MACHINE LEARNING CLASSIFICATION ALGORITHMS AND APPLICATIONS WITH R

Tufan BOSTAN

STATISTICS PROJECT

Department of Statistics
Supervisor: Res. Assist. Dr. Cenk İÇÖZ

Eskisehir

Eskisehir Technical University
Faculty of Science
June 20

ÖZET

MAKİNE ÖĞRENMESİ SINIFLANDIRMA ALGORİTMALARI VE R İLE UYGULAMALARI

Tufan BOSTAN

İstatistik Bölümü
Eskişehir Teknik Üniversitesi, Haziran 2021

Danışman: Araş. Gör. Dr. Cenk İÇÖZ

Diyabet teşhisi konan hasta sayısı dünya genelinde hızla artmaya devam etmektedir. Diyabet hastalığı pankreasın yeterli insülin sağlayamaması ya da vücudun oluşturduğu insülini etkili bir biçimde tüketememesi sonucu ile oluşan kronik bir rahatsızlıktır. Belirli bir zamandan sonra göz rahatsızlıkları, kalp damar rahatsızlıkları, böbrek rahatsızlıkları gibi hastalıkları meydana getiren diyabet, tedavi giderlerinin yüksekliği ve iş gücü yitirme sebebi ile rahatsız olan kişiye sosyoekonomik sıkıntı oluşturmaktadır. Diyabet, çoğunlukla yaş ortalaması küçük olanlarda ve yetişkinlerde görülmektedir. Bu nedenle de erken teşhisi oldukça önemlidir. Diyabet teşhisi için kolay, hızlı ve doğru tanı koyma araçlarına ihtiyaç duyulmaktadır. Erken diyabet teşhisi için makine öğrenimi algoritmalarına dayalı kolay, hızlı ve hassas bir tahmin aracı geliştirmek gerekmektedir. Bu çalışmada, günümüzde yaygın olarak kullanılan Makine Öğreniminin sınıflandırma yöntemlerinden yararlanılarak bu hastalığın erken teşhisinin konulması üzerine çalışılmıştır. Bu bağlamda Lojistik Regresyon, Naïve Bayes, Karar Ağaçları, K-En Yakın Komşu, Destek Vektör Makineleri ve Rastgele Ormanlar sınıflandırma yöntemleri sırasıyla uygulanarak yöntemlerin veri üzerindeki başarısının karşılaştırılıp sunulması amaçlanmaktadır. Çalışmanın ilk aşamasında Makine Öğrenmesi ve metotları anlatılmış, sınıflandırma ve başarı kriterlerine değinilmiştir. Ardından sınıflandırma yöntemleri ayrı ayrı incelenmiştir. Daha sonrasında sınıflandırma yöntemlerinin R ile uygulaması yapılmıştır ve performans ölçütleri hesaplanmıştır. Son olarak da örnekler üzerinden elde edilen sonuçlar karşılaştırılarak çıkarılan sonuçlara değinilmiştir.

Anahtar Kelimeler: Diyabet, Makine Öğrenmesi, Sınıflandırma, Lojistik Regresyon, Naïve Bayes, Karar Ağaçları, K-En Yakın Komşu, Destek Vektör Makineleri, Rastgele Ormanlar

ABSTRACT

MACHINE LEARNING CLASSIFICATION ALGORITHMS AND APPLICATIONS WHIT R

Tufan BOSTAN

Department of Statistics
Eskisehir Technical University, June 2021

Supervisor: Res. Assist. Dr. Cenk İÇÖZ

Number of people diagnosed with diabetes disease continues to rise rapidly along the world. It is a chronic disease that occurs because of the pancreas' inability to provide enough insulin or the body's inability to effectively consume the insulin it produces. Diabetes, which causes diseases such as eye diseases, cardiovascular diseases, kidney diseases after a certain period, creates socioeconomic distress for the person with diabetes due to the high treatment costs and loss of work force. Diabetes is mostly seen in younger people and adults. Therefore, early diagnosis is very important. Diagnosis of diabetes requires easy, fast and accurate diagnostic tools. It is necessary to develop an easy, fast and accurate prediction tool based on machine learning algorithms for early diabetes diagnosis. In this study, we try to make the early diagnosis of this disease by using the classification methods of Machine Learning, which has been widely employed today. In this context, it is aimed to compare the success of the several classification methods of the concerned data set by applying Logistic Regression, Naïve Bayes, Decision Trees, K-Nearest Neighbor, Support Vector Machines and Random Forests respectively. In the first stage of the study, Machine Learning and its methods are explained, classification and success criteria are mentioned. Then, the classification methods are examined separately. Afterwards, the classification methods are applied with R and the performance measures of the data is calculated. Finally, the results obtained from the examples are compared and the results are mentioned.

Key words: Diabetes, Machine Learning, Classification, Logistic Regression, Naïve Bayes, Decision Trees, K-Nearest Neighbor, Support Vector Machines, Random Forests

THANKS

I would like to thank my dear supervisor, Res. Asst. Dr. Cenk İÇÖZ, who supported me at every stage of my research, my dear teachers who contributed a lot to us with their lessons in the education, as well as my classmates and my dear family, who supported me throughout my education life.

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1.INTRODUCTION

Machine learning provides an alternative solution to existing standard predictive modeling and has an informational potential that better explains big data (Obermeyer, 2016). Machine Learning has been developed in computational learning and structure definition studies. These studies are based on a calculation to learn all the complex and nonlinear interactions between variables by minimizing the error between predicted and observed results (Draisaitl and Ohno-Machado, 2002). Machine learning, which is a sub-branch of artificial intelligence, which is has been highly used today, is the modeling of systems that make predictions by making inferences from data with mathematical and statistical operations. In this way, it is widely used in the analysis of big data that is difficult to process.

Machine learning is a set of algorithms that provides insight into the internal nature of a data set. Understanding the internal nature of the data set can be used by humans or other machines in a decision-making process. Examples can be given for machine learning, which is everywhere in our daily life, face/fingerprint recognition on mobile phones or to predict advertisements that will interest users based on their activity and to predict the music, that user may like, based on the music the user is listening to, with data collected from other users. In addition, it is widely used in areas such as health, finance, logistics and production. For example; Early care researchers and computer scientists at the University of Nottingham used Machine Learning algorithms for cardiovascular disease risk assessment. They found that these Machine Learning algorithms are better than experienced medical practitioners at determining heart attack risk (Weng and ark., 2017). Thanks to machine learning, it increases the possibility of benefiting from preventive treatment thanks to early diagnosis. At the same time, unnecessary treatments that may arise with incorrect diagnoses are avoided. (Weng and ark., 2017). Machine Learning is generally used for two purposes. The first is to predict future results, a game of chess can be given as an example. By calculating the next possible moves, the correct moves to beat the opponent can be determined. The other is to classify objects into special groups. As an example, it is possible to develop a Machine Learning algorithm that automatically grades the quality of the product for the company that going to buy the products.

In this study, Classification Algorithms are discussed. In the literature, since each of them has complex theories, they have often been studied separately and many examples

are available for each of them. The aim of this study is to present various examples of these classification algorithms in a more understandable way by applying them with the R Programming Language. First, there will be a short introduction to Machine Learning. Next, each of the Classification Algorithms will be explained separately. Then, examples of Classification Algorithms will be made with the R program and finally these results will be examined.

2. MACHINE LEARNING

2.1. Definition

There is a large amount of data that cannot be processed and analyzed manually in the computer environment. Machine learning methods provide significant and statistical results from this data pool and make predictions for the future from past data. In the early days of technological development, there were usually centers where companies' data were stored and processed. Today, thanks to the extensive usage of personal computers and the rapid development of technology, it is seen that almost every individual has become a data source. For example, a person constantly generates data in his/her daily life with the products he/she purchases, the movies she prefers to watch, or the ideas she expresses about a subject on social media. Let’s consider a company that sells online through a website. The company can determine the product that the person may need by looking at the products that person bought or searches of the customers in the past. In today’s world Google is a perfect example for this. Google estimates the products that may be interest to the person, according to searches that person made on the Google. In this way, it can give personal advertisements. It is known by researchers that this situation does not occur randomly such as “when they buy coke, they also buy chips”. Every data has a specific model.

An algorithm is needed to solve a problem with the computer. Algorithm is a sequence of instructions that must be done to convert input to output. The application of machine learning methods to large databases is called data mining. In data mining, large amounts of data are processed to create a simple model with high prediction accuracy (Alpaydin, 2016).

Some Machine learning application areas are, finance, banks, applications, medical diagnosis, crime detection, education planning and stock. Machine learning is not only a

database problem, but also a part of AI (Artificial Intelligence). The system in a changing environment must be capable of learning. If the system can learn and adapt these changes, the system designer does not need to predict and provide solutions for all possible situations. The system will be able to handle potential problems on its own.

Machine learning is programming computers to optimize a performance metric using sample data or past experiences and uses statistical theory in constructing mathematical models because the main task is to infer from a sample. Computer science has two important roles in machine learning. The first of these is the need for efficient algorithms to solve the optimization problem and to store and process the large amount of data generally possessed during the training phase. Second, after a model is learned, its representation and algorithmic solution must also be effective for its inference (Alpaydin, 2016).

2.2. Machine Learning Methods

Machine learning is divided into two as supervised and unsupervised. In supervised learning, the inputs for each sample belonging to the case and the corresponding output values should introduced to the system. Here, the job of the system is to develop the inputs according to the specified outputs. In this way, the relationships between input and output can be determined. Unsupervised learning is a learning method that consists of only observations with input variables.

2.2.1. Supervised Machine Learning

In the supervised learning technique, the data to be used in the learning phase have observed results. In this case, the data is called as labelled. The corresponding function that matches between input data and output data is performed. This function can be determined by classification or regression algorithms. If the outputs in the data set are categorical, classification algorithms should be used. If they are numerical, regression algorithms should be used. Data analysts determine which features the model needs to analyze, in other words, which data features have an impact on the output and determines which feature group is most effective in modeling output by calculating accuracies for different feature groups. Models created using these features can make better predictions for new data. Some scenarios in which supervised machine learning algorithms are used are; It can be represented as price prediction, trend prediction, retail trade and stock trading. Some of the commonly used supervised machine learning algorithms are Linear

Regression, Logistic Regression, Random Forest, Gradient Boosted Tree, Support Vector Machines, Artificial Neural Networks, Decision Trees, Naive Bayes, K-Nearest Neighbor.

2.2.2. Unsupervised Machine Learning

Unlike the supervised learning technique, where outputs are known for inputs consisting of different feature sets in the data, the desired results in the unsupervised learning technique are not yet defined and unknown. The unsupervised learning technique uses it to predict similar patterns in unlabeled data. For example, if we are talking about a classification problem, it is not clear to say which class the input data belongs to. Unsupervised methods generally classify data using similarities and differences within the data. Unsupervised learning uses two different techniques. These techniques are clustering and dimensionality reduction. Areas where unsupervised machine learning algorithms are used; The situations in which the structure of the information needs to be discovered (separation of the geographical regions photographed from the satellite according to their content), the extraction of valuable information (the presence of different features in the DNA), the detection of patterns (customer segmentation). Some commonly used unsupervised machine learning algorithms are; K-Mean Clustering, T-Distributed Stochastic Neighborhood Burials (T-SNE), Principal Component Analysis, Association rule.

2.3. Holdout Method

The holdout method consists of randomly separating the existing data sample into two subsets: one is used to train the model and the other is used to test it. A common ratio is 70% for training and 30% for testing. If there is a small sample of data, there is a danger of either extracting too much data from a very small test set or a training set. In other words, this method is typically only preferred for very large datasets (Torgo, 2016).

2.4. Classification

Classification aims to assign each vector defined as input to different groups independent of each other. Algorithms written to solve the classification problem are called classifiers. (Alpaydın, 2016; Bishop, 2007; Camastra ve Vinciarelli, 2008)

2.5. Classification Success Criteria

Some criteria have been developed to control the superiority of classifiers relative to each other. These criteria are based on the creation of a confusion matrix.

The representative table formed by the actual output values and the estimated values obtained from the model is given in Table 2.1.

		Actual		
		Positive	Negative	Total
Predict	Positive	True Positive (TP)	False Positive (FP)	TotP
	Negative	False Negative (FN)	True Negative (TN)	TotN
	Total	Positive (P)	Negative (N)	Grand Total (G)

Table 2.1: *Table of Confusions*

According to Table 2.1, classification success criteria are calculated with the following equations. Accuracy and error rate of classification are found with the following equations.

$$Accuracy = \frac{TP + TN}{G} \tag{2.1}$$

$$Error\ Rate = 1 - Accuracy \tag{2.2}$$

The efficiency of the classifier in predicting positive class labels is given by the sensitivity ratio:

$$Sensitivity = \frac{TP}{P} \tag{2.3}$$

The efficiency of the classifier in predicting negative class labels is given by the specificity ratio.

$$Specificity = \frac{TN}{N} \tag{2.4}$$

3. MACHINE LEARNING CLASSIFICATION METHODS

3.1. Logistic Regression

Logistic regression analysis is basically a method similar to linear regression analysis. The purpose of linear regression analysis is to determine the cause-effect relationship between the dependent variable and the independent variable. In linear regression analysis, this purpose is provided by the following equation.

$$E(Y|X) = \beta_0 + \beta_1 x + \epsilon \quad (3.1)$$

In the equation, B0 denotes the initial constant, B1 denotes the expected change in y of the 1-unit x variable increment, and E denotes the error in the estimation. The equation also gives an answer to the question of what is the expected value of y for any x. All variables used in linear regression analysis must be continuous. In this case, this application cannot be done if the dependent variable is categorical. In logistic regression analysis, the dependent variable should be categorical. Therefore, logistic regression analysis has been introduced in order to overcome this deficiency of simple linear regression analysis. In logistic regression analysis, independent variables are calculated with the logarithmic transformation function in order to solve such structural problems. If there is only one independent variable, the logistic regression equation to be created is as follows.

$$\pi(x) = \frac{e^{(\beta_0+\beta_1 X)}}{1 + e^{(\beta_0+\beta_1 X)}} \quad (3.2)$$

Since the equation has an asymptotic structure, the result to be obtained in (3.2) is between 0 and 1. β_0 in the equation is the constant and β_1 is the regression coefficient showing the effect of the independent variable on the dependent variable.

The classification logic of logistic regression is based on the probability of whether the decision-making unit belongs to a group or a cluster. This probability is expressed as the number of odds and formulated as follows.

$$Odds\ Ratio = \frac{p(x)}{1 - p(x)} \quad (3.3)$$

Odds ratio is used in the correct interpretation of B coefficients in logistic regression. Here $p(x)$ shows the probability of any event x, $1 - p(x)$ is the probability that x does not occur. In this case, if the occurrence of x event is evaluated as a 'success', the equation in (3.3) shows the ratio of the probability of success to the probability of failure.

By making logarithmic transformation to the obtained odds function it is ensured that the dependent variable can take values continuously and between $(-\infty, +\infty)$. This equation is called the logit function. This function is illustrated as follows:

$$g(x) = \ln\left[\frac{\pi(x)}{1-\pi(x)}\right] = \beta_0 + \beta_1 x \tag{3.4}$$

As a result of the function, the dependent variable increases and approaches 1 or decreases to 0. The general acceptance opinion is classified as 1 if the value found here is greater than 0.5, and as 0 if it is less.

3.2. Naïve Bayes

The Naive Bayes Classifier is a simple probabilistic classification method based on Bayes' theorem. It is an approach that calculates the probability that a new data belongs to any of the existing classes using the sample data in the current classified state. In this classifier, the attributes are considered independent of each other. The examples are all equally important. The value of one property does not contain information about another property value.

Suppose we are working on a data set, each of which consists of n attributes and belongs to any of the m classes. In this case, when you want to classify a new X example whose class is unknown, the probability of belonging to that class is calculated by using Equation (3.5). The class with the highest probability of these values is considered to be the class to which the sample belongs.

$$P(S_i|X) = \frac{P(X|S_i) * P(S_i)}{P(X)} \tag{3.5}$$

$P(S_i|X)$: The probability of S_i event occurring when X event occurs, $P(X|S_i)$: Probability of X event occurring when Si event occurs, $P(S_i)$,$P(X)$: It is the a priori probability of events Si and X.

The value of $P(X)$ is the same for each sample data, since each X sample is equally important. In this case, Equation (3.5) can be simplified to Equation (3.6).

$$P(S_i|X) = P(X|S_i) * P(S_i) \tag{3.6}$$

After applying Equation (5) for each class and calculating the probabilities, the class to which the sample belongs can be found. (Bermejo vd., 2011)

3.3. Decision Tree

Decision trees are one of the simplest forms of the decision model, and they use sample data properties to create their rules in the form of a tree structure. The origins of decision trees are similar to the way people make decisions. Decision trees are very popular because they present information in a more visually understandable way. It can be used for both regression and classification problems. Decision trees can reduce a data sample to a controllable set of rules that can be used to make a classification decision. (Bohanec and Bratka,1994; Lewis, 2017).

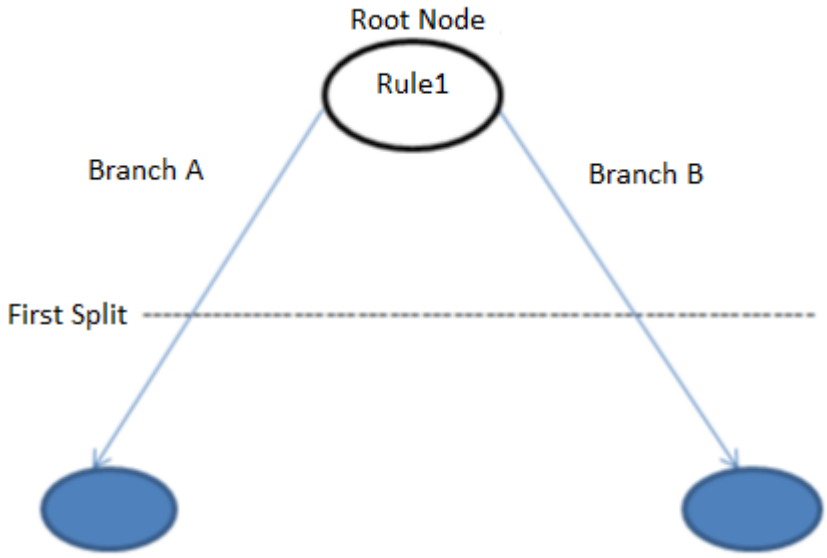


Figure 3.1: *Decision Tree's Root Node*

The goal is to choose a feature and rule that divides the data into correctly classified samples. This process involves defining the property that is most useful for classification and obtaining a decision rule using that property.

All properties are tested to see which feature is best for splitting data. Ideally, a simple rule divides training examples into their own classes. However, in most cases it does not develop as desired, and a rule is chosen that separates samples as precisely as possible. This is Rule-1 in Figure 3.1. After this stage, the tree consists of two new branches with associated nodes.

As seen in Figure 3.2, the nodes of the new branch are determined after the previous stage. In this section, if all instances in a node belong to the same class, then it paused, and each leaf node is labeled with the name of that class.

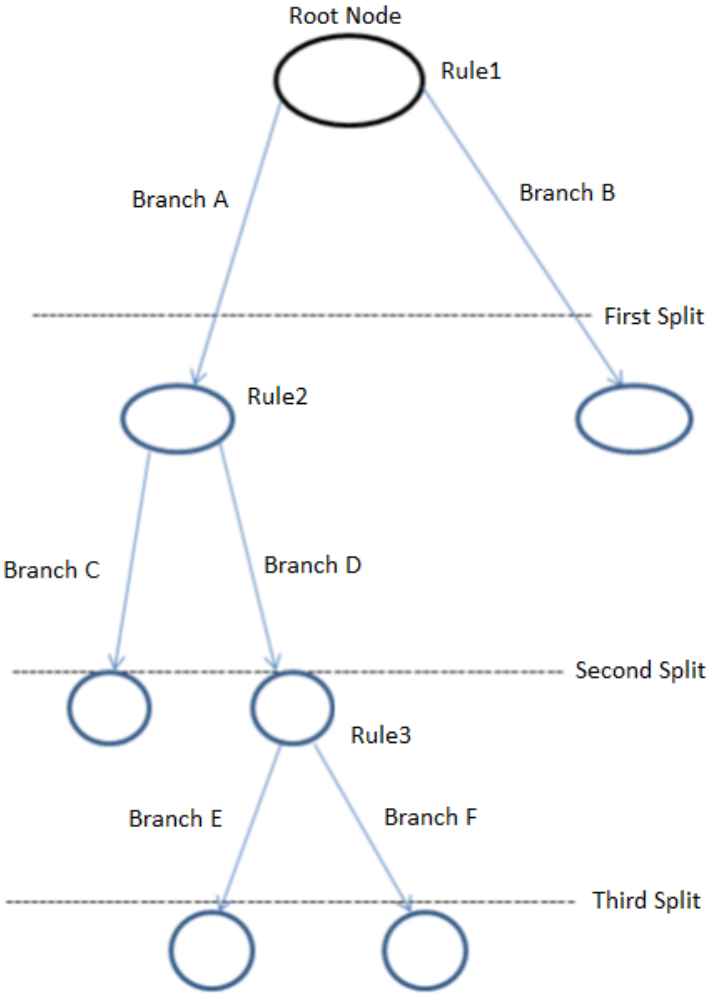


Figure 3.2: *Decision Tree's Root Node Splits*

This tree-building process is repeated for each branch, allowing samples to be split to further improve the classification purity of the data. The process continues up to a stop criterion or ends when all states in the nodes section have been correctly classified. A typical stop criterion is to stop data from splitting when new feature-rule combinations increase the purity of subgroups by a very small amount.

3.4. K-Nearest Neighbors

The closest neighborhood algorithm is one of the machine learning algorithms that is easy to understand and works quite well in practice. This algorithm is a nonparametric algorithm. Here the nonparametric means that, there is no assumption about the probability distribution that produces the sample data. Real world data tend to violate theoretical assumptions and therefore nonparametric algorithms such as KNN are useful.

KNN calculates the distance between the observations to be classified and their neighbors. The estimated class is determined by a majority vote. If we look for an answer to the question of how to determine the proximity of a neighbor; The distance between i

and j can be measured in many ways. Euclidean distance is often used for continuous features and KNN calculates the distance between the observations to be classified and their neighbors. The estimated class is determined by a majority vote. If we look for an answer to the question of how to determine the proximity of a neighbor; The distance between i and j can be measured in many ways. Euclidean distance is generally used for continuous properties and denoted as $D(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$, where n is the number of features. Another popular measure is Manhattan Distance calculated as $D(x_i, x_j) = \sum_{m=1}^{n}(|x_{im} - x_{jm}|)$. The Minkowski Distance is obtained by the equation $D(x_i, x_j) = \sum_{m=1}^{n}\left(|x_{im} - x_{jm}|^q\right)^{1/q}$. Another distance, called the Canberra Distance, is found by $D(x_i, x_j) = \frac{\sum_{m=1}^{n}(|x_{im} - x_{jm}|)}{\sum_{m=1}^{n}(|x_{im} + x_{jm}|)}$. Both Euclidean and Manhattan distance are considered special forms of Minkowski distance. The Minkowski distance is Euclidean when q = 2 is selected and is the weighted version of the Manhattan distance when q = 1. If you change a distance function you change how the data points are classified (Zhang and ark., 2018).

3.5. Support Vector Machines

The support vector machine is capable of dividing data into two or more classes with separation mechanisms in the form of linear in two-dimensional space, planar in three-dimensional space and hyperplane in multi-dimensional space. The method, which is frequently used to determine linearly decomposable classes, is successfully used in classification of nonlinear data by moving the input space that cannot be parsed linearly through kernel functions to higher dimensional linearly decomposable space. Assuming $\theta = \{x_i, y_i\}, i = 1,2, \dots, N$ for the data consisting of N elements to be used for education, x_i indicates the feature vector, and $y_i \in \{-1,1\}$ indicates the class values. In the case of linear separation, these bivalent data can be separated directly by a plane. Although an infinite number of multiple planes can be drawn that can divide the data set into classes, the purpose is to select the hyperplane that will make the unknown classification error the smallest. As can be seen in Figure 3.3, $f(\vec{x}) = \vec{w}^T\vec{x} + b \geq 1$ represents the first class, $(y_i = 1)$ and $f(\vec{x}) = \vec{w}^T\vec{x} + b \leq -1$ represents the second class $(y_i = -1)$.

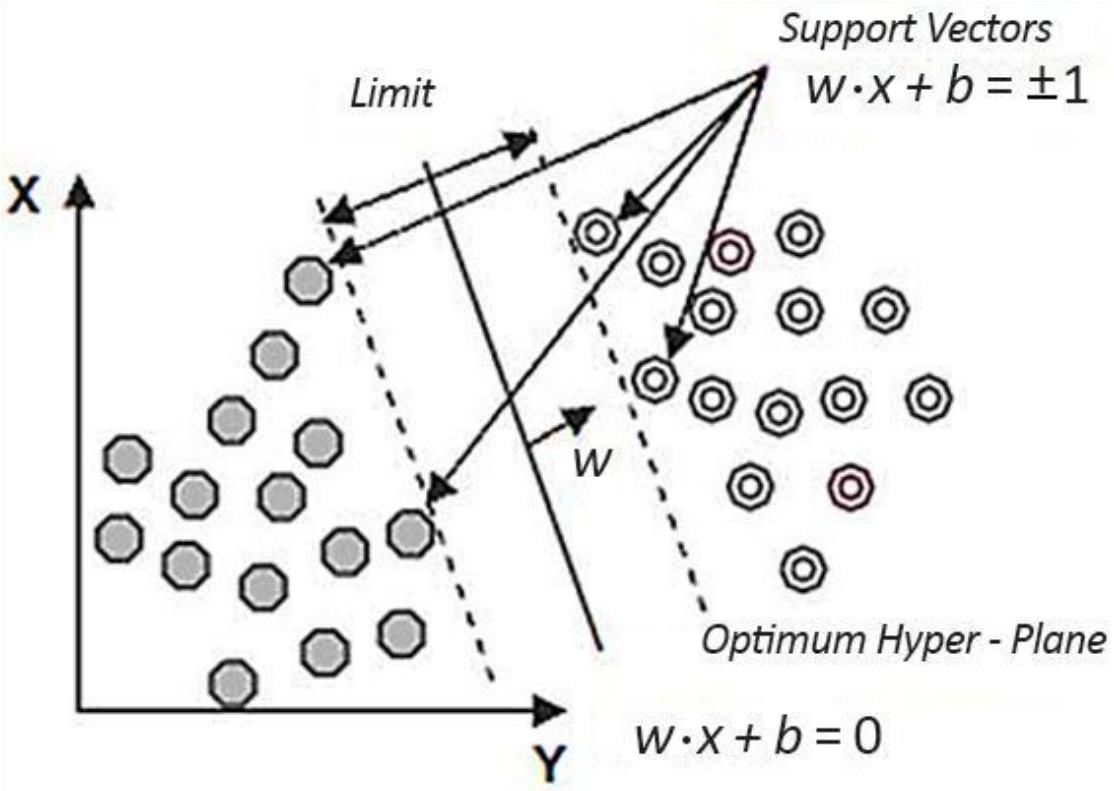


Figure 3.3: *Support Vector Machines and Hyperplane Selection*

The distance between two boundaries is expressed by the formula $\lambda = 2/\vec{w}^2$. Since the aim is to make the value of λ maximum, the expression $1/\lambda$ should be minimum. The related limitation is $y_i(w^T x_i + b) - 1 \geq 0, y_i \in \{-1, +1\}$. The dual of the related problem is given in Equation (3.7). The problem in equality is solved with the help of Lagrange equations, the constraints of "Karush-Kuhn-Tucker (KKT)" given in Equation (3.8) and Equation (3.9).

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} a_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1], a_i \geq 0, \forall i \tag{3.7}$$

$$\frac{\partial L}{\partial w_j} = 0, \forall i \tag{3.8}$$

$$\frac{\partial L}{\partial w_n} = 0 \tag{3.9}$$

Nonlinear transformations can be made using the kernel function. This allows it to be separated linearly in high dimensions. The most common kernel functions are Gauss, Polynomial and Sigmoid functions.

3.6. Random Forest

Random Forest is an algorithm that achieves a high level of success rate in classification using many decision trees structures (Breiman, 2001).

In the random forest algorithm, similar to other decision trees, it is important to determine branching criteria and to choose an appropriate pruning method. The Gini Coefficient method is used to determine branching criteria (Mather, 2005).

In order to create the tree structure in the algorithm, the number of instances to be used in each node and the number of trees to be created must be determined. During classification, the decision forest is created from K trees determined by the user. When a new object is to be classified, it is processed by these K decision trees and the class is determined by selecting the highest one among the ratios obtained from each tree (Pal, 2005; Çölkesen, 2009).

If the training and test data for the data set to be used in the random forest is not determined beforehand, considering the class rates in the data set, 2/3 of the whole data set is used as training (inBag) and 1/3 as test data (OutOfBag, OOB). For K decision trees that will form the decision forest, again, a sample is created using K bootstrap techniques, and inBag and OOB data are separated for each sample. All trees are tested with the allocated OOB data, the error rate is calculated and then the OOB error of the decision forest is calculated by taking the average of these error rates. A weight is given to all trees according to the calculated OOB error rate. The error rate and the weight given to the tree are inversely proportional. The decision tree with the highest error rate takes the lowest weight, while the decision tree with the lowest error rate gets the highest weight. All trees go through a voting process for classification according to the determined weights. The tree with the highest number of votes is determined as a class estimate (Akman et al., 2011).

4. APPLICATIONS

In this section, the usage of the method used in the thesis is shown on an example. First, the data set used and the attributes in the related data set will be introduced. Then the data split into two parts and some machine learning classification methods which are Logistic Regression, Naïve Bayes, Decision Trees, K-Nearest Neighbor, Support Vector Machines and Random Forests applied to the data set and results of these methods will be examined. This application is written in R language. The popularity of R for machine

learning is one of the reasons to use this language. R Studio was used as the compiler. From machine learning classification algorithms, Logistic Regression, Naïve Bayes, Decision Trees, K-Nearest Neighbor, Support Vector Machines and Random Forests were used. In addition to these algorithms, support was received from the "e1071", "rpart", "rpart.plot", "neuralnet" and "randomForest" libraries in the development of the project.

4.1. Dataset

To estimate the probability of having diabetes, a dataset containing data from patients who have new or have diabetes is required. In this study a data set used which has been collected using direct questionnaires from the patients of Sylhet Diabetes Hospital in Sylhet, Bangladesh and approved by a doctor. This dataset was donated to the "UCI Machine Learning Repository" web page on 2020-07-12 and downloaded for use in this study on 2021-05-27, to access the web page of the dataset or get more information about it, see 15. Reference. The data set will be analyzed with Logistic Regression Algorithm, Naïve Bayes Algorithm, Decision Trees Algorithm, K-Nearest Neighbor Algorithm, Support Vector Machines Algorithm and Random Forests Algorithm. In this data set, the algorithm with the best accuracy will be tried to be found. Each attribute in the dataset is shown in Table 4. 1.

Attribute	Explanation
Age	age of participant;
Sex	participant's gender; Female or Male
Polyuria	Urination of more than 3 liters per day in an adult patient is called polyuria; Yes or No
Polydipsia	Abnormally great thirst as a symptom of disease (such as diabetes) or psychological disturbance; Yes or No
sudden weight loss	Unexplained weight loss is a decrease in body weight, when participants did not try to lose the weight on their own; Yes or No
weakness	the state or condition of being weak; Yes or No
Polyphagia	excessive eating or appetite, especially as a symptom of disease; Yes or No
Genital thrush	a common yeast infection that affects men and women; Yes or No
visual blurring	Lack of sharpness of vision with, as a result, the inability to see fine detail; Yes or No
Itching	an uneasy irritating sensation in the upper surface of the skin; Yes or No

Irritability	becoming frustrated or upset easily; Yes or No
delayed healing	that the any part of the body took or is taking longer to heal than expected; Yes or No
partial paresis	the weakening of a muscle or group of muscles; Yes or No
Muscle stiffness	the muscles feel tight and difficult to move, particularly after resting; Yes or No
Alopecia	the partial or complete absence of hair from areas of the body where it normally grows; baldness; Yes or No
Obesity	abnormal or excessive fat accumulation that presents a risk to health; Yes or No
Class	diagnosis of diabetes; Negative or Positive

Table 4.1: *Attributes of the data set and their explanations*

4.1.1. Preparation Of the Dataset

First, the dataset was downloaded from its source and uploaded to the R program. Then the variables in the data set were redefined as factors. This process has been done to eliminate the errors that will occur in case of incorrect type of the variables.

Then, it was checked whether there was any missing value in the data, and it was concluded that there was no missing value. No missing value provides more reliable results.

In the next step, some descriptive statistics were obtained for the dataset to gather more information about the data.

```
#data summary/descriptive statistics
summary(data)

##       Age           Gender    Polyuria  Polydipsia sudden weight loss weakness
##  Min.   :16.00   Female:192   No :262   No :287        No :303        No :215
##  1st Qu.:39.00   Male  :328   Yes:258   Yes:233        Yes:217        Yes:305
##  Median :47.50
##  Mean   :48.03
##  3rd Qu.:57.00
##  Max.   :90.00
##
##  Polyphagia  Genital thrush visual blurring Itching   Irritability
##   No :283    No :404        No :287         No :267     No :394
##   Yes:237    Yes:116        Yes:233         Yes:253     Yes:126
##
##
##  delayed healing partial paresis muscle stiffness Alopecia  Obesity
##  No :281         No :296         No :325          No :341   No :432
##  Yes:239         Yes:224         Yes:195          Yes:179   Yes: 88
##
##
##     class
##  Negative:200
##  Positive:320
##
```

Figure 4.1: *Summary of the Dataset*

```
str(data)

## tibble [520 x 17] (S3: tbl_df/tbl/data.frame)
##  $ Age               : num [1:520] 40 58 41 45 60 55 57 66 67 70 ...
##  $ Gender            : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2
...
##  $ Polyuria          : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 2 2 2 2 1 ...
##  $ Polydipsia        : Factor w/ 2 levels "No","Yes": 2 1 1 1 2 2 2 2 2 2 ...
##  $ sudden weight loss: Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 2 1 2 ...
##  $ weakness          : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Polyphagia        : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 2 1 2 2 ...
##  $ Genital thrush    : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 2 1 2 1 ...
##  $ visual blurring   : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 2 1 2 1 2 ...
##  $ Itching           : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 1 2 2 2 ...
##  $ Irritability      : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 2 2 2 ...
##  $ delayed healing   : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 2 1 1 1 ...
##  $ partial paresis   : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 1 2 2 2 1 ...
##  $ muscle stiffness  : Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 1 2 2 1 ...
##  $ Alopecia          : Factor w/ 2 levels "No","Yes": 2 2 2 1 2 2 1 1 1 2 ...
##  $ Obesity           : Factor w/ 2 levels "No","Yes": 2 1 1 1 2 2 1 1 2 1 ...
##  $ class             : Factor w/ 2 levels "Negative","Positive": 2 2 2 2 2 2 2
2 2 2 ...
```

Figure 4.2: *Structure of the Dataset*

Here, information about the number of observations for each variable in the data set can be obtained. According to the results obtained in Figure 4.1, the age variable takes values between 16 and 90 and its average is 48.03. From this it can be concluded that the data set was obtained from people between the ages of 16 and 90. The gender variable is a categorical variable with 2 levels and 192 women, and 328 men are observed. It can be said that the number of males is dominant in this data set. The remaining variables are similarly categorical variables with 2 levels and the last one, "class" variable is the response variable. This variable also has 2 levels, and they are “Negative” and “Positive”, as seen in both outputs and histogram of the “class” variable in Figure 4.3. Here, a negative indicates that the person does not have diabetes, while a positive indicates that the person has diabetes. The data types and the range or levels of the variables can be clearly seen in Figure 4.2. While age is a numeric variable, all other variables are defined as factors.

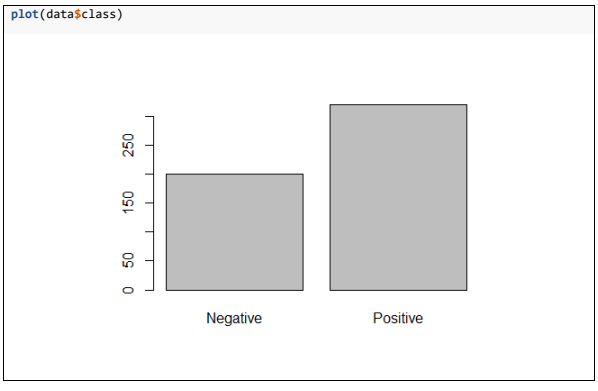


Figure 4.3: *Histogram of class Variable*

4.1.2. Data Split

At this stage, the data set is split into two parts. 70% of the data set was used for training and 30% for testing. This selection is made randomly. The training set will be used to train the models and the test set will be used to measure the success of the trained models. Here, the ratio used when splitting the data set varies according to the analyzer. In general, 70-80% are reserved for train. This separation process has a direct impact on the performance of the model. If too much data is reserved for training, the model can learn training data well, but this may also reduce its performance on the test set. For this reason, an overfitting problem may be encountered, and as a result, it should be considered as it may negatively affect its performance in new datasets.

```
nrow(train);nrow(test)

## [1] 364

## [1] 156
```

Figure 4.4: *Row count of the dataset after it split.*

```
summary(train$class);summary(test$class)

## Negative Positive
##      141      223

## Negative Positive
##       59       97
```

Figure 4.5: *Summary of the train and test sets.*

The data set is split into two parts. After the data set is split, according to output in Figure 4.4 the number of observations to be used for training is 364 and the number of observations to be used for testing is 156. In addition, according to output in Figure 4.5, response variable "class" has 141 negative and 223 positive observations in the "train" set and 59 negative and 97 positive for the test. Here, it is seen that the number of positive observations is higher than the number of negative observations. This may cause models to learn the positive class better but not the negative class enough. In this case, it may be necessary to consider that it may reduce the success in predicting the negative class.

4.2. Function for Performance Calculation

In this section, a function named “performance” is created to calculate the performance of the models. This function takes the matrix input, which is confusion matrix and returns TP, FN, TN, FP, accuracy, error rate, sensitivity and specificity values to be used in performance comparison.

4.3. Application of algorithms to the dataset

In this section, the train set and models will be generated for each algorithm and the predictive values for the response variable will be obtained with the models. Then, accuracy values and confusion matrices for the test set will be calculated to see the success on the train and test sets. Finally, the values to be used in the comparison will be obtained with the "performance" function and stored for evaluation in the result section.

4.3.1. Logistic Regressıon Application

In this study, the “glm()” function was used to apply Logistic regression. The response variable and the explanatory variables to be used are introduced to this function first. All variables in this data set will be used. The class variable as the response and the remaining variables as the explanatory variables. As mentioned before, the dataset was trained with the training set (Figure 4.6).

```
#...Training Log. Reg. Model...
model_LRM <-  glm(class~. ,data = train, family = "binomial")
```

Figure 4.6: *Training Logistic Regression model in R*

The model has been trained, now using this model the prediction probabilities for the train and test sets calculated. The same process was applied for the test set and accuracy values were obtained for both data sets and confusion matrix obtained for the test set, results are as in Figure 4.7.

```
## [1] "Train"             "0.950549450549451"

## [1] "Test"              "0.935897435897436"

##           actual
## predicted  Negative Positive
##   Negative       54        5
##   Positive        5       92
```

Figure 4.7: *Accuracy Scores and Confusion Matrix for Logistic Regression Model*

According to Figure 4.7, the accuracy values obtained for the train and test sets are 0.95 and 0.93, respectively. It can be said to be quite successful. It also made 10 misclassifications according to the confusion matrix.

4.3.2. Naïve Bayes Application

For this method, the "naiveBayes()" function from the "e1071" library is used. As previous model, train set used for training the model. The class variable as the response and the remaining variables as the explanatory variables (Figure 4.8).

```r
library(e1071)

#...Training Naive Bayes Model...
model_NB <- naiveBayes(x=train[-17], y=train$class)
```

Figure 4.8: *Training Naïve Bayes model in R*

The model has been trained and predictive values for train and test sets obtained using this model. Here, as before, accuracy values for the train and test sets and the confusion matrix for the test set were obtained, results are as in Figure 4.9.

```
## [1] "Train"             "0.881868131868132"

## [1] "Test"              "0.846153846153846"

##           actual
## predicted  Negative Positive
##   Negative       55       20
##   Positive        4       77
```

Figure 4.9: *Accuracy Scores and Confusion Matrix for Naïve Bayes Model*

According to Figure 4.9, the accuracy values of the train and test sets are 0.88 and 0.84, respectively. It is a little bit low compared to the previous model. The model also made 24 misclassifications according to the confusion matrix. It predicted as negative the patients who should mostly be positive and need treatment. This is a difficult situation to accept, especially in the healthcare field. Therefore, we can say that the success of this model is low for now.

4.3.3. Decision Tree Application

The "rpart" library is used for this method. The function used here is "rpart()". As previous models, “class” used as the response variable, and all remaining variables as explanatory variables and train set of the data to train model (Figure 4.10). In addition, the decision tree was obtained and displayed in Figure 4.11.

```r
#...Training Decision Tree Model...
library(rpart)

library(rpart.plot)

model_DT <- rpart(class ~., method = "class", data = train)
rpart.plot(model_DT)
```

Figure 4.10: *Training Decision Tree model in R*

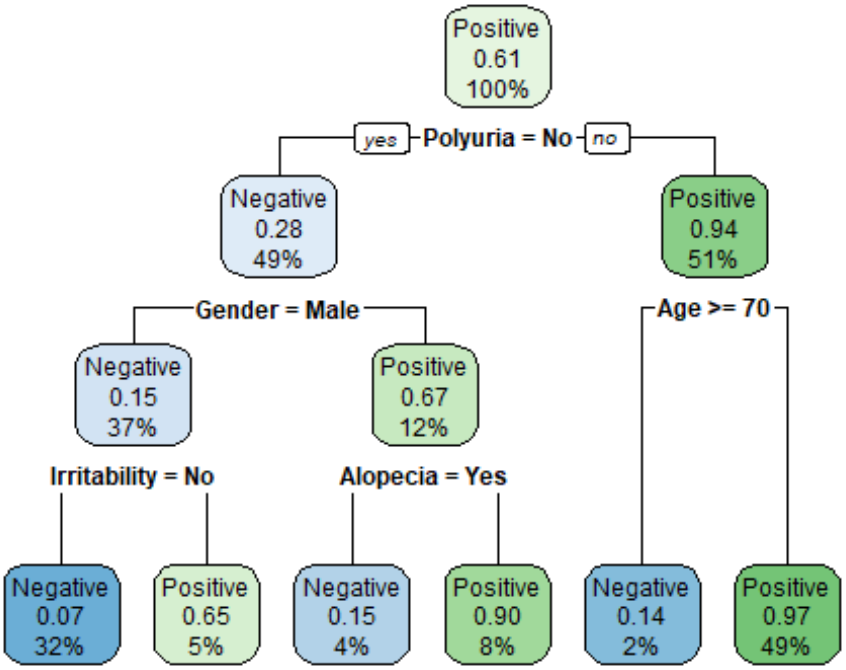


Figure 4.11: *Decision Tree*

In Figure 4.11, decision there were displayed. Positive was the most observed in the test part of the data so in root node, Positive were observed. The colors of the nodes are depending on the levels and the nodes take the color of the most observed level. For example, in the first node which called as root node, 61% of the data is Positive, so the node has light green color. This ratios on second row of each node represents chance of having diabetes on that node. Percentages are indicated on the last line of each box; these are indicating what percentage of all data is in that node. For root node, this percentage is 100 because the data did not split yet. If the Polyuria observed as No, then it goes left. If it is observed as Yes, then it goes right. Let's consider Polyuria is observed as No then it goes to left as mentioned before. That box colored as blue which means, Negative is most observed level here. The probability that participants in this node have diabetes is 0.28 and the percentage that written of last row of this box 49% so the main data split into two parts and 49% of them on the left and 51% of them on the right side. At the top of tree, it is seen that only 61% of all participants have diabetes. This is directly related to the observations in the data used. It is the result that is inferred from the data when the effect of no variable is included. In the event that symptoms (variables) are observed or not, some decisions made by the algorithm are shown in the next branches of the tree.

Polyuria is in the root node, and its asks whether the participant does have polyuria. If yes, then it goes to the right node. In this node it seen that participant have diabetes whit 0.94 probability. From this, it can be concluded that polyuria is one of the most important symptoms of diabetes. If we continue on the same branch, next distinction will be age. If participant age is smaller than 70, then this ratio rises to 97%. On the other hand, participants who show signs of polyuria and are over 70 years of age are less likely to have diabetes, with a ratio of 0.14. It can be concluded that, if the person who has polyuria symptom and under 70, S/he has a 97% chance of having diabetes. Now, let's check the left side of the root node, this happens when the participant does not have polyuria. In second node of the left side, asks whether gender of the participants male, if it yes then it goes left again and reaches to third node. In this node also negative as dominant. Also 12% of the participants without polyuria were female and 37% were male compared to all participants, according to third nodes of the left side. If we keep going on left, next node will ask whether patricians have irritability. If it is no, then it goes left and finally reaches to leaf node which is at the bottom left corner. The conclusion to be obtain here is that the probability of having diabetes in male participants without polyuria and irritability is 0.07 which is quite low. If we go back to second node again and consider that the answer not male, then it goes right. In this node again positive is dominant. It is seen that female participants without polyuria symptoms are more likely to have diabetes than male participants without polyuria symptoms. If we continue from there, next question will whether participant does have alopecia. If she does have alopecia, it goes left and reaches the last node (leaf node), then probably she does not have diabetes because the probability of having diabetes in this node is 0.15.

In the next step, the values related to the performance of the model and the confusion matrix were obtained, results are as in Figure 4.12.

```
## [1] "Train:"            "0.925824175824176"

## [1] "Test:"             "0.858974358974359"

##
## pred_labels_dt_test Negative Positive
##            Negative       50       13
##            Positive        9       84
```

Figure 4.12: *Accuracy Scores and Confusion Matrix for Decision Tree Model*

According to Figure 4.12, the accuracy values obtained for train and test are 0.93 and 0.85, respectively. Although the model is successful on the training set, the accuracy

value of the test set is way lower than the training set. In this case, it is possible to err in the estimations made on new patients. On the other hand, the model made 22 misclassifications according to the confusion matrix.

4.3.4. K-Nearest Neighbors Application

For this method, the "knn()" function from the "class" library is used. As previous models, train set used for training the model and “class” variable as the response and the remaining variables as the explanatory variables. Also, the number of neighbors is set as 5 (Figure 4.13).

```
library(class)
#...Training K-Nearest Neighbors Model...
model_KNN <- knn(train=trainKNN, test=testKNN, cl=trainKNN[,17], k=5)
```

Figure 4.13: *Training K-Nearest Neighbors model in R*

Then, confusion matrix was created with the predicted values obtained from the model and its accuracy on the test was calculated, results are as in Figure 4.14.

```
## [1] "Test"              "0.865384615384615"

##    model_KNN
##       1  2
##   1 57  2
##   2 19 78
```

Figure 4.14: *Accuracy Score and Confusion Matrix for K-NN Model*

According to Figure 4.15, the success of this model on the test set is 0.87. It is quite low compared to previous models. In the confusion matrix here, 1 represents “Negative” and 2 “Positive” values. It is seen that there are 21 incorrect classifications. It can be a difficult situation to accept in the healthcare field, as in the Naïve Bayes model. Therefore, this model may not have sufficient success for this study.

4.3.5. Support Vector Machines Application

Here, the "e1071" library, which was also used for Naive Bayes, was used. The function used is “svm()”. For this model, as in other models, the class variable is used as the response variable and the explanatory variable for all the remaining variables and train set was used to train the model (Figure 4.15).

```
library(e1071)
#...Training Supp. Vec. Mach. Model...
model_SVM <- svm(formula=class~., data=train, type="C-classification", kernel = "linear")
```

Figure 4.15: *Training Support Vector Machines model in R*

The model has been trained and predictive values for train and test sets obtained using this model. Accuracy values for the train and test sets and the confusion matrix for the test set were obtained, results are as in Figure 4.16.

```
## [1] "Train"             "0.936813186813187"

## [1] "Test"              "0.923076923076923"

##           actual
## predicted  Negative Positive
##   Negative       53        6
##   Positive        6       91
```

Figure 4.16: *Accuracy Score and Confusion Matrix for Support Vector Machines Model*

According to Figure 4.16, the accuracy values obtained for train and test are 0.94 and 0.92, respectively. It can be said to be quite successful. At the same time, the model made 12 misclassifications according to the confusion matrix.

4.3.6. Random Forest Application

For this method, the "randomForest()" function from the "randomForest" library is used. As previous models, train set used for training the model and class variable as the response and the remaining variables as the explanatory variables. Also, the number of trees is set as 10 for this study (Figure 4.17).

```
library(randomForest)

#...Training Random Forest Model...
model_RF <- randomForest(x = train[-17],
                         y = train$class,
                         ntree = 10)
```

Figure 4.17: *Training Random Forest model in R*

Then, values related to the performance of the model and the confusion matrix were obtained, results are as in Figure 4.18.

```
## [1] "Train"             "0.994505494505495"

## [1] "Test"              "0.948717948717949"

##           actual
## predicted  Negative Positive
##   Negative       56        5
##   Positive        3       92
```

Figure 4.18: *Accuracy Score and Confusion Matrix for Random Forest Model*

According to Figure 4.18, the accuracy values obtained for train and test are 0.99 and 0.95, respectively. The model has learned the train set quite well. In addition, its

success on the test set is also very high. At the same time, only 8 misclassifications were made according to the confusion matrix. In this case, we can say that the model classified class variable well.

4.4. Obtaining the Comparison Chart

In this section, the accuracy, error rate, sensitivity and specificity scores of the models are obtained. Here, accuracy represents how accurately the model predicts the diagnosis of the disease. Error rate represents the extent to which the model erroneously predicts the diagnosis of the disease. Sensitivity is the ability of a test to correctly identify patients with a disease and specificity is the ability of a test to correctly identify people without the disease. Here, the results of the "performance" function that have been applied for each model before are presented. Accuracy, error rate, sensitivity and specificity scores and the elements of the confusion matrix of each model are included. The relevant output is shown in Figure 4.19.

```
##                     TP FN TN FP  Accuracy Error.Rate Sensitivity Specificity
## Random Forest       92  5 56  3 0.9487179 0.05128205   0.9484536   0.9491525
## Logistic Regression 92  5 54  5 0.9358974 0.06410256   0.9484536   0.9152542
## Support V. Mach.    91  6 53  6 0.9230769 0.07692308   0.9381443   0.8983051
## K-Nearest Neighbors 78  2 57 19 0.8653846 0.13461538   0.9750000   0.7500000
## Decision Trees      84 13 50  9 0.8589744 0.14102564   0.8659794   0.8474576
## Naive Bayes         77 20 55  4 0.8461538 0.15384615   0.7938144   0.9322034
```

Figure 4.19: *Performance Scores For all Models*

5. CONCLUSION

The applications of machine learning algorithms in the healthcare field are increasing day by day. In the health sector, machine learning-based decision support systems are used to obtain more accurate results in the diagnosis and treatments made by doctors, to prevent human-induced errors and to help the doctor's decision. Diabetes, which is one of the most common diseases of today's world, significantly increases the patient's chance of getting rid of this disease with an early and correct diagnosis. In this study, data obtained from a hospital were classified with 6 different models using more than one classifier method and the success rates of the results were compiled in Table 5.1.

Model	TP	FN	TN	FP	Accuracy	Error Rate	Sensitivity	Specificity
Random Forest	92	5	56	3	0.9487179	0.05128205	0.9484536	0.9491525
Logistic R.	92	5	54	5	0.9358974	0.06410256	0.9484536	0.9152542
SVM	91	6	53	6	0.9230769	0.07692308	0.9381443	0.8983051
K-NN	78	2	57	19	0.8653846	0.13461538	0.9750000	0.7500000
Decision Trees	84	13	50	9	0.8589744	0.14102564	0.8659794	0.8474576
Naïve Bayes	77	20	55	4	0.8461538	0.15384615	0.7938144	0.9322034

Table 5.1: *Performance Scores For all Models*

According to the Accuracy values in the Table 5.1, the success rate of Random Forest in diagnosing the disease is approximately 95%, Logistic Regression is approximately 94% and Support Vector Machines is approximately 92%. In the other 3 models, this value is below 90% and it is seen that it performs worse than the first three models in the table. The error rates of these three models, approximately 0.05, 0.06 and 0.08, respectively. The conclusion can be obtaining here is that Random Forest has a 5% chance, Logistic Regression 6% chance, and Support Vector Machines 8% chance of misdiagnosing. Also, the false negative numbers of these three models are 5, 5, and 6, respectively. In addition, although the number of false negative predictions of the K-NN model is quite low, the false positive value of 19 makes it difficult to choose this model. The number of false positive predictions is 3, 5, and 6 for the first three models in the table, respectively. Again, these values are better than the other three models. Like the situation in K-NN is also seen in the Naive Bayes model. Although the number of false positive predictions is as low as 4, the number of false negative predictions is quite high, such as 20, so we do not prefer this model over other models. Sensitivity and Specificity values are also related to these values. According to the sensitivity ratio, 4 models with less false-negative observations shine out. These models are Random Forest, Logistic Regression, Support Vector Machines and K-NN. The sensitivity rate of each of these models is greater than 90%, and in this case, the probability of diagnosing diabetes in participants who do not have diabetes is low. On the other hand, Random Forest, Logistic Regression, Support Vector Machines and Naive Bayes false-positive observations come first according to the Specificity ratios as they are low. The specificity of each of these models is approximately 90% or greater, and in this case participants with diabetes are unlikely to be diagnosed with diabetes. Although the sensitivity ratio of the K-NN model is high, the specificity value is low, similarly, the sensitivity ratio of the Naive Bayes model is high while the sensitivity ratio is low, so these two models should not be preferred.

As a result, Random Forest, Logistic Regression and Support Vector Machine models were more successful in diagnosing diabetes on this dataset than other models. According to the accuracy values, the Random Forest model is the best among the 6 models. Each misclassification is very important since the data about diagnosis of a

disease. In order to eliminate these, these models can be developed or models that can classify more successfully can be trained.

6.REFERENCES

1. ARSLAN, H., KAYNAR, O., YÜKSEK, A.G. &; GÜN, O. “Kurumsal Kolektif Süreçler için E-Posta İletilerinden Görev Keşfi ve Gerçek Zamanlı Görev Yönetim Sisteminin Geliştirilmesi,” Global Journal on Technology, Kasım 2015.
2. UYSAL M., DOĞRUSÖZ Ö., Güran, Aysun, "Destek Vektör Makineleri Parametre Optimizasyonunun Duygu Analizi Zerindeki Etkisi," DEÜ Mühendislik Fakültesi Mühendislik Bilimleri Dergisi 48, 2014, pp. 87- 88.
3. BERMEJO P., GÁMEZ J. A., PUERTA J. M. (2011): "Improving the performance of Naive Bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets", Expert Systems with Applications, Volume 38, No. 3, p.2072–2080.
4. ZHANG, S., LI, X., ZONG, M., ZHU, X., & WANG, R. (2018). Efficient knn classification with different numbers of nearest neighbors. *IEEE transactions on neural networks and learning systems*, *29*(5), 1774-1785.
5. BREIMAN L. (2001): "Random Forests", Machine Learning, Cilt 45, No. 1, s.5–32.
6. MATHER P. M. (2005): "Computer Processing of Remotely-Sensed Images".
7. PAL M. (2005): "Random Forest classifier for remote sensing classification", International Journal of Remote Sensing, Cilt.26, No.1, s.217–222.
8. ÇÖLKESEN I. (2009): " Uzaktan Algılamada İLeri Sınıflandırma Tekniklerinin Karşılaştırılması ve Analizi", Gebze Yüksek Teknoloji Entitüsü, Jeodezi ve Fotogrametri Mühendisliği, Y.Lisans Tezi.
9. AKMAN M., GENÇ Y., ANKARALI H. (2011): "Random Forests Yöntemi ve Sağlık Alanında Bir Uygulama", Türkiye Klinikleri Biyoistatistik Dergisi, Cilt 3, No. 1, s.36–48.
10. HAN J., KAMBER M. (2006): "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers.
11. GERSHENSON C., "Artificial Neural Networks for Beginners", http://arxiv.org/ftp/cs /papers/0308/0308031.pdf
12. ALPAYDIN E. (2010): "Introduction to Machine Learning", The MIT Press Cambridge, Massachusetts London, England.
13. TEBELSKIS J. (1995): "Speech Recognition using Neural Networks", Carnegie Mellon University Pittsburgh, Pennsylvania.
14. KESKİN, A. K. (2018). Makine Öğrenmesi Sınıflandırma Algoritmalarının İncelenmesi. Yüksek Lisans Tezi, Sinop Universitesi, Sinop.
15. Data Source: ISLAM, MM Faniqul, et al. 'Likelihood prediction of diabetes at early stage using data mining techniques.' Computer Vision and Machine Intelligence in Medical Image Analysis. Springer, Singapore, 2020. 113-125. *https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.#*

7.APPENDICES

```
library(readxl)
#.....data preparation.....
data <- read_xlsx("C:/Users/tfn/Documents/diabetes_data_upload.xlsx") #data a
saigned to "data" variable
#redefining the data type process...
data$Gender <- as.factor(data$Gender)
data$Polyuria <- as.factor(data$Polyuria)
data$Polydipsia <- as.factor(data$Polydipsia)
data$`sudden weight loss` <- as.factor(data$`sudden weight loss`)
data$weakness <- as.factor(data$weakness)
data$Polyphagia <- as.factor(data$Polyphagia)
data$`Genital thrush` <- as.factor(data$`Genital thrush`)
data$`visual blurring` <- as.factor(data$`visual blurring`)
data$Itching <- as.factor(data$Itching)
data$Irritability <- as.factor(data$Irritability)
data$`delayed healing` <- as.factor(data$`delayed healing`)
data$`partial paresis` <- as.factor(data$`partial paresis`)
data$`muscle stiffness` <- as.factor(data$`muscle stiffness`)
data$Alopecia <- as.factor(data$Alopecia)
data$Obesity <- as.factor(data$Obesity)
data$class <- as.factor(data$class)
```

```
#end of the process
#missing value check
anyNA(data)

## [1] FALSE
```

```
#data summary/descriptive statistics
summary(data)

##       Age       Gender    Polyuria  Polydipsia sudden weight loss weakness
##  Min.   :16.00   Female:192   No :262   No :287    No :303        No :215
##  1st Qu.:39.00   Male  :328   Yes:258   Yes:233    Yes:217        Yes:305
##  Median :47.50
##  Mean   :48.03
##  3rd Qu.:57.00
##  Max.   :90.00
##  Polyphagia Genital thrush visual blurring Itching   Irritability
##  No :283    No :404        No :287         No :267   No :394
##  Yes:237    Yes:116        Yes:233         Yes:253   Yes:126
##
##  delayed healing partial paresis muscle stiffness Alopecia  Obesity
##  No :281         No :296         No :325          No :341   No :432
##  Yes:239         Yes:224         Yes:195          Yes:179   Yes: 88
##
##       class
##  Negative:200
##  Positive:320
```

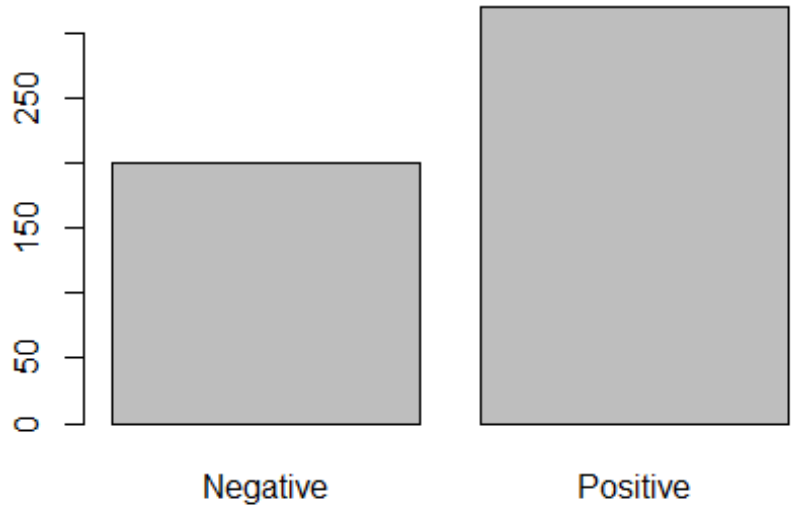
```r
str(data)
```

```
## tibble [520 x 17] (S3: tbl_df/tbl/data.frame)
##  $ Age               : num [1:520] 40 58 41 45 60 55 57 66 67 70 ...
##  $ Gender            : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Polyuria          : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 2 2 2 2 1 ...
##  $ Polydipsia        : Factor w/ 2 levels "No","Yes": 2 1 1 1 2 2 2 2 2 2 ...
##  $ sudden weight loss: Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 2 1 2 ...
##  $ weakness          : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Polyphagia        : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 2 1 2 2 ...
##  $ Genital thrush    : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 2 1 2 1 ...
##  $ visual blurring   : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 2 1 2 1 2 ...
##  $ Itching           : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 1 2 2 2 ...
##  $ Irritability      : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 2 2 2 ...
##  $ delayed healing   : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 2 1 1 1 ...
##  $ partial paresis   : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 1 2 2 2 1 ...
##  $ muscle stiffness  : Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 1 2 2 1 ...
##  $ Alopecia          : Factor w/ 2 levels "No","Yes": 2 2 2 1 2 2 1 1 1 2 ...
##  $ Obesity           : Factor w/ 2 levels "No","Yes": 2 1 1 1 2 2 1 1 2 1 ...
##  $ class             : Factor w/ 2 levels "Negative","Positive": 2 2 2 2 2 2 2 2 2
2 ...
```

```r
head(data)
```

```
## # A tibble: 6 x 17
##     Age Gender Polyuria Polydipsia `sudden weight loss` weakness Po
lyphagia
##   <dbl> <fct>  <fct>    <fct>      <fct>                <fct>    <f
ct>
## 1    40 Male   No       Yes        No                   Yes      No
## 2    58 Male   No       No         No                   Yes      No
## 3    41 Male   Yes      No         No                   Yes      Ye
s
## 4    45 Male   No       No         Yes                  Yes      Ye
s
## 5    60 Male   Yes      Yes        Yes                  Yes      Ye
s
## 6    55 Male   Yes      Yes        No                   Yes      Ye
s
## # ... with 10 more variables: Genital thrush <fct>, visual blurring
<fct>,
## #   Itching <fct>, Irritability <fct>, delayed healing <fct>,
## #   partial paresis <fct>, muscle stiffness <fct>, Alopecia <fct>,
## #   Obesity <fct>, class <fct>
```

```r
plot(data$class)
```



FUNCTION FOR PERFORMANCE CALCULATION

```r
performance <- function(cm){
  TN <- cm[1,1]
  TP <- cm[2,2]
  FN <- cm[1,2]
  FP <- cm[2,1]

  accuracy = sum(diag(cm))/sum(cm)
  errorrate = 1-accuracy
  sensitivity = TP /(TP + FN)
  specificity = TN / (TN + FP)
  return(data.frame(TP,FN,TN,FP,accuracy,errorrate,sensitivity,specificity))
}
```

DATA SPLITTING

```
#...Data Splitting...
set.seed(2380) #for reproducibility
data_split <- sample(nrow(data), nrow(data) * 0.7) #Spliting data into two pa
rts
train <- data[data_split,]
test <- data[-data_split,]
nrow(train);nrow(test)

## [1] 364

## [1] 156

summary(train$class);summary(test$class)

## Negative Positive
##      141      223

## Negative Positive
##       59       97
```

LOGISTIC REGRESSION App.

```
#...Training Log. Reg. Model...
model_LRM <-  glm(class~. ,data = train, family = "binomial")

#...Performance of the Model...
#For Train
predicted_probs_train <- predict(model_LRM,type = "response")
predicted_class_train <- ifelse(predicted_probs_train > 0.5, "Positive", "Neg
ative")
print(c("Train",mean(predicted_class_train == train$class)))

## [1] "Train"             "0.950549450549451"

#For Test
predicted_probs_test <- predict(model_LRM, test, type = "response")
predicted_class_test <- ifelse(predicted_probs_test > 0.5, "Positive", "Negat
ive")
print(c("Test",mean(predicted_class_test == test$class)))

## [1] "Test"              "0.935897435897436"

#Confusion Matrix
cm_LR <- table(predicted=predicted_class_test,actual=test$class)
performace_LR <- performance(cm_LR)
cm_LR

##           actual
## predicted  Negative Positive
##   Negative       54        5
##   Positive        5       92
```

NAIVE BAYES App.

```
library(e1071)

## Warning: package 'e1071' was built under R version 4.0.5

#...Training Naive Bayes Model...
model_NB <- naiveBayes(x=train[-17], y=train$class)

#...Performance of the Model...
#For Train
y_pred_TRAIN = predict(model_NB, newdata = train[-17])
print(c("Train",mean(y_pred_TRAIN == train$class)))

## [1] "Train"             "0.881868131868132"

#For Test
y_pred_TEST = predict(model_NB, newdata = test[-17])
print(c("Test",mean(y_pred_TEST == test$class)))

## [1] "Test"              "0.846153846153846"

#Confusion Matrix
cm_NB <- table(predicted=y_pred_TEST,actual=test$class)
performance_NB <- performance(cm_NB)
cm_NB

##           actual
## predicted  Negative Positive
##   Negative       55       20
##   Positive        4       77
```

DECISION TREE

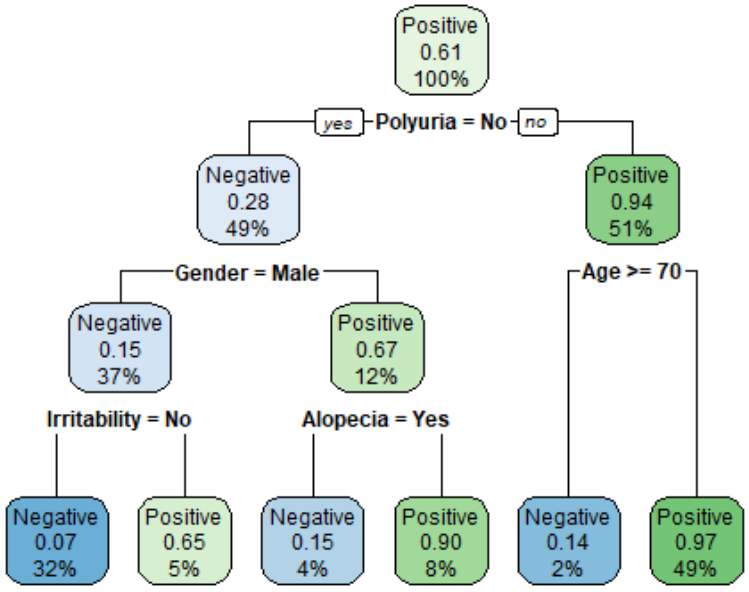
```
#...Training Decision Tree Model...
library(rpart)

## Warning: package 'rpart' was built under R version 4.0.5

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.5

model_DT <- rpart(class ~., method = "class", data = train)
rpart.plot(model_DT)
```




```
#...Performance of the Model...
#FOR TRAIN
pred_labels_dt_train <- predict(model_DT, train, type = "class")
conf_mat_dt_train <- table(pred_labels_dt_train, train$class)
print(c("Train:",sum(diag(conf_mat_dt_train))/sum(conf_mat_dt_train)))

## [1] "Train:"            "0.925824175824176"

#FOR TEST
pred_labels_dt_test <- predict(model_DT, test, type = "class")
cm_DT <- table(pred_labels_dt_test, test$class)
print(c("Test:",sum(diag(cm_DT))/sum(cm_DT)))

## [1] "Test:"             "0.858974358974359"

#Confusion Matrix
performance_DT <- performance(cm_DT)
cm_DT

##
## pred_labels_dt_test Negative Positive
##            Negative       50       13
##            Positive        9       84
```

K-NEAREST NEIGHBORS

```
library(class)
#...Training K-Nearest Neighbors Model...
trainKNN <- data.matrix(train)
trainKNN <- data.frame(trainKNN)
testKNN <- data.matrix(test)
testKNN <- data.frame(testKNN)
model_KNN <- knn(train=trainKNN, test=testKNN, cl=trainKNN[,17], k=5)

#...Performance of the Model...
#Confusion Matrix
cm_KNN <- table(testKNN[,17],model_KNN)
performance_KNN <- performance(cm_KNN)
print(c("Test", sum(diag(cm_KNN)/sum(cm_KNN))))
```

```
## [1] "Test"              "0.865384615384615"
```

```
cm_KNN
```

```
##    model_KNN
##       1  2
##   1 57  2
##   2 19 78
```

SUPPORT VECTOR MACHINES

```
library(e1071)
#...Training Supp. Vec. Mach. Model...
model_SVM <- svm(formula=class~., data=train, type="C-classification", kernel
= "linear")

#...Performance of the Model...
#For Train
pred_svm_TRAIN = predict(model_SVM, newdata = train[-17])
print(c("Train",mean(pred_svm_TRAIN == train$class)))
```

```
## [1] "Train"             "0.936813186813187"
```

```
#For Test
pred_svm_TEST = predict(model_SVM, newdata = test[-17])
print(c("Test",mean(pred_svm_TEST == test$class)))
```

```
## [1] "Test"              "0.923076923076923"
```

```
#Confusion Matrix
cm_SVM <- table(predicted=pred_svm_TEST,actual=test$class)
performace_SVM <- performance(cm_SVM)
cm_SVM
```

```
##           actual
## predicted  Negative Positive
##   Negative       53        6
##   Positive        6       91
```

RANDOM FOREST

```
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

#...Training Random Forest Model...
model_RF <- randomForest(x = train[-17],
                         y = train$class,
                         ntree = 10)

#...Performance of the Model..
#For Train
pred_train_RF <- predict(model_RF, train[,-17])
print(c("Train",mean(pred_train_RF == train$class)))

## [1] "Train"             "0.994505494505495"

#For Test
pred_test_RF <- predict(model_RF, newdata = test[-17])
print(c("Test",mean(pred_test_RF == test$class)))

## [1] "Test"              "0.948717948717949"

#Confusion Matrix
cm_RF <- table(predicted=pred_test_RF,actual=test$class)
performance_RF <- performance(cm_RF)
cm_RF

##           actual
## predicted  Negative Positive
##   Negative       56        5
##   Positive        3       92
```

```r
comp <- data.frame(
            "TP" = c(performace_LR$TP,performace_SVM$TP,
                           performance_DT$TP,performance_KNN$TP,
                           performance_NB$TP,performance_RF$TP),
            "FN" = c(performace_LR$FN,performace_SVM$FN,
                           performance_DT$FN,performance_KNN$FN,
                           performance_NB$FN,performance_RF$FN),
            "TN" = c(performace_LR$TN,performace_SVM$TN,
                           performance_DT$TN,performance_KNN$TN,
                           performance_NB$TN,performance_RF$TN),
            "FP" = c(performace_LR$FP,performace_SVM$FP,
                           performance_DT$FP,performance_KNN$FP,
                           performance_NB$FP,performance_RF$FP),
            "Accuracy" = c(performace_LR$accuracy,performace_SVM$accuracy,
                           performance_DT$accuracy,performance_KNN$accuracy,
                           performance_NB$accuracy,performance_RF$accuracy),
            "Error Rate" = c(performace_LR$errorrate,performace_SVM$errorrate,
                           performance_DT$errorrate,performance_KNN$errorrate,
                           performance_NB$errorrate,performance_RF$errorrate),
            "Sensitivity"=c(performace_LR$sensitivity,performace_SVM$sensitivi
ty,
                           performance_DT$sensitivity,performance_KNN$sensitiv
ity,
                           performance_NB$sensitivity,performance_RF$sensitivi
ty),
            "Specificity"=c(performace_LR$specificity,performace_SVM$specifici
ty,
                           performance_DT$specificity,performance_KNN$specific
ity,
                           performance_NB$specificity,performance_RF$specifici
ty),
            row.names = c("Logistic Regression", "Support V. Mach.", "Decision
Trees",
                          "K-Nearest Neighbors", "Naive Bayes", "Random Forest
"))
comp[order(comp$Accuracy,decreasing = T),]
```

```
##                     TP FN TN FP  Accuracy Error.Rate Sensitivity Specifici
ty
## Random Forest       92  5 56  3 0.9487179 0.05128205   0.9484536   0.94915
25
## Logistic Regression 92  5 54  5 0.9358974 0.06410256   0.9484536   0.91525
42
## Support V. Mach.    91  6 53  6 0.9230769 0.07692308   0.9381443   0.89830
51
## K-Nearest Neighbors 78  2 57 19 0.8653846 0.13461538   0.9750000   0.75000
00
## Decision Trees      84 13 50  9 0.8589744 0.14102564   0.8659794   0.84745
76
## Naive Bayes         77 20 55  4 0.8461538 0.15384615   0.7938144   0.93220
34
```