ensure all samples from the training set are correctly classified. In order to ensure this using a linear margin, one will often have to increase the dimension of the problem. The combination of the problem with constraints can be solved using quadratic programming. Yet one issue that must be taken into consideration is that the run-time will increase dramatically as the dimension of the problem increases.

Random Forest. Random forest classifier is an algorithm based bootstrap aggregation which randomly select the data with replacement. For random forest, it will have M individual decision trees. At each step for constructing the tree, it will consider at most L randomly selected

features instead of all the available features. The training phase will train all M decision trees, and will make prediction by performing majority voting among the M decision tree. The core idea is to improve the accuracy by combining different trees.

3.0.2 Evaluation Method

In this project, the research group used stratified k-fold cross validation method to evaluate the classifiers. The reason of adopting this approach is that the data sample is rather limited and the cross-validation which implements a re-sampling procedure can simulate the unknown data well." This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds."(9) The general steps of implementing stratified 10-fold cross validation method is as follows:

- Split the data set into 10 randomized parts within which the data are of same distribution;
- Take one subset without repeating each time as the test set, use the other 9 sub-sets as the training set to train the model, and then calculate the model on the selected test set;
- Calculate the average of the Mean Squared Error(MSE) in 10 randomized tests

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

3.1 Evaluation Results

To have a better understanding of the classifiers, the research group test both Weka built-in classifiers and self-implemented classifiers. The results generated by stratified 10-fold cross validation method are further integrated to several data tables as follows:

TABLE 3.1: Weka Classifier Accuracy Table 1

	ZeroR	1R	1NN	5NN	NB
No feature Selection	65.1042%	70.8333%	67.8385%	74.4792%	75.1302%
CFS	65.1042%	70.8333%	69.0104%	74.4792%	76.3021%

TABLE 3.2: Weka Classifier Accuracy Table 2

	DT	MLP	SVM	RF
No feature Selection	71.7448%	75.3906%	76.3021%	74.8698%
CFS	73.3073%	75.7813%	76.6927%	75.9115%

TABLE 3.3: My Classifier Accuracy Table

	My1NN	My5NN	MyNB
No feature Selection	68.6329%	75.3862%	75.3828%
CFS	68.4979%	75.9211%	76.6883%

3.2 Evaluation on Classification Results

3.2.1 Results Accuracy Analysis

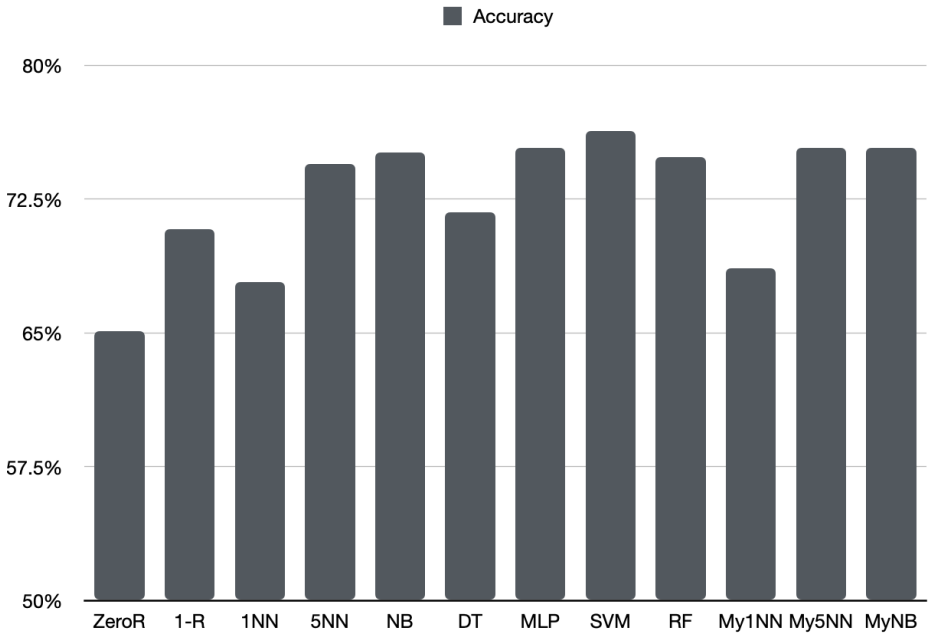


FIGURE 3.1: Accuracy Bar Plot

Combining Table 3.1, Table 3.2 and Table 3.3, we can plot a bar plot in regard of the accuracy of the machine learning models. From this plot, all the accuracy of different algorithms are above 65 percent, with five algorithms whose accuracy are above 72.5 percent, namely 5-Nearest Neighbor(5NN), Naive Bayes(NB), Multi-layer Perceptron(MLP), Support Vector Machine(SVM) and Random Forest(RF). For those algorithms whose accuracy is above 72.5 percent, the gap among them is rather small but for those algorithms whose accuracy is below 72.5 percent, the accuracy difference is obvious especially for ZeroR and 1NN classifier. Therefore, in regard of the accuracy, we do not recommend ZeroR and 1NN classifier for this data set.

Moreover, for all the classifiers, the accuracy is below 80%. In regard of the medical practices, this precision is not satisfactory. This finding suggests that finer tunning is required to optimize the performance of the classifiers until they can be put into practice. For the purpose of this discussion, thus far, MLP and SVM can compete for the best classifier on this data set.

There are also some more interesting findings when observing the performance of Naive Bayes and comparing RF and DT. As mentioned before Naive Bayes is most accurate based on the assumption that the variable are independent and equally important. But for this data-set, one example which can possibly violate this assumption is the feature Age and Pregnancies. As people are possibly more likely to have higher pregnancies as they are older, this breaks the independence of the features. Thus a more extension analysis was conducted on the effect of CFS on NB which will be discussed in future session.

On the other hand, the accuracy result of RF out performed DT by almost 3%. This is supported by the lecture that RF usually outperforms a single DT. One possible explanation for this result is that RF reduce the over fitting problem which might occurs in DT. DT will often be influence

by noisy information which in this case can be BMI or other less significant features which are filtered out by CFS. Yet, it should be kept in mind that Random forest have the randomness factor when selecting the features and the training data. This implies that more trial and errors need to be made to optimize the performance of the classifier.

3.2.1.1 Worst classifier and Best Classifier Analysis

In regard of the accuracy test, the worst classifier is ZeroR while the best classifier is Support Vector Machine. As examples, we will examine those two algorithms in details.

ZeroR. ZeroR algorithm is a very simple prediction model, which will create a frequency table for the data set and select the class with the higher frequency as the prediction value. In this data set, as mentioned in Chapter 2, the class results are in favor of "No" class, and therefore the ZeroR algorithm can get an accuracy of approximately 65 percent. However, it is expected that if the data set is larger and more balanced, ZeroR classifier will work even worse.

Support Vector Machine. SVM algorithm is a non-linear prediction algorithm which has good generalization ability. Its optimization goal is to minimize structural risk, not empirical risk and therefore this algorithm performs well in small data set.(10) According to past research on SVM, this classifier performs better when the expected class is binary (only two possible results).(9) In this project, the expected results are "yes" or "no", which corresponds to the empirical observation. This feature also explains why SVM performs well in this data set.

3.2.2 K-NN: K value Analysis

Another finding that caught our attention is the different performance for the different K-NN classifier. As for our classifiers experiments, both 1-NN and 5-NN classifiers use the same K-NN(k-Nearest Neighbor) algorithm but with different parameters. 1-NN records and calculates only 1 neighboring data while 5-NN records and calculate 5 neighboring data. Both My1NN and My5NN classifiers are the research group's implementation on K-NN algorithm while the 1NN and 5NN are Weka's implementation on K-NN algorithm.

Classifier	Accuracy	Running Time	False Positive	False Negative
1NN	69.0104%	0	119	122
5NN	74.4792%	0	88	107
300NN	71.4844%	0	22	197

TABLE 3.4: K-NN Classifier Table

From the table above, for this data set, Weka's classifiers suggests that 5-NN implementation works better than 1-NN implementation. However, it should be noted that this finding does not mean that k value and performance are positive correlated. If k value is too small, the classification result is easily affected by noise points. However, if k is too large, there may be too many points of irrelevant categories in the neighbor lists. For example we did a 300NN classifier test, the results is worse than 5NN. Therefore, to measure the effectiveness of k-nn classifiers, it is significant to analyse the real data performance in each model.

3.2.3 Confusion Matrix Analysis

Though accuracy is a significant criterion to judge the classifier, it is not the whole story. Considering that this data set is of medical purposes, the difference between false positive cases and false negative cases are very important. Because it is assumed that someone who have diabetes but classified as healthy and missed treatment is more serious than if someone is classified as unhealthy and decided to take more precise diagnosis. When we run Weka classifiers, we collect the confusion matrix generated for each classifier and make a data table and plot as follows:

Classifier	False Positive	False Negative
ZeroR	0	268
1-R	83	141
1NN	125	122
5NN	89	107
NB	85	106
DT	119	98
MLP	97	92
SVM	56	126
RF	93	100

TABLE 3.5: Confusion Matrix Table

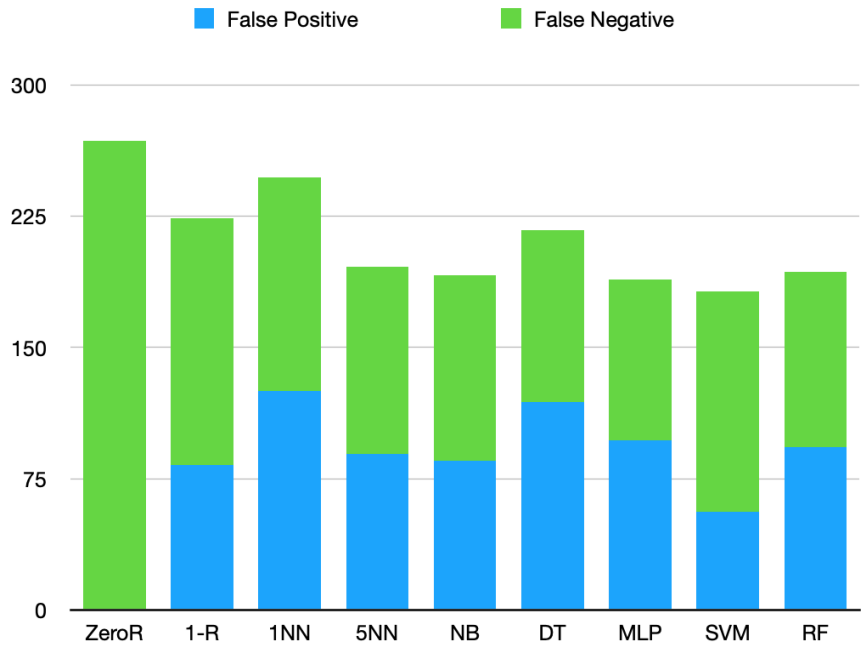


FIGURE 3.2: Confusion Cases Plot

The performance of difference classifiers in regard of the confusion cases distribution varies a lot. The most unbalanced distribution comes from ZeroR classifier while the results of 1NN, 5NN and MLP classifiers are distributed rather evenly. Also we notice that the performance of NB classifier and DT classifier are almost equal in this respect but there are more false negative cases in NB results. Therefore we think DT classifier performs better than NB classifier in term of confusion matrices.

3.2.3.1 Extensive Analysis by Taking Confusion Matrix into Consideration

ZeroR. Just as exaplained in accuracy analysis, the over-simplified algorithm of ZeroR makes it less accurate than others. In this case, it should be noted that ZeroR has no False Positive cases. The reason is that this algorithm predicts all input to be of Negative("No") class, and therefore there is no false positive case. This will result in some serious consequence in medical science as many patients might miss their chance of being diagnosed and be treated at the suited time.

SVM VS MLP. From previous discussions on accuracy, the researchers favored MLP and SVM as SVM is the best classifier with fewest number of confusion cases and MLP comes the second. However, within the confusion cases of these two classifiers, the distribution differs. The major confusion cases of SVM are false negative while those of MLP are false positive. In medical field, the false negative cases are more serious than false positive cases because the health condition of patients may deteriorate largely because of a false-negative diagnosis. Therefore, in this data set, in regard of confusion cases, we think MLP classifier works better than SVM classifier.

3.2.4 Running Time Analysis

When we run Weka built-in classifiers, the program will output the running time to build the model. This output value can be used to indicate the training time of each classifier. The program is running on a MacBook Pro(Mid 2015) of 16 GB 1600 MHz DDR3 memory and 2.5 GHz Quad-Core Intel Core i7 CPU and the data is as follows:

Classifier	ZeroR	1-R	1NN	5NN	NB	DT	MLP	SVM	RF
Time To Build Model(Sec.)	0	0	0	0	0.01	0.04	0.37	0.06	0.4

TABLE 3.6: Running Time Table

From the data table above, the running time of classifiers are very different. ZeroR, 1-R, 1NN, and 5NN classifiers run very fast with approximately 0 seconds to build while MLP and RF classifiers need 0.4s to build. Although this running time difference is rather small in this project, if we implement the classifiers to larger data set rather than this Pima Indian Diabetes data set, or run the clinical prediction in real-world situations, this running time difference should be taken into serious consideration.

To offer some further discussion on the topic, we have taken some further analysis of the possible implementation of the different algorithms. Each of the algorithms will be described in terms of it's worst-case performance at training and testing. For the sake of simplicity and consistency, the following discussion will refer to the number of samples as n and the number of features as f.

To start with zero R and one R. Zero R is an algorithm that simply return the majority class. In other word, one simply need to calculate the class with the most amount of count in the the training stage which takes $O(n)$ time. The test stage is also very straight forward as it only need to return the class with maximum number of category, i.e. in $O(1)$ time.

One R has a slightly more complicated training stage as compared to 0 R because it need to iterate over all the features and construct a tree for each of the respective features before selecting the best performing tree. Thus, the overall run-time for training will be $O(fn)$. However,

similar to 0R, the decision part is pretty trivial as one just needs to make predication based on the value of the best performing feature. This infers that the runtime for predicting a test can be done in $O(1)$ if the condition is categorical, and $O(log(q))$ where q is possible paths of the tree if the condition is nominal.

1NN and 5NN both have a training time of $O(nf)$ as they simply need to insert the sample of the training set into a data structure for fast look-up. However, one variation to improve the testing performance is to sort the training data which takes $O(fnlogn)$ time. In our example, as n is so small (around 800), $log(800)$ will be 9. This is equivalent to f in this data. The major overhead for kNN algorithm comes from the testing stage. In order to make a prediction, one must select the k closest training sample. Thus, to find the k closest tree using binary search, the run time will be $O(klogn)$.

By looking at the run time analysis above we can see that most of these algorithms have a training time bounded by $O(nf)$. As both n and f are small, the time taken to build model will be small, thereby justifying the result of our experiment.

In order to build the naive Bayes classifier, the classifier needs to calculate the mean and standard deviation for each of the features. Although this is also bounded by $O(nf)$ there is some extra computation needed to compute the mean and standard deviation which explains why the training time for this model is slightly longer than the previous approaches. The testing stage for the classifier runs in $O(f)$ as it simply computes the probability using all the features.

Here we conclude that the testing time for NB is faster for kNN where as the training time for NB is slower than kNN.

The run-time for decision tree is somewhat more complicated to analyze as it varies based on the heuristic used. For the sake of simplicity, we assume the heuristic used is entropy which has a run time of $O(nf)$. In the worst case, the tree will separate only one training sample and need to continue to divide the other $n-1$ samples leading to a left/right deep tree. Thus the height will be $O(n)$ making the training stage run in $O(n^2f)$ time. However, there are simpler decision trees that run in $O(fnlogn)$ time. The testing time will be simple to analyze as in the worst case, it will traverse through the node with maximum height which is $O(n)$ or $O(log(n))$. This explains why the training time for DT is the longest thus far.

MLP is a classifier that is very difficult to analyze as the run-time varies based on the number of epochs, the optimization technique applied, the number of hidden layers, the number of nodes in each hidden layer, and many other parameters. MLP is known notoriously for it's high run time to optimize each of the weights. This claim is supported by our experiment which demonstrated that MLP takes the longest time to train. For test time, it is also very hard to define a formula for the run time for a similar reason stated as before. In general, the test time will be predominated by the number of weight in the neuron network W which is very often very large.

The training time of SVM is $O(n^3)$. This is much longer than the other classifier that we have discussed so far apart from MLP. However, the testing time is still $O(f)$ as we just need to consider all the features and fit accordingly to the decision boundary. This is also consistent with our experiment results.

The last classifier that was used is RF. The complicity of RF is related to the maximum number of features being considered for branching l, the number of trees that are used m. Combine this with our analysis on the runtime for DT we can conclude that the training time is $O(mln^2)$ or $O(mlnlog(n))$. The testing time will be equivalent to the total time needed to run each of the decision trees which is equivalent to $O(n)$ or $O(log(n))$.

All in all, after comparison, it was discovered that all classifiers will run in acceptable training time except for MLP and RF which took significantly longer time. However, in terms of testing time, we propose that except for MLP, all will run in an acceptable time frame. Thus, with all those in mind, we might favor SVM instead of MLP as the best performing classifier as it not only has high performance but also have acceptable training and test time.

3.2.5 Feature Selection Analysis

As introduced in Chapter 2, feature selection is an important parameter tuning technique in machine learning field to improve the performance. Generally speaking, by selecting significant features, the data set can reduce the dimension and therefore decrease the running time of modelling and improve the overall accuracy. Nevertheless, reducing the number of features can also avoid the problem of over fitting as classifier will be less influence by noise that are irrelevant to the actual hidden pattern. However, the feature selection process is rather complicated because the change of a single feature has huge influence on the machine learning model as a whole. (6)

In this project, we are using best-first search method to implement correlation-based feature selection(CFS). CFS primarily determines the number of features in a selected subset by evaluating feature subsets and ranking feature subsets rather than individual feature. CFS method first calculates the feature-class and feature-feature correlation matrices from the training data set, and then searches the feature subset space with best first search algorithm. The CFS method starts with an empty set(S) generating all possible single feature and calculates the Merit Value of the feature.

$$Merits_s = \frac{kr_{cf}^{-}}{\sqrt{k + k(k-1)r_{ff}^{-}}}$$

The algorithm will add the one with the largest merit value Enter to set S, and then add the second largest feature to set S. If the features selected have a merit value less than the original merit values in set S, this method will remove the feature with the second largest merit value. Then it will traverse all the features to find the feature combination with largest feature value. The time complexity of this CFS method is bounded by $O(m \cdot n^2)$.

$$O(m, n) = m \cdot \frac{n(n-1)}{2}$$

In the equation above, m is the number of features in the subset, n is the number of all features. Therefore, the CFS method is an efficient algorithm with acceptable time complexity, which can be implemented to data set with reasonable time. In this project, we used Weka built-in implementation to do CFS. It evaluated all the nominal data in the training data set and selected five best variables, which are glucose, insulin, BMI, pedigree and age. Then we use the new data set after CFS to run the machine learning models. The results will be analyzed in the following sections.

3.2.5.1 Accuracy Rate Difference

The table and plot below show the accuracy change before and after the CFS. The improvement is a relative proportion calculated based on the data before CFS.

From the data above, for most classifiers, the accuracy of the prediction improves after CFS except for ZeroR, 1-R, 5NN and My1NN while My1NN is the only classifier whose performance decreases after the CFS. Generally speaking, in regard of the accuracy, to implement the

Model Name	Accuracy	Accuracy(CFS)	Improvement
ZeroR	65.1042%	65.1042%	0%
1-R	70.8333%	70.8333%	0%
1NN	67.8385%	69.0104%	1.7274%
5NN	74.4792%	74.4792%	0%
NB	75.1302%	76.3021%	1.5598%
DT	71.7448%	73.3073%	2.1778%
MLP	75.3906%	75.7813%	0.5182%
SVM	76.3021%	76.6927%	0.5119%
RF	74.8698%	75.9115%	1.3913%
My1NN	68.6329%	68.4979%	-0.1967%
My5NN	75.3862%	75.9211%	0.7095%
MyNB	75.3828%	76.6883%	1.7318%

TABLE 3.7: CFS Accuracy Table

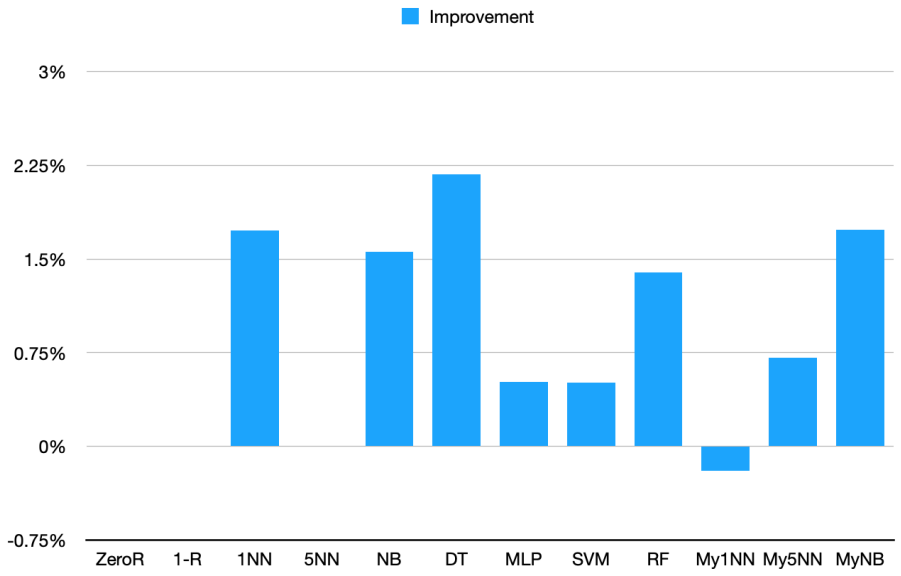


FIGURE 3.3: CFS Accuracy improvement Rate Bar Plot

feature selection before running classifier is, in most cases a recommended approach. Especially for those large data sets with a great many different features, to use feature selection in pre-processing the data will reduce the dimension and therefore decrease running time. This is consistent with our hypothesis in section 2.2. But it should be noted that for some special classifier implementations and data sets, this way of pre-processing the data may backfire. Another important factor to take into consideration is that since RF has some random factor in it, the comparison might not be appropriate as different features might be selected for constructing the decision tree and different training data might be included in training.

Naive Bayes Classifier. Moreover, from the two graphs above depicting the performance improvement after CFS, it is obvious that the Naive Bayes Classifier achieved the biggest progress. Specifically, the BN classifier improved by 1.5 percent in accuracy. We think this improvement is due to the important hypothesis of NB algorithm: features are equally weighted and each

feature is independent. Before applying CFS, there are many irrelevant features which breaks the assumption of naive Bayes. By CFS method, less relevant features are filtered and relatively, NB classifier will focus more on significant features.

3.2.5.2 Confusion Cases Difference

After looking at the change in accuracy, the next step is to observe how CFS affects the ratio of false positive and false negative cases when predicting the data.

Classifier	FP	FP(CFS)	Improvement	FN	FN(CFS)	Improvement
ZeroR	0	0	0%	268	268	0%
1-R	83	83	0%	141	141	0%
1NN	125	119	4.8000%	122	119	2.4590%
5NN	89	88	1.1236%	107	108	-0.9346%
NB	85	67	21.1764%	106	115	-8.4906%
DT	119	108	9.2436%	98	97	1.0204%
MLP	97	90	7.2165%	92	96	-4.3478%
SVM	56	53	5.3571%	126	126	0%
RF	93	83	10.7527%	100	102	-2.0000%

TABLE 3.8: CFS Confusion Cases Table

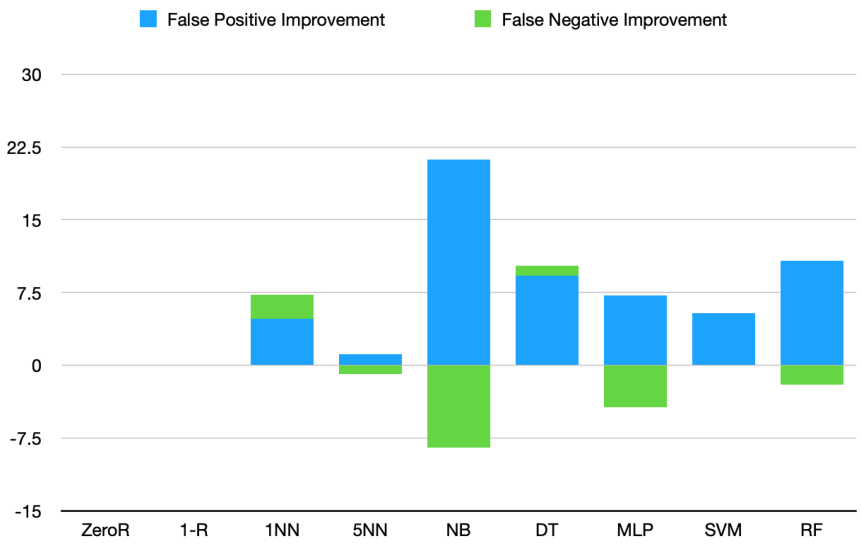


FIGURE 3.4: CFS Confusion cases improvement Rate Bar Plot

It should be noted that for NB, MLP and RF classifier, the false positive confusion cases decrease but the false negative confusion cases increase. Considering the fact that in medical field, the false negative situation is more serious that the false positive situation, it is complicated to analyse whether CFS really improves the confusion situations for those three classifiers. Therefore, in regard of the confusion cases, the CFS may lead to the good performance improvement but for some specific classifiers, more data should be tested to analyse the real situation. ·

	1NN	My1NN	5NN	My5NN	NB	MyNB
Without CFS	67.8385%	68.6329%	74.4792%	75.3862%	75.1302%	75.3828%
With CFS	69.0104%	68.4979%	74.4792%	75.9211%	76.3021%	76.6883%

TABLE 3.9: My Implementation VS Weka Implementation(Accuracy)

3.2.6 Self-implementation Classifier VS Weka-implementation Classifier

In this project, the research group implemented three different classifiers using naive bayes algorithm and k-nearest neighbors algorithm, which are My1NN, My5NN and MyNB classifier. From the table 3.9, before CFS all three classifiers are slightly more accurate than the Weka-implementation classifier. This difference is caused due to our implementation of tie breaker. After applying CFS to the data set, My5NN classifier and MyNB classifier still perform better than Weka Counterparts but My1NN classifier's performance drops and perform worse than Weka's 1NN classifier.

CHAPTER 4

Conclusion

This project mainly evaluated various different machine learning classifiers, trying to figure out the most suitable classifier for Pima Indian Diabetes data set. In addition, the project also discussed some simple optimisation techniques including parameter tuning and feature selection. The detailed conclusions are as follows:

4.1 Classifier Selection

According to our evaluation process described in Chapter 2, we first evaluated the prediction accuracy of the classifiers. We find out that Support Vector Machine(SVM) classifier is the most accurate, Multi-layer Perceptron(MLP) classifier comes the second while the ZeroR classifier produces least satisfactory results. Then we analyse the confusion matrices of the respective classifiers. SVM, though enjoys highest prediction accuracy, performs unwell. SVM produces a great many false negative cases and given the fact that false negative results are "fatal" in medical practices, we recommend to adopt MLP approach instead. Nonetheless, when we evaluate the running time of the classifiers, the disadvantage of MLP classifier exposes. The MLP classifier requires relatively long time to build the model and train. However, after time complexity analyses on algorithm, though MLP trains more slowly than other classifiers, we agree that by taking prediction time into consideration, the time complexity of MLP classifier is acceptable.

To sum up, after careful analyses and observations, we recommend to choose MLP classifier to predict Type 2 Diabetes with the Pima data set. But it should be noted that if the time efficiency is the absolute priority in prediction, the practitioners should consider Naive Bayes classifier or 5-NN classifier instead.

4.2 Classifier Optimisation

Besides all the analyses and comparisons among different classifiers, this project also briefly discusses a couple of feasible strategies to improve the classifier performance. By comparing different k-NN classifiers, we find out that by tuning parameters of machine learning algorithm, the classifier performance will improve. Then, we extend the optimisation research to feature selection methods. In this project, we implemented Correlation-based Feature Selection and the find out that this method improves the overall accuracy performance. However, in regard of the confusion cases, the CFS method actually increases the possibility of false negative cases. This performance drop indicates that more delicate optimisation should be implemented to CFS procedure. Also, for those people who care about their health conditions, they should pay more attention on the features selected by CFS method, which are glucose, insulin, BMI, pedigree, and age because according to this data set, these features are more related to type 2 diabetes.

4.3 Future Work

Though with the data set and various machine learning classifiers provided, we have already mined a lot of findings, it should be acknowledged that there are many limitations in this project and accordingly, there are much to do in the future research.

Inclusive Data Set. The data set used in this project is a relatively small data set, which only records data of female Pima people who are above 21 years old. The limits in race, gender and age may lead to unavoidable bias for machine learning classifiers. Also, the data set magnitude is rather small with only 768 entries. We believe that with larger and more inclusive data set, the characteristics of different classifiers will be more obvious and it is highly possible to get more details about classifiers with larger data set.

Sophisticated Feature Selection. The feature selection method discussed in this project, which only takes independent variables into account are rather rough and limited and there are much more to consider for feature selection. One of the possible approach is to extract the combination of different features and observe the combined effect on the results.

Other Classifier Optimisations. The evaluated prototype of those classifiers are all very basic implementations of respective machine learning algorithms. Except for the methods discussed in 4.2, there are a great many other optimisation techniques, especially for MLP algorithm. We believe that with more appropriate optimisations, all of the classifiers evaluated in this project will perform better.

CHAPTER 5

Reflection

Thanks to this Pima Indian Diabetes research project, we made comprehensive progress in the artificial intelligence field by exploring various academic resources and implementing required data classifiers.

In the preparation stage, we read extensively on various academic sources not only about machine learning algorithms or classifier implementations but also the background information about Pima People and Type 2 diabetes. This learning process helps us understand the issue more thoroughly and makes us realize the inter-connectivity between machine learning theory and real world data modelling. In other words, by demonstrating the real world application of artificial intelligence, this project stimulates our enthusiasm to use the machine learning knowledge to resolve the real-world problem.

During the project process, we learned various features of different classifiers and algorithms. Not only do we realize the performance differences among those algorithms, we also recognize that there is no such "perfect" machine learning model. Every single model has its advantages and disadvantages and we need to implement the model with the real data set to test the real performance. For example, in this project, the accuracy of Naive Bayes classifier is lower than that of Multi-layer Perceptron but it runs much faster. In the cases when we need to pursue time-efficiency, we will consider the trade-off and choose Naive Bayes classifier.

Moreover, this project demonstrates how feature selection before model calculation can improve the classifier performance. Though our feature selection implementation in this project is somewhat naive and simplified, the effectiveness of the feature selection procedure has already amazed us, especially when we take confusion matrix into consideration. This approach inspired us to learn more on parameter tuning of a specific machine learning model and shed the light on future individual research.

Bibliography

[1] D. Australia, "Type 2 diabetes," 2020. [Online]. Available: https://www.diabetesaustralia.com.au/type-2-diabetes

[2] *Prevention of type 2 diabetes : from science to therapy*. New York, NY: Springer.

[3] L. Schulz and L. Chaudhari, "High-risk populations: The pimas of arizona and mexico," *Current Obesity Reports*, vol. 4, no. 1, pp. 92–928, 2015.

[4] Q. Zou, K. Qu, Y. Luo, D. Yin, Y. Ju, and H. Tang, "Predicting diabetes mellitus with machine learning techniques." *Frontiers in genetics*, vol. 9, p. 515, 2018. [Online]. Available: http://search.proquest.com/docview/2136552357/

[5] U. M. Learning, "Pima indians diabetes database," 2016. [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database

[6] G. Claeskens, *Model selection and model averaging*, ser. Cambridge series on statistical and probabilistic mathematics ; 27. Cambridge: Cambridge University Press.

[7] T. A. Alhaj, M. M. Siraj, A. Zainal, H. T. Elshoush, and F. Elhaj, "Feature selection using information gain for improved structural-based alert correlation," *PLoS ONE*, vol. 11, no. 11, p. e0166017, 2016. [Online]. Available: https://doaj.org/article/d8791c0570254338ac8ddeeaee976ed3

[8] Anonymous, "cfs illustration figure," 2017. [Online]. Available: https://blog.csdn.net/littlely_ll/article/details/71545929

[9] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*, ser. Springer Texts in Statistics. New York, NY: Springer New York, 2013, vol. 103.

[10] S. Abe, *Support Vector Machines for Pattern Classification*, ser. Advances in Pattern Recognition. London: Springer London.