Predicting Diabetes Using Tree-based Methods

By Hyunjin Nam

Department of Statistics

Uppsala University

Supervisor: Måns Thulin

2019

Abstract

The aim of this study is to develop a statistical model to predict type 2 diabetes based on the tree-based model. Furthermore, the aim to compare classification with current medical criteria. Used 60,318 patient's data with demographic factors and laboratory measurements from MIMIC III database. 12,933 patients are pre-diagnosed as having diabetes and will implement supervised learning based on tree models. Decision Tree, Random Forest, Boosting with a XGBoost algorithm is used as a classification method to predict diabetes. The results show XGBoost outperformed the two other models in yielding highest classification rate, with a 84.6% test accuracy. However, the two other methods also show relevantly high accuracy, which is comparable with the physician's medical approach. Two interesting findings from this paper are: 1) Ensemble methods such as Random forest and boosting can be easily overfitted on training data, but this problem can be solved with correct hyper-parameter tunning. And 2) Tree-based methods such as XGboost and Random Forest can solve variables' multicollinearity problems.

Keywords— Diabetes, Machine Learning, Tree-based Model, Decision Tree, Random Forest, Boosting, XGBoost

Contents

1 Introduction

1.1 Introduction

Diabetes Mellitus, commonly referred to as diabetes, is a chronic disease that occurs either when the pancreas does not able to produce enough insulin or when the body cannot effectively use the insulin it produces.centre (n.d.) Diabetes can be dived into two broad categories by their cause and development of the disease; 1) Type 1 diabetes, which is also called insulin-dependent, juvenile diabetes. And 2) type 2 diabetes mellitus, which is also called non-insulin-dependent or adult-onset diabetes. In this study, I will concentrate on diagnosing type 2 diabetes. Type 2 diabetes is a metabolic disorder that results in hyperglycemia(high blood glucose levels) due to the body. It is a disease in which the body is ineffective at using the insulin it has produced; also known as insulin resistance or Being unable to produce enough insulin.

Table 1: Clinical guide line for the diagnosis of type 2 diabetes

- An HbA1c level(A1C) $\geq 6.5\%$. The test should be performed in a laboratory using a method that is certified by the National Glycohemoglobin Standardization Program (NGSP) and standardized or traceable to the Diabetes Control and Complications Trial(DCCT) reference assay.* OR
- A fasting plasma glucose (FPG) level $\geq$ 126 mg/dl (7.0 mmol/l). Fasting is defined as no caloric intake for at least 8 h.* OR
- A plasma glucose $\geq$ 200 mg/dl (11.1 mmol/l) during an oral glucose tolerance test (OGTT). The test should be performed as described by the World Health Organization, using a glucose load containing the equivalent of 75 g anhydrous glucose dissolved in water.* OR
- In a patient with classic symptoms of hyperglycemia (i.e., polyuria, polydipsia, polyphagia, weight loss) or hyperglycemic crisis, random plasma glucose geq 200 mg/dl (11.1 mmol/l).

*In the absence of unequivocal hyperglycemia, criteria 1–3 should be confirmed by repeat testing.

Table 1 shows the diagnostic criteria for type 2 diabetes. Besides the medical guideline for the diagnosis of type 2 diabetes, several factors can increase the risk of developing type 2 diabetes. It is well known that the risk of developing diabetes increases with age. According to the CDC report, 4.0 % of people aged 18 to 44 years are living with diabetes, 17 % of those aged 45 to 64 years, and 25.2 % of those aged over 65 years are living with diabetes. Centers for Disease Control and Prevention (2017)

Also, people of a specific ethnicity is more likely to develop type 2 diabetes. African Americans, Mexican Americans, American Indians, Native Hawaiians, Pacific Islanders and Asian Americans are known for having a higher risk for these diseases. This can be explained by genetic reasons, such as food consumption and lifestyle factors that can intrigue type 2 diabetes. Association (2011) Moreover, being overweight or obese, eating an unhealthy diet, physical inactivity, and having high blood pressure or raised cholesterol levels can increase the possibility of developing type 2 diabetes. Community (2018)

Type 2 diabetes is a non-curable chronic disease that can bring many complications which can threaten life, and the early stage of diabetes have no apparent symptoms and is hard to detect. Therefore, early diagnosis to prevent the further process of diabetes, and its complications is a critical issue in the medical field.

As a lot of medical research has produced reliable clinical guidelines, there is good motivation to use a statistical approach. Also, many studies have been shown that machine learning can be precisely applied to various topics in medical health. Hence, in this paper, I would like to develop a statistical model to predict type 2 diabetes based on the machine learning algorithm, and compare this model with preexisting medical pathology to check if the statistical approach with patients' data can detect diabetes correctly.

The objective problem has two main challenges. The first challenge is variable selection. As the different diseases are requiring only a certain amount of lab test, it is crucial to select the right variables and discard the rest of the variables. There is a lot of machine learning technique that can use all of the data to have excellent predict performance. However, we want to use a small number of variables for future predictions so that patients can take the right amount of test to receive the diagnosis. Secondly, the result should be easy to interpret. As it is a medical diagnosis, we need to build a model that is easy to explain the reason why the person is having certain diseases or not.

The methodology that will be mainly used will be the tree-based model, which is one of the most frequently used methods in the medical field. It is easy to interpret, mirrors physician's criteria and gives relatively high accuracy than any other statistical methods. A decision tree is the base-line for the tree-based model, and the two other methods will be modifications of this algorithm. Random forest and Boosting are called Ensemble methods that combine multiple classifiers (several decision trees) in different ways. A random forest can decrease variance, and boosting can reduce bias.

1.2 Previous study

There are several previous studies regarding the diagnosis of type 2 diabetes using machine learning algorithms. Chi et al. (2008) shows type 2 Diabetes diagnosis based on quantum particle swarm optimisation (QPSO) algorithm and weighted least squares support vector machines (WLS-SVM) is presented. This paper shows how to overcome the disadvantage of low diagnostic accuracy caused by lacking the prior knowledge and finite learning samples.

Moreover, according to the study done by Zou et al. (2018), they used a decision tree, random forest and neural network to predict diabetes mellitus. This paper implemented principal component analysis(PCA) and minimum redundancy maximum relevance(mRMR) to reduce the dimensional. Dimensional reduction skills they used in this paper was impressive, but this method cannot be applied in this paper as I want to select a small number of input variables that can effectively predict the output.

Lee et al. (2011) used Genetic and Clinical Data to predict type 2 diabetes. The implemented methods are Support Vector Machine, C4.5, logistic regression, and K-nearest neighbour. The interesting trial they have done is to cluster the data by triacylglycerides and BMI values so that it will predict better for each group. The misclassification rates from this study were from 28.24 to 52.69% in the different models. One of the reasons present by this paper that may explain the higher misclassification rates obtained was in data handling. First, categorical data which should be converted into numerical data to measure the distance with precision were kept in numerical data. Also, small sample size, which was 684 patients with only 212 diabetes patients, was not enough to build the correct model.

From previous results, I found tree-based methods have strengths with predicting diabetes. This is because a greedy algorithm in tree-based makes the model simple and select a good amount of variables. On the other hand, previous researches show this approach can have weaknesses of causing low accuracy. However, it may be overcome with using a large set of data with the correct way of processing. Also, machine learning algorithm like as XGBoost in a tree-based model can give better performance than a simple decision tree by tuning parameters.

2 Background

2.1 Definitions

2.1.1 Train and Test data-set

Before going into the analysis process, a dataset can be divided into training and testing data. Most of the data is used for training, and a smaller amount of the data is used for testing. Training data is used to fit a parameter and builds the models to find an algorithm to map the function $Y = f(X)$ where X is an input vector, and Y is an output vector. In this study, the output vector is the diagnosis of diabetes and $f(X)$ is the decision tree, random forest, and boosting.

Testing data is independent of the training dataset. Test data do not affect the building if the model, but it is instead used to check the performance of the model. In this paper, I divide the full dataset into training and testing data with 80:20 ratio.

2.1.2 k-Fold Cross-Validation

Table 2: Algorithm for k-Fold Cross-Validation Brownlee (2018)

```
1. For b = 1 to B:
   (a) Shuffle the dataset randomly.
   (b) Split the dataset into k groups.
   (c) For each unique group:
         i. Take one group as a hold out or validation data set.
        ii. Take the remaining k-1 groups as a training data set.
       iii. Fit a model on the training set and evaluate it on the validation set.
        iv. Retain the performance score and discard the model.
   (d) Summarize the model by computing average of K models' performance.
```

k-Fold Cross-Validation helps to build a better model by checking the performance of validation dataset and determining how well the model will generalise. This approach requires randomly dividing the set of observations into k different groups of approximately equal amount. The first fold is used as a validation set, and the method is fit on the remaining k 1 folds. The accuracy is then calculated on the observations in the held-out fold. An Introduction to Statistical Learning James et al. (2013)

This procedure is repeated k times. For each time, a different group of observations is treated as a validation set. This process gives k estimates of the validation performance for the result. The k-fold Cross Validation estimate is computed by averaging the performance.

In this paper, Decision tree and boosting are using 10-fold validation to test the model's performance to predict the new data that was not used in training for estimation.

2.1.3 Model Evaluation Error Metrics

Choosing the right evaluation matrix for classification models is vital to building an appropriate model. In this paper, test prediction accuracy is the main criterion to measure the performance of each model. However, other criteria, such as confusion matrix and Kappa, will also be considered. Also, Table 4 shows the terms that will be used as a model's performance measurement.

Table 3: Confusion matrix

	Ref:0	Ref:1
Pred:0	True Negative	False Negative
Pred:1	False Positive	True Positive

Table3 shows the confusion matrix. There are four sectors in the confusion matrix. True positive is defined as the number of diabetes cases that are correctly predicted as diabetes. Also, false positive is defined as the number of healthy cases that are incorrectly predicted as diabetes. True negativ is defined as the number of healthy cases that are correctly predicted as healthy. Lastly, false negative is defined as the number of diabetes cases that are incorrectly predicted as healthy.

Table 4: Definition of terms Ogunleye and Qing-Guo (2019)

- Accuracy : This is used to determine the amount of a particular class that is correctly predicted over the total number of sample. It can be calculated as $\frac{TP+TN}{TP+TN+FP+FN}$.
- Sensitivity : The ratio of the number of correctly predicted diabetes cases over the total number of the diabetes cases. It can be calculated as $\frac{TP}{TP+FN}$.
- Specificity : The ratio of the number of correctly predicted healthy cases over the total number of healthy cases. It can be calculated as $\frac{TN}{FP+TN}$.
- Positive predictive value : The ratio of the number of correctly predicted diabetes cases over the total number of cases predicted as diabetes. It can be calculated as $\frac{TP}{TP+FP}$.
- Negative predictive value : The ratio of the number of correctly predicted as healthy cases over the total number of the cases predicted as healthy. It can be calculated as $\frac{TN}{TN+FN}$.
- Prevalence : The total ratio of the number of diabetes cases. $\frac{FN+TP}{TP+TN+FP+FN}$.
- Detection Rate : The ratio of the number of correctly predicted as diabetes over the total number of cases. $\frac{TP}{TP+TN+FP+FN}$.
- Detection Prevalence : The ratio of the number of predicted as diabetes over the total number of cases. $\frac{FP+TP}{TP+TN+FP+FN}$.
- Balanced Accuracy : The average of the proportion corrects of each class individually $(\frac{TP}{TP+FN} + \frac{TN}{FP+TN})/2$.

When evaluating classification within two different groups, Cohen's Kappa, often called Kappa (κ) is often used. To compute Kappa, first, need to calculate the observed level of agreement by a confusion matrix.

$$P_0 = \frac{TN+TP}{TN+FN+FP+TP} \tag{1}$$

$$P_N = \frac{TN+FP}{TN+FN+FP+TP} \cdot \frac{FN+FP}{TN+FN+FP+TP} \tag{2}$$

$$P_P = \frac{FN+TP}{TN+FN+FP+TP} \cdot \frac{FP+TP}{TN+FN+FP+TP} \tag{3}$$

$$P_e = P_N + P_P \tag{4}$$

$$\kappa = \frac{P_0 - P_e}{1 - p_e} \tag{5}$$

Kappa is always having a less than or equal to 1. If the Kappa value is 1, it means the model fits perfectly, and lesser values imply not in the perfect agreement. Table 5 Altman (1991) shows the rule of thumb to evaluate kappa.

Table 5: Criteria for Kappa

Kappa	Strength of agreement
Less than 0.20	Poor agreement
0.20 to 0.40	Fair agreement
0.40 to 0.80	Moderate agreement
0.80 to 1.00	Very good agreement

2.2 Decision Tree

2.2.1 Introduction

A decision tree is useful for interpretation. It can be displayed graphically and is easily understood even by a non-expert. It also mirrors a physician's criteria to diagnosis diseases. It uses a greedy approach for recursive binary splitting; at each step of the tree-building process, the best split is chosen at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step. James et al. (2013) Therefore, patients can go through laboratory test according to the sequence of the nodes and can terminate sooner if they meet certain conditions.

Unfortunately, a simple decision tree may not perform as well as other Machine Learning algorithms. However, many decision trees using ensemble methods (random forests and boosting); this limitation can be redeemed and provide high prediction accuracy.

2.2.2 Algorithm

Table 6: Algorithm for decision tree

1. Use recursive binary splitting to grow a large tree on the training data.
   - (a) Select one of the inputs X_j $(j \in 1, ..., p)$ and a cut-point s. Partition the input space into two half-spaces, $\{\, X : X_j < s\}$ and $\{\, X : X_j > s\}$
   - (b) Repeat this splitting for each region until stopping criterion is meet.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k = 1, . . .,K:
   - (a) Repeat Steps 1 and two on all but the kth fold of the training data.
   - (b) Evaluate the error on the data in the left-out kth fold, as a function of α. Average the results for each value of α, and pick α to minimise the average error
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

When the dataset is with n patients with p variables, we can divide patients into J different spaces by using a decision tree. The predictor space is decided as the set of possible values for $X_1, X_2, ..., X_p$ into J distinct and non-overlapping regions, $R_1, R_2, ..., R_J$ that minimise the error. The splitting is proceeding in a top-down recursive binary splitting. It starts from the top of the tree where all the observations are in a single region and successively splits the regions into several sub-branches until the stopping criterion is met.James et al. (2013) It is called to be greedy because at each splitting; it picks the very best split point in each time.

For any j and s, the pair of half-planes are defined as

$$R_1(j, s) = \{X|X_j < s\} \text{ and } R_2(j, s) = \{X|X_j > s\}$$

There are no statistical significance test variables in the decision tree. Instead, there is Gini impurity which can compute if the variable is important when constructing the classification tree.

Gini impurity index defines by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where:

K $=$ The number of classes in the dependent variable.

$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$.

If the splitting continues until the stopping criterion is met, then the tree might over-fit on training data. This will result in a too-complicated model with many sub-branches, which can lead to poor test set performance. Then tree pruning is needed, which refers to cutting back the tree to be more simple with low variance.

Cost complexity pruning is a pruning method to select sub-tree that leads to the lowest cross-validation error. First, grow a huge tree T_0 with many sub-trees. The error is evaluated using a cost complexity function. The cost complexity function is defined as

$$\begin{aligned} R_\alpha(T) &= R(T) + \alpha \mid T \mid \\ &= \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) + \alpha \mid T \mid \end{aligned}$$

where:

$R(T)$ $=$ Learning error which corresponds to the Gini impurity index in this classification tree.

$\mid T \mid$ $=$ Number of nodes of the tree T.

α $=$ Tunning parameter that controls a trade-off between the sub-trees complexity and its fit to the training data.

By adding penalty term with tuning parameter α, it controls a trade-off between the sub-trees complexity and its fit to the training data. When $\alpha = 0$, then the sub-tree T will coincide with the first tree T_0. As α increases, the penalty for having a tree with many terminal nodes will consequently increase. By increasing α from zero to 1, branches get pruned from the initial tree. Therefore the best model with the right α can be found by using cross-validation, which minimises the cost complexity the most. An Introduction to Statistical Learning James et al. (2013)

A measure of variable importance is the sum of Gini impurity index for each split for which it was the primary variable, plus goodness for all splits in which it was a surrogate. Therneau and Atkinson (2015) This help to understand which variables are relevant to build the model.

$$ni_j = W_j C_j - W_{left(j)} C_{left(j)} - W_{right(j)} C_{right(j)}$$

where:

ni$_j$	= The importance of node j.
W$_j$	= Weigthed number of samples reaching node j.
C$_j$	= The impurity value of node j.
left(j)	= Child node from left split on node j.
right(j)	= Child node from right split on node j.

$$fi_i = \frac{\sum_{j:\ node\ j splits\ on\ variable\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

where:

fi$_i$	= The importance of variable i.
ni$_j$	= The importance of node j.

This can be normalized to a value between 0 and 1.

$$Norm(fi_i) = \frac{fi_i - min(fi)}{max(fi) - min(fi)}$$

2.3 Random Forest

2.3.1 Introduction

Bagging is an ensemble learning method which is the shortened term for' bootstrap aggregating'. It involves bootstrapping the train data into B different set, and for iteration, it is building different decision trees. Aggregating gives an output of the class that earned the most votes of the B number of trees. Since bagging is aggregating a number of trees, it can reduce variance and helps to avoid over-fitting. Whenever split is decided in bagging, like as the decision tree, every variable is on the consideration, even though it is using a bootstrapped sample, there is a high chance that all of the bagged trees might look similar to each other. In other words, the bagged trees will be highly correlated and averaging many highly correlated quantities does not lead to better performance.

Random forests is a substantial modification technique of bagging that builds a B number of de-correlated sample trees. Random forest builds a B number of decision trees on bootstrapped training samples, which is the same procedure with bagging. However, in Random forest model, when each split is made, a random sample of m (=$\sqrt{p}$) predictors is chosen instead of using the full set of p predictors. Hence, Random forest can give a reduction in variance as averaging many uncorrelated quantities. Bell (2015)

2.3.2 Algorithm

Table 7: Algorithm for Random Forest

```
1. For b = 1 to B:
   (a) Draw a bootstrap sample $Z^*$ of size $N$ from the training data
   (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for
       each terminal node of the tree, until the stopping criterion is met.
       i. Select $m(\approx \sqrt{p})$ variables at random from the $p$ variables
       ii. Pick the best variable/split-point among the m
       iii. Split the node into two daughter nodes
   (c) Output the ensemble of trees $\{T_b\}_1^B$
   (d) Prediction:
       i. Result for the prediction is $\hat{G}^*_{bag}(x) = argmax_k \hat{f}_{bag}(x)$
```

Given a set of n independent observations $Z = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$. Bagging is re-sampling training data by bootstraping. Then the boostrap sample can be define as $Z^{*b} = \{(x_1^*, y_1^*), (x_2^*, y_2^*), ..., (x_N^*, y_N^*)\}$, where $b = 1, 2, ..., B$. For every sample $Z^{*b}, b = 1, 2, ..B$, we can calculate build classification tree T_b.

Every tree in Bagging and Random Forest have grown a rooted tree without pruning. Hence, aggregating them in a single model will produce a low-variance statistical learning model. In Random Forest, Before each split, select $m(\approx \sqrt{p})$ of the input variables at random as candidates for splitting. This idea is to improve the variance reduction by reducing the correlation between the trees without increasing the variance too much. Hastie, Tibshirani, and Friedman (2017).

The prediction at input vector x is the majority votes from the B trees. It can be written as below.

$$\hat{G}^*_{bag}(x) = argmax_k \hat{f}_{bag}(x)$$

where:

$\hat{f}_{rf}(x) = K$ vector $[p_1(x), p_2(x), \ldots, p_k(x)]$.
$P_k(x)$ $=$ The proportion of trees prediction class k at x.

Variance Importance

The variance importance in Random forest follows the same procedure with the Decision tree. Importance in Random forest is mean of all tree's importance and normalising it. It can be written as below. Ronaghan (2018)

$$RFfi_i = \frac{\sum_{j \in all\ trees} fi_{ij}}{T}$$

where:

RFfi$_i$	$=$ The importance of variable calculated from all trees in the Random Forest model
fi$_{ij}$	$=$ the variable importance for i in tree j.
T	$=$ Total number of trees.

This can be normalized to a value between 0 and 1.

$$Norm(RFfi_i) = \frac{RFfi_i - min(RFfi)}{max(RFfi) - min(RFfi)}$$

$\boldsymbol{Out\ of\ Bag(OOB)}$

Random forest is the technique that trees are repeatedly fit to bootstrapped subsets of the observations. Hastie, Tibshirani, and Friedman (2017) In each bootstrap training set, about two-thirds of the samples are used to fit the tree. The remaining one-third of the observations are referred to as the out-of-bag observations. With out-of-bag observation, predict the response using the tree built with training samples. By averaging the result, we can compute an out-of-bag error. If the B is sufficiently large, an out-of-bag error can be equivalent to cross-validation error.

2.4 Boosting

2.4.1 Introduction

Boosting is an ensemble technique that new models are added to fix the errors made by existing models. Models are added recursively till no noticeable improvements can be detected. Sundaram (2018) Each tree in boosting has high bias, but by effectively combining these weak trees, it will produce a low bias and low variance result. In contrast to Random Forest, it will grow simpler trees with fewer splits.Chen and Guestrin (2016)

There are several boosting algorithms which are Adaboost, Gradient Boosting Machine(GBM), and Extreme Gradient Boosting(XGBoost). Adaboost is an algorithm that provides the advantages of more straightforward implementation, simpler variable selection, and reasonably good generalisation. However, they generate only sub-optimal solutions and are sensitive to outliers and noisy data. Kadiyala and Kumar (2018) gradient boosting is an algorithm in which new models are created that predict the residuals of prior models and then added together to make the final prediction. It uses a gradient descent algorithm to minimise loss function concerning the previous iteration's value.Ogunleye and Qing-Guo (2019)

XGBoost is an implementation technique of gradient boosting machines created by Tianqi Chen. The difference in modelling details is that Xgboost used a more regularised model formalisation to control over-fitting, which gives it better performance.Chen (2015) The XGBoost algorithm combines many week classifiers, and by combining them, it gets better performance. Each tree has a high bias with weak performance. It starts by building an initial tree with high bias, which has a poor performance by itself. Then it sequentially builds next tree which is trained to predict what the previous tree was not able to fitted well and is itself a weak learner too. It continues this procedure until stopping criteria met. Spark (2017)

XGBoost is well known technique that can provide better solutions than other machine learning algorithms. Also, since after it shows good results in several Kaggle's competition, it has become the popular machine learning algorithm to analyse with structured data. I chose to XGBoost for boosting algorithm to implemented in this study. There are several advantages of using XGBoost over other algorithms. It is having faster and gives comparatively giving better performance. Also, it has a parallelizable algorithm and outperforms other algorithm methods, as it has present better performance on a variety of machine learning with big datasets. Kadiyala and Kumar (2018) Therefore, `XGBoost` package in R is used for building boosting model. Also `Caret` package in R is used for tuning parameters.

2.4.2 Algorithm

For a given data set with n examples and M variables, $D = \{(x_i, y_i)\}$ where $|D| = n, x_i \in R^m, y_i \in R$. A tree ensemble model uses k addictive functions to predict the output.

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in F.$$

where F is the space of decision tree, and each f_x corresponds to an independent tree structure q and leaf weight w.

XGboost build model which is minimizing the following regularized objective

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

where:

$\sum_i \Omega(f_k)$	$=$ Regularization term which is $\sum_i \Omega(f_k) = \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2$ when w is the vector of scores on leaves
$\sum_i l(\hat{y}_i, y_i)$	$=$ Training loss which is $L(\phi) = \sum[y_i ln(1 + e^{-\hat{y_i}}) + (1 - y_i) ln(1 + e^{\hat{y_i}})]$, to be used for binary logistic regression in this case.
γ	$=$ Parameter which indicated the threshold for the gain.
λ	$=$ Parameter which indicated regularizaion term.

It is intractable to learn all the trees at once. Boosting is trying to fix the trees from the trees built in the previous steps. Therefore, it uses an addictive strategy to fix what trees have learned, and add one new tree at each time that can adjust the errors. Then the prediction value at step t as $\hat{y}_i^{(t)}$

$$\begin{aligned} \hat{y}_i^{(0)} &= 0 \\ \hat{y}_i^{(1)} = f_1(x_i) &= \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) &= \hat{y}_1^{(1)} + f_2(x_i) \\ &\vdots \\ \hat{y}_i^{(t)} = \sum_{t=1}^{t} f_k(x_i) &= \hat{y}_i^{(t-1)} + f_t(x_i) \end{aligned}$$

$\hat{y}_i^{(t)}$ is the prediction of the i-th instance at the t-th iteration. Then we need to add f_t to minimize the objective function $L^{(t)}$. As Equation 2.4.2 includes functions that are hard to optimised using traditional optimisation methods in Euclidean space, it needs to use a second-order approximation to be optimised.

$$\begin{aligned} L^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \\ &\approx \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \end{aligned}$$

where $\mathrm{g}_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

Also, we can remove the constant terms to obtain the following simplified objective at step t. Also Define $I_j = \{i \mid q(x_i) = j\}$ as the instance set of leaf j. Then we can also compute the optimal weight w_j^* of leaf j.

$$\begin{aligned} \widetilde{L}^{(t)} &= \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \\ &= \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{i=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i) + \lambda) w_j^2] + \gamma T \end{aligned}$$

For each leaf j, to find optimal weight, compute $\frac{\partial L^{(t)}}{\partial w_j} = 0$. Then,

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

By substituting weights into the objective function, we can calculate the corresponding optimal value.

$$\begin{aligned} \widetilde{L}^{(t)}(q) &= -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \\ &= -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \end{aligned}$$

where:

$\mathrm{G}_j = (\sum_{i \in I_j} g_i)^2$
$\mathrm{H}_j = (\sum_{i \in I_j} h_i)^2$

XGBoost does not explore all possible tree structures but builds a tree greedily. A greedy algorithm that starts from a single leaf and iteratively adds branches to the trees is used. Assume that L and R are the instance sets of left and right nodes after the split, and $I = L \bigcup R$, then the loss reduction after the split is given by

$$L_{split} = \frac{1}{2} [\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \lambda$$

This formula can be decomposed as 1) the score on the new left leaf, 2) the score on the new right leaf, 3)the score on the first leaf, and 4) regularisation on the additional leaf. Similar to pruning, if the gain is smaller than γ, the new branch will not be added.

This formula is also used in practice for evaluating the split candidates. Therefore, the variance importance is the average gain (L_{split}) across all split the variable is used in. The concept is as follows: before adding a new split on a feature X to the branch, there were some wrongly classified elements; after adding the split on this feature, there are two new branches, and each of these branches is more accurate.

3 Data

3.1 Data Source

Data from the MIMIC III database has been used to analyse the diagnosis of diabetes. It is an open-access research database that covers patients with hospital admissions admitted to medical and surgical care at the Beth Israel Deaconess Medical Center in Boston, Massachusetts, between 2001 and 2012.

The data collection project was approved by the Institutional Review Boards of Beth Israel Deaconess Medical Center(Boston, MA) and the Massachusetts Institute of Technology(Cambridge, MA). The data includes vital signs, medications, laboratory measurements, observations and notes charted by care providers, fluid balance, procedure codes, diagnostic codes(ICD-9-CM), imaging reports, hospital length of stay, and survival data.

3.2 Data process

3.2.1 Data imputation

Training a model with a dataset that contain a lot of missing values (NA) can impact the quality of a statistical model. Missing value can be categorised as three different types: missing completely at random, missing at random, and informative missingness.

Missing completely at when the data is completely independent of observable variables and parameters of interest. Missing at random is when the probability of being missing depends on the value of another, non-missing, attribute. For example, males are less likely to fill in a depression survey, however this is related with their level of depression, after accounting for maleness.Obadia (2017) On the other hand, the probability of being missing depends on the actual value of the attribute itself. Little (2019) For example, people tended not to answer the survey depending on their depression level.

There are generally several imputation methods to replace NA with some other values. For instance, NA can be replaced with 0, mean, or median values. Also, regression imputation, which refers to predicting the NA by regressing it from other related variables in the same dataset, is another solution.

In this data set, NA is informative missingness. As the patients do not need unnecessary lab tests, NA means the patients' lab test. Therefore, all the NA in this paper is replaced with 0. As the NA implies some information, the NA is kept as a distinct number so that classifier can detect the difference between the real values and NA, which is encoded as 0.

3.2.2 One hot encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to machine learning algorithms to do a better job in prediction. It is also the same as full dummy coding for a categorical variable which converts one variable consists of a n number of categories into a n number of variables that can be represented using dichotomous columns.

For example, $ethnicity$ is encoded as 1 for 'Indian Alaska Native' and 2 for 'Asian Indian'. However, it does not mean that Indian Alaska is having a higher value than Asian Indian. One hot encoding is needed to eliminate the ordinality from a categorical variable. As shown in Table 10, there are two categorical variables, and they will be converted into numerical variables with One-hot encoding.

Table 8: Before and After One Hot Encoding

Ethnicity
1 (Indian/Alaska)
2 (Asian Indian)
3 (Cambodian)
4 (Chinese)
$\vdots$

Indian/Alaska	Asian Indian	Cambodian	Chinese	$\cdots$
1	0	0	0	$\cdots$
0	1	0	0	$\cdots$
0	0	1	0	$\cdots$
0	0	0	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

4 Results

4.1 Study population

I treated patients with a different visit as distinct patients. For example, if patients A visit two time and had a diagnosis, then it was counted as two distinct patients. According to Table 9, There are 12,933(21.411%)patients having diabetes mellitis type 2 Diabetes. There are 616 laboratory measurements in the MIMIC-III database. However, as 134 variables are null for type 2 patients, only 482 variables will be used.

Table 9: Number of diabetes patients

	obs	percentage
Diabetes	$12,933$	21.441
Not Diabetes	$47,385$	78.559
Total	$60,318$	100

Data can be categorised into two parts: 1) Demographic data and 2) Laboratory measurements. Demographic data is consists of patients' information, such as sex, age, ethnicity, and BMI. On the other hand, laboratory measurements are showing patients' medical test result such as glucose, haemoglobin A1c, white blood cells.

Table 10: Categorical Variables in the dataset

Variable	Complete perc.	Unique Factors	Top Counts
ETHNICITY	1	41	WHI: 41979, BLA: 5538, UNK: 4654, HIS: 1721
GENDER	1	2	M: 33755, F: 26563

Table 10 shows all patients' information about categorical data which are assigned a certain unit of observation to a particular category based on some qualitative property. Two variables which are $Ethinicity$ and $Gender$ is categorical data which assign a certain unit of observation to a particular category based on some qualitative property. There are 41 categories for $Ethnicity$ and most common ethnicity is 'White'. No observations are missing for all categorical data. However, 4654 patients are having ethnicity "Unknown", which is almost the same thing as a missing value.

Table 11: Numerical Variables in the dataset

variable	ICD-9-CM	Complete perc.	mean	sd	p0	p25	p50	p75	p100
Glucose	2345-7	87.112	131.22	39.62	3	107.68	121.6	143	798
Age		100.000	51.23	27.97	0	38	59	73	89
% Hemoglobin A1c	4548-4	11.544	6.61	1.76	3.8	5.6	6	6.9	22
Glucose	2339-0	47.009	147.23	65.99	12	117	133.4	156.75	1359
Glucose	5792-7	10.115	410.73	363.14	70	100	250	750	1000
Urea Nitrogen	3094-0	88.130	25.45	18.39	1.42	13.5	19.5	31.26	232.04
Magnesium	2601-3	86.062	2.02	0.26	0.4	1.87	2	2.15	8.98
INR(PT)	5895-7	83.514	1.44	0.77	0	1.1	1.24	1.49	47.65
PT	5902-2	83.509	15.33	4.97	8.57	12.97	13.93	15.72	150
Osmolality, Measured	2692-2	8.951	296.59	24.67	203	280.58	295	309.28	434.5
Creatinine	2160-0	88.110	1.39	1.68	0	0.72	0.94	1.38	208.55
Potassium	2823-3	91.236	4.16	0.46	1.6	3.87	4.1	4.36	9
MCH	785-6	97.986	30.85	2.98	0	29.22	30.61	32.21	45.01
BMI		47.770	40.85	1295.67	0	21.65	26.14	30.75	216828.68
RDW	788-0	97.972	15.4	2.11	0	13.8	15	16.62	31.95
White Blood Cells	804-5	98.072	11.44	7.61	0.1	8	10.4	13.44	467.67
Potassium	2823-3	91.236	4.16	0.46	1.6	3.87	4.1	4.36	9
				⋮					

Table11 shows statistics information about numerical data. Complete perc. Indicates the percentage of complete data – that is data which is not null. For example, 87.112% of data is complete for the Glucose variable, which is encoded as 2345-7 in ICD-9-CM. However, Hemoglobin A1c has a lot of missing values; only 11.544% of the data is complete. The p columns show their respective quantile values. For example, a minimum value of age is 0, and approximately 1/4 of the ascending orders of values of the whole age is 38, and the oldest age is 89.

4.2 Desicion Tree

`rpart` package in R is used for building decision tree models.

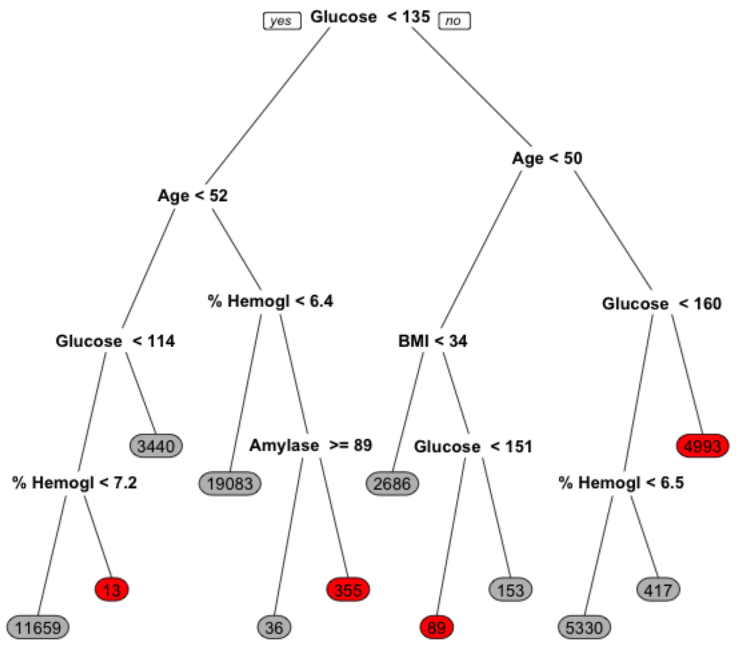

Figure 1: Not pruned tree

Figure 1 shows the result for the decision tree. A node, which is divided into sub-nodes, is named a parent node of sub-nodes where a sub-nodes are the child of the parent node. For in this case, There are 11 nodes included roots nodes, which is $Glucoes$. In the Left split of the root node, Age is root nodes for this sub-trees and $glucose$ and $Hemoblobin$ are the sub-nodes. The tree's root node can be interpreted like "Is the glucose level under 135(mg/dl)?". If the patients have under 135(mg/dl) glucose level, then it should follow the left-node.

On this step, tree grows with 11 splits with using 5 different variables which is $Glucose(mg/dL)$, Age, $Hemoglobin(\%)$, BMI, and $Amylase$. This result seems to be similar to the clinical guidelines from Table 1. However, to improve this model to prevent over-fitting, the tree needs to be checked if pruning can give a better result.

Table 12

CP	nsplit	rel error	xerror	xstd
0.039	0	1	1	0.009
0.024	3	0.882	0.892	0.008
0.010	4	0.858	0.866	0.008
0.002	6	0.838	0.846	0.008
0.001	8	0.833	0.845	0.008
0.0002	9	0.833	0.845	0.008
0	11	0.832	0.846	0.008

n= 48254, Root node error: 10355/48254 = 0.21459

Table 12 shows the information to help prune the tree. cp is corresponding with α mentioned in the Background part. Rel error(relative error) is $1 - R^2$ root mean square error, which is the error for predictions of the data that were used to estimate the model. Also, the x-error is the cross-validation error, and node error is the percent of correctly sorted records at the first root splitting node. Sydney (2019)

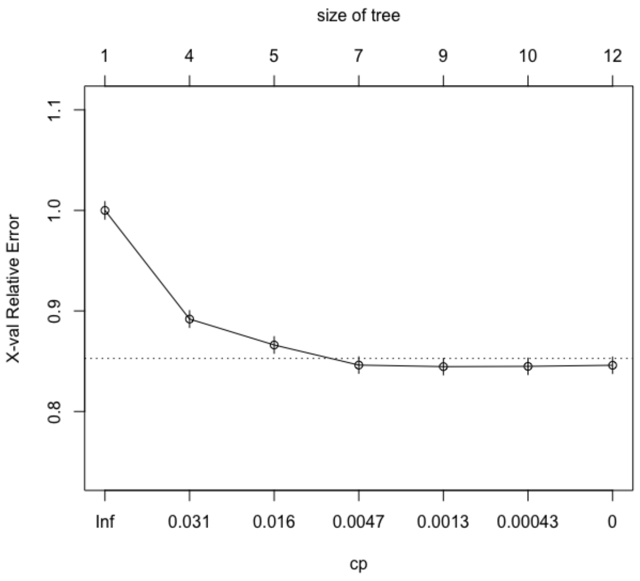

Figure 2: cp plot for Desicion Tree

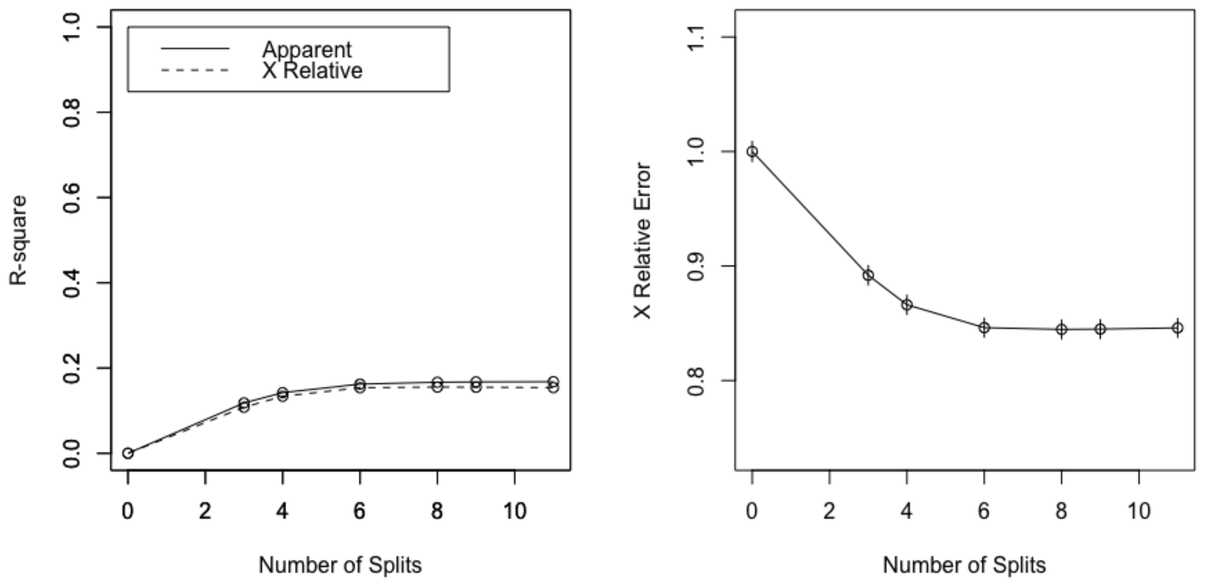

Figure 3: Split

According to Figure 2, when cp is larger, the model shows better performance with lower relative error. This means making a tree more complex without pruning is showing better performance. Figure 3 also shows the trade-off between model complexity and performance. When cp increase, the tree will grow deeper with more splits, then the R^2 will increase with lower relative error.

Tree is pruned by using cost complexity criterion. The horizontal line in Figure 2 shows the smallest cross validation error which is 0.845 when $cp = 0.001$ with 8 splits. Therefore, the final decision tree model is the model pruned with $cp = 0.001$. The pruned tree with $cp = 0.001$. has a misclassification rate of 0.21459 ·0.83332 $\cdot$ 100% = 17.889% (i.e. 82.121% of prediction accuracy) in training data. Also the tree has a misclassification rate of 0.21459 ·0.84462 $\cdot$ 100% = 18.121% (i.e. 81.875% of prediction accuracy) in cross-validation.

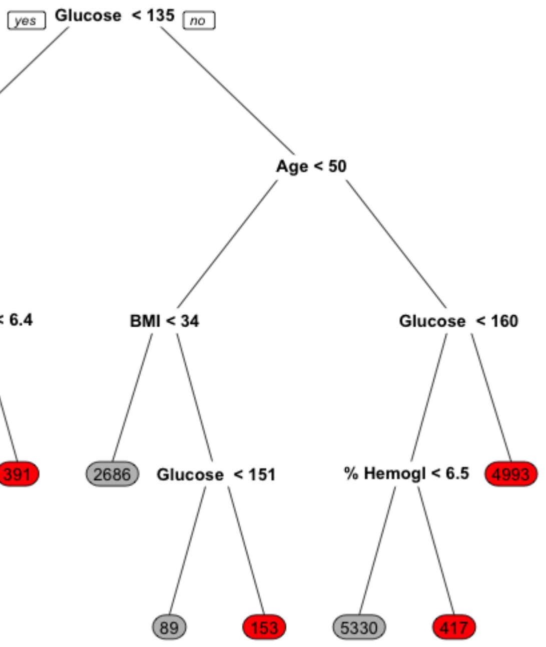

Figure 4: Pruned tree

According to Figure 4, the final pruned tree grows with 8 splits with using 4 different variables which is $Glucose(mg/dL)$, Age, $Hemoglobin(\%)$, and BMI. By comparing with the previous tree, we can see that sub-tree under $Age < 52$ and $Hemoglobin(\%) < 6.5$ has been pruned. By comparing with the clinical guideline from Table 1, The result seems similar to the actual doctor's diagnosis procedure. $Glucose$ and $Hemoglobin$ is a necessary lab test to tell if the person has diabetes or not. One difference part that makes hard to compare with clinical guideline and this analysis is that our data does not have information about if the glucose is with fasting or not. But the only information the dataset has is that this glucose is coded as 2345-7 in LOINC code which is Glucose [Mass/volume] in Serum or Plasma. But assuming that root node is split when $glucose < 135$, this could be $Glucose$ with fasting plasma instead of $Glucose$ without fasting.

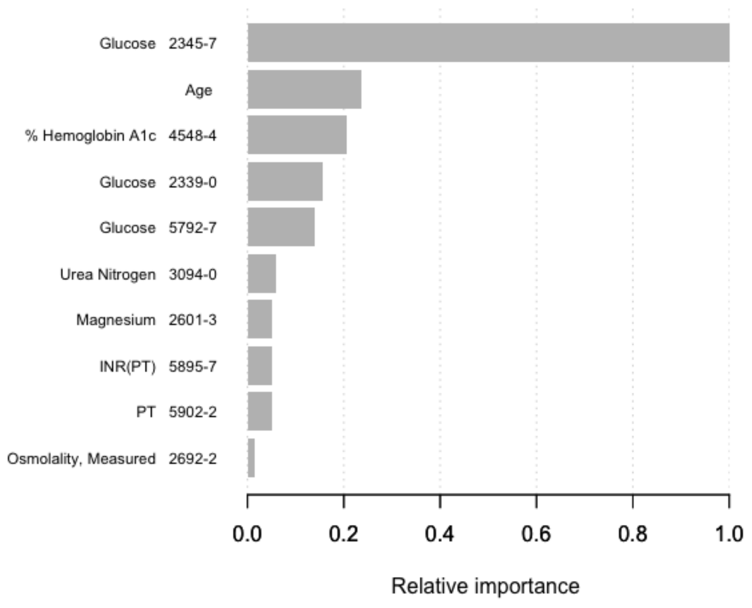

Figure 5: Importance for Decision tree

Figure 5 shows the relative importance for the variables.

Even if some of the variables have high importance, the classification error rate will not be reduced, splitting with those variables. Decision trees make no assumptions on relationships between variables. For in this case, we can see $Glucose$ 23390 is not in the slit point but included as a top 10 variables for the variance importance. On the other hand, BMI was used for the split point but was not included as a top 10 variables for the variance importance

This is how the decision tree is overcoming multicollinearity. As Decision tree is based on the greedy algorithm, if variable A, B are heavily correlated, no information can be gained from splitting on B after having split on A. So it would choose to split with C instead of B even if B has higher importance value. Jonner (2018) Therefore, There are several variables that have higher importance than BMI, but the split is made on BMI instead. This is because $Glucose$ 23390,for instance, is highly correlated with $Glucose$ 23457, so there will be not much information gain by slitting with $Glucose$ 23390 again.

Table 13: Confusion matrix for Decision tree

	Ref:0	Ref:1
Pred:0	8930	1623
Pred:1	556	955

Table 14: Accuracy for Decision tree

Statistics	Test Data
Accuracy	0.8194
Kappa	0.3672
Sensitivity	0.63203
Specificity	0.84620
Pos Pred Value	0.37044
Neg Pred Value	0.94139
Prevalence	0.12525
Detection Rate	0.07916
Detection Prevalence	0.21369
Balanced Accuracy	0.73912

With the pruned final Decision tree, we will predict test data and check if the model can be generalised. According to Table 13, the test data prediction accuracy is 81.938%. According to the Table 14, $Kappa$ is 0.3672, which is about fir agreement according to Table5.

4.3 Random Forest

First of all, `randomforest` package in R is used for building decision tree models.

Hyper-parameters tuning refers to the settings of an algorithm that can be adjusted to optimise performance. While model parameters, as the slope and intercept in linear regression, are learned during training, hyper-parameters must be set by the statistician before training.Koehrsen (2018) The choice of hyper-parameter has a significant role to play in the model's performance. However, best hyper-parameters are usually impossible to determine ahead of time. It rather relies more on experimental results than theory. Therefore, the best method to determine the optimal settings is to try many different combinations and evaluate the performance of each model. Koehrsen (2018)

In Random forest, hyper-parameters include $ntree$, which is the number of decision trees in the model and $mtry$, which is the number of variables considered by each tree when splitting a node. In this study, I will fix the $ntree$ as 300 to have reasonable time to compute by computer and tune only $mtry$. Before each split, select m number of $mtry$ for the input variables at random as candidates for splitting. The rule of thumb for the m is $\sqrt{p}$. However, I have tried from 17 to 26 to check if there is a difference when $mtry$ value is changing.

Table 15: Tuning $mtry$

$mtry$	Accuracy	Kappa
17	0.8299830	0.3249754
18	0.8325942	0.3434220
19	0.8323662	0.3444441
20	0.8350603	0.3586585
21	0.8349567	0.3612232
22	0.8350603	0.3627620
23	0.8375471	0.3795159
24	0.8379409	0.3833879
25	0.8388320	0.3886770
26	0.8388320	0.3889512

According to Leo Breiman and Adele Cutler, the founders of random forests, "There is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error." Breiman (2001). Therefore, all the accuracy in the Random forest is with Out-of-bag instead of cross-validation.

According to Table15, it shows the out of bag accuracy depending on the $mtry$. The default value for $mtry$ in R package `RandomForest` is $\sqrt(p)$ which is 22.02272 in this case. however, Table15 show that accuracy grows if $mtry$ increase from 17 to 25. However, the accuracy stop increase when the $mtry$ become 25 and stay the same when it increases to 26. Therefore, The final hyper-parameter tuned by this trial is $mtry$ with 25, and $ntree$ with 300.

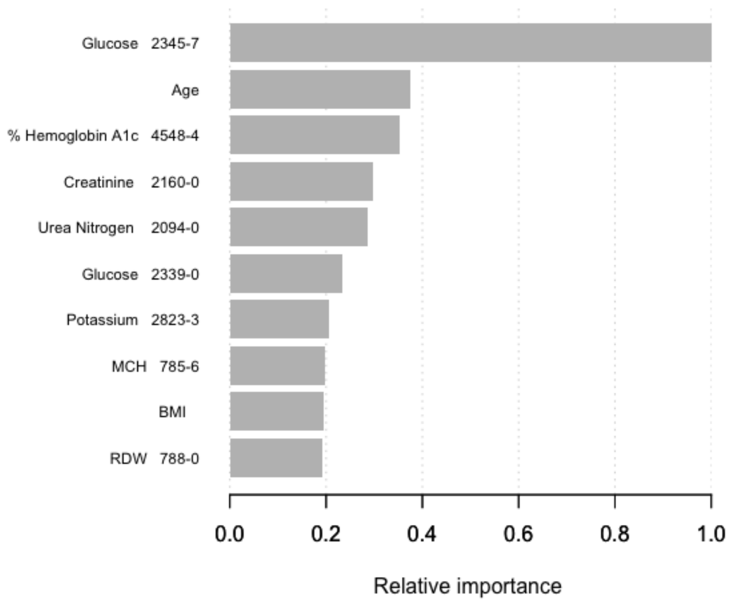

Figure 6: Importance for Random forest

Figure 6 shows the relative importance for the variables. It seems like the most important variables are similar to the decision tree from Figure5.

Table 16: Confusion matrix for Random forest

	Ref:0	Ref:1
Pred:0	9288	1711
Pred:1	198	867

Table 17: Accuracy for Random forest

Statistics	Test Data
Accuracy	0.8418
Kappa	0.4012
Sensitivity	0.33631
Specificity	0.97913
Pos Pred Value	0.81408
Neg Pred Value	0.84444
Prevalence	0.21369
Detection Rate	0.07187
Detection Prevalence	0.08828
Balanced Accuracy	0.65772

Using the testing set, we can check if the prediction of out of bag estimate of error can be confirmed, and see if the model can be generalised.

Table 16 shows that confusion matrix with test data predicted with random forest. The test accuracy for the prediction is 84.07659%.

Table 18: Validation accuracy for Random forest

	Train Data	Validation Data	Test Data
Random Forest	99.98%	83.88%	84.18%

Table 18 shows the 3 type of different accuracy. It shows that train data has 99.98 % accuracy, but the out-of-bag validation accuracy and Test data accuracy shows more than 10 % lower than the training accuracy. We can conclude that Random forest over-fitted on train data. However, as there is not so much difference between out-of-bag accuracy and test accuracy, we can say that out-of-bag accuracy is reliable to use as a validation method.

4.4 Boosting

XGBoost is providing the advantages of higher execution speeds, better model performance, enabling parallelised computations, cache optimisation, and out-of-core computing for huge databases. However, to achieve optimal performance, the model has to be tuned carefully. Tuning XGBoost is a task that needs a lot of trial because of the number of hyperparameters it has. Ogunleye and Qing-Guo (2019) XGBoost parameters can be divided into three categories. 1)General Parameters: Controls the booster type in the model which eventually drives overall functioning. 2) Booster Parameters: Controls the performance of the selected booster. 3) Learning Task Parameters: Sets and evaluates the learning process of the booster from the given data Chen and Guestrin (2016) For general parameters, I set the booster type as $gbtree$. For the Learning Task Parameters, I set objective as $binary : logistic$ to build the model with binary classification. For tuning booster Parameters, it has around 13 different, and it is quite tricky to find the best values for each parameter. So I try to tune four different parameters which can affect the most for the model.

First of all, I will start tuning hyper-parameter with the default values which is $eta = 0.3$, $minimum\, hild\, weight = 1$, $gamma = 0$, and $max\ depth = 6$. Furthermore, adjust each hyper-parameter step by step with trying different values.

Eta is the learning rate, which is the step size shrinkage used in the update to prevent over-fitting. After each boosting step, we can get the weights of new variables, and eta shrinks the weights of the variables to make the boosting process more conservative "Release 0.90 xgboost developers" (2019) For example, if one step is made at eta = 1.00, the step weight is 1.00. If the step is made at eta = 0.25, the step weight is 0.25.

To get the most of XGBoost, the learning rate eta should be set as low as possible with a large number of trees. If eta increases, computation become faster, but it might not be able to reach the best optimum. On the other hand, if eta decreases, computation becomes slower, but it will make easier reaching the best optimum. Laurae (n.d.)

To get a reasonable running time while testing hyper-parameter, I set the maximum number of trees as 300. As a maximum amount of the tree is restricted, a lower learning rate eta will slowly converge to the optimum.

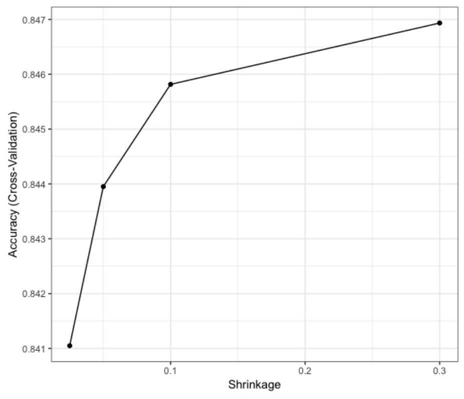

Figure 7: Shrinkage for XGBoost

Table 19: Tuning eta

eta	Accuracy	Kappa
0.025	0.8410494	0.4530215
0.050	0.8439507	0.4726837
0.100	0.8458157	0.4846230
0.300	0.8469347	0.4991396

Figure 19 shows the 10 fold cross-validation accuracy when eta is in different values. It shows that higher eta shows better performance. This is understandable as the number of trees is restricted as 300, smaller eta compute slowly and cannot reach the optimal if the number of trees is not enough. Therefore, the best eta is 0.3 when $ntree$ is 300.

The next parameter to controls is the $min\ child\ weight$. It is the minimum sum of instance weight needed in a child node. If the tree partition step reach a leaf node with the sum of instance weight less than $min\ child\ weight$, then the building process will stop further partitioning and making trees smaller. "Release 0.90 xgboost developers" (2019)

With the results from previous tuning step with $eta = 0.3$, will try to find the optimal value for the $min\ child\ weight$ by setting it as 1, 2, 3 when the default is 1.

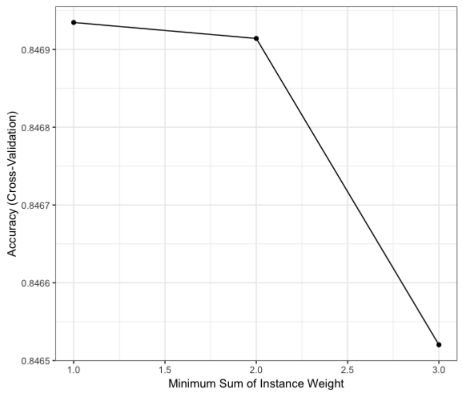

Figure 8: $minimum\ child\ weight$ for XGBoost

Table 20: Tuning $minimum\ child\ weight$

$minimum\ child\ weight$	Accuracy	Kappa
1	0.8465618	0.4995619
2	0.8455255	0.4960586
3	0.8445516	0.4920074

Figure 21 shows the 10 fold cross-validation accuracy when $min\ child\ weight$ is in different values. It shows that lower $min\ child\ weight$ have better performance. This is understandable as larger the $min\ child\ weight$, the more conservative the algorithm will be.

According to table **??**hild weighttab:min child weight results shows that the default value(1) shows the best performance. Therefore, the optimal parameter values so far is $eta = 0.3$ and $min\ child\ weight = 1$.

$gamma$ is the min split loss, which is the minimum loss reduction required to make a further partition on a leaf node of the tree. The larger $gamma$ is, the higher the regularisation and the more conservative the algorithm will be. "Release 0.90 xgboost developers" (2019) Both $gamma$ and $min\ child\ weight$ are methods to prune the trees. $min\ child\ weight$ is the controller to force pruning using derivatives. On the other hand, $gamma$ is controller to force pruning of the pure weights. Laurae (2016)

With the best values from the previous step, will test whether changing the gamma has any effect on the model's prediction accuracy. The default value is 0, with no regularisation. Starting with value 0; I will try to increase the value up to 1 and check how 10 fold cross-validation accuracy will be affected.

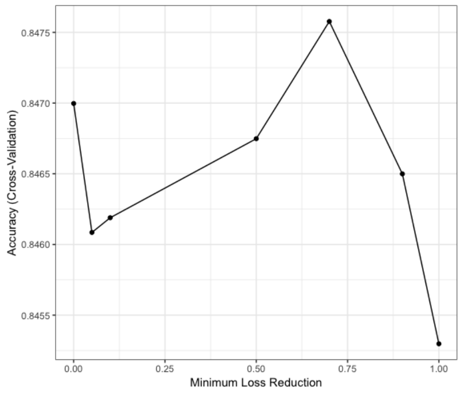

Figure 9: Minimum Loss Reduction for XGBoost

Table 21: Tuning $gamma$

$gamma$	Accuracy	Kappa
0.00	0.8469971	0.4992787
0.05	0.8460851	0.4965060
0.10	0.8461890	0.4958556
0.50	0.8467481	0.5000168
0.70	0.8475771	0.5018122
0.90	0.8464993	0.4995402
1.00	0.8452975	0.4944811

According to table**??**, the results shows that the 0.7 shows the best performance. Therefore, having regulation for the right amount can give better result for the prediction. Therefore, final values for the XGBoost model is parameter with eta = 0.3, $min\ child\ weight$ = 1 , and $gamma$ = 0.7.

For the last step, I tried tuning hyper-parameter $max\ depth$. In Random Forest, Tree grows without tuning so it can have deep depth. However, in XGBoost, it is a rather better choice to tuned trees so that it will not overfit.

$max\ depth$ is the value restricting the depth of a tree. More deep-rooted trees have more terminal nodes and fit more data. Increasing $max\ depth$ will make the model more complex and more likely to over-fit. Convergence also requires fewer trees if we grow them deeply. However, as Boosting is building trees subsequently, if the trees are too deep, then the first tree will use too much information and final trees would support the loss function. The Boosting benefits from using information from many trees. Therefore, it is intuitive that huge trees are not desirable. Vasconcelos (2018)

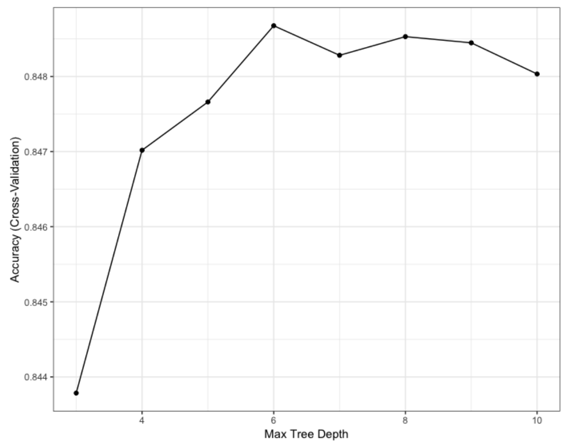

Figure 10: $max\ depth$ for XGBoost

Table 22: Tuning $max\ depth$

$max\ depth$	Accuracy	Kappa
3	0.8437852	0.4840687
4	0.8470181	0.4995679
5	0.8476606	0.5030520
6	0.8486759	0.5081955
7	0.8482819	0.5055336
8	0.8485308	0.5052345
9	0.8484481	0.5040177
10	0.8480334	0.5021124

According to Figure 22, cross-validation accuracy seems to increase until six and decrease again after six. This can happen because before six, the model is too simple to have an excellent performance. However, after six, it over-fits on the training data and shows bad accuracy with cross-validation.

Hence, the final values used for the model with $nrounds$ = 300 is eta = 0.3, $min\ child\ weight$ = 1, $gamma$ = 0.7, and $max\ depth$ = 6. And this model has a test accuracy around 84.823% and Kappa 0.508 which is moderate agreement according to Table 5.

Table 23: Final parameters for XGBoost

	Range	Default	Comparison	Best result
eta	[0,1]	0.3	0.025, 0.025, 0.1, 0.3	0.3
$mimimum\ child\ weight$	[0,∞]	1	1, 2, 3	1
$gamma$	[0,∞]	0	0, 0.05, 0.1, 0.5, 0.7, 0.9, 1	0.7
$max\ depth$	[0,∞]	6	3, 4, 5, 6, 7, 8, 9, 10	6

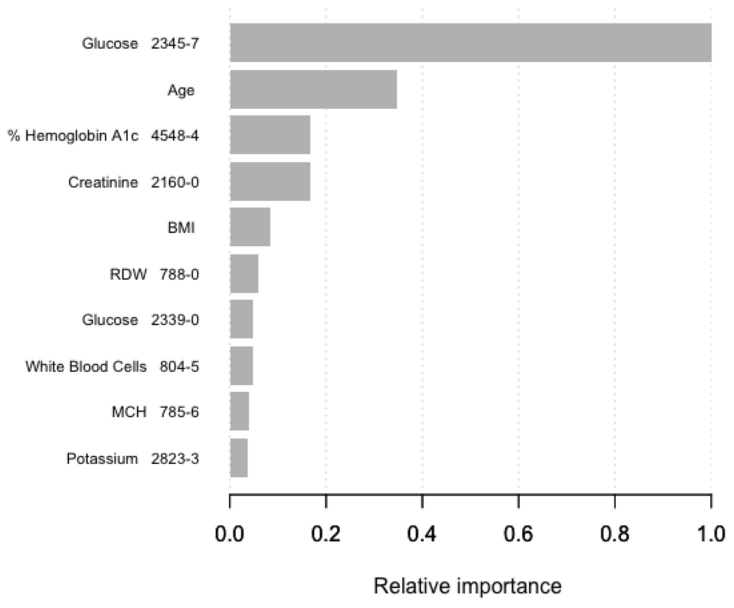

Figure 11: Importance for XGBoost

According to Figure 11, it shows the top 10 variables according to the importance, which is the gain that gives the most improvement in accuracy brought by a variable to the branches it is on. By comparing with Table 1, the result seems to be similar to the actual doctor's consideration. $Glucose$ and $Hemoglobin$ accounts for two of the most critical factors.

Table 24: Confusion matrix for XGBoost

	Ref:0	Ref:1
Pred:0	8875	1248
Pred:1	611	1330

Table 25: Accuracy for XGBoost

Statistics	Test Data
Accuracy	0.8459
Kappa	0.4961
Sensitivity	0.5159
Specificity	0.9356
Pos Pred Value	0.6852
Neg Pred Value	0.8767
Prevalence	0.2137
Detection Rate	0.1102
Detection Prevalence	0.1609
Balanced Accuracy	0.7257

Table 24 shows that confusion matrix with test data predicted with random forest. The test accuracy for the prediction is 84.59%. Also, we can check that false negative is higher than false positive, which means there is a higher risk of prediction patients as having diabetes even though it is not.

5 Discussion

This paper aimed to build a predictive model for type 2 diabetes using patients' demographic factors and laboratory data. Using tree-based models, which are decision tree, random forest, and boosting, around 85 % of patients can be correctly diagnosed with a statistical approach.

If the model is built on the training data, then the model will give very well result on the training set, but will not be able to generalise to new data, such as in a test set. When a model performs highly on the training set but poorly on the test set, this is called over-fitting. An over-fit model looks well fitted on the training set but will generalise poorly to other datasets and is useless in a real application. Koehrsen (2018)

Over-fitting can be screened through cross-validation and tuning parameters. If the training accuracy and cross-validation accuracy are differing too much, it means the model is over-fitted. Furthermore, it can be controlled with parameter-tuning in Random Forest and Boosting. For example, the model grows too many trees without pruning or restriction due to the loss threshold not reached.

According to Table 26, decision tree seems to be correctly fitted, as the accuracy across train data and test data clearly do not differ. However, and boosting seem an over-fitting problem. Therefore, tuning more parameters with different grids are required for future study development.

Table 26: Overall Accuracy

	Train Data	Cross Validation	Test Data
Decision Tree	82.121%	81.875%	81.938%
Random Forest	99.98%	83.88%	84.18%
Boosting	98.2571 %	84.87%	84.5905%

A statistical diagnosis based on tree based model is presented in this paper. Applying the method in diagnosing type 2 diabetes, it shows that the decision tree model is quick, and the diagnosis accuracy is not much lower compared to two other methods. Random forest improved performance up to 84% by combining several trees than a decision tree. However, in terms of accuracy, Boosting shows the best result in this paper. Boosting require a lot of time for hyper-parameter tuning, but it can give higher performance than other methods. However, considering the decision tree and random forest does not require too much of adjustment by the analyst, it is also impressive that they can give almost as good accuracy as XGBoost.

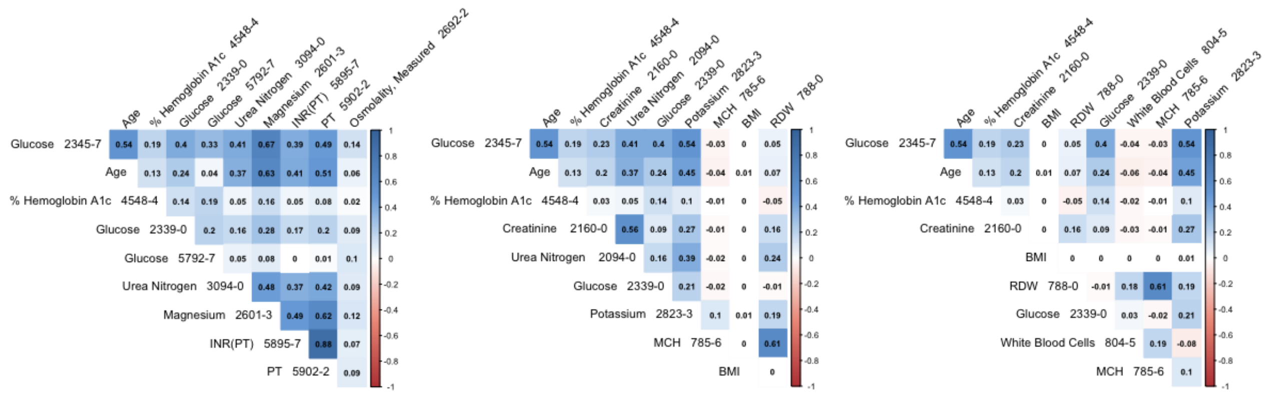

Figure 12: Correlation between variables

It was interesting how multicollinearity can be solved in the tree-based model. The multicollinearity between variables can be found by looking at the correlation between variables that count for high importance in each model. Figure 12 shows the correlation between top 10 importance variables for the Decision tree, Random forest, and XGBoost consequently. Decision tree calculates variance importance based on the sum of the goodness of split measures from each split for which it was the primary variable. However, decision trees make no assumptions on relationships between variables. Assuming there are two correlated variables which are variable A and variable B, the variance importance will still show that A and B account for the high importance. However, the split will be made on only one of them because there will be no information can be obtained from splitting on B after having split on A. This was an interesting point of the tree-based model because other models such as Logistic regression would use both the variables and couldn't solve the multicollinearity problem.

In Random forest, it variables with high importance values are not highly correlated. This is because when each split is decided, the random forest used only m predictors as a consideration. If the split is decided with every variable on the consideration like as bagging, there is a high chance that all of the ensembles trees might look similar to each other. On the other hand, Random forest builds a B number of decision trees on bootstrapped training samples with a random sample of m predictors instead of using the full set of p predictors. Hence, Random forest can have more trial to see which variable enhance performance more within A and B. Therefore, if A and B are completely correlated, approximately 50% of the trees will choose variable A, and the other 50% will choose variable B. So the importance of the information contained in A and B. **Tianqi10**

Furthermore, in boosting, a tree is growing sequentially; when a specific link between variable and outcome have been learned by the algorithm, it will try to not refocus on the correlated variable. Therefore, all the importance and split will be on either variable A or variable B but not both. Furthermore, it will not bother the modelling because the boosting algorithm will focus more on reducing the bias that has not been solved from the previous tree. "Release 0.90 xgboost developers" (2019)

The limitation of this study is that all the dataset is from a hospital instead of a primary care clinic. Also, it is from a single location, so it can be hard to generalise in a different context. In the analyse part, I set the number of trees in 300 and tried tuning parameter sequentially. However, trying tuning with grids in different combinations of several parameters can give better performance. Potential future research that can come from this research is to generalise the topic to other disease using these tree-based methods.

Appendices

.1 ICD-9-CM encoding

The diagnostic codes are based on ICD-9-CM, which is the International Classification of Diseases developed by the World Health Organization in 1975. It is the official system of assigning codes to diagnoses and procedures associated with hospital utilisation in the United States. Each code corresponds to a single diagnostic result.

Table 27 shows the ICD-9-CM codes which correspond with T2D. For example, patients who diagnosis as 250.50 has diabetes with ophthalmic manifestations.

Table 27: ICD-9-CM code for diabetes Health Statistics (2015)

ICD-9-CM	Description
250.00	Diabetes mellitus without mention of complication, type II or unspecified type, not stated as uncontrolled
250.02	Diabetes mellitus without mention of complication, type II or unspecified type, uncontrolled
250.10	Diabetes with ketoacidosis, type II or unspecified type, not stated as uncontrolled
250.12	Diabetes with ketoacidosis, type II or unspecified type, uncontrolled
250.20	Diabetes with hyperosmolarity, type II or unspecified type, not stated as uncontrolled
250.22	Diabetes with hyperosmolarity, type II or unspecified type, uncontrolled
250.30	Diabetes with other coma, type II or unspecified type, not stated as uncontrolled
250.32	Diabetes with other coma, type II or unspecified type, uncontrolled
250.40	Diabetes with renal manifestations, type II or unspecified type, not stated as uncontrolled
250.42	Diabetes with renal manifestations, type II or unspecified type, uncontrolled
250.50	Diabetes with ophthalmic manifestations, type II or unspecified type, not stated as uncontrolled
250.52	Diabetes with ophthalmic manifestations, type II or unspecified type, uncontrolled
250.60	Diabetes with neurological manifestations, type II or unspecified type, not stated as uncontrolled
250.62	Diabetes with neurological manifestations, type II or unspecified type, uncontrolled
250.70	Diabetes with peripheral circulatory disorders, type II or unspecified type, not stated as uncontrolled
250.72	Diabetes with peripheral circulatory disorders, type II or unspecified type, uncontrolled
250.80	Diabetes with other specified manifestations, type II or unspecified type, not stated as uncontrolled
250.82	Diabetes with other specified manifestations, type II or unspecified type, uncontrolled
250.90	Diabetes with unspecified complication, type II or unspecified type, not stated as uncontrolled
250.92	Diabetes with unspecified complication, type II or unspecified type, uncontrolled

.2 LOINC

LOINC is universal code names and identifiers to medical terminology related to electronic health records. The purpose of this is to assist in the electronic exchange and gathering of clinical outcomes such as laboratory tests, clinical observations, outcomes management and research. Loinc (2019)

For example, 2345-7 stands for Glucose [Mass/volume] in Serum or Plasma, and 1558-6 stands for Fasting glucose [Mass/volume] in Serum or Plasma.

References

Altman, Douglas G. (1991). "Practical statistics for medical research / Douglas G. Altman". In: *London : Chapman and Hall, 1991*.

Association, American Diabetes (2011). *Age, Race, Gender Family History*. http://www.diabetes.org/are-you-at-risk/lower-your-risk/nonmodifiables.html.

Bell, Jason (2015). *Machine Learning with R*, pp. 315–348. ISBN: 9789811068072. DOI: 10.1002/9781119183464.ch12.

Breiman, Leo (2001). "RANDOM FORESTS Leo". In: pp. 1–33.

Brownlee, Jason (2018). *A Gentle Introduction to k-fold Cross-Validation*. https://machinelearningmastery.com/k-fold-cross-validation/.

Centers for Disease Control and Prevention, CDC (2017). "National Diabetes Statistics Report: Estimates of Diabetes and Its Burden in the United States. Atlanta, GA: Centers for Disease Control and Prevention; 2017". In: *US Department of Health and Human Services* Cdc, pp. 2009–2012. ISSN: 1943-4693. DOI: 10.1177/1527154408322560. URL: https://www.cdc.gov/diabetes/pdfs/data/statistics/national-diabetes-statistics-report.pdf.

centre, WHO Media (n.d.). *Diabetes*. URL: https://web.archive.org/web/20130826174444/http://www.who.int/mediacentre/factsheets/fs312/en/.

Chen, Tianqi (2015). *What is the difference between the R gbm (gradient boosting machine) and xgboost (extreme gradient boosting)?* https://www.quora.com/What-is-the-difference-between-the-R-gbm-gradient-boosting-machine-and-xgboost-extreme-gradient-boosting.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: DOI: 10.1145/2939672.2939785. arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754%7B%5C%%7D0Ahttp://dx.doi.org/10.1145/2939672.2939785.

Chi, Yue, Xin Liu, Kewen Xia, and Chang Su (2008). "An intelligent diagnosis to type 2 diabetes based on QPSO algorithm and WLS-SVM". In: *Proceedings - 2nd 2008 International Symposium on Intelligent Information Technology Application Workshop, IITA 2008 Workshop* 1.1, pp. 117–121. DOI: 10.1109/IITA.Workshops.2008.36.

Community, The Global Diabetes (2018). *Type 2 Diabetes*. https://www.diabetes.co.uk/type2-diabetes.html.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2017). "The Elements of Statistical Learning The Elements of Statistical Learning". In: URL: https://web.stanford.edu/%7B~%7Dhastie/Papers/ESLII.pdf.

Health Statistics, National Center for (2015). *International Classification of Diseases,Ninth Revision, Clinical Modification (ICD-9-CM)*. https://www.cdc.gov/nchs/icd/icd9cm.html.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Vol. 103. ISBN: 978-1-4614-7137-0. DOI: 10.1007/978-1-4614-7138-7. URL: http://link.springer.com/10.1007/978-1-4614-7138-7.

Jonner (2018). *Multicollinearity in Decision Tree*. https://datascience.stackexchange.com/questions/31402/multicollinearity-in-decision-tree.

Kadiyala, Akhil and Ashok Kumar (2018). "Applications of python to evaluate the performance of decision tree-based boosting algorithms". In: *Environmental Progress and Sustainable Energy* 37.2, pp. 618–623. ISSN: 19447450. DOI: 10.1002/ep.12888.

Koehrsen, Will (2018). *Hyperparameter Tuning the Random Forest in Python*. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74.

Laurae (2016). *xgboost: "Hi I'm Gamma. What can I do for you?"and the tuning of regularization*. https://medium.com/data-design/xgboost-hi-im-gamma-what-can-i-do-for-you-and-the-tuning-of-regularization-a42ea17e6ab6.

– (n.d.). *Let me learn the learning rate (eta) in xgboost! (or in anything using Gradient Descent optimization)*. https://medium.com/data-design/let-me-learn-the-learning-rate-eta-in-xgboost-d9ad6ec78363.

Lee, Juyoung, Bhumsuk Keam, Eun Jung Jang, Mi Sun Park, Ji Young Lee, Dan Bi Kim, Chang Hoon Lee, Tak Kim, Bermseok Oh, Heon Jin Park, Kyu Bum Kwack, Chaeshin Chu, and Hyung Lae Kim (2011). "Development of a Predictive Model for Type 2 Diabetes Mellitus Using Genetic and Clinical Data". In: *Osong Public Health and Research Perspectives* 2.2, pp. 75–82. ISSN: 22109099. DOI: 10.1016/j.phrp.2011.07.005. URL: http://dx.doi.org/10.1016/j.phrp.2011.07.005.

Little RJA, Rubin (2019). "Statistical Analysis with Missing Data". In: *John Wiley Sons, Incorporated*.

Loinc (2019). *What LOINC is*. https://loinc.org/get-started/what-loinc-is/.

Obadia, Yohan (2017). *The use of KNN for missing values*. https://towardsdatascience.com/the-use-of-knn-for-missing-values-cf33d935c637.

Ogunleye, Adeola Azeez and Wang Qing-Guo (2019). "XGBoost Model for Chronic Kidney Disease Diagnosis". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 5963.c, pp. 1–1. ISSN: 1545-5963. DOI: 10.1109/TCBB.2019.2911071. URL: https://ieeexplore.ieee.org/document/8693581/.

"Release 0.90 xgboost developers" (2019). In:

Ronaghan, Stacey (2018). *The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-learn and Spark*. https://medium.com/@srnghn/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3.

Spark, Cambridge (2017). *Understand your dataset with Xgboost*. https://blog.cambridgespark.com/getting-started-with-xgboost-3ba1488bb7d4.

Sundaram, Ramya Bhaskar (2018). *An End-to-End Guide to Understand the Math behind XGBoost*. https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/.

Sydney (2019). *Understanding the Outputs of the Decision Tree Tool*. `https://community.alteryx.com/t5/Alteryx-Knowledge-Base/Understanding-the-Outputs-of-the-Decision-Tree-Tool/ta-p/144773`.

Therneau, Terry M. and Elizabeth J. Atkinson (2015). "An Introduction to Recursive Partitioning Using the RPART Routines". In: *Mayo Clinic Division of Biostatistics*, pp. 1–62.

Vasconcelos, Gabriel (2018). *Tuning xgboost in R: Part I*. `https://insightr.wordpress.com/2018/05/17/tuning-xgboost-in-r-part-i/?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com`.

Zou, Quan, Kaiyang Qu, Yamei Luo, Dehui Yin, Ying Ju, and Hua Tang (2018). "Predicting Diabetes Mellitus With Machine Learning Techniques". In: *Frontiers in Genetics* 9.November, pp. 1–10. DOI: `10.3389/fgene.2018.00515`.