Predicting Diabetes Diagnosis with Medication Purchasing Habits

by

Louie-Marie Dreyer

Submitted in partial fulfillment of the requirements for the degree
Masters in Information Technology (Big Data Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

January 2019

Publication data:

Louie-Marie Dreyer. Predicting Diabetes Diagnosis with Medication Purchasing Habits. Masters dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, April 2008.

Electronic, hyperlinked versions of this dissertation are available online, as Adobe PDF files, at:

http://upetd.up.ac.za/UPeTD.htm

Predicting Diabetes Diagnosis with Medication Purchasing Habits

by

Louie-Marie Dreyer
E-mail: lumidreyer@gmail.com

Abstract

Diabetes has become a global health problem with an alarming rate of undiagnosed patients. Machine learning applications have successfully been used in medical research to assist with early diagnosis of diseases, and predicting the onset of diabetes is one of the focus areas in diabetes research. The emerging advances in electronic health records is creating opportunities for new machine learning applications that are based on the abundance of health information available. Medication purchases is one of the data components of electronic health records and is administrated by community pharmacists, who form the third largest group of health care providers. The focus of this research is to determine the feasibility of using the medication purchasing habits of a patient to build a machine learning model that can aid community pharmacists in the diagnosis of diabetes. Decision tree and decision tree ensemble models were trained on the medication purchasing data across 622 pharmacies in South Africa with 3 years of data. The models could accurately predict diabetes diagnosis in a range of 88% to 95% on a testing set that included over 30 000 patients.

Keywords: Machine learning, Diabetes, Random Forests.

Supervisors : Prof. J. Grobler
Prof. A.P Engelbrecht
Department : Department of Computer Science
Degree : Master of Information Technology

Acknowledgements

I would like to express my sincere gratitude to the following people for their assistance, direct or indirect, during the production of this research work:

- Prof. Jacomine Grobler and Prof. Andries P. Engelbrecht, my supervisors, for all their insight and support;
- Johan Du Preez, my mentor and research sponsor, for all his support, insight and motivation;
- Deon van Niekerk, my mentor and colleague, for all his support and guidance;
- Christoff Du Preez, my friend and colleague, for all his support and assistance in creating the machine learning model;
- Andrew Cowgill, my friend and engineer, for all his believe, support and motivation;
- Amory Tudhope, my study partner, for all her help and support in completing this degree together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Diabetes has become a global health problem and a major cause of death worldwide [1]. One of the biggest problems of diabetes is the number of undiagnosed patients. Machine learning has been widely used in medical sciences to assist with early diagnosis and prediction of diseases [34].

Section 1.1 presents the motivation for this dissertation and the objectives are listed in Section 1.3. The methodology used is discussed in Section 1.2, while the contributions made are summarised in Section 1.4. The outline of the dissertation is discussed in Section 1.5.

1.1 Motivation

Diabetes is a chronic illness that occurs due to raised levels of glucose in the blood. The disease affects nearly one out of 11 adults, with a total number of 425 million people being diagnosed in 2017. This is 10 million more adults with diabetes than in 2015 [1]. The rate of undiagnosed diabetes cases is alarmingly high, with one out of two adults remaining undiagnosed [1]. A study has indicated that the clinical diagnosis of Type 2 diabetes is done only four to seven years after the onset of the disease [22]. During this time, complications such as cardio-vascular problems begin, which can be easily prevented with earlier diagnosis and the correct treatment [22].

One of the many successful applications of machine learning has been in the health care environment. Machine learning has been used to build prediction models with clinical information to assist in the diagnosis and treatment of various diseases [30]. Three surveys and review papers [29, 30, 33] on the application of machine learning in diabetes research were studied during this research, which included 204 published cases of machine learning and data mining applications in diabetes research. The cases ranged from predicting diagnosis to treatment plans and complication management.

There are two main data sets that have been used for prediction models for the onset of diabetes in the reviewed research, namely the Pima Indian dataset [51] and Electronic health records (EHR) [27]. EHR is more recent data used for creating machine learning models due to the technological advances in the digitisation of health records [27]. EHR consists of varies data components such as medication administration, physical assessments, laboratory tests, diagnoses, medical history, immunisation history and various other components [27]. EHR is a very new area of research with a large number of focus areas for the different data components that have not been studied.

Medication administration is one example of such a focus area, with no current work that has been done on this potential data source for predicting the onset of diabetes. Medication administration data can be used to build prediction models that can aid community pharmacists [35], who are responsible for dispensing the medication. Pharmacists are the third largest group of health care providers, with doctors and nurses being respectively, first and second [35]. Community pharmacists operate as retailers outside of the hospital environment. There is an emerging consensus within the professional orginisations, academics and policy makers that pharmacists could contribute to beter and safer treatment of chronic diseases [35]. Community pharmacists have primarily been responsible for retailing and dispensing of medication. Various professional pharmacy regulatory authorities have been expanding the responsibilities of pharmacists due to the vital role in patient care that they have [48].

The purpose of this dissertation is to determine the feasibility of using the medication purchasing habits of a patient to build a machine learning model that can aid community

pharmacists in the diagnosis of diabetes. As far as the knowledge from the literature study completed for the purpose of this dissertation, a machine learning model based on medication purchasing habits have not yet been studied to predict diabetes diagnosis.

1.2 Objectives

The research objectives of this dissertation are summarised as follows:

- To investigate the current machine learning applications in the field of diabetes research.
- To test the feasibility of using the data collected of medication administration for the development of an effective prediction model for diabetes diagnosis.

1.3 Methodology

The methodology used in this dissertation incorporates the following:

- A literature survey related to machine learning and the application of machine learning in diabetes research.
- An experimental investigation into the use of medication administration history for predicting the diagnosis of diabetes.

1.4 Contribution

The novel contribution of this dissertation is creating the first machine learning model for the prediction of diabetes based on medication purchasing habits of a patient.

1.5 Dissertation outline

The list below presents a summary of the remaining chapters in this dissertation:

- **Chapter 2** broadly outlines machine learning, with a focus on supervised machine learning and decision trees as this was the approach followed for this dissertation.
- **Chapter 3** discusses machine learning application in diabetes research and reviews studies done on the prediction of diabetes diagnosis. The chapter focuses on the two main data sources that have been used in current literature, namely the Pima Indian data set and EHR.
- **Chapter 4** provides a summary of the specific medication purchasing data used for this research.
- **Chapter 5** presents the results of the experimental machine learning models that were created.
- **Chapter 6** provides a summary of the general conclusions of this dissertation.

Chapter 2

Classifier machine learning models

A classifier machine learning model is a popular machine learning application and a widely research field. Section 2.1 introduces the different machine learning techniques. Section 2.2 summarises the components of a classifier model. Section 2.3 provides a framework for modeling a supervised machine learning model. Section 2.4 introduces the different machine learning algorithms that can be used for a classifier model.

The focus for this research is on decision trees and ensemble models for decision trees. This is based on the evaluation of the supervised machine learning algorithms, as summarised in Section 2.4. Section 2.5 introduces decision trees and Section 2.6 provides an overview of ensemble models for decision trees.

2.1 Machine learning techniques

Learning techniques can be supervised, unsupervised, or based on reinforcement learning [16]. Supervised learning entails training conducted with a data set that has pre-defined classes for each data input consisting of multiple features [25]. A regression model is built for a continuous label by finding the functional mapping that describes the relationships between the feature space and scalar floating point space. A classifier model is created by finding hyperplanes that can separate the different classes of the dataset by determining mutually distinguishing features of a class [53].

Unsupervised learning entails training conducted with a training set that contains no pre-defined classes. An unsupervised machine learning algorithm finds patterns in the dataset and builds a description for each pattern [25, 53].

The training approach for reinforcement learning is based on a reward and punish process where correctly predicted outcomes are rewarded by the model and incorrect outcomes are punished [16].

2.2 A classifier model

A classifier model is created with a supervised machine learning approach and is illustrated in Figure 2.1. The goal of supervised machine learning is to build a classifier model that can assign class labels based on the input features of the instance. The classifier model is then used to assign class labels to instances based on the input features, where the class label is unknown.

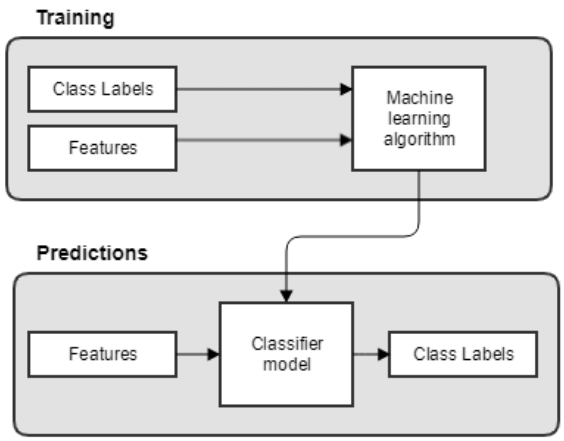

Figure 2.1: Illustration of a classifier model, adopted from [32].

2.3 Framework for supervised machine learning models

The process of creating a classifier model with supervised machine learning is described in Figure 2.2. The classifier is trained using the training dataset. This classifier can then be used to generalise and predict the class for new instances [32]. This process is discussed in detail in the following sections.

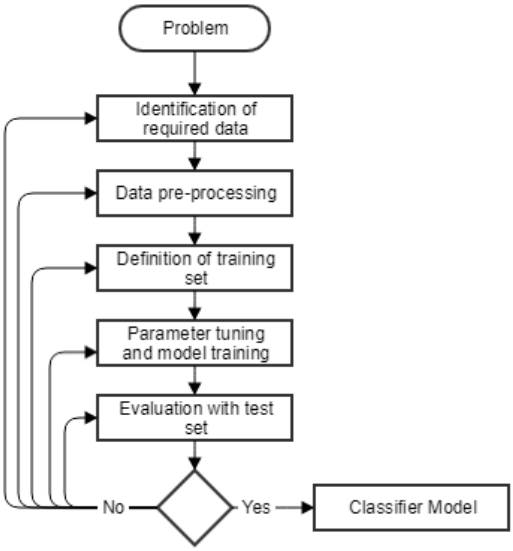

Figure 2.2: A model representation of an environment, adopted from [32].

Identification of required data

Identifying and collecting the required data is the first step in the machine learning process. Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models [32]. The goal of feature engineering is to create new input features that can be used to determine strong and simple relationships that can be used to predict the output feature.

Feature engineering may include steps such as [32]:

- Feature construction: the manual construction of new features from raw data.
- Feature importance: an estimate of the usefulness of a feature.
- Feature selection: many features converted to a few that are important and relevant.

Various methods exist to determine feature importance and selection, however these methods are outside the scope of this dissertation.

Data pre-processing

Pre-processing of data helps to improve the quality of data used and includes factors such as completeness, consistency, accuracy, timeliness, interpretability and believability [21]. Data pre-processing typically involves data cleaning and data transformation [53], which is discussed in the following sections.

Data Cleaning: The quality of data is typically affected due to data being incomplete, inconsistent and noisy. Data cleaning is done to improve the quality of data by addressing these problems [21]. Missing or incomplete attributes can either be removed or the attributes can be fixed by replacing the value. The new value for the attribute can be the mean value of the feature or the most probable value can be determined statistically.

Another task is noise reduction, which involves removing random errors and variances caused by outliers by either removing the outliers or using smoothing techniques [53]. One method of determining outliers is using statistical description techniques and visualisations to identify the outliers. An example of such a method is using box-plots to determine the outliers. Outliers of a box-plot are data points outside the whiskers and are data points not within 99,3% of the data range. Data smoothing can also be performed by techniques such as binning and regression [21].

Data transformation: The selected algorithm will determine what data transformation is required. Data scaling is a common task that is required for some algorithms. Many algorithms perform better if the data values have the same scale, because some

algorithms can assume that the data is normally distributed with zero mean and unit variance. A feature can dominate the algorithm training if the variance magnitude is much higher than the others [40].

One of the last data transformation tasks is to binary encode nominal attributes by replacing such attributes with separate features for each value [53]. This task is also called creating dummy variables for hot encoding these attributes [40].

Definition of training set

The same data that is used to train the model cannot be used to test the model because that will not provide an indication of how well the model performs. Poor generalisation of a model occurs when the model fails to predict instances that are unseen [7]. A common practice to overcome poor generalisation is to split the available data into a training and test set. There are also instances that would require splitting the training set into a third set, namely a validation set, that is used to validate hyper-parameter tuning. The problem with split tests are deciding what threshold to use. Different splits on the same data will have different results and cause model variance [7].

A widespread strategy to overcome the limitation of the size of the splits is to use cross-validation (CV) for training and testing the model, as illustrated in Figure 2.3. The *k-fold* CV approach splits the set into k smaller sets following the procedure below for each of the *k-folds* [7]:

1. The model is trained using *k-fold* of the folds of the training data.
2. The resulting model is scored on the remaining fold.
3. The accuracy of the final model is the average score for each run.

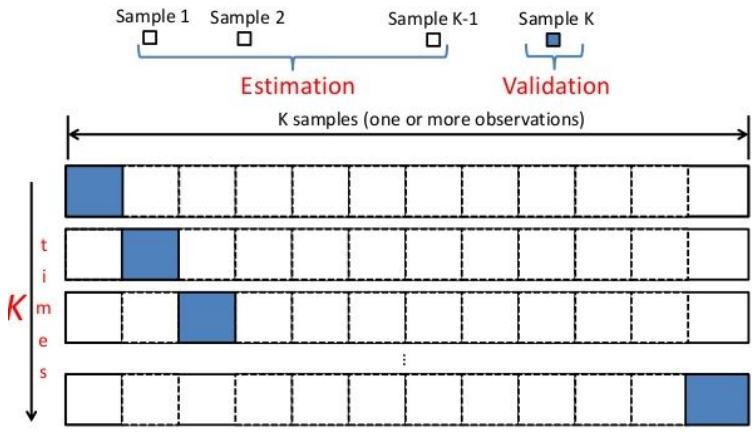

Figure 2.3: K-fold cross validation, adopted from [7].

Hyper-parameter tuning and model training

Parameters include all internal variables of the model that are estimated during training of the model. Hyper-parameters, on the other hand, are external variables that need to be set before training of the model [31]. The relationship between model tuning, which includes determining the hyper-parameters, and model training, is illustrated in Figure 2.4.

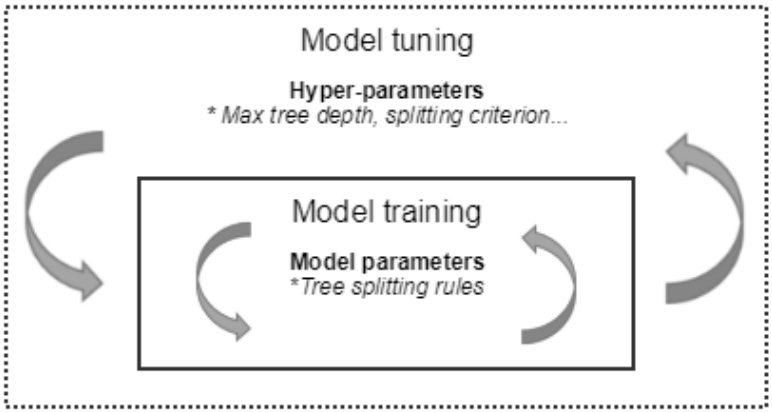

Figure 2.4: Model tuning vs model training, adopted from [31].

The process of finding the optimal set of values for the hyper-parameters of a model

is called parameter tuning. The values of the hyper-parameters affect the performance of the training process as well as the accuracy of the model. The best values for hyper-parameters are dependent on the dataset and need to be determined before model training [31]. Some of the most common approaches for parameter tuning are illustrated in Figure 2.5 and include:

- Grid search: Each hyper-parameter is assigned a range of values and the model is trained and assessed across all combinations of the hyper-parameters. The disadvantage of grid search is that the search needs to be "coarse" to identify the set of parameters that will improve the model results [31].
- Random search: Using random combinations of hyper-parameters have been proven by Bergstra and Bengio [9] to be an effective alternative to performing an entire grid search. The effectiveness of the random search is dependent on the uniformity and size of the samples. Candidate combinations of hyper-parameters can be concentrated in an area that is omitted by the random search combinations [31].
- Latin hypercube: A more structured approach to random search is using Latin hypercube samples, where the sample combinations are still random combinations, but are uniform across each parameter. This approach aims at using the sample space more effectively [31].

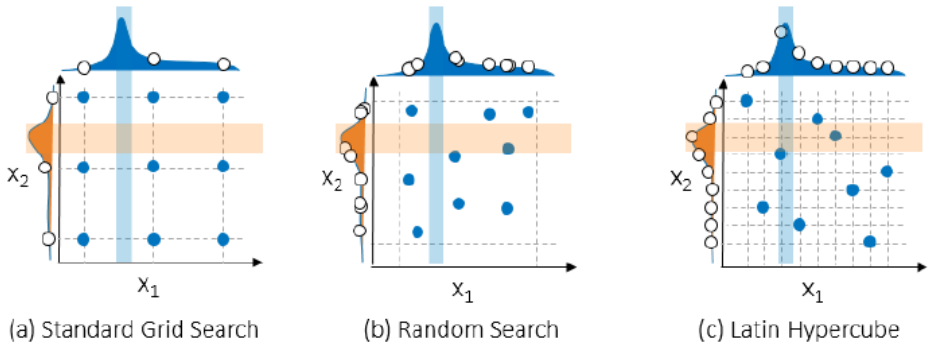

Figure 2.5: Parameter tuning with grid search, random search and Latin Hypercube, adopted from [31].

Model evaluation

The generalisation performance is evaluated by scoring the predicted classes of the test dataset that was determined by the trained model, against the actual classes of the test dataset. The results of the scored dataset can be evaluated through various metrics. The confusion matrix, as illustrated in Figure 2.6, is used to calculate some of the metrics that is discussed next.

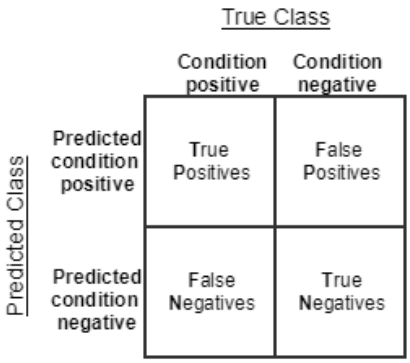

Figure 2.6: Confusion matrix, adopted from [18].

Metrics for a classification machine learning model include the following [40]:

- Accuracy measures the goodness of a classification model as the proportion of correctly predicted results to total instances of the test.
- Precision is the proportion of true positive results over all positive results, which includes the sum of true positives and false positives.
- Recall is the fraction of all correct results returned by the model, which is calculated as the true positive results over the true positives and false negatives.
- The F-score is computed as the weighted average of precision and recall, giving a value between 0 and 1. The ideal F-score value is 1.
- The area under curve (AUC) measures the area under the receiving operating characteristics (ROC) curve. The ROC curve is commonly used to visualise the

performance of a classification model by plotting the true positive rate on the y-axis and false positive rate on the x-axis [18]. True positive rate is also known as the sensitivity or hit rate of the model and is calculated as shown in Equation (2.1). False positive rate is also known as the specificity or false alarm rate of the model [18] and is calculated as shown in Equation (2.2). If the classification algorithm is performing well, the AUC will be closer to 1; a random guessing algorithm's AUC score will be closer to 1/2.

$$True\ positive\ rate = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.1}$$

$$False\ positive\ rate = \frac{False\ Positives}{False\ Positives + True\ Negatives} \tag{2.2}$$

The ROC curve visualises the trade-offs between the benefits or true positives of the model and the costs or false positives, as illustrated in Figure 2.7. A poor classification model is one that randomly guesses the classification model and will have an ROC curve that hugs the diagonal line. The ROC curve in the perfect scenario, where all the classes where correctly predicted, is 1.0, the area of the entire graph, whereas for a poor classifier the ROC curve hugs the diagonal and the area under the curve is 1/2 [18].

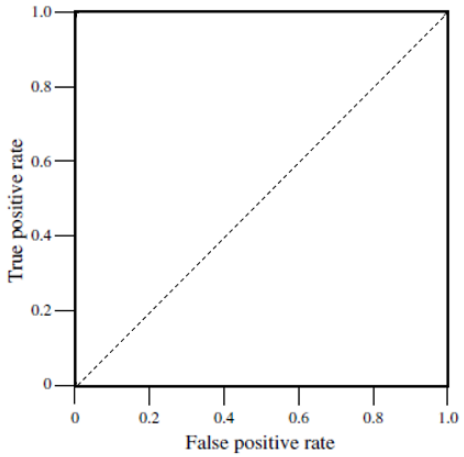

Figure 2.7: An example of a ROC graph.

2.4 Classifier machine learning algorithms

Section 2.4.1 provides an overview of supervised machine learning algorithms for classification tasks. The algorithms are evaluated in Section 2.4.2 to determine which algorithms should be the focus for this research.

2.4.1 Supervised machine learning algorithm classification

Kotsiantis [32] completed a review of the most important supervised machine learning algorithms for classification tasks. The algorithms can be grouped into five categories, namely information theory learning, perceptron based, statistical learning algorithms, instance based learning and support vector machines, as illustrated in Figure 2.8.

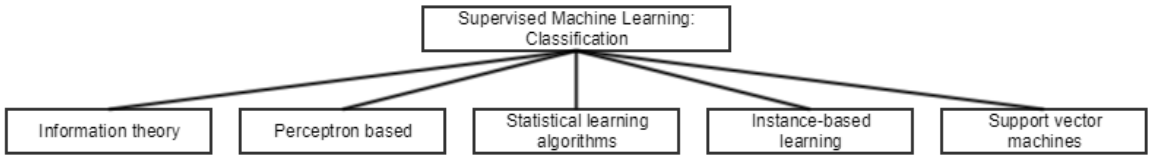

Figure 2.8: Supervised machine learning algorithms for classification, adopted from [32].

Information theory includes methods such as decision trees and rule-based classifiers. Decision trees is a useful and easy to understand machine learning technique that can be used for describing, classification and generalisation of data by constructing a prediction model [36]. Rule-based algorithms derive rules directly from the training dataset that are used to classify instances [32].

A vast growing research field is that of perceptron based algorithms, which include single and multi-layered neural networks. Neural networks have been modelled on natural and biological intelligence and have had great successes in machine learning tasks [16].

Statistical learning algorithms used for classification tasks use probability models that calculate the probability of an instance belonging to a class. One of the most well known statistical learning algorithms is the Bayesian network, which is based on the probability relationships between features [32].

Instance-based learning algorithms include algorithms such as k-Nearest Neighbour (kNN). The principle of the kNN algorithm is that similar instances will generally exist in close proximity to each other [32].

The last category of supervised machine learning is that of support vector machines (SVM). SVM creates hyperplanes by finding the largest possible margin that separates the data classes. The aim of SVM training is to find the maximum margin of the hyperplane as that will separate the data more and improve the generalisation error of the model [32].

2.4.2 Evaluation of supervised machine learning algorithms for classification

The choice of algorithm depends on the classification task and the dataset that is used to construct the model. The type of task and the attributes of the dataset determines which factors should be considered when evaluating supervised machine learning algorithms, which may include [32]:

- Accuracy;
- Speed of learning;
- Tolerance to missing values;
- Tolerance to irrelevant features;
- Tolerance to redundant features;
- Tolerance to highly interdependent features;
- Tolerance to noise and outliers;
- Dealing with discrete, binary or continuous features;
- Dealing with the danger of overfitting;

- Incremental learning ability;
- Interpretablity of model.

The classification task for this research is to predict the onset of diabetes using medication purchasing habits, which is a high-dimensional dataset with a large number of attributes. The dataset is also very sparse, with a high number of missing data values for each medication type. Therefore, the factors chosen to evaluate classification algorithms for this research included accuracy, handling missing values and tolerance to irrelevant features, because these are the main factors influenced by the high-dimensional and sparse dataset.

Kotsiantis [32] compared algorithms from the five different groups of supervised machine learning classification algortithms that were discussed in Section 2.4.1. Kotsiantis [32] allocated a score out of four based on evidence of existing empirical and theoretical studies for the evaluating factors of the algorithms. The results from Kotsiantis' study for the important evaluation factors for the classification task of this research are summarised in Table 2.1.

	Decision Trees	Neural Networks	Naive Bayes	kNN	SVM
Accuracy	2	3	1	2	2
Handling missing values	3	1	4	1	2
Tolerance to irrelevant features	3	1	2	2	3
Total	8	5	7	6	7

Table 2.1: Evaluating supervised machine learning algorithms, adopted from [32].

From Table 2.1, it is expected that decision trees will perform the best when the dataset has missing values and the algorithm also has a good tolerance to irrelevant features for a high-dimensional dataset. These two factors contributed to decision trees obtaining the highest score of eight and is discussed in more detail in the following section.

2.5 Decision trees

Section 2.5.1 describes how a decision tree is constructed. Section 2.5.2 summarises the hyper-parameters of a decision tree. Section 2.5.3 discusses the advantages and disadvantages of a decision tree.

2.5.1 Constructing a decision tree

A decision tree, as illustrated in Figure 2.9, contains the following elements:

- Root: A root node is the first node of the tree that contains all the classes [36].
- Child nodes: A node with two or more child nodes are referred to as internal nodes. Leaf nodes are the end nodes of the tree and each leaf node has a class associated with it [36].
- Split function and arcs: Splits test the value for a feature for an expression. Arcs to the next node depicts the distinct outcomes of the node test [36].

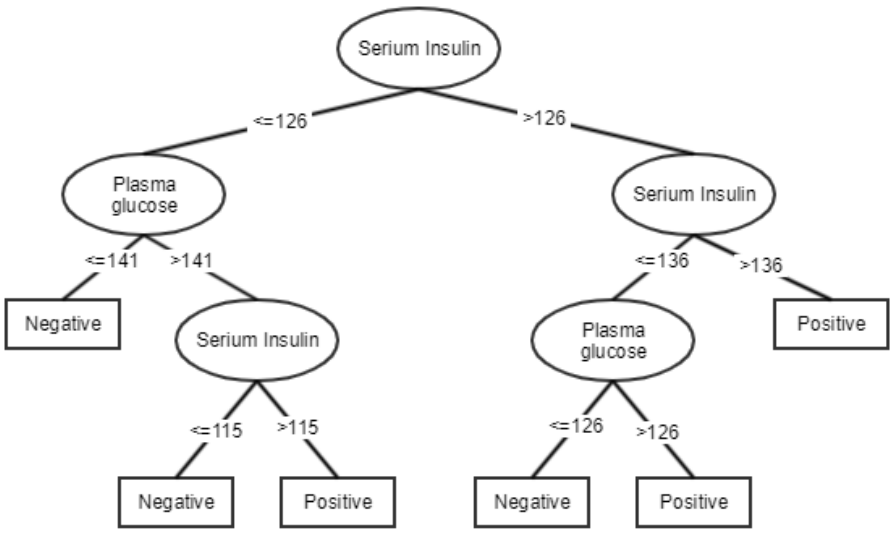

Figure 2.9: Example of a decision tree predicting diabetes diagnosis, adopted from [8].

The construction of a decision tree has been referred as induction, tree growing or tree building. The steps involved in the induction of a tree includes [36]:

1. Determine a split test and score each of the possible splits using a goodness measure score, for example information gain.

2. Choose the split with the best result for the goodness measure for the current node and create child nodes for each distinct outcome of the split. Partition the training instances using the split test into the child node with the same outcome.

3. If all the training instances at the current node belongs to the same class, create a leaf node otherwise repeat steps 1 and 2 until a leaf node is created.

Determining the best split involves finding the best combination of features and rules that can divide the data into the different classes. The features need to be evaluated in order to determine which are the best features to use for the node split. The most common feature evaluation method used to determine the goodness measure of the feature is information gain [36]. Information gain is based on Shannon's entropy and can be defined as the difference between the entropy of the original dataset and the weighted sum of entropies for the resulting datasets [26]. During induction of the tree, the aim is to maximise the mutual information gain at each node or level of the tree.

Information gain forms the basis for the most common tree induction methods such as ID3 and C4.5. These tree induction methods can be summarised as follow:

- ID3 was invented by J.R Quilan [42] in 1986. The ID3 algorithm builds the tree by maximising the information gain at each split.

- C4.5 was an improvement on the ID3 algorithm that was presented by Quilan [26] in 1993. The goodness measure for C4.5 is gain ratio, which is also an information based measure that takes the number of test outcomes into account [43].

Classification and regression trees (CART) [36] uses a simpler alternative to information gain, which is based on the gini index. CART was presented in 1984 by Brieman *et al.* [46] and uses the gini index as the goodness measure to determine which feature to

use at the split. The gini index is a distance measure that refers to the distance between the probability distributions of the classes.

2.5.2 Decision tree hyper-parameters

There are four main decision tree hyper-parameters, which are summarised in Table 2.2. These hyper-parameters of decision trees are dependent on the classification tasks and the data used to construct the decision tree.

Parameter	Description
Maximum depth of the decision trees	Increasing the depth of the tree might increase precision but at the risk of overfitting on the training data.
Goodness measure criteria to determine the quality of split	The two main measures used are entropy or the gini index. Entropy is the criterion used to determine information gain. The gini index is used to determine the gini impurity measure.
The minimum number of instances required to split an internal node	The minimum number of instances parameter is the minimum number of samples required to split a node and create another branch, rather then leaving the node as n leaf node.
Maximum number of features to consider at a node for a split	The maximum number of features to consider at a node for splitting determines the subset size of features that are evaluated for the split.

Table 2.2: Decision tree hyper-parameters, adopted from [40].

2.5.3 Advantages and disadvantages of a decision tree

There are several advantages of the decision tree classification technique such as:

- A decision tree model is easy to understand as it is intuitive and easy to interpret by domain experts [36, 41, 45]
- Decision tree methods are exploratory and non-parametric methods, as opposed to inferential methods. Therefore, a wide range of data distributions can be used as there are no assumptions that need to be made for the model and the data [36].
- Feature selection is integrated in the training and classification processes and the model can accommodate many features [40].
- Non-linear decision boundaries are created and thus very little data pre-processing is required [36].
- Various data types can be used as input data such as numeric, nominal and textual data [45].
- Decision trees can be used for various types of machine learning such as classification, clustering and feature selection [45].

There are several disadvantages of decision trees. The following are some of the disadvantages of a single decision tree:

- Decision tree can easily overfit on the training data and become a less generalised and unstable model if pruning is not done after model training [45].
- Fragmentation of the data happens when many features are tested along a path that partitions the data into smaller fragments that contain a small number of instances. The prediction confidence of a node is limited when there is a small number of instances and also adds to the problem of overfiting on the training data [45].

- A replication of subtrees within a model is caused when there are complex interactions among the features. The feature space at a node is divided into mutually exclusive regions and replication of the same feature may be necessary to construct the classifier [39, 45].

The disadvantages of a decision tree can be overcome by pruning or using ensemble models. Pruning is a method used to obtain the right sized tree [46]. Pruning involves first building a complete tree, where the splitting of a leaf node does not improve on the generalistion accuracy of the tree. The next step is to remove all subtrees that do not improve the generalisation accuracy of the model. This method of pruning is referred to as reduced error pruning [44] and is the simplest approach to pruning [17].

2.6 Ensemble models

Section 2.6.1 introduces the different approaches for creating ensemble models. Section 2.6.2 provides an overview of algorithms used for creating decision tree ensemble models.

2.6.1 Ensemble models approaches

Ensemble methods are based on the general principle that rather than relying on a single model, better results and a more generalised model can be created by combining multiple related models [45]. The main principle of an ensemble model is to weight the results of several models and then to combine the models to make a better decision than that obtained by a single model [45].

The two approaches for constructing ensemble models are bagging and boosting [49]. These approaches can be summarised as follow:

- Bagging is a bootstrap method used for creating an ensemble model. The individual models in the ensemble model is trained on a random redistribution of the training set [37]. Each model's training set randomly draws and replace examples of the original training set. Each training set for the individual models can contain

multiple of the same examples of the original training set, while other examples are omitted, because of the resampling with replacement method [37].

- The focus of the boosting method, that was introduced by Freund and Schapire *et al.* [49], is to create a series of models. The resampling of examples from the original training set for the models are based on the performance of the previous model. The examples that were incorrectly predicted by the previous model is used more in the training set for the next model. The aim of boosting is to train new models that can improve the accuracy of the previous model [37].

2.6.2 Decision tree ensemble models

Some of the algorithms for ensemble models with decision trees that use bagging methods include the following:

- Random decision forests is an ensemble learning algorithm intended for classification tasks that was proposed by Breiman in 2001 [11]. The random forests algorithm is one of the most used ensemble learning algorithms.
- Extremely randomised trees adds more randomness to the random decision forests algorithm in the way that the splits functions are computed [20] for each individual decision tree.

Some of the algorithms for ensemble models with decision trees that use boosting methods include the following:

- Adaptive boosting (AdaBoost) is a popular boosting algorithm for classification tasks and is based on the work done by Freund and Schapire *et al.* [49].
- Gradient boosted trees is a boosting algorithm that was originally designed for regression tasks [19], but variations on the loss function used by the algorithm has allowed for classification tasks [45].

The four mentioned ensemble models for decision trees are discussed in more detail in the following sections.

Random forests

The random forests algorithm consists of multiple decision trees which are trained on a random subset of the training set. The random subset is created with bagging or bootstrap of the training set [11]. The generalisation error of random forests is decreased by introducing the randomness of features used in constructing each decision tree. The randomness introduced in each decision tree of the algorithm improves both the bias and variance component of the generalisation error. The bias error is the systematic or persistent error that the inducer of the decision tree makes. The variance error is caused by the variation of training data used for the algorithm. The performance of the model is better than a single tree as it overcomes the bias error that exists with a single decision tree [11].

The different implementations of the random forests algorithm can vary with regard to the following steps of the algorithm [10]:

- The decision tree algorithm used to construct each individual decision tree.
- The method used to sample the subset of the training dataset that is used to construct the individual decision trees.
- The method used to combine the predictions of each individual tree to produce the final predicted outcome of the random forests algorithm.

The steps of the random forests algorithm, as introduced by Brieman [11], is illustrated in Figure 2.10. The original random forests algorithm used the CART method to construct each individual tree. The gini index is used as the goodness measure criterion for finding the best feature to be used for the split function at each node.

The bootstrap method is used to create a random subset of the training dataset to train the individual decision tree models. The training data that was not selected for the subset is used as an internal validation dataset to calculate the out-of-bag (OOB) error. The OOB estimate is the average of the error obtained by each tree on the validation

sets [11]. The training of the random forests can be terminated at the number of trees where the OOB error stabilises [24].

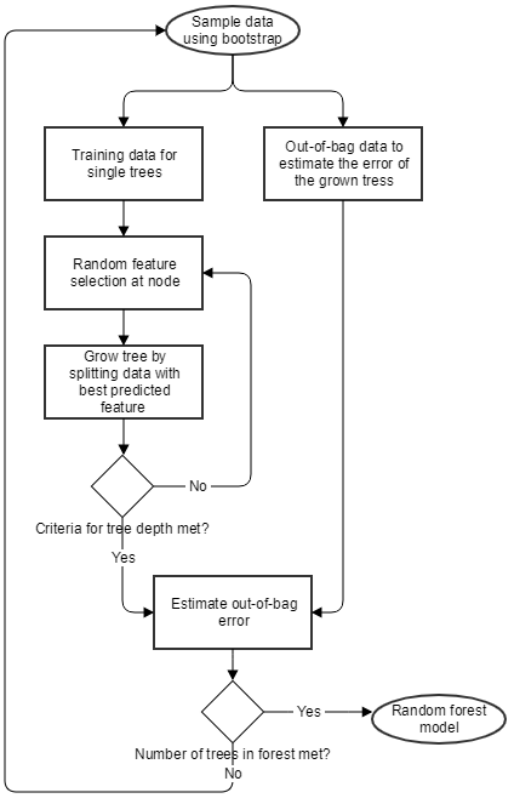

Figure 2.10: Random forests algorithm flow for training, adopted from [10].

The method used to combine the multiple decision trees is through voting. The voting is done as follows [40]:

1. Each tree in the decision forests outputs a non-normalised frequency histogram of labels.

2. The aggregation process sums these histograms and normalises the result to obtain the probabilities for each label.

3. The trees that have high prediction confidence will have a greater weight in the final decision of the ensemble.

The hyper-parameters of a random forests algorithm are summarised in Table 2.3. The hyper-parameters includes all the hyper-parameters of a decision tree model, as well as an additional hyper-parameter for the number of trees in the random forests algorithm.

Parameter	Description
Number of decision trees	Maximum number of trees for the ensemble method. More trees will increase accuracy but training time will also increase. The accuracy of the model will stabilise after a certain number of trees, which is dependent on the classification task and data that is used. The number of decision trees where the OOB stabilises can be used to determine the number of decision trees for the random forests algorithm.
Maximum depth of the decision trees	Increasing the depth of the tree might increase precision but at the risk of overfitting and increased training time.
Goodness measure criteria to determine the quality of split	The two main measures used is entropy or the gini index. Entropy is the criterion used to determine information gain. The gini index is used to determine the gini impurity measure.
The minimum number of instances required to split an internal node	The minimum number of instances required to split a node and create another branch, rather then leaving the node as n leaf node.
Maximum number of features to consider at a node for a split	The maximum number of features to consider at a node for a split determines the subset size of features that are evaluated for the split.

Table 2.3: Random forests hyper-parameters, adopted from [24, 40].

Extremely randomised trees

The extremely randomised trees algorithm only has one variation on the random forests algorithm that adds more randomness to how the individual decision trees are constructed [45]. The goal of adding more randomness to the model is to improve the generalisation error of the random forests algorithm.

The additional randomness is added when the split for the node is determined. The splits in the CART trees used in the random forests algorithm is determined by finding the best feature in the random subset of features with a corresponding split value that has the best results in the goodness measure of the split. The variation of the extremely randomised trees algorithm is that the corresponding split value is also selected at random, instead of being determined and evaluated against the goodness measure of the split [20].

The disadvantage of the extremely randomised trees method, is that it tends to have a higher bias in the individual decision tree models [20]. However, the bias can be overcome by adding enough decisions trees to the extremely randomised trees model to compensate for the bias in the individual decision tree models.

AdaBoost

The decision trees in the AdaBoost model are iteratively built on subsets of the training data that is determined through the boosting method. A higher weight is given to the misclassified instances of the decision tree and the resampling method focuses on creating a subset of training data for the higher weighted misclassified instances in the training data [49]. The AdaBoost algorithm is illustrated in Figure 2.11.

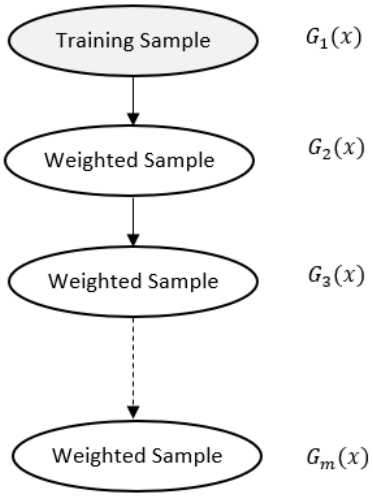

Figure 2.11: AdaBoost algorithm flow, adopted from [24].

The final classifier model, $G(x)$, is created by combining the individual classifier models through a weighted majority vote, as shown in Equation 2.3.

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)] \tag{2.3}$$

with,

$$m = 1, 2, ..., M \tag{2.4}$$

where M is the number of classifiers models and α_m is the weight determined by the boosting algorithm for each classifier model. The weighted voting method allows for more accurate classifier models to carry more weight in the final prediction of the classification task [24].

The steps in creating the classifiers, $G_m(x)$, can be summarised as follow [24]:

1. Set all the instance weights to $w_i = 1/N$, where $i = 1, 2, ..., N$ and N is the number of instances.
2. Create the classifier model, $G_m(x)$, using w_i weights.
3. Calculate the error rate on the training sample as $error_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_1}$ where y_i is the class of the instance in the training sample.

4. Calculate the classifier weight as $\alpha_m = log((1 - error_m)/error_m)$

5. Calculate new instance weights as $w_i \leftarrow w_i \cdot [\alpha_m \cdot I(y_i \neq G_m(x_i))]$ with $i = 1, 2, ..., N$.

6. Repeat steps two to five for $m = 1$ to M and calculate the final model $G(x)$.

The hyper-parameters of a AdaBoost algorithm are summarised in Table 2.4. The base estimator for the AdaBoost algorithm is the decision tree. The hyper-parameters of the decision tree also needs to be tuned before the hyper-parameters in Table 2.4 should be tuned [40].

Parameter	Description
Number of decision trees	Maximum number of trees for the ensemble method. More trees will increase accuracy but training time will also increase. The accuracy of the model will stabilise after a certain number of trees, which is dependent on the classification task and data that is used.
Learning rate	The learning rate shrinks the contribution of each tree that is added to the model in order to prevent overfitting that can be caused by too many trees in the AdaBoost algorithm

Table 2.4: AdaBoost hyper-parameters, adopted from [24, 40].

Gradient boosted trees

The gradient boosting tree algorithm is similar to the AdaBoost algorithm, but with the exception of the loss function that is minimised during training [45]. The AdaBoost algorithm's aim is to reduce the misclassification of instances in order to optimise the accuracy of the model.

Loss functions used for classifications are based on the margin function, namely $y \cdot f(x)$, which is comparable to the residuals $y - f(x)$ of regression tasks. The class

predicted by the algorithm is $f(x)$ and the true class of the instance is y. The response of a classification task is either positive or negative, with positive classifications denoted by 1 and negative classifications by -1 [24]. According to the classification rule, $G(x) = sign[f(x)]$, the margins can be used as follow [24]:

- $y_i \cdot f(x_i) > 0$ are correctly classified instances.
- $y_i \cdot f(x_i) < 0$ are incorrectly classified instances.

The goal of a classification algorithm is to increase the number of correctly classified instances by producing positive margins as frequently as possible. The misclassification loss function, as seen in Equation 2.5, adds a penalty, I, for all incorrectly classified instances [24].

$$L(y, f(x)) = I(y_i \cdot f(x_i) < 0) \tag{2.5}$$

The deviance loss functions are both approximations of the misclassification loss function. The difference between the algorithms is the degree of the loss function that is added. The exponential loss functions increases the penalty exponentially, whereas the deviance loss functions increases linearly. The deviance loss function is more advantageous then the exponential loss function, as it is more robust to noise in the data [24].

$$L(y, f(x)) = exp(-yf(x)) \tag{2.6}$$

$$L(y, f(x)) = log(1 + exp(-2yf(x)) \tag{2.7}$$

The hyper-parameters of a gradient boosted algorithm are summarised in Table 2.4.

Parameter	Description
Number of decision trees	Maximum number of trees for the ensemble method. More trees will increase accuracy but training time will also increase. The accuracy of the model will stabilise after a certain number of trees, which is dependent on the classification task and data that is used.
Learning rate	The learning rate shrinks the contribution of each tree that is added to the model in order to prevent overfitting that can be caused by too many trees in the gradient boosted algorithm

Table 2.5: Gradient boosted hyper-parameters, adopted from [24, 40].

The hyper-parameters for Gradient boosted trees are the same as the hyper-parameters of the AdaBoost algorithm. The hyper-paramters include the number of trees and the learning rate of the algorithm, as summarised in Table 2.5.

2.7 Summary

The machine learning technique that is used to create a classifier model is a supervised machine learning algorithm. The machine learning algorithm is trained on input features with assigned class labels. The classifier model predicts and assigns class labels to unseen instances based on the input features. There are various steps involved in creating a machine learning model. There are also various algorithms that can be used for classification. Decision trees and ensemble models for decision trees were chosen as the algorithms for this study, because these algorithms perform better then other algorithms with data that contains irrelevant features and missing values.

Chapter 3

Machine learning in diabetes research

The previous chapter discussed how machine learning can be used for classification tasks. This chapter reviews the application of machine learning in the research area of diabetes. Section 3.1 provides an introduction to diabetes mellitus. Section 3.2 discuses the different areas of research with regard to diabetes and machine learning. The focus of this study is on predicting diabetes, therefore Section 3.3 considers studies that have been done in this area of diabetes research.

3.1 Diabetes mellitus

Diabetes mellitus is a chronic disease that is caused by raised levels of glucose and it occurs when the body is not able to produce enough of the hormone insulin, which regulates the body's glucose levels [1]. Diabetes can be classified into two common types, namely Type 1 diabetes (T1D) and Type 2 diabetes (T2D).

T2D is the most common type of diabetes as it accounts for more than 90% of all cases [1]. The raised level of glucose is the result of the body being insulin resistant, which occurs when either not enough insulin is being produced, or the body is not responding to the insulin being produced. Studies have shown that some of the main causes of T2D

are due to modifiable lifestyle risk factors which include a poor diet, physical inactivity, obesity, and smoking [1].

T2D is the result of the body not being able to use the insulin that is produced, while T1D occurs when the body cannot produce insulin. T1D is caused when the immune system attacks the cells that produce insulin [1]. T1D is an autoimmune disease where strong genetic components and environmental triggers have been identified as potential causes [55]. The treatment for T1D is daily insulin injections in order to maintain a healthy level of glucose, because the body is not able to produce any insulin [1]. T1D can occur at any age, but is more frequently diagnosed in children and adolescents.

3.2 Machine learning applications in diabetes research

There are various machine learning applications in diabetes research. A study performed in 2016 by Kavakiotis *et al.* [30] reviewed machine learning and data mining applications within diabetes research. The study reviewed research published in PubMed, a widely used search engine for biomedical and life sciences journal literature, and the DBLP Computer Science Bibliography. The study considered research that was published between 2011 and 2016. The results for searches using the key terms "machine learning", "data mining" and "diabetes" are shown in Figure 3.1. The results were classified into five areas, namely biomarkers identification and prediction of diabetes, diabetic complications, drugs and therapies, genetic background and environment, and health care management.

Biomarkers identification and prediction of diabetes had the highest percentage of results with 47%. There were two focus areas identified within this area, namely determining diagnostic and predictive biomarkers as well as disease prediction. Biomarkers, or biological molecules, are indicators of a medical condition that can be tested and observed from outside of the patient [52], such as testing bodily fluids, for example blood, saliva or urine.

The second highest area of research focuses on diabetic complications. Diabetes has

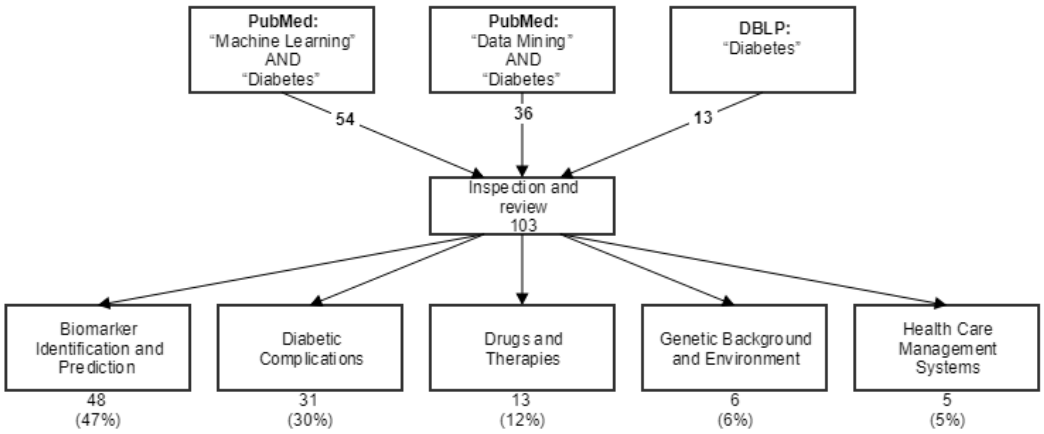

Figure 3.1: Literature review of machine learning and diabetes, adopted from [30].

been associated with major health complications, such as strokes and coronary artery disease [12]. The high morbidity and mortality rate for both T1D and T2D patients are directly and indirectly caused by the effects and complications of hyperglycaemia [30].

Medication is used to treat diabetes by maintaining the blood sugar levels, with insulin being the most common type of medication [30]. Drugs and therapies used for the treatment of diabetes is the third area of diabetes research, and the fourth area focuses on determining the genetic background and environmental risk factors involved in the onset of both T1D and T2D.

The last area is in health care management systems as health care institutions are investing significant resources in diabetes health care due to the vast number of people needing diabetes treatment [30]. The number of people diagnosed with diabetes is set to increase from 171 million in 2000 to 366 million people by the year 2030 [1] and will impact the demand on health care institutions.

The focus area of this research is the prediction of diabetes diagnosis and is discussed next.

3.3 Prediction of diabetes diagnosis

The first prediction model for the onset of diabetes was published in 1988 by Smith *et al.* [51]. The model consisted of a neural network model that was trained on the Pima Indian dataset and had an accuracy of 76%. There have since been numerous studies using machine learning algorithms improving the classifications results using the Pima Indian dataset, which is discussed in Section 3.3.1. Section 3.3.2 discusses studies done using more recent data sources, namely EHR.

3.3.1 Pima Indian dataset

The Pima Indian diabetes dataset was created from studies completed by the National Institute of Diabetes, Digestive and Kidney Diseases since 1965 [51]. The studies involved people over five years of age from the Pima Indian population in Phoenix, Arizona. The data was collected from medical examinations that were done every two years, which included an oral glucose tolerance test that was used to determine diabetes diagnosis [51].

The dataset was donated in May 1990 to the UCI Machine Learning Repository, which contains over 350 publicly available datasets that are widely used for research in the field of machine learning [14]. The dataset consists of 768 patients of which 268 were diagnosed with diabetes. There are eight features per patient, including age, number of pregnancies, glucose intolerance test results, diastolic blood pressure, tricep skin fold thickness, body mass index, diabetes pedigree function and two-hour serum insulin results.

The advantage of the Pima Indian dataset is that it has been used in various studies to evaluate and compare the performance of different classification models. A study done by Ozcift and Gulten [38] compared the performance of 30 classification algorithms, and another study by Wu *et al.* [54] compared 15 algorithms. Table 3.1 contains a summary of recent studies completed with a focus on random forest and decision tree algorithms.

Table 3.1: Decision trees and random forests studies on PIMA Indian dataset.

Publication	Year	Model and algorithms	Performance
[3]	2018	Random forest	71.6%
[13]	2017	Hybrid model with k-means for data reduction and a decision tree classifier	90.4%
[23]	2017	C4.5 decision tree algorithm	90.4%
[47]	2014	Random forest and CART	83.8%
[28]	2011	Weka classifier J48 decision tree algorithm	78.2%

3.3.2 Electronic health records

The emerging advances in the digitisation of records and information has allowed for significant technological advances and has enabled the adoption of EHR [6]. There are various data components available in EHR, such as medication administration, physical assessments, laboratory tests, diagnoses, medical history, immunisation history and various other components [27]. EHR has become a tool that is used to capture all clinical information during the treatment of a patient [5]. EHR is a significant data source of high-dimensional patient information. Table 3.2 contains a summary of diabetes prediction studies that have been done using EHR as the data source.

Table 3.2: Summary of diabetes research done with electronic health records.

Publication	Year	Model and algorithms	Dataset size	Performance
[5]	2017	Feedforward Neural Network	2280	95%
[15]	2016	Logistic Regression Model	739	90.4%
[4]	2016	Multivariate logistic regression and a random forest probabilistic model	9948	84.9%
[50]	2012	Mixture of expert systems	1500	99.36 %

The results of the studies indicate that EHR datasets can be used to train models

that can perform well and predict a diabetes diagnosis. One of the advantages of EHR is that it contains a history of the patient's medical information, which can be used to train models that can predict the onset of diabetes based on historic behaviour and trends.

3.4 Limitations of diabetes prediction studies

A limitation of studies completed using the Pima Indian dataset is the relevance of the dataset used to train the models. The dataset was created more than 50 years ago and since then there could have been many lifestyle and environmental changes in the population that was used for the dataset. Diabetes is a disease that is impacted by lifestyle and environmental factors, as mentioned in Section 3.1.

There is limited studies that have been completed using EHR data, because it is such a new field. All the studies completed on EHR data has been on the entire data with all the data components. There are no current studies that focuses on the feasibility of using the individual data components of EHR to predict the onset of diabetes, such as using only medication purchases.

3.5 Summary

This chapter discussed the research that has been done in diabetes research with machine learning methods. The majority of machine learning applications in diabetes research has been in predicting diabetes and biomarker identification. Most of the studies for predicting diabetes diagnosis have used either the Pima Indian dataset or EHR data. The Pima Indian dataset is not a relevant dataset anymore, because it is based on studies done in the 1960's. EHR is a new field of study, because it is data collected and stored using emerging technologies that are digitising medical information records. Medication administration is a data component of EHR data and there has been no studies done in determining the feasibility of only using medication administration data to predict the onset of diabetes.

Chapter 4

Medication purchasing data

Successful studies have created models that can predict a diabetes diagnosis on the EHR of a patient. The focus of this study is to determine if it is feasible to use a part of EHR, namely the history of medication administration, to create a predictive model that can assist the community pharmacists in their role in patient treatment. Section 4.1 gives an overview of the available raw data of medication purchases. Section 4.2 provides the detail of the first step in creating the machine learning model, namely identification of the required data, which describes how the raw data was used to create the dataset for training the machine learning model.

4.1 Overview of available medication purchasing data

The data available for this research includes raw transactional data of medications that were dispensed at 622 pharmacies across South Africa from 2013 to 2017. During exploration of the data, it was determined that the data contains over 215 million medication dispenses over this period. The raw data is described in the dictionary shown in Table 4.1. The data provided was already anonymous. The patients, doctors or pharmacists have no unique identifiers, such as an ID number in the data. There is also not enough data to re-engineer and determine a unique identifier, and therefore, there is no risk of unethically using personal information of either the patients, doctors, or pharmacists.

Field	Description
Patient ID	Unique and anonymous patient key.
Transaction ID	Unique and anonymous transaction key.
Dispensing date	The date that the medication was dispensed.
Dispensing pharmacy	Unique and anonymous pharmacy key where the medication was purchased.
NAPPI code	Globally unique key for the medication.
Prescribing doctor	Unique and anonymous key of the doctor who prescribed the script with the medication.
Quantity	The total number of units of the medication dispensed.
Repeats left	The number of repeats left of the script for medication.
Script number	The unique key of the script that was originally used by a doctor to prescribe the medication.
ICD10	ICD-10 is the International Classification of Diseases and Related Health Problems (10th revision). It is a coding system developed by the World Health Organisation (WHO) that translates the written description of medical and health information into standard codes.
Script starting date	The starting date of the script as the script can have multiple repeats.
Manufacturer code	The unique identifier of the manufacturer of the medication.
Drug schedule	Drugs and other substances that are considered controlled substances under the Controlled Substances Act (CSA).
MIMS ID	The Monthly Index of Medical Specialties (MIMS) ID can be used as a lookup to a table with description of what medical condition the medication is used for.
Gender	The gender of the patients, with 1 representing male and 0 representing female.

Table 4.1: Data dictionary

4.2 Identification of required data

There were two steps involved in identifying the data required for model training, namely determining the classification label and engineering the features, which is discussed in the following sections.

4.2.1 Determining diabetic label and diabetic patients

MIMS is a reference guide to the pharmaceutical products that are currently available in South Africa and it is used in various medical professions as a prescription tool [2]. The MIMS reference guide includes unique ID's for medical conditions such as Pain Syndromes, Infectious Diseases, Women's Health, Psychiatry, Cardiovascular and Metabolic Diseases, Aesthetic Medicine, and Oncology [2]. For each of these medical conditions, the MIMS ID has the information on most prescription medicines available in South Africa that can be used for treatment of the condition. The MIMS IDs associated with the treatment of diabetes were determined as 19.1.1, 19.1.2 and 19.1.3. The MIMS IDs are broken down into the three levels, and can be defined as:

- Level 1: "19" denotes medical treatment of conditions for endocrine systems. Endocrine systems include the organs in the human body that produce the hormones used to chemically control various functions of cells, tissues and other organs. The pancreas is one of the endocrine systems in the body. The pancreas is responsible for producing the hormone insulin, which regulates the body's glucose or sugar levels. As mentioned previously, diabetes occurs due to raised levels of glucose in the blood [1].
- Level 2: "1" is specific for anti-diabetic agents which are used to regulate insulin and glucose levels.
- Level 3: "19.1.1" are insulin specific prescription medications, "19.1.2" are oral agents, and "19.1.3" are all other medications used as anti-diabetic agents.

Therefore, the diabetic label for a patient was set to true if the patient was prescribed any medication with a MIMS ID equal to 19.1.1, 19.1.2, or 19.1.3.

The date of diagnosis for diabetic patients was set to the first date when medication for diabetes MIMS classes was dispensed. It is important to have a history of medication purchases before diagnosis for model training, therfore, it was decided to only use patients who were diagnosed in the year 2017. This decision ensured that the patients used for training had a history of three years of medication purchases.

4.2.2 Feature engineering

Features regarding the purchasing habits of patients had to be extracted from the raw line item purchases in the data set. The goal of feature engineering is to provide strong and, ideally, simple relationships between new purchasing habit features and the output label for the supervised learning algorithm to model. A total of 416 features were constructed and split into six categories, as summarised in Table 4.2.

Category	Features
Patient features	Age; Gender
Doctor features	Number of prescribing doctors
Scripts features	Total number of scripts; Number of scripts with no repeats; Number of scripts with less than 6 repeats; Number of scripts with repeats between 7 and 12; Number of scripts with more than 12 repeats
Pharmacy visits	Number of pharmacy visits; Maximum number of pharmacy visitv per month
Medication features	Number of days medication was used before diagnosis for each MIMS class

Table 4.2: Summary of features engineered

4.3 Summary

The final dataset consisted of 416 features, which is a high-dimensional dataset that is also sparse, as there were missing values for some of the 395 features for medications used for each MIMS ID. As discussed in Section 2.4.2, decision trees and random forests are good choices of classification algorithm due to these attributes of the dataset. The following chapter discusses the experimental results of the random forest classification model.

Chapter 5

Empirical analysis

The scope of this study is to build a machine learning model that will attempt to predict a diabetes diagnosis based on the medication purchasing habits of patients. Section 5.1 discusses the experimental environment. The steps of and results for the machine learning model are discussed in Section 5.2.

5.1 Experimental environment

The platforms and systems used for the experiments are shown in Figure 5.1. The raw transactional data is stored in a PostgreSQL database. PostgreSQL is an open source object-relational database. Data extraction was done with scripting in the PostgreSQL database. The prescription table is the main table, with each medication dispensed to a patient as a line item. Scripting was done to create a raw extraction of all medication prescribed to patients, with all relevant data needed for the feature engineering to create the training and testing datasets. The raw extraction was exported as a CSV file.

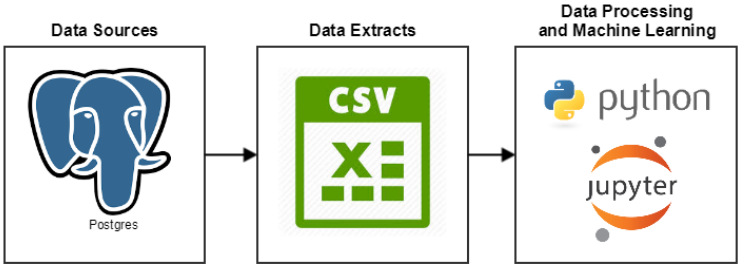

Figure 5.1: Environment and platforms used for experiments.

Jupyter notebooks were created to run the experiments in the Jupyter web application. The notebook kernel runs on Python 3. The Python packages used for the project are described in Table 5.1. Python was chosen for this research as it has become one of the most popular languages for scientific computing [40]. The Python packages used for the experiments are summarised in Table 5.1.

Packages	Description
NumPy	Fundamental package for scientific computing with a powerful N-dimensional array object.
Pandas	Module used for manipulation of data.
Matplotlib	Module used to create data visualisations.
Scikit-Learn	Module used for machine learning modelling.

Table 5.1: Python packages used during experiment

5.2 Machine learning experiments

The framework discussed in Section 2.3 for building a supervised machine learning model was followed in creating the classification model. There were no tasks required for data pre-processing. There was no incomplete data in die medication purchasing data. Also, decision trees and decision tree ensemble models perform well with noisy data. Therefore, there was no requirement to remove incomplete or noisy data.

Section 5.2.2 shows the results of parameter tuning and model training and Section 5.2.3 discusses the evaluation of the final model against the test data set.

5.2.1 Definition of training set

The data set was split randomly into a training set containing 70% of the data and a test set containing 30% of the data. The training and test sets are summarised in Table 5.2.

Data set	Diabetic	Non-diabetic	Total
Training	36 066	36 623	72 689
Test	15 475	15 677	31 152

Table 5.2: Summary of patients used for training and test data set

5.2.2 Parameter tuning and model training

Parameter tuning and model training were done in five rounds:

- Round 1: Decision tree model
- Round 2: Random forests model
- Round 3: Extremely randomised model
- Round 4: Adaboost model
- Round 5: Gradient boosting model

Round 1: Decision tree model

Pruning of the decision tree model was done to determine what depth of the tree should be used. The only hyper-parameter that needs to be tuned when the pruning method is used, is the maximum depth of the tree. Other parameters like minimum sample split or minimum sample leaf should be set to values that would not cause a stopping rule

when growing the tree. The steps followed for hyper-parameter tuning of the decision tree model is summarised as follow:

1. Created a range of tree depths of one to 100.
2. Trained a decision tree classifier for each depth in the tree depth range using the training dataset.
3. The accuracy score of each trained model was determined using the validation dataset.

A decision tree model was created using the decision tree classifier in the Scikit machine learning package [40]. The default values for the other parameters that of the Scikit decision tree classifier were used and are summarised in Table 5.3.

The results for the different decision tree models with different depths are summarised in Figure 5.2. The maximum accuracy of the validation set is between 88% and 89%. The decision tree model had a first value for accuracy score in this range at depth 11, which is 88.05%. The trees with depths more than 11 does not improve on the accuracy and only reaches 89% at depth 72. Therefore, the tree was pruned to a depth of 11.

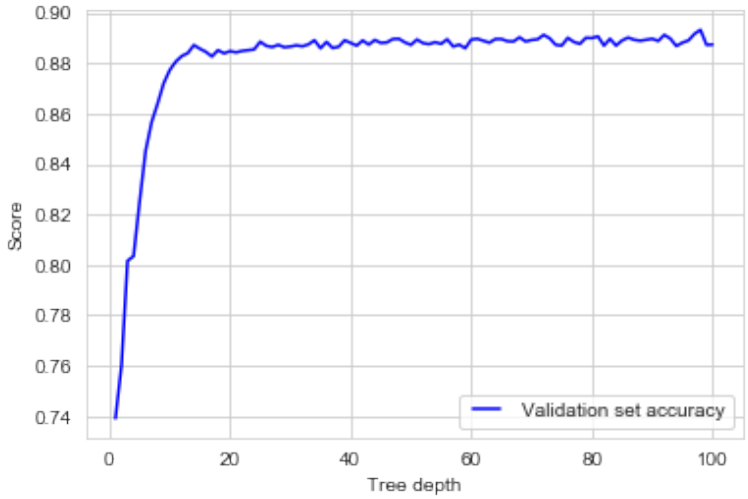

Figure 5.2: Parameter tuning results for tree depth for decision tree.

Parameters	Description	Default Value
Criterion	The function used to measure the quality of the split	Gini
Splitter	Strategies used to determine split, which include determining the best or determining a random split	Best
Minimum samples split	The minimum number of instances required to split an internal node	2
Minimum samples leave	The minimum number of instances required to be a leave node	1
Minimum weight fraction leaf	The minimum weighted fraction of the sum total of weights of all the input instance required to be at a leaf node. Samples have equal weight when sample weight is not provided.	0
Random state	The seed used for the random number generator.	None
Maximum leaf nodes	The maximum number of leaf nodes for the tree.	None
Minimum impurity decrease	A node will be split if this split induces a decrease of the impurity greater than or equal to this value.	0
Minimum impurity split	Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.	0
Class weight	Weights associated with classes.	None
Presort indicator	Whether to presort the data to speed up the finding of best splits in fitting.	False

Table 5.3: Decision tree classifier parameters and values

Round 2: Random forests model

The two main hyper-parameters that affect the accuracy of a random forest model are [40]:

- Number of trees in the model: The size of the forests is determined by the number of decision trees in the forest. The accuracy of the random forest model will increase up to a point as the number of trees in the forest increases.
- Maximum number of features considered for splitting a node: The lower the number of maximum features that can be used to split the lower the variance, but the higher the bias of the model.

Parameter tuning was done in two stages. The number of trees was tuned first, followed by the maximum number of features. The default values from the Scikit model was used for the other parameters, as summarised in Table 5.4.

Number of decision trees: The bootstrap method selects a random subset of the training dataset to train the model. The training data that was not selected for the subset is used as an internal validation dataset to calculate the out-of-bag (OOB) error. The results are shown in Figure 5.3. The OOB estimate stabilises at 160 trees and is the number of trees for the random forest algorithm.

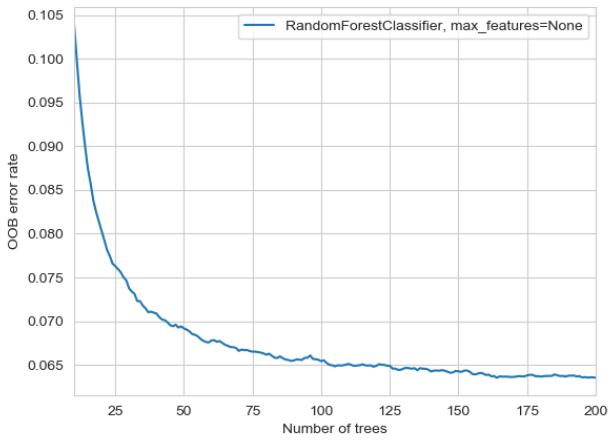

Figure 5.3: Parameter tuning results for number of trees in random forests model.

Parameters	Description	Default Value
Criterion	The function used to measure the quality of the split	Gini
Splitter	Strategies used to determine split, which include determining the best or determining a random split	Best
Maximum depth	The maximum depth of the tree.	None
Minimum samples split	The minimum number of instances required to split an internal node	2
Minimum samples leave	The minimum number of instances required to be a leave node	1
Minimum weight fraction leaf	The minimum weighted fraction of the sum total of weights of all the input instance required to be at a leaf node. Samples have equal weight when sample weight is not provided.	0
Random state	The seed used for the random number generator.	None
Maximum leaf nodes	The maximum number of leaf nodes for the tree.	None
Minimum impurity decrease	A node will be split if this split induces a decrease of the impurity greater than or equal to this value.	0
Minimum impurity split	Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.	0
Class weight	Weights associated with classes.	None
Presort indicator	Whether to presort the data to speed up the finding of best splits in fitting.	False

Table 5.4: Random forests classifier parameters and values

Maximum number of features The Scikit random forest classifier model has three different options for setting the maximum number of features, namely:

- Sqrt: Maximum number of features = $\sqrt{Number\ of\ features}$
- Log2: Maximum number of features = $\log_2(Number\ of\ features)$
- None: Maximum number of features = $Number\ of\ features$

The exhaustive grid search with cross validation function of the Scikit library was used to determine the best value for the maximum number of features. The parameter grid included all three values for the maximum number of features. The grid search was completed with a 10-fold cross validation with the training set. The results of the grid search is summarised in Table 5.5. The best value for the maximum number of features is the square root of the total number of features.

Value	Average cross validation score	Rank
Sqrt	93.53%	1
Log2	93.30%	2
None	93.05%	3

Table 5.5: Grid search results for maximum number of features for random forest model

Round 3: Extremely randomised model

The only variation between the random forest algorithm and the extremely randomised algorithm is the randomness added when a node split is determined from a random subset of features. The same parameters as the random forest model, namely number of trees and maximum number of features, were tuned. The extra trees classifier from the Scikit library was used to create the extremely randomised model [40]. The default values for the other parameters is the same as the random forest model and are summarised in Table 5.4.

Number of decision trees: The OOB estimate for the extremely randomised model was determined for a range of 10 to 200 number of decision trees and is displayed in Figure 5.4. The OOB estimate stabilises at 170 trees and is therefore the best value to be used for the number of trees for the extremely randomised model.

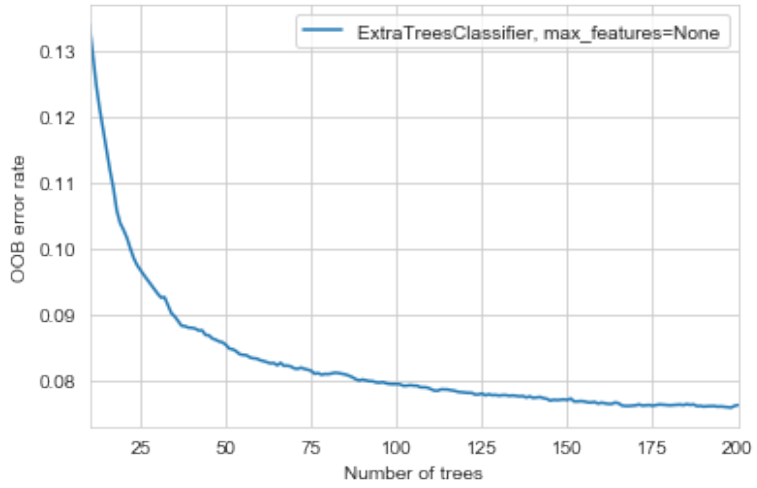

Figure 5.4: Parameter tuning results for number of trees in extremely randomised model.

Maximum number of features: The exhaustive grid search with cross validation function of the Scikit library was used to determine the best value for the maximum number of features for the extremely randomised model. The parameter grid included all three values for the maximum number of features. The grid search was completed with a 10-fold cross validation with the training set. The results of the grid search is summarised in Table 5.6. The best value for the maximum number of features is to use all the features and have no maximum value for number features to consider for the split functions.

Round 4: AdaBoost model

The AdaBoost algorithm can be used with various base estimators. The decision tree model trained in round 1 was used as the base estimator. The depth of the decision tree was set to 11, as this was the depth obtained from parameter tuning. The two hyper-parameters that needs to be tuned for an AdaBoost model include [40]:

Value	Accuracy score	Rank
None	92.97%	1
Sqrt	92.36%	2
Log2	91.61%	3

Table 5.6: Grid search results for maximum number of features for extremely randomised model

- Number of trees in the model: The number of trees where boosting is terminated.
- Learning rate: The learning rate shrinks the contribution of each tree that is added to the model.

There is a trade-off between the number of trees and the learning rate. The lower the learning rate, the more trees can be used before the model overfits. The higher the learning rate, the fewer trees can be used before the model overfits.

The exhaustive grid search with 10-fold cross validation function of the Scikit library was used to determine the best value for the number of trees and the learning rate. The grid included the following values for the hyper-parameters:

- Number of trees in the model: 100; 200; 300; 350; 400
- Learning rate: 1; 0.5; 0.25; 0.1

The grid resulted in four models, one for each of the learning rates. The average accuracy results from the 10-fold cross-validation for the models with different number of trees, are shown in Figure 5.5. A learning rate of 0.5 resulted in the best model across all the values for the number of trees in the model. There is a slight decrease in the accuracy at 350 trees for the model with a learning rate of 0.5. The decrease in the results is indicative of over fitting of the model on the training data. The boosting of the final model with learning rate of 0.5 is therefore terminated at 300 trees.

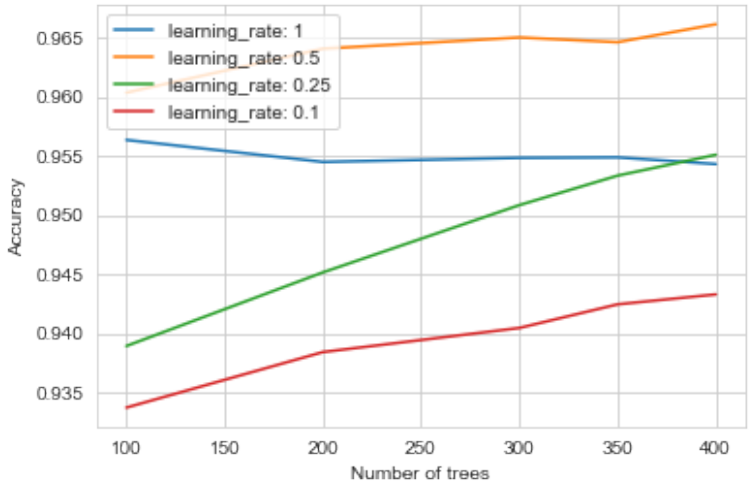

Figure 5.5: Parameter tuning results for AdaBoost model.

Round 5: Gradient boosting model

The gradient boosting model has the same parameters and hyper-parameters as the AdaBoost model. The same grid with cross validation search was used as for the AdaBoost model. The grid resulted in four models, one for each of the learning rates. The average accuracy results from the 10-fold cross-validation for the models with different number of trees, are shown in Figure 5.6. The highest accuracy was obtained with a model with a learning rate of 0.25 at 300 trees. There is a decrease in the accuracy of the model at 350 trees, which is indicative of over fitting of the model on the training data. The boosting of the final model with learning rate of 0.25 is therefore terminated at 300 trees.

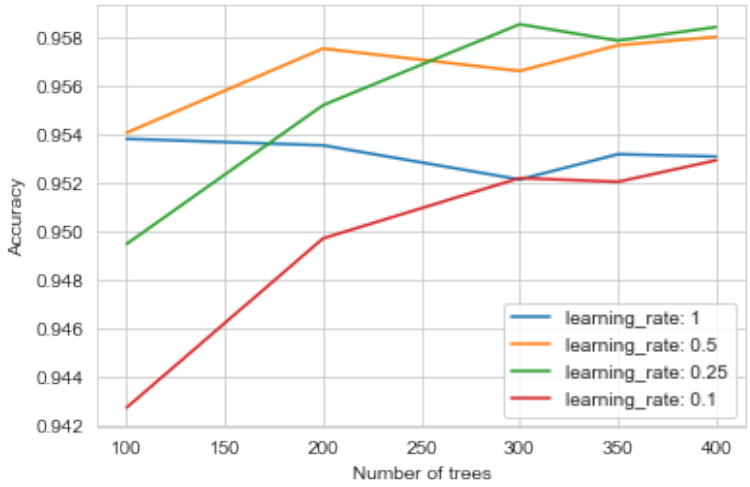

Figure 5.6: Parameter tuning results for gradient boosting model.

5.2.3 Model evaluation

All five models were evaluated on the testing dataset with 10-fold cross validation. The results are summarised in Table 5.7. As expected, the decision tree was the worst performing model with an accuracy of 88%. The boosting algorithms performed better then the bagging algorithms. Both the boosting algorithms, namely the adaboost and gradient boosting trees, had an accuracy of 95%.

Model	Accuracy score	AUC score
Decision tree	88%	0.94
Random forests	93%	0.98
Extremely randomised	92%	0.98
Adaboost	95%	0.98
Gradient boosting	95%	0.98

Table 5.7: Avergae accuracy and AUC scores for 10-fold cross validation.

The ROC curve for the four bagging and boosting algorithms were the same. The ROC curve for the Adaboost algorithm is shown in Figure 5.7. The model is predicting the diabetes diagnosis accurately with an average AUC of 0.98 for the 10-fold cross-validation.

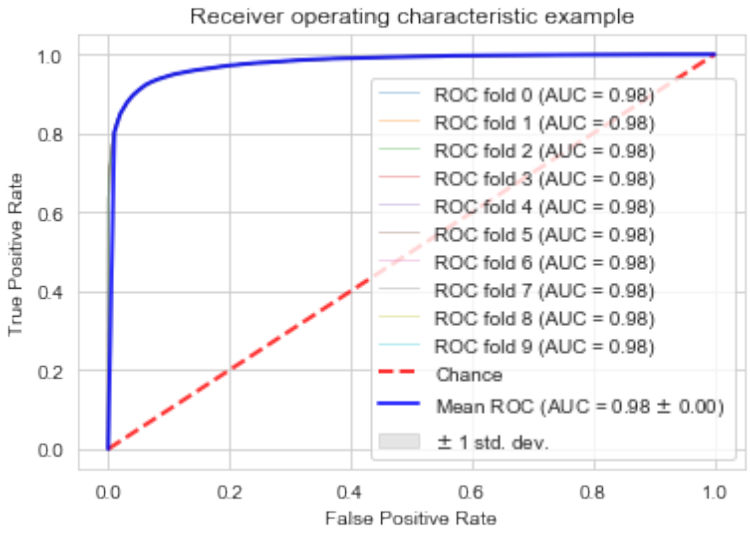

Figure 5.7: ROC curve for Adaboost model.

5.3 Summary

Five classifier models were trained and evaluated on the medication purchasing habits. 10-fold cross-validation was used for both hyper-parameter tuning and model evaluation. The decision tree was the worst performing model with an accuracy of 88%. As expected, the ensemble models improved the accuracy of the single decision tree. The bagging algorithms, namely random forest and extremely randomised model, had an accuracy of 92% and 93%. The boosting algorithms, namely AdaBoost and gradient boosting trees, had both an accuracy of 95%.

Chapter 6

Conclusions

This chapter discusses the summary of conclusions in Section 6.1 and possible future areas of research in Section 6.2.

6.1 Summary of Conclusions

There were two objectives for this research, namely investigating current machine learning applications in the field of diabetes research and testing the feasibility of using medication history for the development of an effective prediction model for diabetes diagnosis.

6.1.1 Machine learning application in the field of diabetes research

Machine learning applications have successfully been used in the field of diabetes research, with the prediction of the onset of diabetes as the main focus area. There have been two main datasets used for the prediction of diabetes, namely the PIMA Indian dataset from 1965 and EHR. EHR is an emerging area of interest due to technology advances and is a potential data source that can be used to train machine learning models. The accuracy of machine learning models used for predicting the onset of diabetes ranged from 84% to 99%. Medication administration is a part of the EHR and a ma-

chine learning model for predicting diabetes diagnosis based on medication purchases is a novel contribution, as it has not been done before.

6.1.2 Machine learning model based on medication purchasing habits

Medication purchases from 622 pharmacies across South Africa between 2013 and 2017 was used to create a dataset of medication purchasing features for diabetic and non-diabetic patients. The dataset is very sparse, with a high number of missing data values for each medication type. Various classification algorithms were evaluated and decision trees performs the best when the dataset has missing values and the algorithm also has a good tolerance to irrelevant features for a high-dimensional dataset.

There were five decision tree based algorithms evaluated for the machine learning model. The first was a decision tree model that was pruned during training. The other algorithms were decicion tree ensemble models. Random forests and extremely randomised model use bagging methods to add randominsation to the machine learning model to improve the generalisation error of a single decision tree. AdaBoost and gradient boosting algorithms use boosting methods to build the ensemble model. The results of the emperical analysis of these algorithms with the dataset is summarised in Table 6.1

Model	Accuracy score	AUC score
Decision tree	88%	0.94
Random forests	93%	0.98
Extremely randomised	92%	0.98
Adaboost	95%	0.98
Gradient boosting	95%	0.98

Table 6.1: Avergae accuracy and AUC scores for 10-fold cross validation.

The decision tree was the worst performing model with an accuracy of 88%. As expected, the ensemble models improved the accuracy of the single decision tree. The

boosting algorithms performed the best, namely AdaBoost and gradient boosting trees, had both an accuracy of 95%.

6.2 Future Work

Two future research areas have been identified through the course of this dissertation, as summarised below:

- Determining the contributing medication purchasing features that can be used to create prevention plans for diabetes.
- Validating the model by performing clinical tests on a sample set of patients to determine if the prediction of diabetes diagnosis based on medication purchases was accurate.

Bibliography

[1] *IDF Diabetes Atlas*. 8th edition, 2017.

[2] MIMS South Africa. *MIMS Annual: 2018*. MIMS South Africa, 2018.

[3] K. Akyol and B. Sen. Diabetes mellitus data classification by cascading of feature selection methods and ensemble learning algorithms. *International Journal of Modern Education and Computer Science*, 10(6):10 – 16, 2018.

[4] A.E. Anderson, W.T. Kerr, A. Thames, T. Li, J. Xiao, and M.S. Cohen. Electronic health record phenotyping improves detection and screening of type 2 diabetes in the general united states population: A cross-sectional, unselected, retrospective study. *Journal of Biomedical Informatics*, 60:162 – 168, 2016.

[5] J.P. Anderson, J.R. Parikh, D.K. Shenfeld, V. Ivanov, C. Marks, B.W. Church, J.M. Laramie, J. Mardekian, B. Piper, R.J. Willke, and et al. Reverse engineering and evaluation of prediction models for progression to type 2 diabetes. *Journal of Diabetes Science and Technology*, 10(1):6 – 18, 2015.

[6] C.M. Angst and R. Agarwal. Adoption of electronic health records in the presence of privacy concerns: The elaboration likelihood model and individual persuasion. *MIS Quarterly*, 33(2):339–370, 2009.

[7] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, July 2009.

[8] M.S. Barale and D.T. Shirkec. Cascaded modeling for pima indian diabetes data. *International Journal of Computer Applications*, 139:1–4, 2016.

[9] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[10] A. Boulestiex, S. Janitza, J. Kruppa, and I.R. Konig. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. Technical report, Department of Statistics, University of Münich, Germany.

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

[12] W. T. Cade. Diabetes-related microvascular and macrovascular diseases in the physical therapy setting. *Physical Therapy*, 88 11:1322–35, 2008.

[13] W. Chen, S. Chen, H. Zhang, and T. Wu. A hybrid prediction model for type 2 diabetes using k-means and decision tree. In *The 8th IEEE International Conference on Software Engineering and Service Science*, pages 386–390, 2017.

[14] D. Dau and E. Karra Taniskidou. UCI machine learning repository, 2017.

[15] M.N Devi, A.A. Balamurugan, and M.R. Kris. Developing a modified logistic regression model for diabetes mellitus and identifying the important factors of type 2 diabetes. *Indian Journal of Science and Technology*, 9(4):1–8, 2016.

[16] A.P. Engelbrecht. *Computational Intelligence: An Introduction.* John Wiley and Sons Chichester, England, second edition, 2007.

[17] F. Esposito, D. Malerba, G. Semeraro, and J. Kay. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

[18] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[19] J.H Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

[20] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[21] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan Kaufmann, 2012.

[22] M. I. Harris, R. Klein, T. A. Welborn, and M. W. Knuiman. Onset of NIDDM occurs at least 4-7 yr before clinical diagnosis. *Diabetes Care*, 15(7):815 – 819, 1992.

[23] E. K. Hashi, M. S. U. Zaman, and M. R. Hasan. An expert clinical decision support system to predict disease using classification techniques. In *International Conference on Electrical, Computer and Communication Engineering*, pages 396–400, 2017.

[24] T. Hastie, R. Tibshirani, and J.H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2008.

[25] M. Holsheimer and A. P. J. M. Siebes. Data mining: The search for knowledge in databases. Technical report, Centrum voor Wiskunde en Informatica, P. O. Box 94079, NL-1090 GB Amsterdam, The Netherlands, 1994.

[26] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali. A comparative study of decision tree ID3 and C4.5. *International Journal of Advanced Computer Science and Applications*, 4(2):13–19, 2014.

[27] K. Hyrinen, K. Saranto, and P. Nyknen. Definition, structure, content, use and impacts of electronic health records: A review of the research literature. *International Journal of Medical Informatics*, 77(5):291 – 304, 2008.

[28] A. Jarullah. Decision tree discovery for the diagnosis of type II diabetes. In *International Conference on Innovations in Information Technology*, pages 303–307, 2011.

[29] N. Jayanthi, B. Vijaya Babu, and N. Sambasiva Rao. Survey on clinical prediction models for diabetes prediction. *Journal of Big Data*, 4(1):2 – 6, 2017.

[30] I. Kavakiotis, O. Tsave, A. Salifoglou, N. Maglaveras, I. Vlahavas, and I. Chouvarda. Machine learning and data mining methods in diabetes research. *Computational and Structural Biotechnology Journal*, 15:104 – 116, 2017.

[31] P. Koch, B. Wujek, O. Golovidov, and S. Gardner. *SAS Global Forum 2017 Conference*. SAS Institute Inc., 2017.

[32] S.B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatics*, 31:249–268, 2007.

[33] K. Anandha Kumar and A. Bharathi. A survey on diabetes mellitus prediction using machine learning techniques. 2016.

[34] S. Liao, P. Chu, and P. Hsiao. Data mining techniques and applications: A decade review from 2000 to 2011. *Expert Systems with Applications*, 39(12):11303 – 11311, 2012.

[35] E. Mossialos, E. Courtin, H. Naci, S. Benrimoj, M. Bouvy, K. Farris, P. Noyce, and I. Sketris. From retailers to health care providers: Transforming the role of community pharmacists in chronic disease management. *Health Policy*, 119(5):628 – 639, 2015.

[36] S.K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[37] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169 – 198, 1999.

[38] A. Ozcift and A. Gulten. Classifier ensemble construction with rotation forest to improve medical diagnosis performance of machine learning algorithms. *Computer Methods and Programs in Biomedicine*, 104(3):443 – 451, 2011.

[39] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, Mar 1990.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[41] V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman. Decision trees: An overview and their use in medicine. *Journal of Medical Systems*, 26(5):445–463, Oct 2002.

[42] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[43] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77 – 90, 1996.

[44] J.R Quinlan. Simplifying decision trees. *International Journal of Human-Computer Studies*, 51(2):497 – 510, 1999.

[45] L. Rokach. Decision forest: Twenty years of research. *Information Fusion*, 27:815 – 819, 2016.

[46] J. Van Ryzin. Classification and regression trees. *Journal of the American Statistical Association*, 81(393):253–253, 1986.

[47] M. T. Mira Kania Sabariah, S. T. Aini Hanifa, and M. T. Siti Sa'adah. Early detection of type II diabetes mellitus with random forest and classification and regression tree (CART). In *International Conference of Advanced Informatics: Concept, Theory and Application*, pages 238 – 242, 2014.

[48] W. Salih. Outcomes of pharmacist interventions to reduce cardiovascular risk factors in diabetes: An evidence-based review. *South African Journal of Diabetes*, 10(2):17 – 20, 2017.

[49] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1657 – 1686, 1998.

[50] D.O Shankaracharya, S. Samanta, and A.S. Vidyarthi. Computational intelligence-based diagnosis tool for the detection of prediabetes and type 2 diabetes in India. *The Review of Diabetic Studies*, 9(1):55 – 62, 2012.

[51] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 261–265. American Medical Informatics Association, 1988.

[52] K. Strimbu and J.A. Tavel. What are biomarkers? *Current Opinion in HIV and AIDS*, 5(6):463 – 466, 2010.

[53] W.S. van Heerden. Self-organizing feature maps for exploratory data analysis and data mining: A practical perspective. Master's thesis, Faculty of Engineering Built Environment and Information Technology University of Pretoria, Pretoria, 2017.

[54] H. Wu, S. Yang, Z. Huang, J. He, and X. Wang. Type 2 diabetes mellitus prediction model based on data mining. *Informatics in Medicine Unlocked*, 10:100 – 107, 2018.

[55] W. You and M. Henneberg. Type 1 diabetes prevalence increasing globally and regionally: the role of natural selection and life expectancy at birth. *BMJ Open Diabetes Research and Care*, 4(1):1 – 7, 2016.

Appendix A

Acronyms

This appendix alphabetically list the acronyms used in this dissertation.

- AdaBoost: Adaptive Boosting
- AUC: Area under curve
- CART: Classification and regression trees
- CV: Cross-validation
- EHR: Electronic health records
- kNN: k-Nearest Neighbour
- MIMS: Monthly Index of Medical Specialties
- ROC: Receiving operating characteristics
- T1D: Type 1 diabetes
- SVM: Support vector machines
- T2D: Type 2 diabetes