A PROJECT REPORT

ON

LIVER DISEASE PREDICTION USING MACHINE LEARNING

Submitted to

Osmania University
in partial fulfilment of the requirements for the award of

MASTER OF SCIENCE
IN
STATISTICS

DEPARTMENT OF STATISTICS UNIVERSITY
COLLEGE OF SCIENCE OSMANIA UNIVERSITY
HYDERABAD – INDIA

By

K. JAMES ALEXANDER	**Roll No: 1007-17-508-005**
J. HARSHA CHANDANA	**Roll No: 1007-17-508-004**
B. JYOTHI	**Roll No: 1007-17-508-006**
N. SANDYA RANI	**Roll No: 1007-17-508-001**
S. SAI PRIYANKA	**Roll No: 1007-17-508-002**
SAMREEN	**Roll No: 1007-17-508-010**
P. NAGARAJU	**Roll No: 1007-17-508-008**

Under the Supervision of
Ms. T. SANDHYA
2018

A PROJECT REPORT

ON

LIVER DISEASE PREDICTION USING MACHINE LEARNING

Submitted to

Osmania University
in partial fulfillment of the requirements for the award of

MASTER OF SCIENCE
IN
STATISTICS

DEPARTMENT OF STATISTICS UNIVERSITY
COLLEGE OF SCIENCE OSMANIA UNIVERSITY
HYDERABAD – INDIA

By

K. JAMES ALEXANDER	**Roll No: 1007-17-508-005**
J. HARSHA CHANDANA	**Roll No: 1007-17-508-004**
B. JYOTHI	**Roll No: 1007-17-508-006**
N. SANDYA RANI	**Roll No: 1007-17-508-001**
S. SAI PRIYANKA	**Roll No: 1007-17-508-002**
SAMREEN	**Roll No: 1007-17-508-010**
P. NAGARAJU	**Roll No: 1007-17-508-008**

Under the Supervision of
Ms. T. SANDHYA

2018

CERTIFICATE

This is to certify that

Mr. K. JAMES ALEXANDER	**Roll No: 1007-17-508-005**
Ms. J. HARSHA CHANDANA	**Roll No: 1007-17-508-004**
Ms. B. JYOTHI	**Roll No: 1007-17-508-006**
Ms. N. SANDYA RANI	**Roll No: 1007-17-508-001**
Ms. S. SAI PRIYANKA	**Roll No: 1007-17-508-002**
Ms .SAMREEN	**Roll No: 1007-17-508-010**
Mr. P. NAGARAJU	**Roll No: 1007-17-508-008**

have submitted the project titled in partial fulfilment for the degree of Master of Science in Statistics.

Head

Department of Statistics

Internal Examiner External Examiner

DECLARATION

The research presented in this project has been carried out in the **Department of Statistics, Osmania University, Hyderabad.** The work is original has not been submitted so far, in part or full, for any other degree of diploma of any university.

K. JAMES ALEXANDER

J. HARSHA CHANDANA

B. JYOTHI

N. SANDHYA RANI

S. SAI PRIYANKA

SAMREEN

P. NAGARAJU

Department of Statistics

Osmania University

Hyderabad – 500007, T.S.

INDIA

ACKNOWLEDGEMENT

I deem it a great pleasure to express my deep sense of gratitude and indebtedness to my research supervisor **Ms. T. SANDHYA,** Statistics department, University College of Science, Osmania University for her valuable guidance, and enlightening discussions throughout the progress of my project work.

I also express my sincere and heartfelt thanks to **Prof. C. JAYALAKSHMI, Head** of Department, Department of Statistics, Osmania University for providing the necessary support and facilities in the department for completion of this work successfully.

It is indeed with great pleasure I record my thanks to **Prof. G. JAYASREE,** Chairperson, Board of Studies , Department of Statistics , Osmania University for having provided with all the facilities to carry out our work.

I thank **Prof. N. Ch. BHATRACHARYULU**, **Prof. K. VANI, Prof. S.A. JYOTHI RANI, Prof. G. SIRISHA, Ms. J.L. PADMA SHREE,** for their encouragement and constant help during the research.

I would like to express my deepest gratitude to **Mr. M. VENUGOPALA RAO** and **Mr. BALA KARTHIK** for their advice, guidance and involvement at various stages of this work, I would also like to thank them for their understanding and constant encouragement throughout this project.

I thank all Non-Teaching members of the Department of Statistics, who helped me during my thesis work.

I am thankful to the Osmania University for permitting me to carry out this work.

CONTENTS

	Page No.
1. INTRODUCTION AND SCOPE OF THE PROBLEM	01 -04
1.1. Scope of the Problem	02
1.2. Data Description	02-03
1.3. Review of chapters	04
2.REVIEW OF MACHINE LEARNING TECHINIQUES	05-22
2.0 Need of machine learning	06
2.1 Machine learning	06-08
2.1.1 Business understanding.	07
2.1.2 Data understanding.	07
2.1.3 Data preparation.	07
2.1.4 Modelling.	07-08
2.1.5 Evaluation.	08
2.1.6 Deployment.	08
2.2 Types of machine learning	08-11
2.2.1 Supervised learning.	9
2.2.2 Unsupervised learning.	10
2.2.3 Reinforcement learning.	11
2.3 Choosing the algorithm	12-15
2.3.1 Types of Regression algorithm.	12-13
2.3.2 Types of Classification algorithm.	13-14
2.3.3 Types of Un supervised algorithm.	14-15
2.4 Choosing and Comparing models through Pipelines.	15-17
2.4.1 Model validation .	16-17
2.5 Model diagnosis with overfitting and under fitting.	17-20
2.5.1 Bias and variance.	17-19
2.5.2 Model performance matrix.	19-21
2.6 overall process of machine learning.	22
3. Machine learning at Work	23-43

4. Summary 44-45
5. Appendix 46-80
R-code 46-53
Data set 54-79
6. Bibliography 80

CHAPTER 1

SCOPE OF THE PROBLEM

1.1 SCOPE OF THE PROBLEM

The problem is related to classifying a person whether he has liver disease or not, using the various diagnostic values of liver functional components in blood.

Source:

The dataset was downloaded from,
UCI ML Repository : Lichman, M.(2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA:University of California, School of Information and Computer Science.

Broadly the objective of the problem are:

This dataset is used to evaluate prediction algorithms in an effort to reduce burden on doctors.

- The attributes that we use here besides age,gender are proteins and liver enzymes whose level controls the condition of liver.(Abnormally high or low level of presence of these enzymes or proteins indicates liver damage).
- Based on the attributes we need to perform algorithms to come up with an appropriate classification model for classifying whether a patient has liver disease or not.
- When a new data is given, we use the model obtained andpredict whether the person needs to be diagnosed or not.

1.2 DATA DESCRIPTION:

The data is extracted from UCI repository, data has 1169 observations and 11 attributes. A brief description of the variables in the dataset:

- **age :** Integer value of age of patient (*Any patient whose age exceeds 90 is given as 90 years old*).
- **gender** : Female patient represented by 1 and male patient represented by 2.
- **total_brubin:** The amount of direct and indirect bilirubin in blood, measured in milligrams per deciliter (*Bilirubin isa substance made when the body breaks down old RBC*).
- **direct_brubin:** The amount of direct(*conjugated*) bilirubin in blood, measured in milligrams per deciliter.
- **alk_phos:** Amount of Alkaline Phosphotase(*It is a enzyme that breaks down proteins*) in blood, measured in International Units per liter.
- **alam_amin:** Amount of Alamine Aminotransferase (*It is an enzyme which helps to digest food*) in blood, measured in International Units per liter.
- **asp_amin:** Amount of Aspartate Aminotransferase (*It is a pyridoxal phosphate-dependent transaminase enzyme*) in blood, measured in International Units per liter.
- **total_proteins:** Amount of Total Proteins in blood, measured in grams per deciliter.
- **albumin:** Amount of Albumin (*It is the main protein produced in the liver whose main function is to regulate the oncotic pressure of blood*) in blood, measured in grams per deciliter.

- **alb_glob_ratio:** Albumin to Globulin Ratio.

Label:

ldisease : Field used to split the data into two sets

- 1 – represent person with liver disease.
- 2- represent person without liver disease.

1.3 Review of the Chapters

Chapter 2 gives the brief introduction about machine learning techniques like need of ML today, types of ML Algorithms and various models in each algorithm and what technique to use when and how to validate, Tune the ML algorithms and how to measure the performance of the ML model.

Section 3 describes the various results obtained for the problem.This section contains all the outputs generated through the ML algorithms applied on the data as well as validation and performance matrices.

Section 4 describes the summary and conclusions followed by Bibliography.

APPENDIX:

It describes the dataset and R code used.

CHAPTER 2

REVIEW OF MACHINE LEARNING PROCESS

REVIEW OF MACHINE LEARNING PROCESS

2.0 NEED OF MACHINE LEARNING

In this age of modern technology, there is one resource that we have in abundance a large amount of structured and unstructured data. In the second half of the twentieth century, machine learning evolved as a subfield of artificial intelligence that involved the development of self-learning algorithms to gain knowledge from that data in order to make predictions. Instead of requiring humans to manually derive rules and build models from analyzing large amounts of data, machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models, and make data-Profiven decisions. Not only is machine learning becoming increasingly important in computer science research but it also plays an ever greater role in our everyday life.

2.1 Machine Learning Process

The CRISP-DM (Cross-Industry Standard Process for Data Mining) Process was designed specifically for the data mining. However, it is flexible and thorough enough that it can be applied to any analytical project whether it is predictive analytics, data science, or Machine learning. The Process has the following six phases

- Business Understanding
- Data Understanding
- Data preparation
- Modeling
- Evaluation
- Deployment

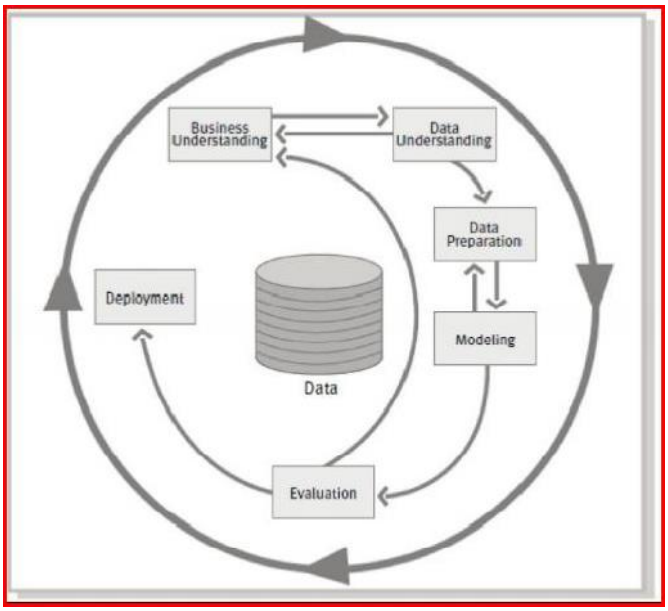


Fig. 2.1 CRISP-DM diagram

And, each phase has different steps covering important tasks which are mentioned below:

2.1.1 Business Understanding

It is very important step of the process in achieving the success. The purpose of this step is to identify the requirements of the business so that you can translate them into analytical objectives. It has the following tasks:

1) Identify the Business objective
2) Assess the situation
3) Determine the Analytical goals
4) Produce a project plan

2.1.2 Data Understanding

After enduring the all-important pain of the first step, you can now get your hands on the data. The task in this process consist the following

1) Collect the data
2) Describe the data
3) Explore the data
4) Verify the data Quality

2.1.3 Data Preparation

This step is relatively self-explanatory and in this step the goal is to get the data ready to input in the algorithms. This includes merging, feature engineering, and transformations. If imputation for missing values / outliers is needed then, it happens in this step. The key five tasks under this step are as follows:

1) Select the data
2) Clean the data
3) Construct the data
4) Integrate the data
5) Format the data

2.1.4 Modeling

Oddly, this process step includes the consideration that you already thought of and prepared for. In this, one will need at least a modicum of an idea about how they will be modeling. Remember, that this is flexible, iterative process and some strict linear flow chart such as an aircrew checklist.

Below are the tasks in this step:

1) Select a modeling technique
2) Generate a test design
3) Build a model
4) Assess a Model

Both cross validation of the model (using train/test or K fold validation) and model assessment which involves comparing the models with the chosen criterion (RMSE, Accuracy, ROC) will be performed under this phase.

2.1.5 Evaluation

In the evaluation process, the main goal is to confirm that the work that has been done and the model selected at this point meets the business objective. Ask yourself and others, have we achieved the definition of success? And, here are the tasks in this step:

1) Evaluate the results
2) Review the process
3) Determine the next steps

2.1.6 Deployment

If everything is done according to the plan up to this point, it might come down to flipping a switch and your model goes live. Here are the tasks in this step:

1) Deploying the plan
2) Monitoring and maintenance of the plan
3) Producing the final report

2.2 Types of Machine Learning

Broadly, the Machine Learning Algorithms are classified into 3 type

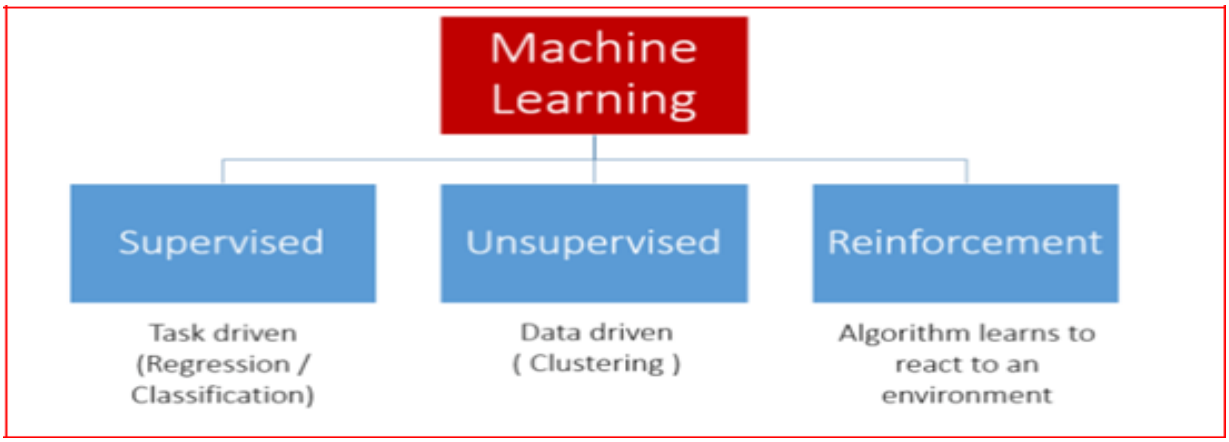


Fig. 2.2 Types of Machine Learning

2.2.1 Supervised Learning

This algorithm consists of a target / outcome / dependent variable which is to be predicted from a given set of predictors / independent variables. Using these set of variables, we generate a function that maps inputs to desired output. The training process continues until the model achieves a desired level of accuracy on the training data.

The process of Supervised Learning model is illustrated in the below picture:

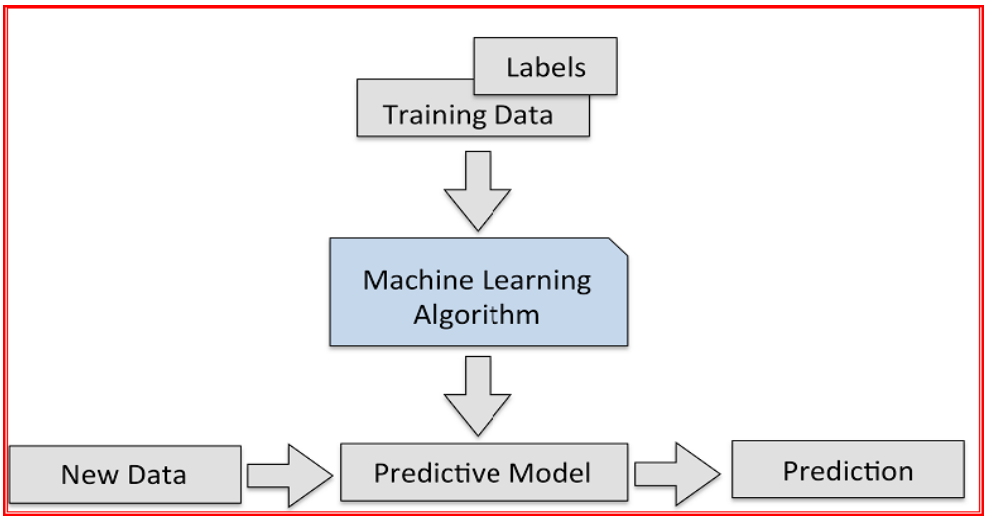


Fig. 2.2.1 Supervised Learning

Examples of Supervised Learning: Regression, Decision Tree, Random Forest, KNN, Logistic Regression,…etc

Classification

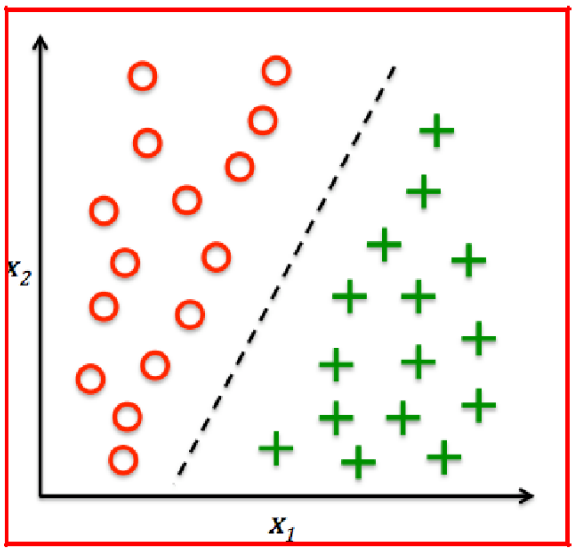


Fig.2.2.1 Classification

Regression

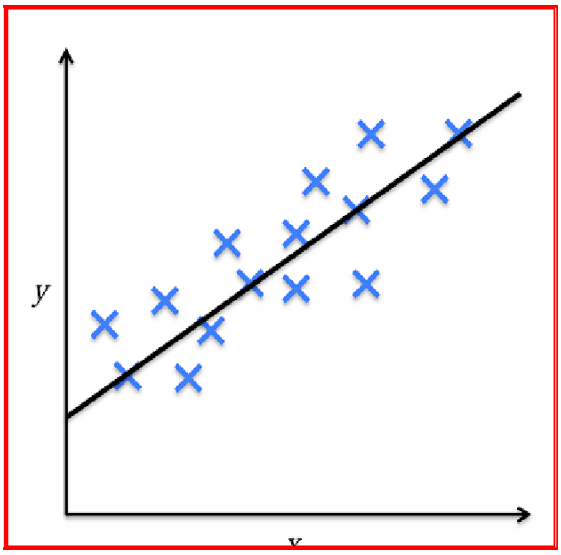


Fig.2.2.1 Regression

2.2.2 Unsupervised Learning

In this algorithm, we will not have any target or outcome variable to predict / estimate. It is used for clustering population into different groups, which is widely used for segmenting customers in different groups for specific intervention. (More of Exploratory Analysis)

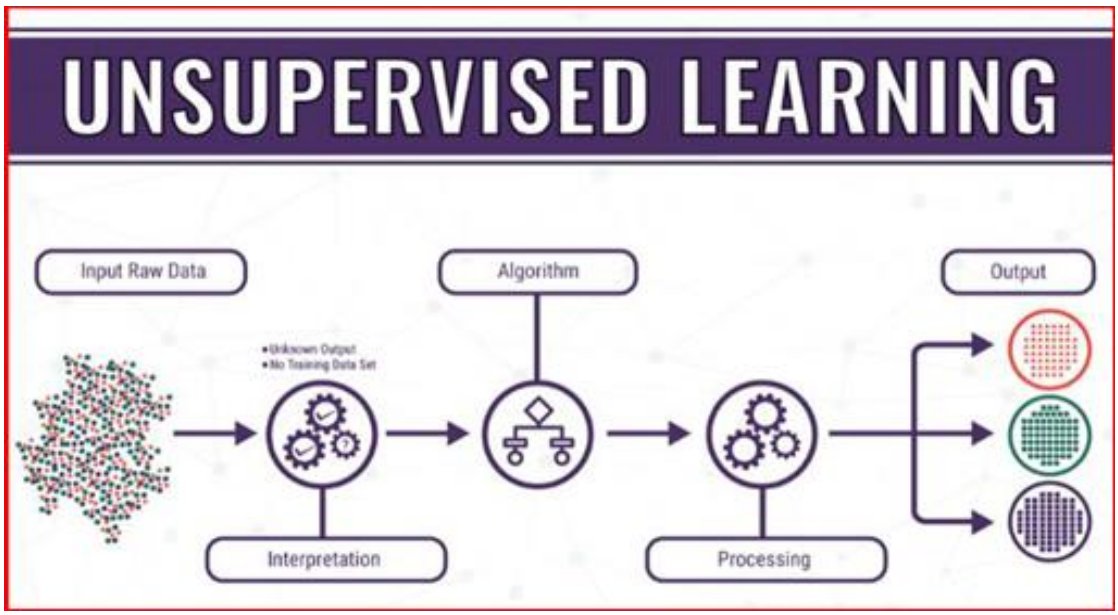


Fig.2.2.2 Unsupervised Learning

Examples of Unsupervised Learning: Data reduction techniques, Cluster Analysis, Market Basket Analysis,…etc

Cluster Analysis **Data Reduction Techniques**

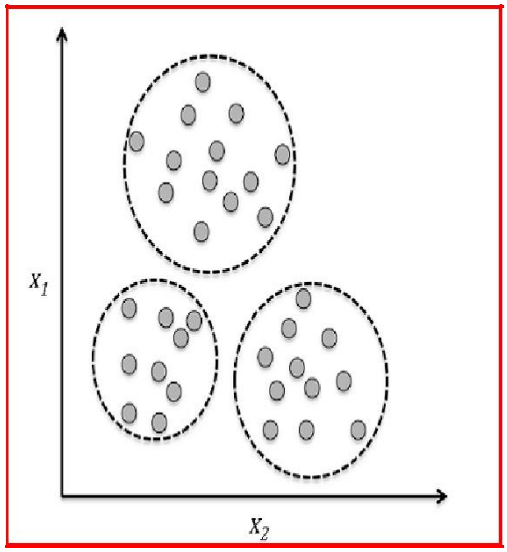


Fig.2.2.2 Cluster Analysis

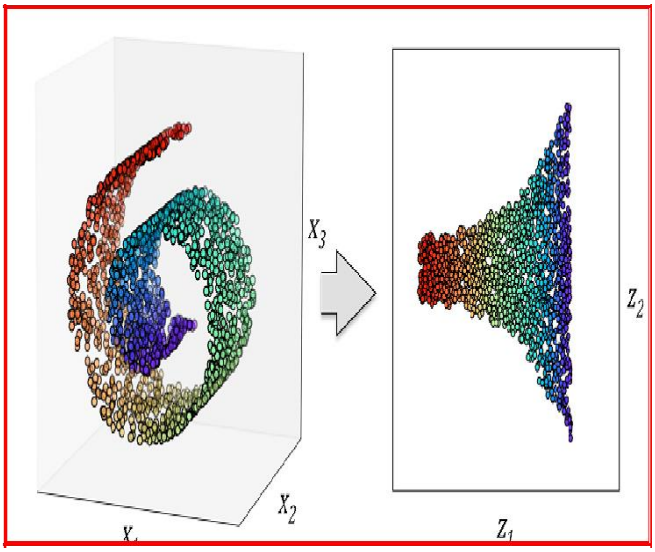


Fig. 2.2.2 Data Reduction Techniques

2.2.3 Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions.
The process of reinforcement learning is illustrated in the below picture:

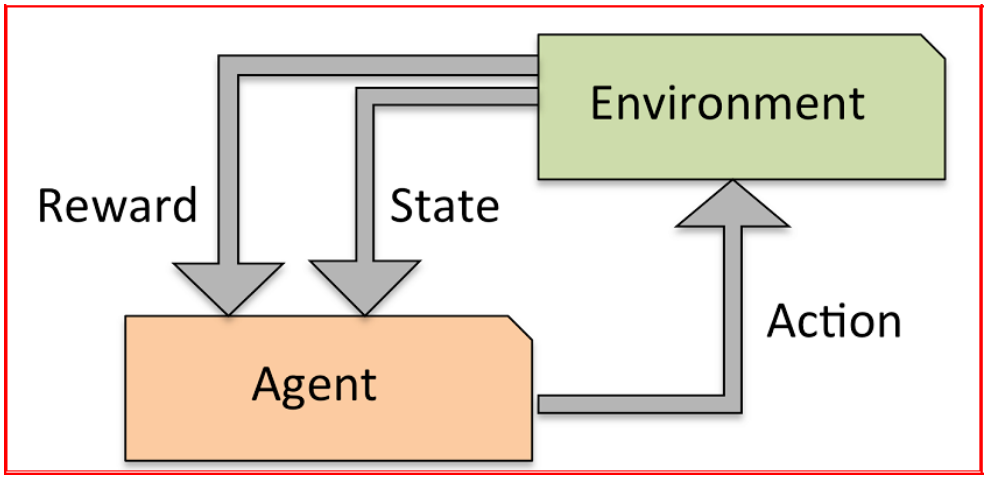


Fig.2.2.3 Reinforcement learning process

Examples of Reinforcement Learning: Markov Decision Process, Self-Profivingcars,...etc

2.3 Choosing the algorithm

Choosing the right algorithm will depend on the type of the problem we are solving and also depends on the scale of the dependent variable. In case of continuous target variable, we will use regression algorithms and in case of categorical target, we will use classification algorithms and for the model which doesn't have target variable, we will use either cluster analysis / data reduction techniques.

Below picture describes the process of choosing the right algorithm:

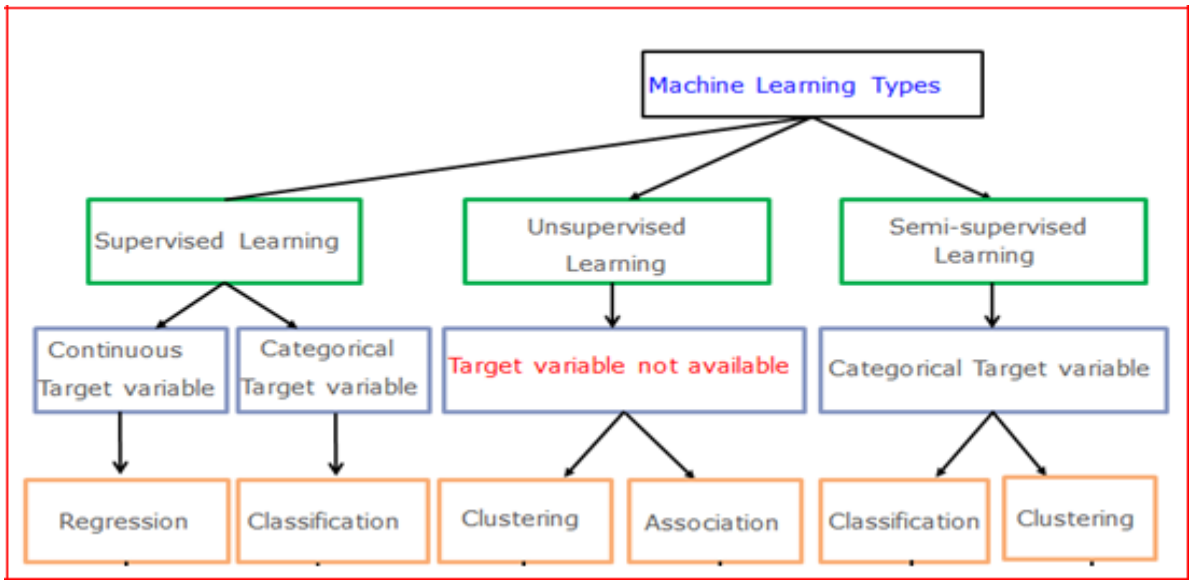


Fig.2.3 Choosing the algorithm

2.3.1 Types of Regression Algorithms

There are many Regression algorithms in machine learning, which will be used in different regression applications. Some of the main regression algorithms are as follows:

a) **Simple Linear Regression:-**In simple linear regression, we predict scores of the variable from the data of second variable. The variable we are forecasting is called the criterion variable and referred to as Y. The variable we are basing our predictions on is called the predictor variable and denoted as X.

b) **Multiple Linear Regression:-**Multiple linear regression is one of the algorithms of regression technique, and is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one dependent variable with two or more independent variables. The independent variables can be either continuous or categorical.

c) **Polynomial Regression:-**Polynomial regression is another form of regression in which the maximum power of the independent variable is more than 1.

In this regression technique, the best fit line is not a straight line instead it is in the form of a curve.

d) **Support Vector Machines:-**Support Vector Machines can be applied to regression problems as well as Classification. It contains all the features that characterizes maximum margin algorithm. Linear learning machine maps a non-linear function into high dimensional kernel-induced feature space. The system capacity will be controlled by parameters that do not depend on the dimensionality of feature space.

e) **Decision Tree Regression:-**Decision tree builds regression models in the form of a tree structure. It breaks down the data into smaller subsets and while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

f) **Random Forest Regression:-**Random Forest is also one of the algorithms used in regression technique. It is very a flexible, easy to use machine learning algorithm that produces, even without hyper -parameter tuning, a great result most of the time. It is also one of the most widely used algorithms because of its simplicity and the fact that it can used for both regression and classification tasks. The forest it builds is an ensemble of Decision Trees, most of the time trained with the "bagging" method.

Other than these we have regularized regression models like **Ridge, LASSO** and **Elastic Netregression** which are used to select the key parameters and these is also **Bayesian regression** which works with the Bayes theorem.

2.3.2 Types of Classification Algorithms

There are many Classification algorithms in machine Learning, which can be used for different classification applications. Some of the main classification algorithms are as follows:

a) **Logistic Regression/Classification**:-Logistic regression falls under the category of supervised learning; it measures the relationship between the dependent variable which is categorical with one or more than one independent variables by estimating probabilities using a logistic/sigmoid function. Logistic regression can generally be used when the dependent variable is Binary or Dichotomous. It means that the dependent variable can take only two possible values like "Yes or No", "Living or dead".

b) **K-Nearest Neighbors:-** k-NN algorithm is one of the most straight forward algorithms in classification, and it is one of the most used ML algorithms. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also use for regression-output is the value of the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

c) **Naive Bayes:-**Naive Bayes is a type of Classification technique based on Bayes theorem, with an assumption of independence among predictors. In simple terms, a

Naive Bayes classifier assumes that the presence of a Particular feature in a class is unrelated to the presence of any other function. Naive Bayes model is accessible to build and particularly useful for extensive datasets.

d) **Decision Tree Classification:-**Decision tree builds classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The first decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

e) **Support Vector Machines:-**A Support Vector Machine is a type of Classifier, in which a discriminative classifier is formally defined by a separating hyperplane. The algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space, this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

f) **Random Forest Classification:-**Random Forest is a supervised learning algorithm. It creates a forest and makes it somehow random. The forest it builds is an ensemble of Decision Trees, most of the times the decision tree algorithm trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. And Random Forest is also very powerful to find the variable importance in classification/ Regression problems.

2.3.3 Types of Unsupervised Learning

Clustering is the type of unsupervised learning in which an unlabelled data is used to profaw inferences. It is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the data points and group similar data points together and also to figure out which cluster should a new data point belong to.

Types of Clustering Algorithms:-There are many Clustering algorithms in machine learning, which can be used for different clustering applications. Some of the main clustering algorithms are as follows:

a) **Hierarchical Clustering:-**Hierarchical clustering is one of the algorithms of clustering technique, in which similar data is grouped in a cluster.

It is an algorithm that builds the hierarchy of clusters. This algorithm starts with all the data points assigned to a bunch of their own. Then, two nearest groups are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

It starts by assigning each data point to its bunch. Finds the closest pair using Euclidean distance and merges them into one cluster. This process is continued until all data points are clustered into a single cluster.

b) K–Means Clustering:-K-Means clustering is one of the algorithms of clustering technique, in which similar data is grouped into a cluster. K-means is an iterative algorithm that aims to find local maxima in each iteration. It starts with K as the input which is the desired number of clusters. Input k centroids in random locations in your space. Now, with the use of the Euclidean distance method, calculates the distance between data points and centroids, and assign data point to the cluster which is close to its centroid. Re calculate the cluster centroids as a mean of data points attached to it. Repeat until no further changes occur.

Types of Dimensionality Reduction Algorithms:-There are many dimensionality reduction algorithms in machine learning, which are applied for different dimensionality reduction applications. One of the main dimensionality reduction techniques is Principal Component Analysis (PCA) / Factor Analysis.

Principal Component Analysis (Factor Analysis):-Principal Component Analysis is one of the algorithms of Dimensionality reduction. In this technique, it transforms data into a new set of variables from input variables, which are the linear combination of real variables. These Specific new set of variables are known as principal components. As a result of the transformation, the first primary component will have the most significant possible variance, and each following component in has the highest possible variance under the constraint that it is orthogonal to the above components. Keeping only the best m < n components, reduces the data dimensionality while retaining most of the data information.

2.4 Choosing and comparing models through Pipelines

When you work on machine learning project, you often end up with multiple good models to choose from. Each model will have different performance characteristics. Using resampling methods like k-fold cross validation; you can get an estimate of how accurate each model may be on unseen data. You need to be able to use these estimates to choose one or two best models from the suite of models that you have created.

2.4.1 Model Validation

When you are building a predictive model, you need to evaluate the capability or generalization power of the model on unseen data. This is typically done by estimating accuracy using data that was not used to train the model, often referred as cross validation.

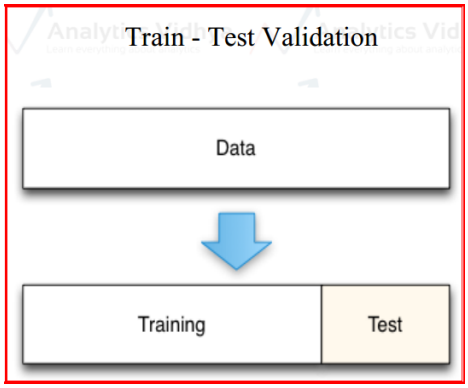


Fig. 2.4.1 Model Validation

A few common methods used for Cross Validation:

1) The Validation set Approach (Holdout Cross validation)

In this approach, we reserve large portion of dataset for training and rest remaining portion of the data for model validation. Ideally people will use 70-30 or 80-20 percentages for training and validation purpose respectively.

A major disadvantage of this approach is that, since we are training a model on a randomly chosen portion of the dataset, there is a huge possibility that we might miss-out on some interesting information about the data which, will lead to a higher bias.

2) K-fold cross validation

As there is never enough data to train your model, removing a part of it for validation may lead to a problem of under fitting. By reducing the training data, we risk losing important patterns/ trends in data set, which in turn increases error induced by bias. So, what we require is a method that provides ample data for training the model and also leaves ample data for validation. K Fold cross validation does exactly that.

In K Fold cross validation, the data is divided into k subsets. Now the holdout method is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model. As can be seen, every data point gets to be in a validation set exactly once, and gets to be in a training set k-1 times. This significantly reduces the bias as we are using most of the data for fitting and also significantly reduces variance as most of the data is also being used in validation set. Interchanging the training and test sets also adds to the effectiveness of this method. As a general rule and empirical evidence, K = 5 or 10 is preferred, but nothing's fixed and it can take any value.

Below are the steps for it:

Randomly split your entire dataset into k"folds"

- For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for k^{th} fold.
- Record the error you see on each of the predictions.
- Repeat this until each of the k-folds has served as the test set.
- The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model.

Below is the visualization of a k-fold validation when k=5.

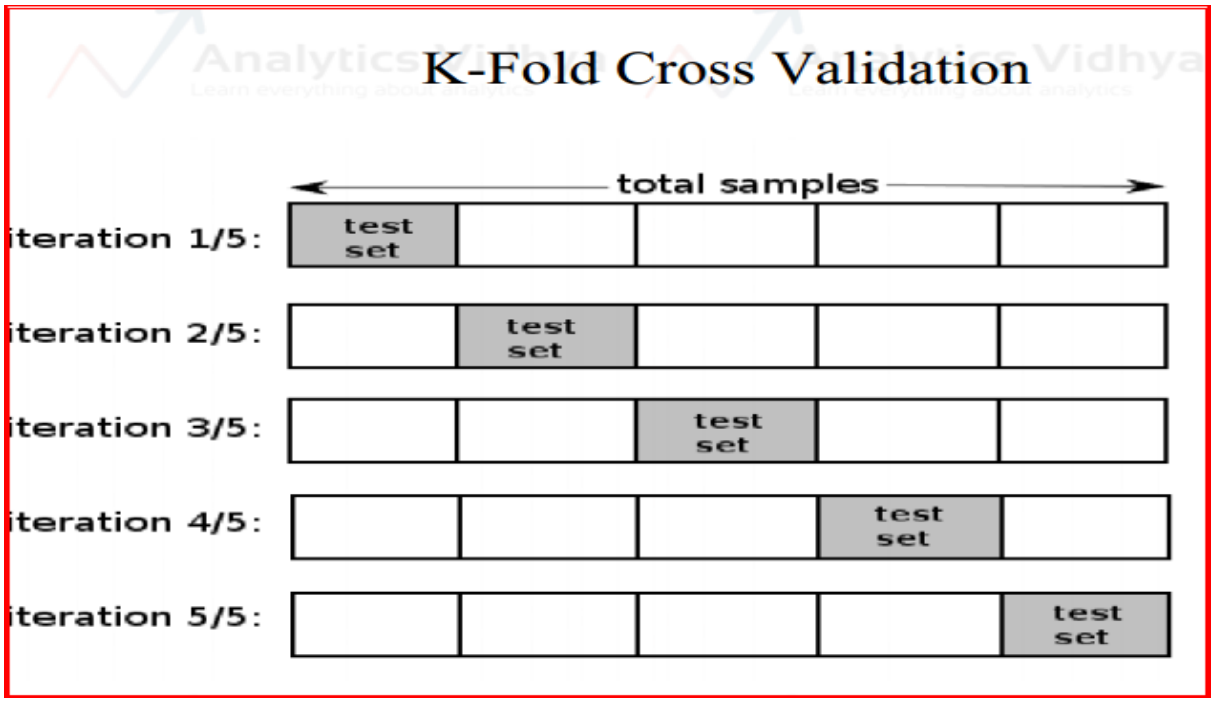


Fig.2.4.1 K- Fold Crosss Validation

How to choose K:

- Smaller dataset: 10-fold cross validation is better
- Moderate dataset: 5 or 6 fold cross validation works mostly
- Big dataset: Train – Val split for validation

Other than this, we have Leave one out cross validation (LOOCV), in which each record will be left over from the training and then, the same will be used for testing purpose. This process will be repeated across all the respondents.

2.5 Model Diagnosis with over fitting and under fitting

2.5.1 Bias and Variance

A fundamental problem with supervised learning is the bias variance trade-off. Ideally, a model should have two key characteristics

1) Sensitive enough to accurately capture the key patterns in the training dataset.
2) Generalized enough to work well on any unseen dataset.

Unfortunately, while trying to achieve the above-mentioned first point, there is an ample chance of over-fitting to noisy or unrepresentative training data points leading to a failure of generalizing the model. On the other hand, trying to generalize a model may result in failing to capture important regularities.

If model accuracy is low on a training dataset as well as test dataset, the model is said to be under-fitting or that the model has high bias. The **Bias** refers to the simplifying assumptions made by the algorithm to make the problem easier to solve. To solve an under-fitting issue or to reduce bias, try including more meaningful features and try to increase the model complexity by trying higher-order interactions

The **Variance** refers to sensitivity of a model changes to the training data. A model is giving high accuracy on a training dataset, however on a test dataset the accuracy proofs prophetically then, the model is said to be over-fitting or a model that has high variance.

To solve the over-fitting issue try to reduce the number of features, that is, keep only the meaningful features or try regularization methods that will keep all the features. Ideal model will be the trade-off between Underfitting and over fitting like mentioned in the below picture.

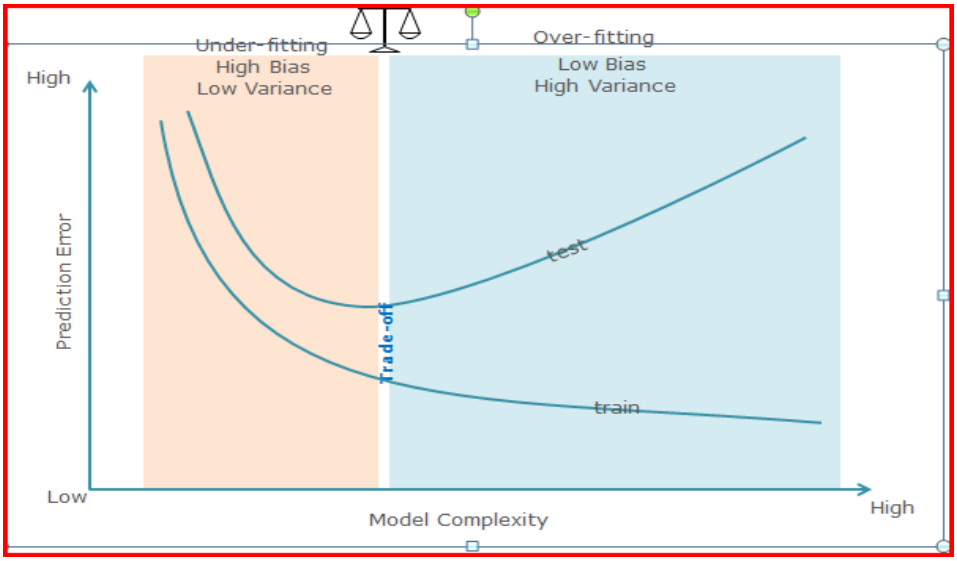


Fig. 2.5.1 Under-fitting vs Over –fitting

And, the Hyperparameters will be tuned in the below mentioned ways to reach the optimal solution:

1) Grid Search
2) Random Search
3) Manual Tuning

2.5.2 Model Performance Matrix

Model evaluation is an integral part of the model development. Based on model evaluation and subsequent comparisons, we can take a call whether to continue our efforts in model enhancement or cease them and select the final model that should be used / deployed.

1. Evaluating Classification Models

Confusion Matrix

Confusion matrix is one of the most popular ways to evaluate a classification model. A confusion matrix can be created for a binary classification as well as a multi-class classification model.

A confusion matrix is created by comparing the predicted class label of a data point with its actual class label. This comparison is repeated for the whole dataset and the results of this comparison are compiled in a matrix or tabular format.

Table: 2.5.2 Confusion Matrix

		Predicted classed		
Actual class		Positive (C_0)	Negative (C_1)	
	Positive (C_0)	a = number of correctly Classified c0 cases	c = number of c_0 cases Incorrectly classified as c_1	Precision = a/(a + c)
	Negative (C_0)	b = number of c_1 cases Incorrectly classified as c_0	d = number of correctly classified c_1 cases	
		Sensitivity (Recall) = a/(a+b)	Specificity = d/c+d	Accuracy = (a+b)(a+b+c+d)

Specificity : The ratio of actual negative cases that are identified correctly.

shows an example confusion matrix.

. Example of classifications Accuracy measurement

		Predicted classed		
Actual class		Positive (C_0)	Negative (C_1)	
	Positive (C_0)	80	30	Precision = 70/110=0.63
	Negative (C_1)	40	90	
		Recall=80/120=0.67	Specificity = 90/240=0.75	Accuracy = 80+90/240=0.71

And, below are the various measures that will be used to assess the performance of the model based on the requirement of the problem and as well as data.

Metric	Description	Formula
Accuracy	What% of predictions were Correct?	(TP + TN)/(TP + TN + EP + FN)
Misclassification rate	What % of prediction is wrong?	(FP + FN)/(TP + TN + FP + FN)
True positive rate OR Sensitivity or recall (completeness)	What % of positive cases did Model catch?	TP/(FN + TP)
False positive Rate	What % 'NO' were predicted as 'Yes'?	FP/FP+TN)
Specificity	What % 'NO' were predicted as 'NO'?	TN/(TN + FP)
Precision(exactness)	What % of positive predictions Were correct?	TP(TP + FP)
FI score	Weighted average of precision And recall	2*((precision*recall)/ (precision + recall))

2. Regression Model Evaluation

A regression line predicts the y values for a given x value. Note that the values are around the average. The prediction error (called as root-mean-square error or RSME) is given by the following formula:

$$RMSE = \sqrt{\frac{\sum_{k=0}^{n} \left(\hat{Y}_k - y_k\right)^2}{n}}$$

And, the regression will also assessed by R square (Co efficient of determination).

3. Evaluating Unsupervised Models

The Unsupervised algorithms will be assessed by the profile of the factors/ clusters which were derived through the models.

2.6 Overall Process of Machine Learning

To put overall process together, below is the picture that describes the road map for building ML Systems

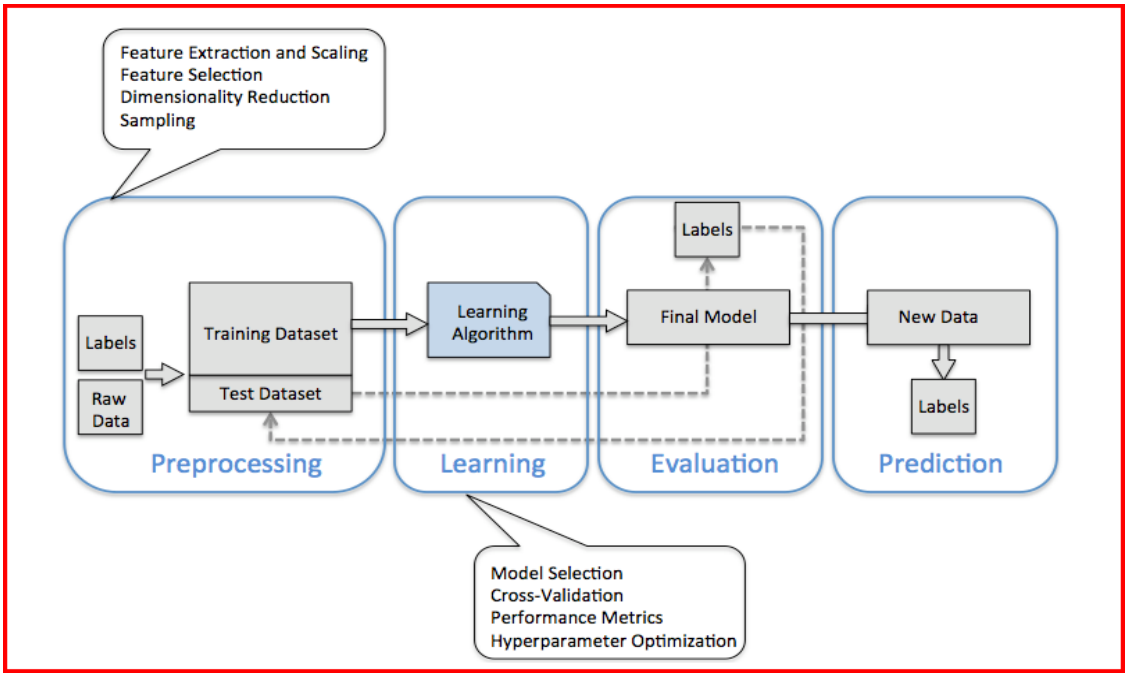


Fig.2.6 Overall Process of Machine learning

CHAPTER 3

MACHINE LEARNING AT WORK

MACHINE LEARNING AT WORK

3.1 An Approach to the Problem:

In order to carry out the analysis, we have extracted 1169 records from the UCI ML Repository and the information of the same is mentioned in Chapter 1.

In this Chapter, we are going to discuss about the results of different Machine Learning methods used in order to obtain the solution for the problem mentioned in Chapter1.

As mentioned in Chapter 2, the first step of a ML Algorithm is Data cleaning and preparing data for the modeling. As a first step, we have to check whether the data was read properly and all the scale types are as per the data.

```
data.frame':   1169 obs. of  11 variables:
$ ldisease      : int  1 1 1 1 1 1 1 1 1 1 ...
$ gender        : int  2 1 2 2 2 1 2 2 1 1 ...
$ age           : int  32 51 38 51 39 22 15 18 42 65 ...
$ total_brubin  : num  32.6 0.9 3.7 0.8 6.6 6.7 0.8 1.4 0.8 0.7 ...
$ direct_brubin : num  14.1 0.2 2.2 0.2 3 3.2 0.2 0.6 0.2 0.1 ...
$ alk_phos      : int  219 280 216 367 215 850 380 215 168 187 ...
$ alam_amin     : int  95 21 179 42 190 154 25 440 25 16 ...
$ asp_amin      : int  235 30 232 18 950 248 66 850 18 18 ...
$ total_proteins: num  5.8 6.7 7.8 5.2 4 6.2 6.1 5 6.2 6.8 ...
$ albumin       : num  3.1 3.2 4.5 2 1.7 2.8 3.7 1.9 3.1 3.3 ...
$ alb_glob_ratio: num  1.1 0.8 1.3 0.6 0.7 0.8 1.5 0.6 1 0.9 ...
```

Output: 3.1.1 Data Frame

Correcting data types:

As we can see from the above table, ldisease and gender which are categorical in nature are read as integer. So, we converted these attributes ldisease and gender from integer to factor (categorical).

```
'data.frame':   1169 obs. of  11 variables:
 $ ldisease      : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ gender        : Factor w/ 2 levels "1","2": 2 1 2 2 2 1 2 2 1 1 ...
 $ age           : int  32 51 38 51 39 22 15 18 42 65 ...
 $ total_brubin  : num  32.6 0.9 3.7 0.8 6.6 6.7 0.8 1.4 0.8 0.7 ...
 $ direct_brubin : num  14.1 0.2 2.2 0.2 3 3.2 0.2 0.6 0.2 0.1 ...
 $ alk_phos      : int  219 280 216 367 215 850 380 215 168 187 ...
 $ alam_amin     : int  95 21 179 42 190 154 25 440 25 16 ...
 $ asp_amin      : int  235 30 232 18 950 248 66 850 18 18 ...
 $ total_proteins: num  5.8 6.7 7.8 5.2 4 6.2 6.1 5 6.2 6.8 ...
 $ albumin       : num  3.1 3.2 4.5 2 1.7 2.8 3.7 1.9 3.1 3.3 ...
 $ alb_glob_ratio: num  1.1 0.8 1.3 0.6 0.7 0.8 1.5 0.6 1 0.9 ...
```

Output: 3.1.2 Correcting data types

Understanding data using Descriptive Statistics:

To understand the data, we will first look at the summary of the data.

```
 ldisease gender       age          total_brubin   direct_brubin      alk_phos
 1:668    1:241   Min.   : 4.00   Min.   : 0.40   Min.   : 0.100   Min.   :  63.0
 2:501    2:928   1st Qu.:32.00   1st Qu.: 0.80   1st Qu.: 0.200   1st Qu.: 174.0
                  Median :43.00   Median : 1.00   Median : 0.300   Median : 205.0
                  Mean   :43.98   Mean   : 3.06   Mean   : 1.294   Mean   : 279.4
                  3rd Qu.:57.00   3rd Qu.: 2.10   3rd Qu.: 1.000   3rd Qu.: 285.0
                  Max.   :90.00   Max.   :75.00   Max.   :18.300   Max.   :1896.0

   alam_amin          asp_amin       total_proteins     albumin
 Min.   :  10.00   Min.   :  10.00   Min.   :2.700   Min.   :0.900
 1st Qu.:  25.00   1st Qu.:  27.00   1st Qu.:5.700   1st Qu.:2.600
 Median :  37.00   Median :  41.00   Median :6.400   Median :3.100
 Mean   :  77.01   Mean   :  94.99   Mean   :6.411   Mean   :3.138
 3rd Qu.:  61.00   3rd Qu.:  73.00   3rd Qu.:7.100   3rd Qu.:3.700
 Max.   :1680.00   Max.   :4929.00   Max.   :9.600   Max.   :5.500

 alb_glob_ratio
 Min.   :0.3000
 1st Qu.:0.8000
 Median :0.9600
 Mean   :0.9695
 3rd Qu.:1.1000
 Max.   :2.8000
 NA's   :9
```

Output: 3.1.3 Summary of the data

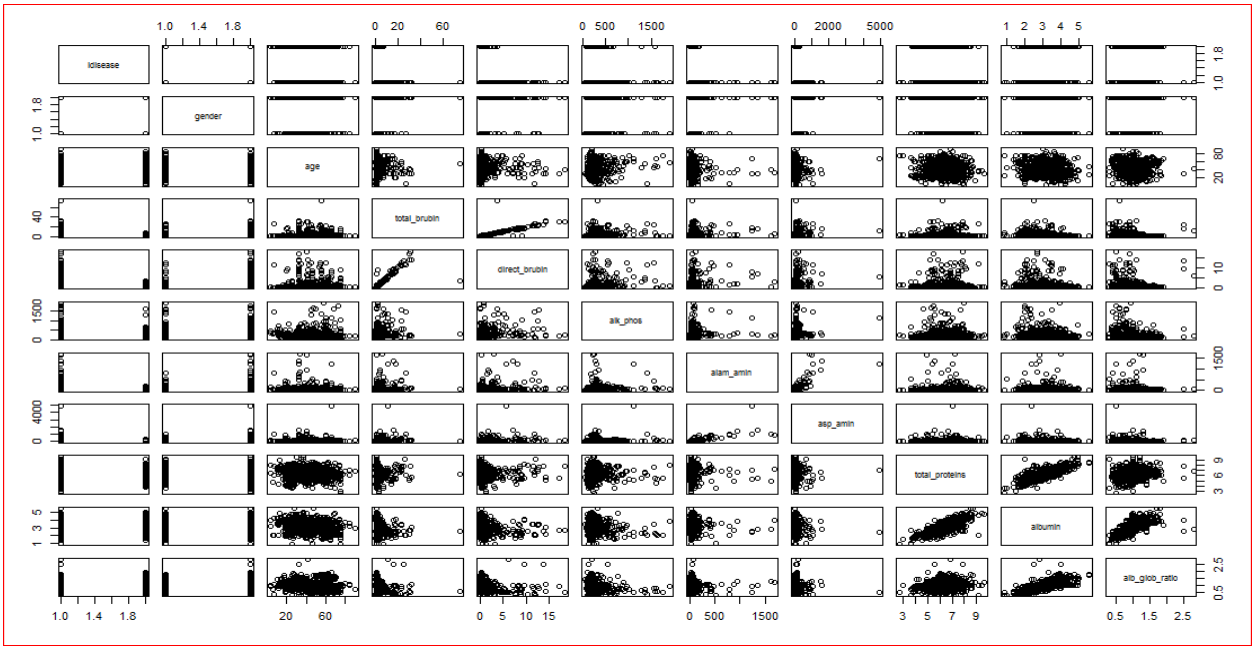
From the summary, we can see that there are

- 241 females and 928 males
- 668 are with liver disease and 501 are without liver disease

Also, we find mean, first quartile, median, third quartile, maximum and minimum values of all the continuous attributes.

Understanding data visually:

Also, look at the data visually to understand the relationships between and within the variables.



Output: 3.1.4 understanding data visually

Checking for missing Values:

We also need to check if the data contains any missing values, which can be done as below

```
      ldisease         gender            age   total_brubin  direct_brubin
     0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
      alk_phos      alam_amin       asp_amin total_proteins        albumin
     0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
alb_glob_ratio
     0.7698888
```

Output: 3.1.5 Checking for missing values

As we don't have any missing values in remaining variables except that there are 0.7698888% i.e., 9 missing values in alb_glob_ratio.

The missing values for the continuous variables will be imputed using Mean / Median value of the valid records and the categorical variables will be imputed using Mode value

Since, alb_glob_ratio is a continuous variable, we impute the missing observations using mean.

Result after imputing missing values:

```
      ldisease         gender            age   total_brubin  direct_brubin
             0              0              0              0              0
      alk_phos      alam_amin       asp_amin total_proteins        albumin
             0              0              0              0              0
alb_glob_ratio
             0
```

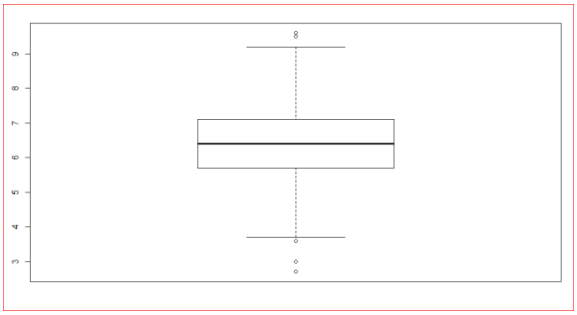
Output: 3.1.6 Result after imputing missing vales

Here we can see that there are no missing values in alb_glob_ratio after imputation.

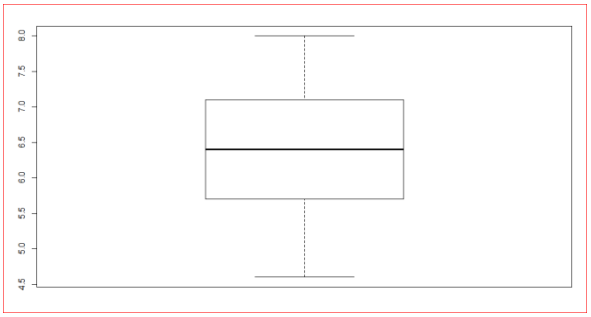
Checking for Outliers:

We used Box-plots to check for Outliers in each of the continuous variables..

Boxplot for total_proteins:



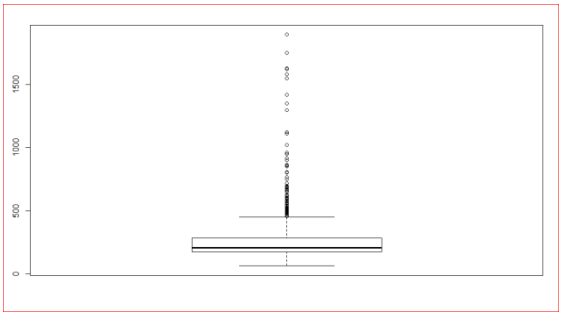
Output: 3.1.7 With outliers



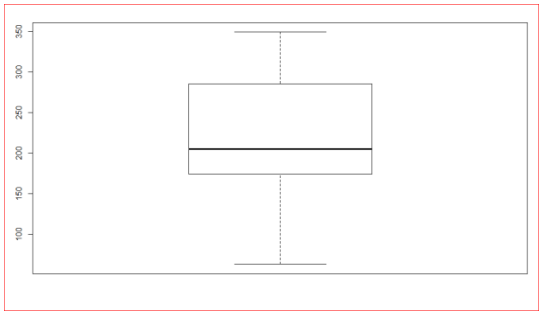
Output: 3.1.8 After replacing outliers

Here, for total_proteins, values more than 95th percentile are imputed using the 95th percentile value and the values less than 5th percentile are imputed using the 5th percentile value in order to remove outliers.

Boxplot for alk_phos:



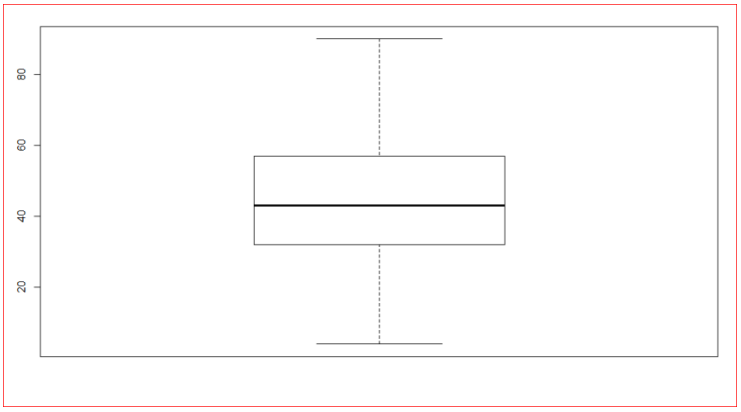
Output: 3.1.9 With outliers



Output: 3.1.10 After replacing outliers

In alk_phos, values more than 85th percentile are imputed using the 85th percentile value in order to remove outliers.

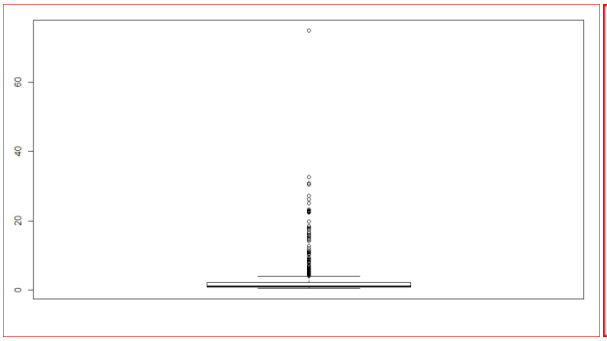
Boxplot for age:



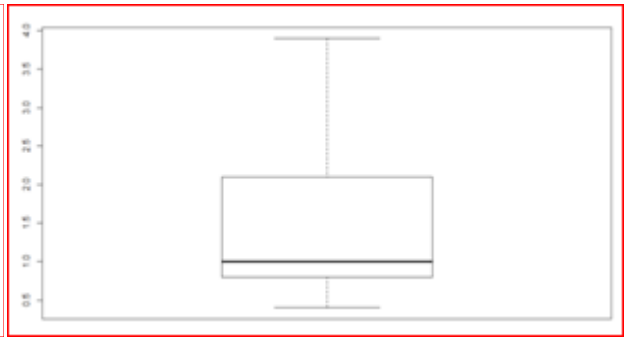
Output: 3.1.11 With no outliers

There are no outliers for age as we have taken a person's age exceeding 90yrs as 90yrs.

Boxplot for total_brubin:



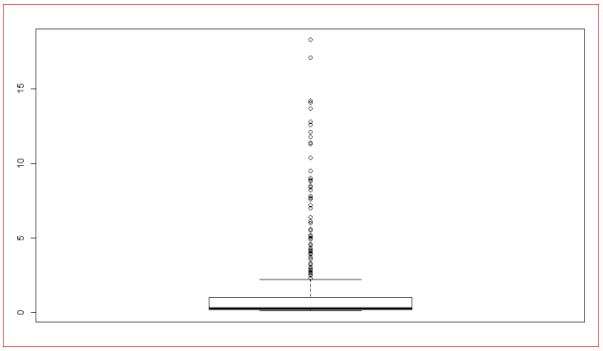
Output: 3.1.12 With outliers



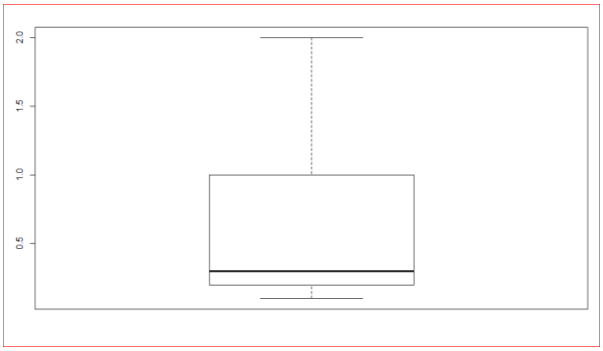
Output: 3.1.13 After replacing outliers

Here, in total_brubin,values more than 85th percentile are imputed using the 85th percentile value in order to remove outliers.

Boxplot for direct_brubin:



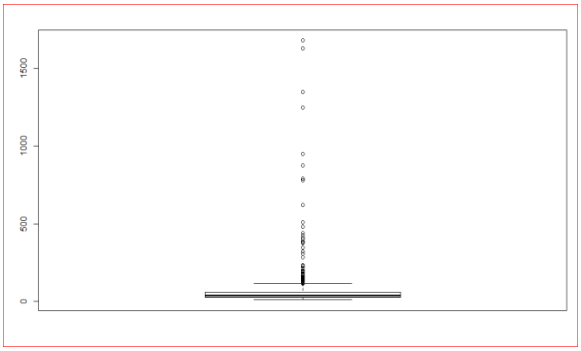
Output: 3.1.14 With outliers



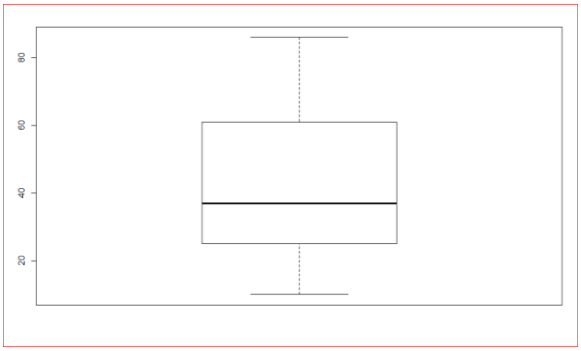
Output: 3.1.15 After replacing outliers

In direct_brubin, values more than 85th percentile are imputed using the 85th percentile value in order to remove outliers.

Boxplot for alam_amin:



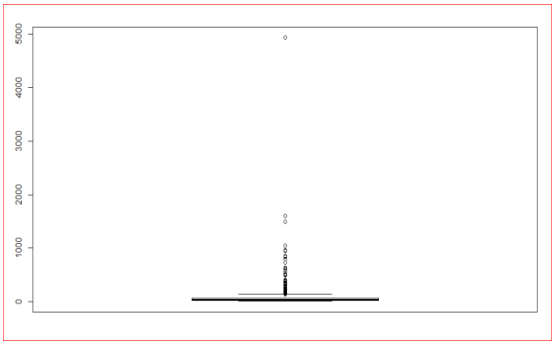
Output: 3.1.16 With outliers



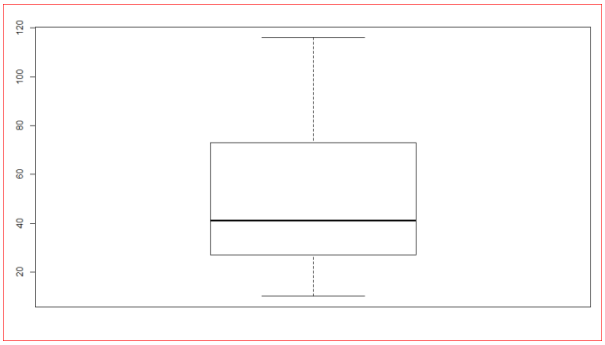
Output: 3.1.17 After replacing outliers

In alam_amin, values more than 85th percentile are imputed using the 85th percentile value in order to remove outliers.

Boxplot for asp_amin:



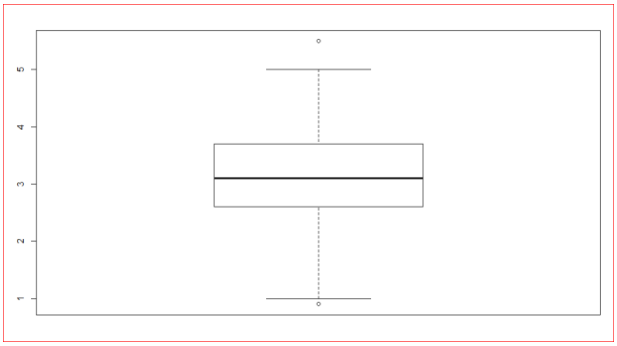
Output: 3.1.18 With outliers



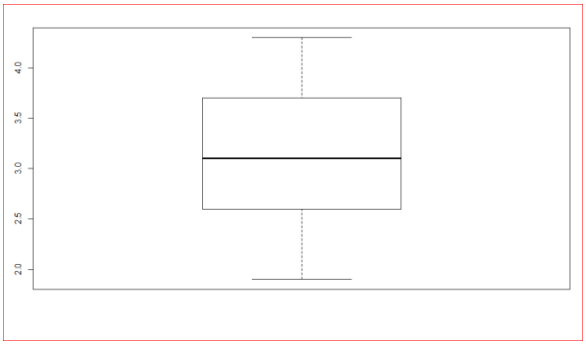
Output: 3.1.19 After replacing outliers

Here, in asp_amin, values more than 85th percentile are imputed using the 85th percentile value in order to remove outliers.

Boxplot for albumin:



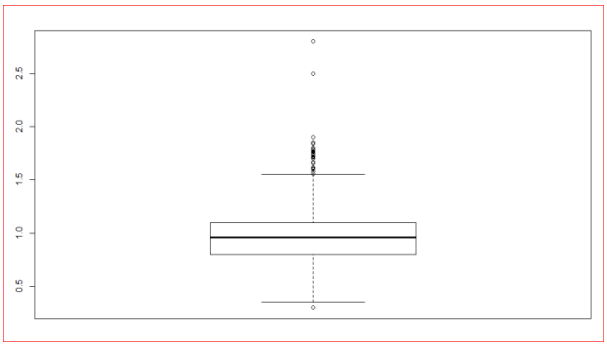
Output: 3.1.20 With outliers



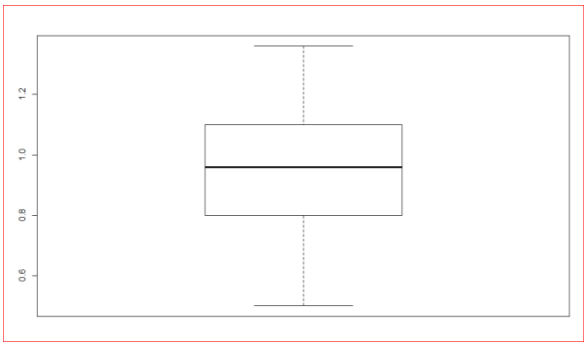
Output: 3.1.21 After replacing outliers

In albumin, values more than 95th percentile are imputed using the 95th percentile value and the values less than 5th percentile are imputed using the 5th percentile value in order to remove outliers.

Boxplot for alb_glob_ratio:



Output: 3.1.22 With outliers

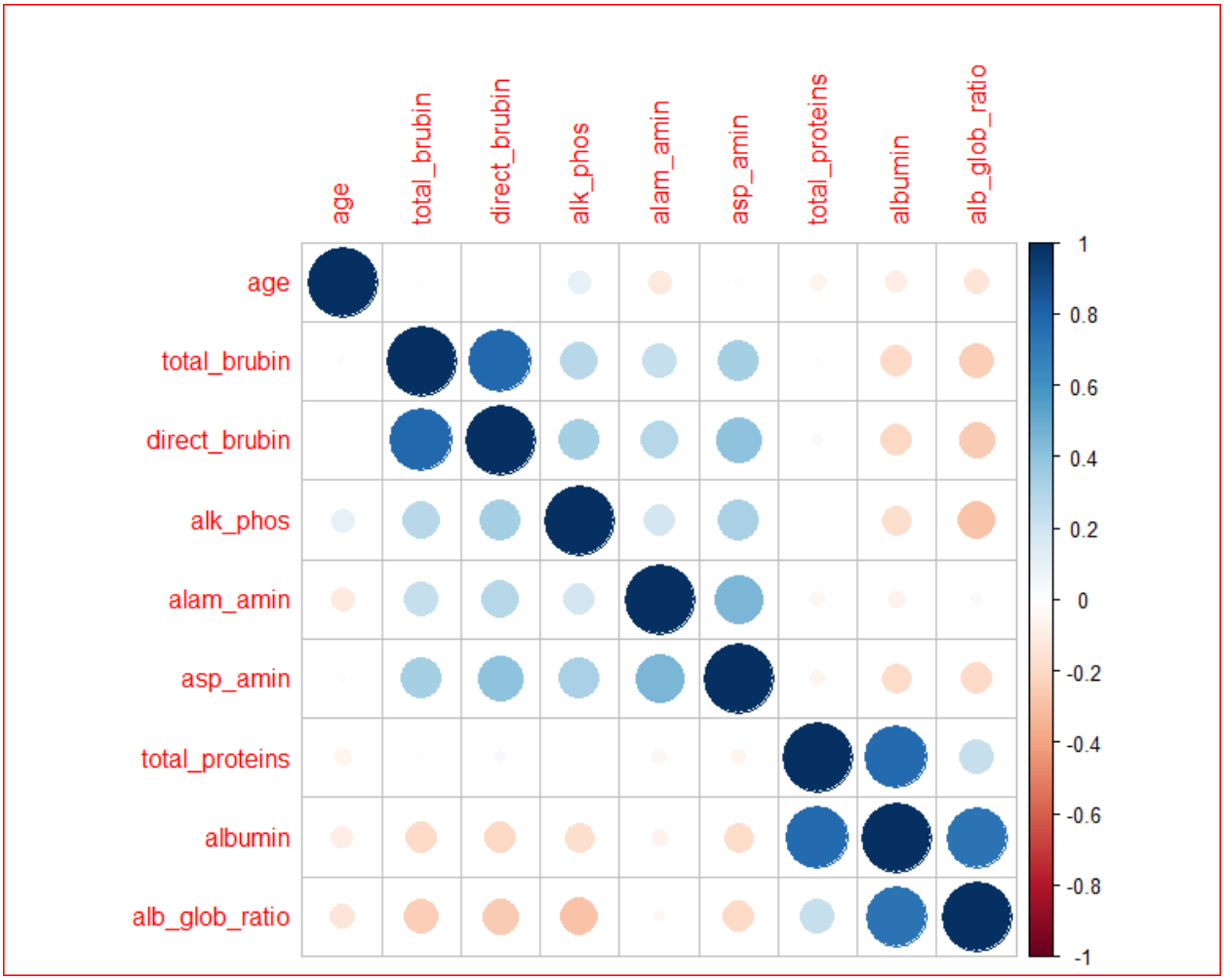


Output: 3.1.23 After replacing outliers

In alb_glob_ratio,values more than 90th percentile are imputed using the 90th percentile value and the values less than 5th percentile are using the 5th percentile value in order to remove outliers.

Understanding relationships between variables:

For the continuous variables, we will look at the Correlation plots to understand the relationships between variables.



Output: 3.1.24 Correlation Plot

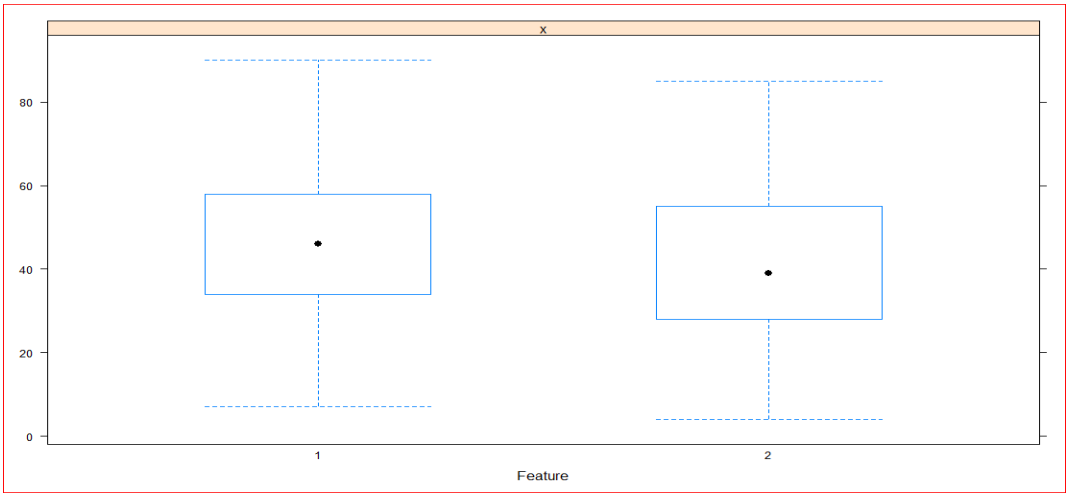
Here, the circle size refers to the strength of the relation and color refers to the direction of the relationship i.e., blue color represents the positive correlation and red color represents negative correlation.

From the plot, we can see that (direct_brubin, total_brubin),(albumin, total_protein), (alb_glob_ratio, albumin) are highly positively correlated.

FEATURE PLOTS:

For the continuous Vs categorical variable, we will look at Feature plots to understand the relationships between variables.

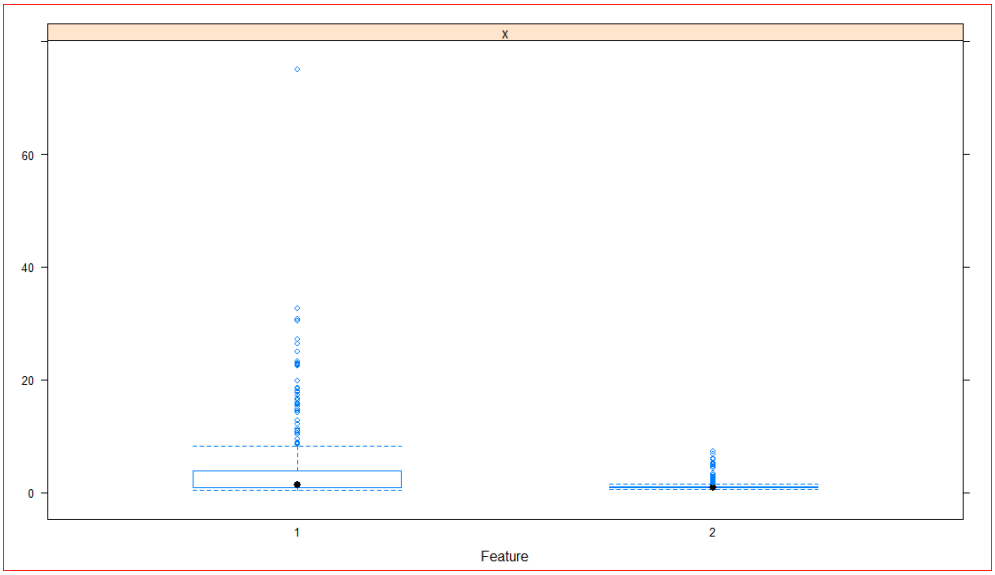
Age Vs ldisease:



Output: 3.1.25 feature plot for age vs. ldisease

We can observe that there is no much difference between age and ldisease.

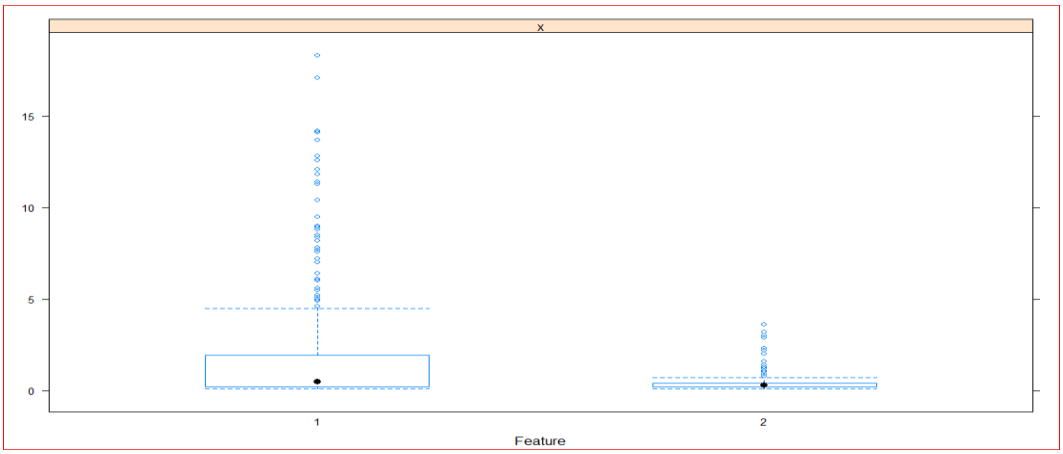
total_brubin Vs ldisease:



Output: 3.1.26 feature plot for total_brubin vs. ldisease

We can observe that there is no much difference between total brubin and ldisease.

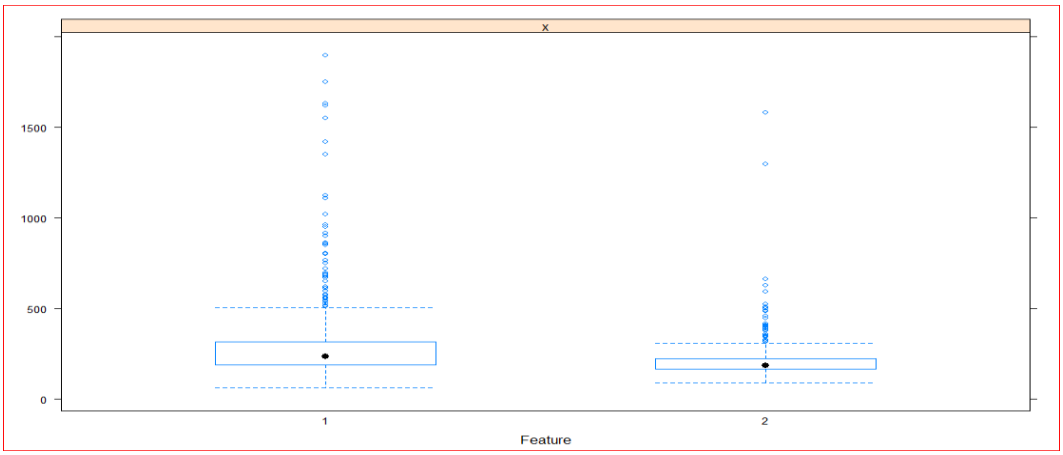
direct brubin Vs ldisease:



Output: 3.1.27 feature plot for direct_brubin vs. ldisease

We can observe that there is no much difference between direct _brubin and ldisease.

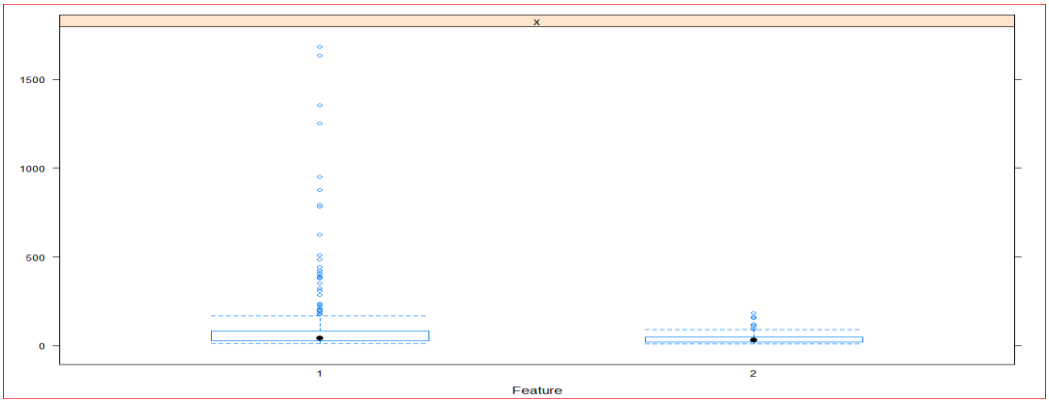
alk_phos Vs ldisease:



Output: 3.1.28 feature plot for alk_phos vs. ldisease

We can observe that there is no much difference between alk_phos and ldisease.

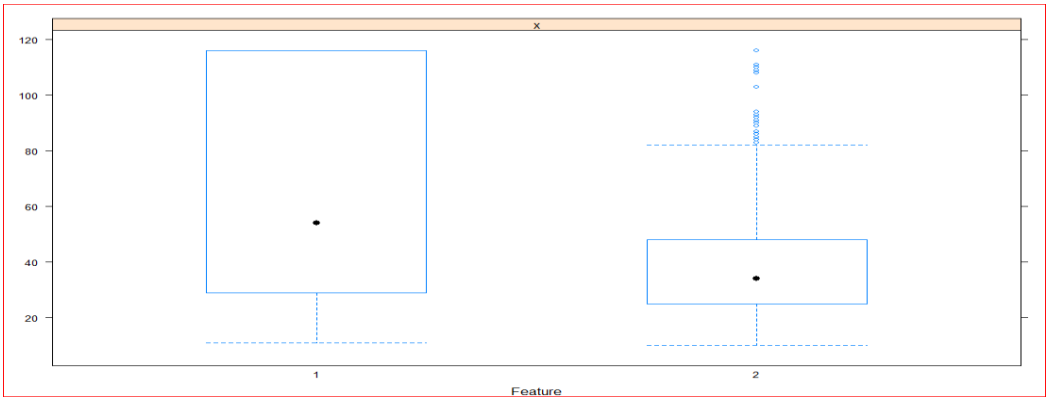
alam_amin Vs ldisease



Output: 3.1.29 feature plot for alam_amin vs. ldisease

We can observe that there is no much difference between alam_amin and ldisease.

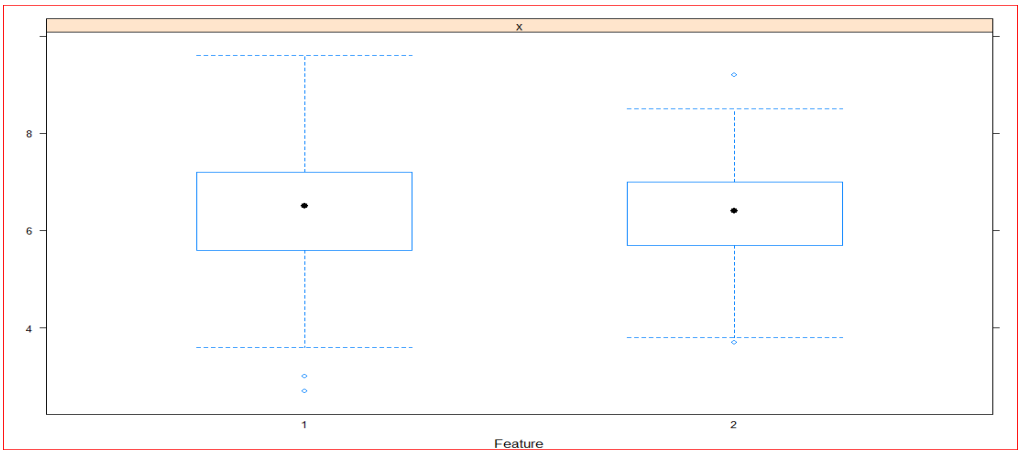
asp_amin Vs ldisease:



Output: 3.1.30 feature plot for asp_amin vs. ldisease

We can observe that there is slight difference between asp_amin and ldisease.

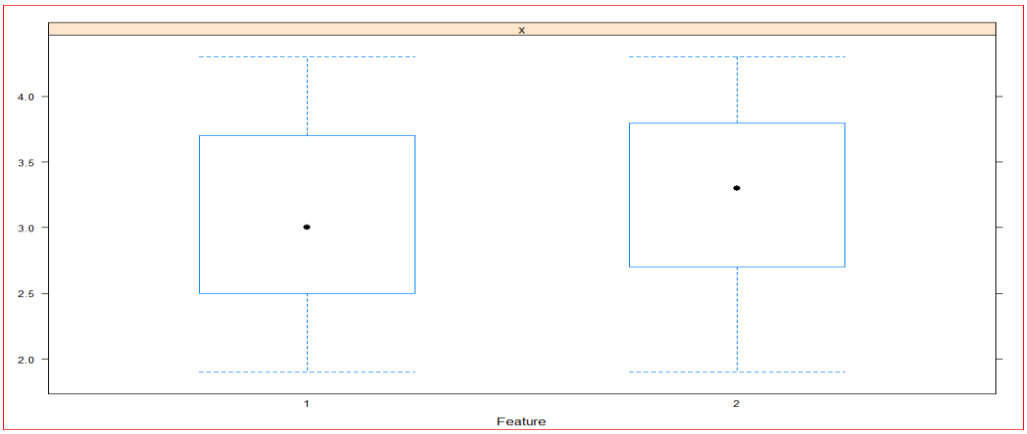
total_protein Vs ldisease:



Output: 3.1.31 feature plot for total_protein vs. ldisease

We can observe that there is no much difference between total protein and ldisease.

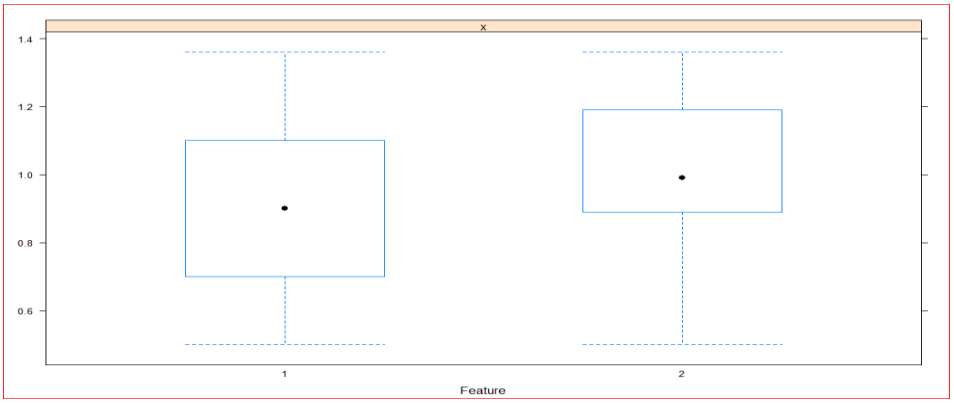
albumin Vs ldisease:



Output: 3.1.32 feature plot for albumin vs. ldisease

We can observe that there is slight difference between albumin and ldisease.

alb_glob_ratio Vs ldisease:



Output: 3.1.33 feature plot for alb_glob_ratio vs. ldisease

We can observe that there is slight difference between alb_glob_ratio and ldisease.

Normalising the Continuous variables:

As our input data is in different units, we have to ideally do normalisation. Hence, we normalized all the continuous variables with mean 0 and variance 1.

```
'data.frame':   1169 obs. of  11 variables:
 $ age             : num  -0.744 0.436 -0.371 0.436 -0.309 ...
 $ total_brubin    : num  4.4901 -0.3283 0.0973 -0.3435 0.5381 ...
 $ direct_brubin   : num  4.88 -0.417 0.345 -0.417 0.65 ...
 $ alk_phos        : num  -0.26545 0.00247 -0.27863 0.38459 -0.28302 ...
 $ alam_amin       : num  0.107 -0.334 0.609 -0.209 0.675 ...
 $ asp_amin        : num  1.85 -0.69 1.85 -1.04 1.85 ...
 $ total_proteins  : num  -0.568 0.269 1.293 -1.127 -2.243 ...
 $ albumin         : num  -0.0517 0.0915 1.6669 -1.627 -1.7703 ...
 $ alb_glob_ratio  : num  0.633 -0.602 1.456 -1.426 -1.014 ...
 $ dataset$ldisease: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ dataset$gender  : Factor w/ 2 levels "1","2": 2 1 2 2 2 1 2 2 1 1 ...
```

Output:3.1.34 Normalising the continuous variables

Cross Validation:

Here we'll perform the train and test split cross validation techniques. So, as part of it, we need to split the original data into train and test considering 90:10 proportion respectively.
Here, we are randomly considering 90% of the original data as train data. And, the dimensions of the train and test data are:

```
> dim(scale_training)
[1] 1169   11
> dim(training)
[1] 1052   11
> dim(test)
[1] 117  11
```

Output: 3.1.35 dimension of train and test data

Further we use k-fold validation for splitting train data into 10 folds as below
control <- trainControl(method="repeatedcv", number=10, repeats=3)

Running Pipeline using k-fold validation:

Here, we will use a pipeline of algorithms for classification to compare accuracies across different methods. As this is a classification problem, we will use Logistic Regression, Decision Tree, SVM, k-NN and Random Forest techniques as apart of the pipeline.

Logistic Regression:

```
Generalized Linear Model

1169 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1052, 1052, 1052, 1052, 1053, 1051, ...
Resampling results:

  Accuracy   Kappa
  0.6977876  0.3980182
```

Output: 3.1.36 Logistic Regression output on train data

When Logistic Model is fitted for the train data, the accuracy obtained is 69.7%.

Decision tree:

```
CART

1052 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 946, 947, 947, 947, 947, 947, ...
Resampling results across tuning parameters:

  cp    Accuracy   Kappa
  0.01  0.7741120  0.5375446
  0.05  0.6891494  0.4003851
  0.10  0.6948607  0.4155173

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01.
```

Output: 3.1.37 CART output on train data

By fitting decision tree the accuracy level obtained is 0.774112 at the hyperparameter value cp=0.01.

Support Vector Machines(SVM):

```
Support Vector Machines with Radial Basis Function Kernel

1052 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 946, 947, 947, 947, 947, 947, ...
Resampling results across tuning parameters:

  sigma  Accuracy   Kappa
  0.01   0.6948547  0.4039813
  0.05   0.7389009  0.4855344
  0.10   0.7591674  0.5217113

Tuning parameter 'C' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1 and C = 1.
```

Output: 3.1.38 SVM output for train data

0.759167 accuracy level is obtained when the sigma value selected is 0.1 and c=1 from a list of parameters using Support Vector Machines.

K – Nearest Neighbourhood:

```
k-Nearest Neighbors

1052 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 946, 947, 947, 947, 947, 947, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  1  0.8814975  0.7583308
  3  0.7918658  0.5726353
  5  0.7738005  0.5378289
  7  0.7668134  0.5239924

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.
```

Output: 3.1.39 k-NN output for train data

Best accuracy i.e.,0.88149 is obtained when k=1 nearest neighbours is choosen from a given list of neighbours by k- Nearest Neighbourhood method.

Random Forest:

```
Random Forest

1052 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 946, 947, 947, 947, 947, 946, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8900689  0.7753851
   6    0.8881791  0.7707522
  10    0.8907128  0.7759190

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 10.
```

Output: 3.1.40 Random Forest output on train data

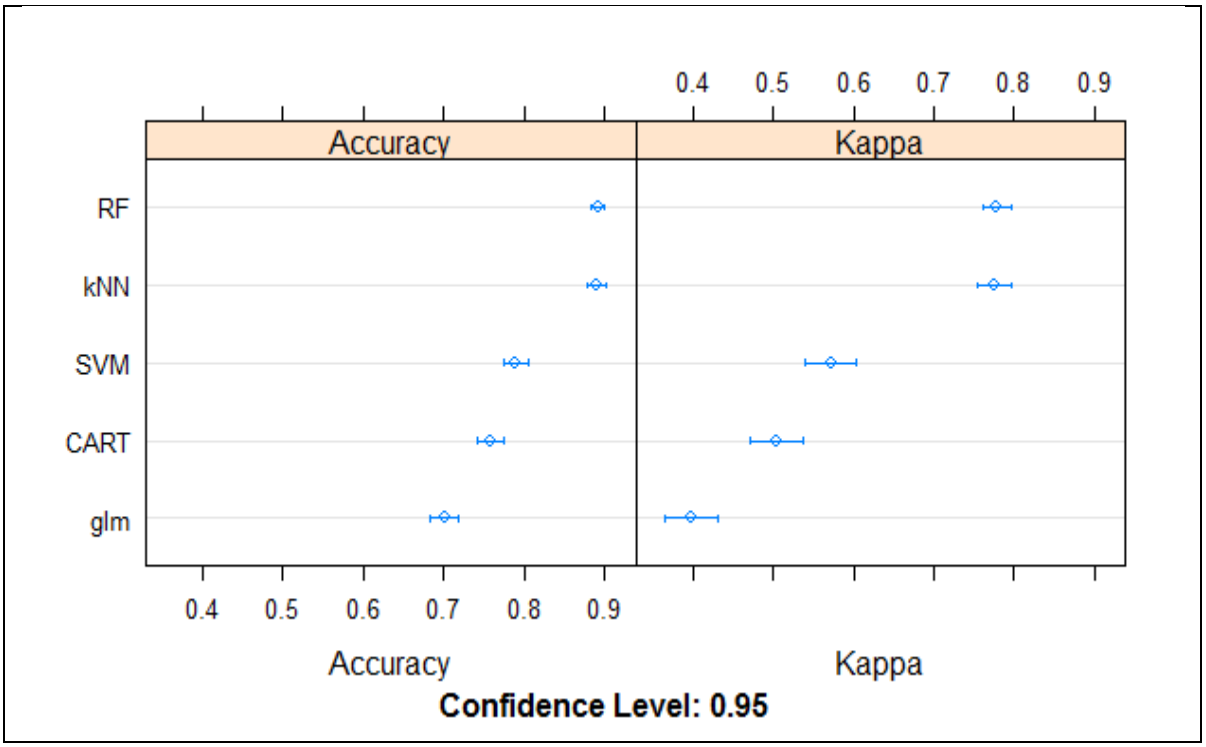
If the number of variables selected for split is 10 then we get the Decision Tree from large number of Decision Trees which are generally used in Random Forest with an accuracy of 0.89071.

Comparing algorithms:

```
Call:
summary.resamples(object = results)

Models: SVM, CART, kNN, glm, RF
Number of resamples: 30

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SVM  0.6952381 0.7428571 0.7571429 0.7591674 0.7886119 0.8207547    0
CART 0.6886792 0.7523810 0.7619048 0.7741120 0.8071429 0.8571429    0
kNN  0.8000000 0.8574798 0.8815364 0.8814975 0.9119048 0.9619048    0
glm  0.5897436 0.6666667 0.6995653 0.6977876 0.7334854 0.7777778    0
RF   0.8207547 0.8761905 0.8952381 0.8907128 0.9119048 0.9433962    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SVM  0.3921569 0.4785941 0.5221067 0.5217113 0.5776507 0.6450476    0
CART 0.3679075 0.4809731 0.5097885 0.5375446 0.6037627 0.7075209    0
kNN  0.5994550 0.7083026 0.7573649 0.7583308 0.8198329 0.9217877    0
glm  0.1865585 0.3424644 0.4020162 0.3980182 0.4694235 0.5527786    0
RF   0.6285504 0.7465181 0.7843137 0.7759190 0.8185589 0.8841952    0
```

Output:3.1.41 Comparing algorithms

When we compare all the models using above, we got best accuracy for Random Forest. So, we used Random Forest model to get the variable importance.
The below plot visualises the accuracies of above five models:



Output: 3.1.42 dot plot

Tuning the parameters in RandomForest:

Hyper parameters in Random Forest are tuned to extract the best paparmeters for final model.

- Number of trees as 100 or 200 or 300.
- The number of variables in each tree could range from 1to 6

```
1052 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 946, 947, 947, 947, 947, 947, ...
Resampling results across tuning parameters:

  mtry  ntree  Accuracy   Kappa
  1     100    0.8872237  0.7695820
  1     200    0.8894250  0.7741431
  1     300    0.8894190  0.7743645
  2     100    0.8909943  0.7770775
  2     200    0.8897424  0.7748254
  2     300    0.8868733  0.7686982
  3     100    0.8856184  0.7654179
  3     200    0.8862624  0.7673783
  3     300    0.8878437  0.7704551
  4     100    0.8837197  0.7619217
  4     200    0.8881491  0.7711308
  4     300    0.8849895  0.7645341
  5     100    0.8824409  0.7588125
  5     200    0.8872087  0.7690611
  5     300    0.8875172  0.7696393
  6     100    0.8821324  0.7582671
  6     200    0.8862564  0.7669790
  6     300    0.8856214  0.7657804

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were mtry = 2 and ntree = 100.
```

Output: 3.1.43 Tuning the parameters in Random Forest

From the above, we found the best accuracy when the number of trees are 100 with each tree containing 2 variables.

Finding key variables using RandomForest:

Variable importance plot:

```
                MeanDecreaseGini
age                    62.662418
total_brubin           51.929490
direct_brubin          70.346490
alk_phos               76.005914
alam_amin              61.163269
asp_amin               79.274515
total_proteins         39.197292
albumin                29.320146
alb_glob_ratio         39.256919
gender                  6.633889
```

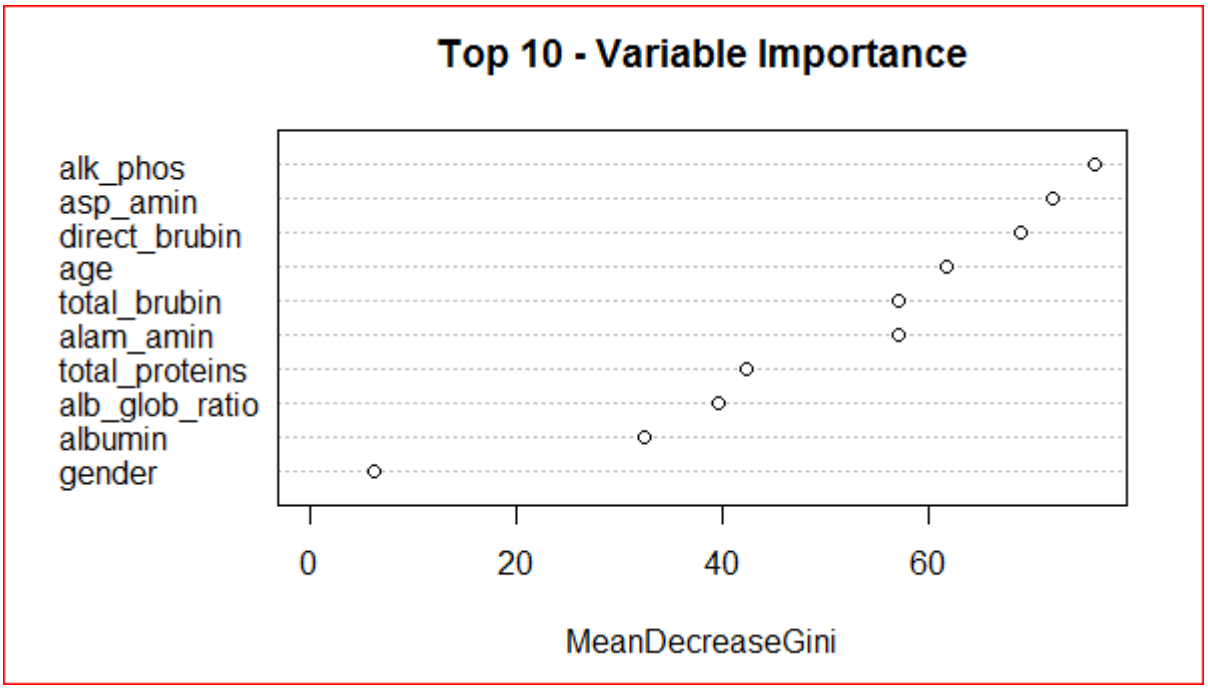
Output: 3.1.44 MeanDecreaseGini values

Above table shows the mean decrease gini index of each variable, based on which we are choosing the key variables. The variable which is having more gini index is more effective.

The increasing order of effective variables is,
asp_amin>alk_phos>direct_brubin>age>alam_amine>total_brubin>alb_glob_ratio>total_protein>albumin>gender

Visualization of key variables:

A plot visualizing the top 10 important variables among the above variables is as follows:



Output:3.1.45 Variable Importance Plot

Above plot describes the importance of top 10 variables actual effecting the prediction of a person having liver disease or not.

Fitting the final model:

We considered the top 6 variables from the Random Forest and fitted Logistic Regression. Summary of the final fitted Logistic model using key variables is shown below.

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7194  -1.0433  -0.0841   0.9938   3.3052

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.97447    0.15161  -6.427 1.30e-10 ***
asp_amin      -0.23996    0.11544  -2.079 0.037650 *
alk_phos      -0.46281    0.14532  -3.185 0.001448 **
total_brubin  -0.10924    0.37911  -0.288 0.773234
alam_amin     -1.17705    0.43836  -2.685 0.007251 **
age           -0.29079    0.07195  -4.042 5.31e-05 ***
direct_brubin -1.71602    0.46840  -3.664 0.000249 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1437.5  on 1051  degrees of freedom
Residual deviance: 1155.6  on 1045  degrees of freedom
AIC: 1169.6

Number of Fisher Scoring iterations: 7
```

Output:3.1.46 Summary of final fitted Logistic model

```
Generalized Linear Model

1052 samples
   6 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 947, 946, 946, 947, 947, 947, ...
Resampling results:

  Accuracy   Kappa
  0.6707996  0.3416298
```

Output: 3.1.47 Accuracy for fitted Logistic model

As this problem is type of classification, we check its accuracy using the confusion matrix.

The confusion matrix for the train data is as follows:

Confusion matrix for Train data:

```
Confusion Matrix and Statistics

          Reference
Prediction   1   2
         1 388 137
         2 212 315

               Accuracy : 0.6683
                 95% CI : (0.6389, 0.6967)
    No Information Rate : 0.5703
    P-Value [Acc > NIR] : 5.063e-11

                  Kappa : 0.3367
 Mcnemar's Test P-Value : 7.460e-05

            Sensitivity : 0.6467
            Specificity : 0.6969
         Pos Pred Value : 0.7390
         Neg Pred Value : 0.5977
             Prevalence : 0.5703
         Detection Rate : 0.3688
   Detection Prevalence : 0.4990
      Balanced Accuracy : 0.6718

       'Positive' Class : 1
```

Output: 3.1.48 Confusion matrix and accuracy for train data

From the above confusion matrix, we obtained the accuracy for the train data as 66.8%. We conclude that 703 observations are correctly classified out of 1052 observations when the Logistic Model is applied for the train data.

Validating the model using Test data:

We fitted Logistic Regression on the test data which is of 117 observations to validate the final model.

```
> dim(test)
[1] 117  11
> names(test)
 [1] "age"            "total_brubin"   "direct_brubin"  "alk_phos"
 [5] "alam_amin"      "asp_amin"       "total_proteins" "albumin"
 [9] "alb_glob_ratio" "ldisease"       "gender"
```

Output: 3.1.49 dimension and names of test data

Confusion matrix on Test data:

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 45 15
         2 23 34

               Accuracy : 0.6752
                 95% CI : (0.5824, 0.7589)
    No Information Rate : 0.5812
    P-Value [Acc > NIR] : 0.02346

                  Kappa : 0.3477
 Mcnemar's Test P-Value : 0.25614

            Sensitivity : 0.6618
            Specificity : 0.6939
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.5965
             Prevalence : 0.5812
         Detection Rate : 0.3846
   Detection Prevalence : 0.5128
      Balanced Accuracy : 0.6778

       'Positive' Class : 1
```

Output: 3.1.50 Confusion matrix and accuracy for test data

Out of 117 observations 79 observations are correctly classified for test data by using the same Logistic Regression model i.e., we attain the accuracy of 67.5% for the test /unseen data.

As the accuracy obtained from train and test data donot differ significantly, the obtained Logistic model is considered as a Generalised model with six important variables – alk_phos, asp_amin, direct_brubin, age, total_brubin, alam_amin.

CHAPTER 4

CONCLUSION

CONCLUSION

In order to identify whether a person has liver disease or not, we developed an algorithm in which we applied pipeline techniques as well as Random Forest to choose the best model and key variables.

Once we identified the key variables, we run the model with only key variables using Train data set and validate the same using Test data set.

The accuracy of Train and Test data are 66.8% and 67.5% respectively.
Since, the accuracy of Train and Test data are almost same, we can say that our model is a Generalized Linear Model.

Hence, we can apply our model for future predictions.

APPENDIX

R - CODE
DATASET
BIBLIOGRAPHY

R Code:

```
getwd()
setwd("C:/Users/USER/Downloads")
getwd()

#####################################################################
#Loading required libraries to perform modeling
#####################################################################
library(mice)
library(randomForest)
library(ggplot2)
library(glmnet)

#####################################################################
# reading data from Folder and checking for the data types
#####################################################################

dataset <- read.csv(file.choose(), header = T)
str(dataset)


table(dataset$ldisease)

dataset$ldisease <- as.factor(dataset$ldisease)
dataset$gender <- as.factor(dataset$gender)
str(dataset)
summary(dataset)
dim(dataset)
head(dataset)
tail(dataset)

#####################################################################
#Checking for Missing value
#####################################################################

print(all(!is.na(dataset)))
#Missing value  Proportion for all the variables
```

```
sapply(dataset, function(df) {
  (sum(is.na(df)==TRUE)/ length(df))*100;
})



########################################################################
# Missing values impute for small proportion of missing values
########################################################################

dataset$alb_glob_ratio[is.na(dataset$alb_glob_ratio)]< -
mean(dataset$alb_glob_ratio,na.rm=T)
sum(is.na(dataset))
dataset$alb_glob_ratio=as.numeric(dataset$alb_glob_ratio)
str(dataset)

########################################################################
#Missing value  Proportion for all the variables
########################################################################

sapply(dataset, function(df) {
  (sum(is.na(df)==TRUE)/ length(df))*100;
})

########################################################################
boxplot(dataset$total_proteins)
boxplot(dataset$alk_phos)
boxplot(dataset$age)
boxplot(dataset$total_brubin)
boxplot(dataset$direct_brubin)
boxplot(dataset$alam_amin)
boxplot(dataset$asp_amin)
boxplot(dataset$albumin)
boxplot(dataset$alb_glob_ratio)

dataset$total_proteins[dataset$total_proteins>quantile(dataset$total_proteins, 0.95)] < -
quantile(dataset$total_proteins, 0.95)
dataset$total_proteins[dataset$total_proteins<quantile(dataset$total_proteins, 0.05)] < -
quantile(dataset$total_proteins, 0.05)
dataset$alk_phos[dataset$alk_phos>quantile(dataset$alk_phos, 0.85)] < -
quantile(dataset$alk_phos, 0.85)
dataset$total_brubin[dataset$total_brubin>quantile(dataset$total_brubin, 0.85)] < -
quantile(dataset$total_brubin, 0.85)
dataset$direct_brubin[dataset$direct_brubin>quantile(dataset$direct_brubin, 0.85)] < -
quantile(dataset$direct_brubin, 0.85)
dataset$alam_amin[dataset$alam_amin>quantile(dataset$alam_amin, 0.85)] < -
quantile(dataset$alam_amin, 0.85)
dataset$asp_amin[dataset$asp_amin>quantile(dataset$asp_amin, 0.85)] < -
quantile(dataset$asp_amin, 0.85)
dataset$albumin[dataset$albumin>quantile(dataset$albumin, 0.95)] < -
quantile(dataset$albumin, 0.95)
```

```
dataset$albumin[dataset$albumin<quantile(dataset$albumin, 0.05)] <-
quantile(dataset$albumin, 0.05)
dataset$alb_glob_ratio[dataset$alb_glob_ratio>quantile(dataset$alb_glob_ratio, 0.90)] < -
quantile(dataset$alb_glob_ratio, 0.90)
dataset$alb_glob_ratio[dataset$alb_glob_ratio<quantile(dataset$alb_glob_ratio, 0.05)] < -
quantile(dataset$alb_glob_ratio, 0.05)

boxplot(dataset$total_proteins)
boxplot(dataset$alk_phos)
boxplot(dataset$age)
boxplot(dataset$total_brubin)
boxplot(dataset$direct_brubin)
boxplot(dataset$alam_amin)
boxplot(dataset$asp_amin)
boxplot(dataset$albumin)
boxplot(dataset$alb_glob_ratio)

pairs(dataset)
pre1=dataset[c(3:11)]
install.packages("corrplot")
library(corrplot)
pre1.cor        =       cor(pre1)
corrplot(pre1.cor,      method="circle")

#####################################################################
# Continuous vs categories
#####################################################################
library(caret)
x <- dataset[,3]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
x <- dataset[,4]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
x <- dataset[,5]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
x <- dataset[,6]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
x <- dataset[,7]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
x <- dataset[,8]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")

x <- dataset[,9]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
x <- dataset[,10]
y <- dataset[,1]
```

```
featurePlot(x=x, y=y, plot="box")
x <- dataset[,11]
y <- dataset[,1]
featurePlot(x=x, y=y, plot="box")
str(dataset)
cont<-subset(dataset,select = -c(gender, ldisease))
str(cont)

######################################################################
# Normalizing input data
######################################################################

scale_training <- as.data.frame(scale(cont[,],
                        center = TRUE, scale = TRUE))
scale_training<-cbind(scale_training,dataset$ldisease,dataset$gender)
str(scale_training)

######################################################################
# we can also set/change the variable names using colnames
######################################################################

colnames(scale_training) <-
c("age","total_brubin","direct_brubin","alk_phos","alam_amin","asp_amin","total_proteins",
"albumin","alb_glob_ratio","ldisease","gender")
names(scale_training)
table(scale_training$ldisease)
write.csv(scale_training,"scaledata.csv")

######################################################################
# Splting data
######################################################################

train_rows<- sample(1:nrow(scale_training), size=0.9*nrow(scale_training))
train_rows
training <- scale_training[train_rows, ]
test <- scale_training[-train_rows, ]
dim(scale_training)
dim(training)
dim(test)
head(test)

######################################################################
# Model Pipleline
######################################################################

library(caret)
# Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10, repeats=3)
```

```
# GLM
set.seed(7)
fit.glm <- train(ldisease~., data=dataset, method="glm", metric="Accuracy",
trControl=control)
print(fit.glm)

# CART
set.seed(7)
grid <- expand.grid(.cp=c(0.01,0.05,0.1))
fit.cart <- train(ldisease~., data=training, method="rpart", metric="Accuracy", tuneGrid=grid,
trControl=control)
print(fit.cart)

# SVM
set.seed(7)
grid <- expand.grid(.sigma=c(0.01,0.05,0.1), .C=c(1))
fit.svm <- train(ldisease~., data=training, method="svmRadial", metric="Accuracy",
tuneGrid=grid, trControl=control)
print(fit.svm)

# kNN
set.seed(7)
grid <- expand.grid(.k=c(1,3,5,7))
fit.knn <- train(ldisease~., data=training, method="knn", metric="Accuracy", tuneGrid=grid,
trControl=control)
print(fit.knn)

#RF
fit.rf <- train(ldisease~., data=training, method="rf", metric="Accuracy", trControl=control)
print(fit.rf)

########################################################################
# Compare algorithms
########################################################################

results <- resamples(list(SVM=fit.svm, CART=fit.cart, kNN=fit.knn, glm=fit.glm, RF=fit.rf))
summary(results)
dotplot(results)

########################################################################
#Tuning Random Forest
########################################################################

customRF <- list(type = "Classification", library = "randomForest", loop = NULL)
customRF$parameters <- data.frame(parameter = c("mtry", "ntree"), class = rep("numeric",
2), label = c("mtry", "ntree"))
customRF$grid <- function(x, y, len = NULL, search = "grid") {}
customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
  randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
}
customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata)
```

```
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes

#####################################################################

library(caret)
library(randomForest)
control <- trainControl(method="repeatedcv", number=10, repeats=3)
tunegrid <- expand.grid(.mtry=c(1:6), .ntree=c(100, 200, 300))
set.seed(100)
custom <- train(ldisease~., data=training,method=customRF, tuneGrid=tunegrid,
trControl=control)
print(custom)

#####################################################################

rf_model3 <- randomForest(ldisease ~ ., data =training, ntree=100, mtry=5)

varImpPlot(rf_model3,
        sort = T,
        n.var = 10,
        main = "Top 10 - Variable Importance")

importance(rf_model3)

#####################################################################
#$Final model With Logistic Regression
#####################################################################

dim(training)
dim(test)

trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)

final.glm <-train (ldisease ~ asp_amin +  alk_phos + total_brubin + alam_amin + age +
direct_brubin, data=training, method="glm",
                trControl=trainControl)

summary(final.glm)
print (final.glm)

predslog <- predict(final.glm, data=training, type = "raw")
tabtrain <- table(Predicted = predslog, Actual = training$ldisease )
caret::confusionMatrix(predslog,training$ldisease)
```

```
#########################################################################
#On testing
#########################################################################

dim(test)
names(test)


p2 <- predict(final.glm,newdata=test,type="raw")

tabtest <- table(Predicted = p2, Actual = test$ldisease)

caret::confusionMatrix(p2,test$ldisease)
#########################################################################
```

Dataset:

ldisease	gender	age	total_bubin	direct_brubin	alk_phos	alam_amin	asp_amin	total_proteins	albumin	alb_glob_ratio
1	2	32	32.6	14.1	219	95	235	5.8	3.1	1.1
1	1	51	0.9	0.2	280	21	30	6.7	3.2	0.8
1	2	38	3.7	2.2	216	179	232	7.8	4.5	1.3
1	2	51	0.8	0.2	367	42	18	5.2	2	0.6
1	2	39	6.6	3	215	190	950	4	1.7	0.7
1	1	22	6.7	3.2	850	154	248	6.2	2.8	0.8
1	2	15	0.8	0.2	380	25	66	6.1	3.7	1.5
1	2	18	1.4	0.6	215	440	850	5	1.9	0.6
1	1	42	0.8	0.2	168	25	18	6.2	3.1	1
1	1	65	0.7	0.1	187	16	18	6.8	3.3	0.9
1	2	38	0.7	0.2	216	349	105	7	3.5	1
1	2	48	2.4	1.1	554	141	73	7.5	3.6	0.9
1	2	60	0.8	0.2	286	21	27	7.1	4	1.2
1	2	42	11.1	6.1	214	60	186	6.9	2.8	2.8
1	2	51	0.8	0.2	230	24	46	6.5	3.1	
1	2	60	3.2	1.8	750	79	145	7.8	3.2	0.69
1	2	72	2.7	1.3	260	31	56	7.4	3	0.6
1	2	53	0.9	0.4	238	17	14	6.6	2.9	0.8
1	2	26	0.6	0.2	120	45	51	7.9	4	1
1	2	42	30.5	14.2	285	65	130	5.2	2.1	0.6
1	1	32	0.6	0.1	176	39	28	6	3	1
1	2	58	1	0.4	182	14	20	6.8	3.4	1
1	2	75	2.9	1.3	218	33	37	3	1.5	1
1	2	40	0.9	0.3	196	69	48	6.8	3.1	0.8
1	2	66	17.3	8.5	388	173	367	7.8	2.6	0.5
1	2	31	1.3	0.5	184	29	32	6.8	3.4	1
1	2	30	1.6	0.4	332	84	139	5.6	2.7	0.9
1	1	48	1	1.4	144	18	14	8.3	4.2	1
1	2	65	4.9	2.7	190	33	71	7.1	2.9	0.7
1	2	51	0.8	0.2	160	34	20	6.9	3.7	1.1
1	2	47	0.9	0.2	192	38	24	7.3	4.3	1.4
1	2	57	0.6	0.1	210	51	59	5.9	2.7	0.8
1	2	32	23	11.3	300	482	275	7.1	3.5	0.9
1	2	40	0.7	0.1	202	37	29	5	2.6	1
1	2	60	0.9	0.3	168	16	24	6.7	3	0.8
1	2	16	2.6	1.2	236	131	90	5.4	2.6	0.9
1	2	60	0.8	0.2	286	21	27	7.1	4	1.2
1	2	42	16.4	8.9	245	56	87	5.4	2	0.5
1	2	60	6.3	3.2	314	118	114	6.6	3.7	1.27
1	1	7	27.2	11.8	1420	790	1050	6.1	2	0.4
1	2	60	1.5	0.6	360	230	298	4.5	2	0.8
1	2	37	1.3	0.4	195	41	38	5.3	2.1	0.6

1	1	34	0.8	0.2	192	15	12	8.6	4.7	1.2
1	2	54	0.8	0.2	218	20	19	6.3	2.5	0.6
1	2	78	1	0.3	152	28	70	6.3	3.1	0.9
1	2	55	14.1	7.6	750	35	63	5	1.6	0.47
1	2	73	1.8	0.9	220	20	43	6.5	3	0.8
1	2	32	15.6	9.5	134	54	125	5.6	4	2.5
1	2	37	0.7	0.2	176	28	34	5.6	2.6	0.8
1	1	54	5.5	3.2	350	67	42	7	3.2	0.8
1	2	54	2.2	1.2	195	55	95	6	3.7	1.6
1	1	36	0.8	0.2	650	70	138	6.6	3.1	0.8
1	2	46	0.6	0.2	290	26	21	6	3	1
1	2	30	0.8	0.2	174	21	47	4.6	2.3	1
1	2	55	4.4	2.9	230	14	25	7.1	2.1	0.4
1	2	58	1	0.4	182	14	20	6.8	3.4	1
1	1	68	0.6	0.1	1620	95	127	4.6	2.1	0.8
1	2	55	75	3.6	332	40	66	6.2	2.5	0.6
1	2	18	0.8	0.2	282	72	140	5.5	2.5	0.8
1	1	48	1	0.3	310	37	56	5.9	2.5	0.7
1	2	33	1.5	7	505	205	140	7.5	3.9	1
1	2	60	2.1	1	191	114	247	4	1.6	0.6
1	2	62	1.8	0.9	224	69	155	8.6	4	0.8
1	2	30	1.6	0.4	332	84	139	5.6	2.7	0.9
1	2	64	3	1.4	248	46	40	6.5	3.2	0.9
1	2	33	0.7	0.2	256	21	30	8.5	3.9	0.8
1	1	34	0.6	0.1	161	15	19	6.6	3.4	1
1	2	50	0.7	0.2	188	12	14	7	3.4	0.9
1	2	49	0.6	0.1	218	50	53	5	2.4	0.9
1	1	42	0.8	0.2	182	22	20	7.2	3.9	1.1
1	2	55	3.6	1.6	349	40	70	7.2	2.9	0.6
1	2	30	1.6	0.4	332	84	139	5.6	2.7	0.9
1	2	40	0.7	0.1	202	37	29	5	2.6	1
1	2	50	0.9	0.3	901	23	17	6.2	3.5	1.2
1	2	75	2.5	1.2	375	85	68	6.4	2.9	0.8
1	1	28	0.9	0.2	316	25	23	8.5	5.5	1.8
1	2	32	0.9	0.3	462	70	82	6.2	3.1	1
1	2	75	0.9	0.2	162	25	20	6.9	3.7	1.1
1	2	75	8	4.6	386	30	25	5.5	1.8	0.48
1	2	50	0.9	0.3	194	190	73	7.5	3.9	1
1	2	72	2.7	1.3	260	31	56	7.4	3	0.6
1	1	49	0.8	0.2	198	23	20	7	4.3	1.5
1	2	33	0.9	0.8	680	37	40	5.9	2.6	0.8
1	1	13	0.7	0.2	350	17	24	7.4	4	1.1
1	1	50	1.7	0.6	430	28	32	6.8	3.5	1
1	2	62	1.2	0.4	195	38	54	6.3	3.8	1.5
1	1	53	0.8	0.2	193	96	57	6.7	3.6	1.16
1	2	75	0.9	0.2	282	25	23	4.4	2.2	1

1	1	48	0.8	0.2	150	25	23	7.5	3.9	1
1	2	33	3.4	1.6	186	779	844	7.3	3.2	0.7
1	2	55	0.7	0.2	290	53	58	6.8	3.4	1
1	2	42	11.1	6.1	214	60	186	6.9	2.8	2.8
1	1	42	0.8	0.2	168	25	18	6.2	3.1	1
1	2	66	15.2	7.7	356	321	562	6.5	2.2	0.4
1	2	58	1	0.5	158	37	43	7.2	3.6	1
1	1	40	0.9	0.2	285	32	27	7.7	3.5	0.8
1	2	29	0.7	0.2	165	55	87	7.5	4.6	1.58
1	2	49	1.1	0.5	159	30	31	7	4.3	1.5
1	2	32	0.7	0.2	276	102	190	6	2.9	0.93
1	2	45	1.7	0.8	315	12	38	6.3	2.1	0.5
1	2	33	2.1	1.3	480	38	22	6.5	3	0.8
1	2	50	1.7	0.8	331	36	53	7.3	3.4	0.9
1	2	42	8.9	4.5	272	31	61	5.8	2	0.5
1	2	33	3.4	1.6	186	779	844	7.3	3.2	0.7
1	2	35	0.7	0.2	198	42	30	6.8	3.4	1
1	2	26	1.7	0.6	210	62	56	5.4	2.2	0.6
1	2	48	1.4	0.6	263	38	66	5.8	2.2	0.61
1	2	57	0.7	0.2	208	35	97	5.1	2.1	0.7
1	1	35	0.9	0.3	158	20	16	8	4	1
1	1	28	0.8	0.2	309	55	23	6.8	4.1	1.51
1	2	55	3.6	1.6	349	40	70	7.2	2.9	0.6
1	2	46	10.2	4.2	232	58	140	7	2.7	0.6
1	2	60	5.7	2.8	214	412	850	7.3	3.2	0.78
1	2	40	1.1	0.3	230	1630	960	4.9	2.8	1.3
1	2	60	2.1	1	191	114	247	4	1.6	0.6
1	2	55	18.4	8.8	206	64	178	6.2	1.8	0.4
1	1	31	0.8	0.2	215	15	21	7.6	4	1.1
1	2	45	2.4	1.1	168	33	50	5.1	2.6	1
1	2	48	5.8	2.5	802	133	88	6	2.8	0.8
1	1	40	2.1	1	768	74	141	7.8	4.9	1.6
1	2	37	1.3	0.4	195	41	38	5.3	2.1	0.6
1	2	51	4	2.5	275	382	330	7.5	4	1.1
1	2	55	0.8	0.2	290	139	87	7	3	0.7
1	2	62	1.8	0.9	224	69	155	8.6	4	0.8
1	2	22	2.4	1	340	25	21	8.3	4.5	1.1
1	2	32	25	13.7	560	41	88	7.9	2.5	2.5
1	2	54	2.2	1.2	195	55	95	6	3.7	1.6
1	1	26	0.7	0.2	144	36	33	8.2	4.3	1.1
1	2	55	3.6	1.6	349	40	70	7.2	2.9	0.6
1	1	34	0.6	0.1	161	15	19	6.6	3.4	1
1	2	32	3.7	1.6	612	50	88	6.2	1.9	0.4
1	2	57	4.5	2.3	315	120	105	7	4	1.3
1	1	31	1.1	0.3	190	26	15	7.9	3.8	0.9
1	2	40	0.7	0.1	202	37	29	5	2.6	1

1	2	64	3	1.4	248	46	40	6.5	3.2	0.9
1	1	19	0.7	0.2	186	166	397	5.5	3	1.2
1	2	68	0.7	0.1	145	20	22	5.8	2.9	1
1	2	31	0.9	0.2	518	189	17	5.3	2.3	0.7
1	2	30	1.6	0.4	332	84	139	5.6	2.7	0.9
1	2	60	5.8	2.7	599	43	66	5.4	1.8	0.5
1	2	31	0.8	0.2	198	43	31	7.3	4	1.2
1	1	58	2.4	1.1	915	60	142	4.7	1.8	0.6
1	2	52	0.8	0.2	245	48	49	6.4	3.2	1
1	2	60	11	4.9	750	140	350	5.5	2.1	0.6
1	2	27	1	0.2	205	137	145	6	3	1
1	2	50	0.6	0.2	137	15	16	4.8	2.6	1.1
1	1	46	0.8	0.2	185	24	15	7.9	3.7	0.8
1	2	65	0.9	0.2	170	33	66	7	3	0.75
1	2	35	26.3	12.1	108	168	630	9.2	2	0.3
1	2	47	0.9	0.2	265	40	28	8	4	1
1	1	48	0.8	0.2	150	25	23	7.5	3.9	1
1	1	13	0.7	0.1	182	24	19	8.9	4.9	1.2
1	2	60	6.8	3.2	308	404	794	6.8	3	0.7
1	2	62	10.9	5.5	699	64	100	7.5	3.2	0.74
1	1	45	0.9	0.3	189	23	33	6.6	3.9	
1	2	42	16.4	8.9	245	56	87	5.4	2	0.5
1	2	39	3.8	1.5	298	102	630	7.1	3.3	0.8
1	2	33	1.5	7	505	205	140	7.5	3.9	1
1	2	40	30.8	18.3	285	110	186	7.9	2.7	0.5
1	2	37	1.8	0.8	145	62	58	5.7	2.9	1
1	2	25	0.8	0.1	130	23	42	8	4	1
1	2	65	0.7	0.1	392	20	30	5.3	2.8	1.1
1	2	18	1.8	0.7	178	35	36	6.8	3.6	1.1
1	1	22	2.2	1	215	159	51	5.5	2.5	0.8
1	2	45	2.2	0.8	209	25	20	8	4	1
1	2	50	0.7	0.2	188	12	14	7	3.4	0.9
1	1	7	27.2	11.8	1420	790	1050	6.1	2	0.4
1	2	22	0.8	0.2	198	20	26	6.8	3.9	1.3
1	2	75	0.9	0.2	206	44	33	6.2	2.9	0.8
1	1	31	0.8	0.2	158	21	16	6	3	1
1	2	38	0.9	0.3	310	15	25	5.5	2.7	1
1	2	47	0.9	0.2	265	40	28	8	4	1
1	2	32	18	8.2	298	1250	1050	5.4	2.6	0.9
1	2	38	3.7	2.2	216	179	232	7.8	4.5	1.3
1	2	40	0.7	0.1	202	37	29	5	2.6	1
1	2	60	2.9	1.3	230	32	44	5.6	2	0.5
1	2	69	0.9	0.2	215	32	24	6.9	3	0.7
1	1	42	0.8	0.2	195	18	15	6.7	3	0.8
1	2	49	0.6	0.1	218	50	53	5	2.4	0.9
1	2	51	2.9	1.3	482	22	34	7	2.4	0.5

1	1	45	1	0.3	250	48	44	8.6	4.3	1
1	2	50	0.9	0.3	194	190	73	7.5	3.9	1
1	2	62	7.3	4.1	490	60	68	7	3.3	0.89
1	2	60	8.9	4	950	33	32	6.8	3.1	0.8
1	2	40	0.7	0.1	202	37	29	5	2.6	1
1	2	70	3.1	1.6	198	40	28	5.6	2	0.5
1	2	54	2.2	1.2	195	55	95	6	3.7	1.6
1	2	49	3.9	2.1	189	65	181	6.9	3	0.7
1	2	50	0.6	0.2	137	15	16	4.8	2.6	1.1
1	2	37	0.7	0.2	235	96	54	9.5	4.9	1
1	1	38	0.8	0.2	185	25	21	7	3	0.7
1	2	34	5.9	2.5	290	45	233	5.6	2.7	0.9
1	2	18	0.8	0.2	282	72	140	5.5	2.5	0.8
1	2	14	1.4	0.5	269	58	45	6.7	3.9	1.4
1	2	60	8.9	4	950	33	32	6.8	3.1	0.8
1	1	54	5.5	3.2	350	67	42	7	3.2	0.8
1	2	38	0.7	0.2	110	22	18	6.4	2.5	0.64
1	1	53	0.8	0.2	193	96	57	6.7	3.6	1.16
1	2	48	3.2	1.6	257	33	116	5.7	2.2	0.62
1	2	26	2	0.9	195	24	65	7.8	4.3	1.2
1	2	66	1	0.3	190	30	54	5.3	2.1	0.6
1	1	48	0.8	0.2	142	26	25	6	2.6	0.7
1	2	21	18.5	9.5	380	390	500	8.2	4.1	1
1	2	37	0.7	0.2	235	96	54	9.5	4.9	1
1	2	32	0.7	0.2	276	102	190	6	2.9	0.93
1	2	40	0.6	0.1	98	35	31	6	3.2	1.1
1	2	26	2	0.9	195	24	65	7.8	4.3	1.2
1	2	60	2.4	1	1124	30	54	5.2	1.9	0.5
1	1	42	0.8	0.2	182	22	20	7.2	3.9	1.1
1	2	37	0.7	0.2	176	28	34	5.6	2.6	0.8
1	2	38	1.1	0.3	198	86	150	6.3	3.5	1.2
1	2	45	2.3	1.3	282	132	368	7.3	4	1.2
1	2	38	1.1	0.3	198	86	150	6.3	3.5	1.2
1	1	42	0.8	0.2	195	18	15	6.7	3	0.8
1	2	42	0.8	0.2	127	29	30	4.9	2.7	1.2
1	1	35	1	0.3	805	133	103	7.9	3.3	0.7
1	1	74	0.9	0.3	234	16	19	7.9	4	1
1	1	34	0.6	0.1	161	15	19	6.6	3.4	1
1	2	45	2.5	1.2	163	28	22	7.6	4	1.1
1	2	32	3.7	1.6	612	50	88	6.2	1.9	0.4
1	2	72	0.6	0.1	102	31	35	6.3	3.2	1
1	2	75	1.4	0.4	215	50	30	5.9	2.6	0.7
1	2	41	1.2	0.5	246	34	42	6.9	3.4	0.97
1	1	32	0.6	0.1	176	39	28	6	3	1
1	2	42	0.8	0.2	127	29	30	4.9	2.7	1.2
1	2	70	0.6	0.1	862	76	180	6.3	2.7	0.75

1	1	48	1.4	0.8	621	110	176	7.2	3.9	1.1
1	2	60	22.8	12.6	962	53	41	6.9	3.3	0.9
1	1	45	23.3	12.8	1550	425	511	7.7	3.5	0.8
1	2	58	1	0.5	158	37	43	7.2	3.6	1
1	1	30	0.7	0.2	63	31	27	5.8	3.4	1.4
1	1	46	14.2	7.8	374	38	77	4.3	2	0.8
1	1	51	0.9	0.2	280	21	30	6.7	3.2	0.8
1	1	48	1.4	0.8	621	110	176	7.2	3.9	1.1
1	2	33	7.1	3.7	196	622	497	6.9	3.6	1.09
1	2	18	0.8	0.2	282	72	140	5.5	2.5	0.8
1	2	38	1.7	0.7	859	89	48	6	3	1
1	1	26	0.9	0.2	154	16	12	7	3.5	1
1	2	55	0.9	0.2	190	25	28	5.9	2.7	0.8
1	2	70	3.1	1.6	198	40	28	5.6	2	0.5
1	2	70	2.7	1.2	365	62	55	6	2.4	0.6
1	2	75	8	4.6	386	30	25	5.5	1.8	0.48
1	2	32	3.7	1.6	612	50	88	6.2	1.9	0.4
1	2	37	0.8	0.2	147	27	46	5	2.5	1
1	2	33	2.1	1.3	480	38	22	6.5	3	0.8
1	2	49	0.6	0.1	218	50	53	5	2.4	0.9
1	2	32	32.6	14.1	219	95	235	5.8	3.1	1.1
1	2	22	0.6	0.2	202	78	41	8	3.9	0.9
1	2	54	2.2	1.2	195	55	95	6	3.7	1.6
1	2	40	0.7	0.1	202	37	29	5	2.6	1
1	1	7	27.2	11.8	1420	790	1050	6.1	2	0.4
1	2	26	6.8	3.2	140	37	19	3.6	0.9	0.3
1	2	65	4.9	2.7	190	33	71	7.1	2.9	0.7
1	2	37	0.8	0.2	125	41	39	6.4	3.4	1.1
1	2	34	6.2	3	240	1680	850	7.2	4	1.2
1	2	50	2.7	1.6	157	149	156	7.9	3.1	0.6
1	2	18	1.4	0.6	215	440	850	5	1.9	0.6
1	1	44	1.9	0.6	298	378	602	6.6	3.3	1
1	2	66	0.7	0.2	239	27	26	6.3	3.7	1.4
1	2	66	11.3	5.6	1110	1250	4929	7	2.4	0.5
1	2	55	3.6	1.6	349	40	70	7.2	2.9	0.6
1	1	54	22.6	11.4	558	30	37	7.8	3.4	0.8
1	2	37	0.8	0.2	147	27	46	5	2.5	1
1	1	42	0.5	0.1	162	155	108	8.1	4	0.9
1	2	31	0.6	0.1	175	48	34	6	3.7	1.6
1	2	75	6.7	3.6	458	198	143	6.2	3.2	1
1	2	52	0.6	0.1	171	22	16	6.6	3.6	1.2
1	2	47	0.9	0.2	265	40	28	8	4	1
1	2	52	0.6	0.1	171	22	16	6.6	3.6	1.2
1	2	45	2.8	1.7	263	57	65	5.1	2.3	0.8
1	2	32	32.6	14.1	219	95	235	5.8	3.1	1.1
1	2	30	1.6	0.4	332	84	139	5.6	2.7	0.9

1	2	40	3.6	1.8	285	50	60	7	2.9	0.7
1	2	60	2.6	1.2	171	42	37	5.4	2.7	1
1	2	38	3.1	1.6	253	80	406	6.8	3.9	1.3
1	2	69	0.9	0.2	215	32	24	6.9	3	0.7
1	2	74	1	0.3	175	30	32	6.4	3.4	1.1
1	1	55	8.2	3.9	1350	52	65	6.7	2.9	0.7
1	2	51	4	2.5	275	382	330	7.5	4	1.1
1	2	70	2.7	1.2	365	62	55	6	2.4	0.6
1	2	33	1.2	0.3	498	28	25	7	3	0.7
1	2	34	5.9	2.5	290	45	233	5.6	2.7	0.9
1	1	26	0.7	0.2	144	36	33	8.2	4.3	1.1
1	2	55	75	3.6	332	40	66	6.2	2.5	0.6
1	2	55	4.4	2.9	230	14	25	7.1	2.1	0.4
1	2	52	1.8	0.8	97	85	78	6.4	2.7	0.7
1	2	40	3.5	1.6	298	68	200	7.1	3.4	0.9
1	2	40	1.1	0.3	230	1630	960	4.9	2.8	1.3
1	2	21	3.9	1.8	150	36	27	6.8	3.9	1.34
1	2	40	0.7	0.1	202	37	29	5	2.6	1
1	2	61	0.8	0.1	282	85	231	8.5	4.3	1
1	2	75	8	4.6	386	30	25	5.5	1.8	0.48
1	1	46	1.4	0.4	298	509	623	3.6	1	0.3
1	2	65	0.7	0.1	392	20	30	5.3	2.8	1.1
1	1	46	1.4	0.4	298	509	623	3.6	1	0.3
1	1	45	0.7	0.2	170	21	14	5.7	2.5	0.7
1	2	34	8.7	4	298	58	138	5.8	2.4	0.7
1	2	75	14.8	9	1020	71	42	5.3	2.2	0.7
1	2	37	0.8	0.2	125	41	39	6.4	3.4	1.1
1	2	75	0.9	0.2	162	25	20	6.9	3.7	1.1
1	2	74	1	0.3	175	30	32	6.4	3.4	1.1
1	2	70	0.6	0.1	862	76	180	6.3	2.7	0.75
1	2	43	22.5	11.8	143	22	143	6.6	2.1	0.46
1	2	18	0.6	0.1	265	97	161	5.9	3.1	1.1
1	2	40	3.6	1.8	285	50	60	7	2.9	0.7
1	2	58	0.4	0.1	100	59	126	4.3	2.5	1.4
1	2	32	12.7	8.4	190	28	47	5.4	2.6	0.9
1	2	28	0.9	0.2	215	50	28	8	4	1
1	2	49	1.1	0.5	159	30	31	7	4.3	1.5
1	2	49	1	0.3	230	48	58	8.4	4.2	1
1	1	31	0.8	0.2	215	15	21	7.6	4	1.1
1	2	50	1.6	0.8	218	18	20	5.9	2.9	0.96
1	1	48	0.9	0.2	173	26	27	6.2	3.1	1
1	2	58	0.4	0.1	100	59	126	4.3	2.5	1.4
1	2	56	17.7	8.8	239	43	185	5.6	2.4	0.7
1	1	58	1.7	0.8	1896	61	83	8	3.9	0.95
1	2	50	0.9	0.2	202	20	26	7.2	4.5	1.66
1	2	53	0.9	0.4	238	17	14	6.6	2.9	0.8

1	2	46	0.8	0.2	160	31	40	7.3	3.8	1.1
1	2	34	4.1	2	289	875	731	5	2.7	1.1
1	2	38	1.1	0.3	198	86	150	6.3	3.5	1.2
1	2	42	0.8	0.2	127	29	30	4.9	2.7	1.2
1	2	65	1.1	0.5	686	16	46	5.7	1.5	0.35
1	2	33	1.8	0.8	196	25	22	8	4	1
1	2	75	0.9	0.2	282	25	23	4.4	2.2	1
1	2	55	0.8	0.2	290	139	87	7	3	0.7
1	1	34	0.6	0.1	161	15	19	6.6	3.4	1
1	2	37	1.3	0.4	195	41	38	5.3	2.1	0.6
1	2	26	6.8	3.2	140	37	19	3.6	0.9	0.3
1	2	38	0.8	0.2	208	25	50	7.1	3.7	1
1	2	12	1	0.2	719	157	108	7.2	3.7	1
1	2	48	5	2.6	555	284	190	6.5	3.3	1
1	1	66	4.2	2.1	159	15	30	7.1	2.2	0.4
1	2	34	4.1	2	289	875	731	5	2.7	1.1
1	1	55	10.9	5.1	1350	48	57	6.4	2.3	0.5
1	2	26	0.6	0.2	120	45	51	7.9	4	1
1	2	78	1	0.3	152	28	70	6.3	3.1	0.9
1	1	40	0.9	0.2	285	32	27	7.7	3.5	0.8
1	2	32	0.9	0.3	462	70	82	6.2	3.1	1
1	2	34	6.2	3	240	1680	850	7.2	4	1.2
1	2	56	17.7	8.8	239	43	185	5.6	2.4	0.7
1	2	52	0.8	0.2	245	48	49	6.4	3.2	1
1	2	57	4	1.9	190	45	111	5.2	1.5	0.4
1	2	49	1.1	0.5	159	30	31	7	4.3	1.5
1	2	36	2.8	1.5	305	28	76	5.9	2.5	0.7
1	2	18	0.7	0.1	312	308	405	6.9	3.7	1.1
1	2	74	1	0.3	175	30	32	6.4	3.4	1.1
1	2	69	0.9	0.2	215	32	24	6.9	3	0.7
1	2	22	0.6	0.2	202	78	41	8	3.9	0.9
1	2	45	2.2	1.6	320	37	48	6.8	3.4	1
1	2	58	0.8	0.2	298	33	59	6.2	3.1	1
1	2	51	0.8	0.2	175	48	22	8.1	4.6	1.3
1	2	75	2.5	1.2	375	85	68	6.4	2.9	0.8
1	2	37	0.8	0.2	125	41	39	6.4	3.4	1.1
1	2	62	6.8	3	542	116	66	6.4	3.1	0.9
1	2	73	1.9	0.7	1750	102	141	5.5	2	0.5
1	2	26	2	0.9	157	54	68	6.1	2.7	0.8
1	2	60	2.3	0.6	272	79	51	6.6	3.5	1.1
1	2	32	12.1	6	515	48	92	6.6	2.4	0.5
1	1	26	0.7	0.2	144	36	33	8.2	4.3	1.1
1	2	65	1.1	0.5	686	16	46	5.7	1.5	0.35
1	1	40	0.9	0.3	293	232	245	6.8	3.1	0.8
1	1	19	0.7	0.2	186	166	397	5.5	3	1.2
1	2	66	0.7	0.2	239	27	26	6.3	3.7	1.4

1	2	62	1.8	0.9	224	69	155	8.6	4	0.8
1	2	70	1.3	0.4	358	19	14	6.1	2.8	0.8
1	2	72	0.6	0.1	102	31	35	6.3	3.2	1
1	1	10	0.8	0.1	395	25	75	7.6	3.6	0.9
1	2	33	7.1	3.7	196	622	497	6.9	3.6	1.09
1	2	26	1.3	0.4	173	38	62	8	4	1
1	2	70	1.3	0.3	690	93	40	3.6	2.7	0.7
1	2	65	7.9	4.3	282	50	72	6	3	1
1	2	31	0.8	0.2	198	43	31	7.3	4	1.2
1	1	29	0.8	0.2	205	30	23	8.2	4.1	1
1	1	68	0.6	0.1	1620	95	127	4.6	2.1	0.8
1	2	18	0.8	0.2	282	72	140	5.5	2.5	0.8
1	2	48	1.4	0.6	263	38	66	5.8	2.2	0.61
1	2	50	1.6	0.8	218	18	20	5.9	2.9	0.96
1	2	55	0.9	0.2	190	25	28	5.9	2.7	0.8
1	1	55	8.2	3.9	1350	52	65	6.7	2.9	0.7
1	2	75	1.8	0.8	405	79	50	6.1	2.9	0.9
1	2	32	18	8.2	298	1250	1050	5.4	2.6	0.9
1	2	18	0.6	0.2	538	33	34	7.5	3.2	0.7
1	2	49	0.6	0.1	218	50	53	5	2.4	0.9
1	2	32	25	13.7	560	41	88	7.9	2.5	2.5
1	2	16	2.6	1.2	236	131	90	5.4	2.6	0.9
1	2	34	3.7	2.1	490	115	91	6.5	2.8	0.7
1	1	7	27.2	11.8	1420	790	1050	6.1	2	0.4
1	1	48	1.1	0.7	527	178	250	8	4.2	1.1
1	2	30	1.3	0.4	482	102	80	6.9	3.3	0.9
1	1	46	4.7	2.2	310	62	90	6.4	2.5	0.6
1	2	37	0.8	0.2	125	41	39	6.4	3.4	1.1
1	2	75	2.9	1.3	218	33	37	3	1.5	1
1	2	65	4.9	2.7	190	33	71	7.1	2.9	0.7
1	2	60	2.1	1	191	114	247	4	1.6	0.6
1	1	48	0.8	0.2	142	26	25	6	2.6	0.7
1	1	45	3.5	1.5	189	63	87	5.6	2.9	1
1	2	38	0.7	0.2	216	349	105	7	3.5	1
1	2	57	4	1.9	190	45	111	5.2	1.5	0.4
1	2	41	1.2	0.5	246	34	42	6.9	3.4	0.97
1	1	68	0.6	0.1	1620	95	127	4.6	2.1	0.8
1	1	31	0.8	0.2	215	15	21	7.6	4	1.1
1	2	50	4.2	2.3	450	69	50	7	3	0.7
1	2	46	15.8	7.2	227	67	220	6.9	2.6	0.6
1	2	48	4.5	2.3	282	13	74	7	2.4	0.52
1	2	57	1.3	0.4	259	40	86	6.5	2.5	0.6
1	1	58	0.7	0.1	172	27	22	6.7	3.2	0.9
1	1	22	2.2	1	215	159	51	5.5	2.5	0.8
1	2	55	0.8	0.2	290	139	87	7	3	0.7
1	2	46	0.6	0.2	290	26	21	6	3	1

1	2	51	2.9	1.2	189	80	125	6.2	3.1	1
1	1	28	0.9	0.2	316	25	23	8.5	5.5	1.8
1	1	20	16.7	8.4	200	91	101	6.9	3.5	1.02
1	2	22	0.6	0.2	202	78	41	8	3.9	0.9
1	2	33	0.7	0.1	168	35	33	7	3.7	1.1
1	2	50	1.6	0.8	218	18	20	5.9	2.9	0.96
1	1	64	0.8	0.2	178	17	18	6.3	3.1	0.9
1	2	40	30.8	18.3	285	110	186	7.9	2.7	0.5
1	2	31	0.8	0.2	198	43	31	7.3	4	1.2
1	2	66	1.1	0.5	167	13	56	7.1	4.1	1.36
1	2	21	3.9	1.8	150	36	27	6.8	3.9	1.34
1	2	21	18.5	9.5	380	390	500	8.2	4.1	1
1	1	58	0.7	0.1	172	27	22	6.7	3.2	0.9
1	2	60	6.3	3.2	314	118	114	6.6	3.7	1.27
1	1	54	22.6	11.4	558	30	37	7.8	3.4	0.8
1	2	34	4.1	2	289	875	731	5	2.7	1.1
1	2	34	4.1	2	289	875	731	5	2.7	1.1
1	2	38	3.1	1.6	253	80	406	6.8	3.9	1.3
1	2	60	5.8	2.7	204	220	400	7	3	0.7
1	2	18	0.6	0.1	265	97	161	5.9	3.1	1.1
1	1	38	0.8	0.2	185	25	21	7	3	0.7
1	1	31	0.8	0.2	158	21	16	6	3	1
1	2	46	0.6	0.2	115	14	11	6.9	3.4	0.9
1	1	42	0.5	0.1	162	155	108	8.1	4	0.9
1	2	75	10.6	5	562	37	29	5.1	1.8	0.5
1	2	26	1.7	0.6	210	62	56	5.4	2.2	0.6
1	2	38	3.1	1.6	253	80	406	6.8	3.9	1.3
1	2	60	11	4.9	750	140	350	5.5	2.1	0.6
1	2	51	0.8	0.2	230	24	46	6.5	3.1	
1	2	75	8	4.6	386	30	25	5.5	1.8	0.48
1	2	66	17.3	8.5	388	173	367	7.8	2.6	0.5
1	2	50	2.7	1.6	157	149	156	7.9	3.1	0.6
1	2	37	0.7	0.2	235	96	54	9.5	4.9	1
1	2	60	5.8	3	257	107	104	6.6	3.5	1.12
1	1	58	1.7	0.8	1896	61	83	8	3.9	0.95
1	2	56	17.7	8.8	239	43	185	5.6	2.4	0.7
1	2	45	2.2	1.6	320	37	48	6.8	3.4	1
1	2	40	0.6	0.1	171	20	17	5.4	2.5	0.8
1	2	60	6.3	3.2	314	118	114	6.6	3.7	1.27
1	2	32	30.5	17.1	218	39	79	5.5	2.7	0.9
1	2	57	1.4	0.7	470	62	88	5.6	2.5	0.8
1	2	45	3.2	1.4	512	50	58	6	2.7	0.8
1	2	51	0.7	0.1	180	25	27	6.1	3.1	1
1	2	52	1.8	0.8	97	85	78	6.4	2.7	0.7
1	2	18	0.6	0.1	265	97	161	5.9	3.1	1.1
1	2	72	2.7	1.3	260	31	56	7.4	3	0.6

1	2	68	1.8	0.5	151	18	22	6.5	4	1.6
1	1	54	23.2	12.6	574	43	47	7.2	3.5	0.9
1	2	33	2	1	258	194	152	5.4	3	1.25
1	2	70	0.6	0.1	862	76	180	6.3	2.7	0.75
1	2	60	2.4	1	1124	30	54	5.2	1.9	0.5
1	1	66	4.2	2.1	159	15	30	7.1	2.2	0.4
1	2	52	0.9	0.2	156	35	44	4.9	2.9	1.4
1	2	45	2.9	1.4	210	74	68	7.2	3.6	1
1	1	22	6.7	3.2	850	154	248	6.2	2.8	0.8
1	2	28	0.8	0.3	190	20	14	4.1	2.4	1.4
1	2	21	18.5	9.5	380	390	500	8.2	4.1	1
1	2	60	22.8	12.6	962	53	41	6.9	3.3	0.9
1	2	55	0.6	0.2	220	24	32	5.1	2.4	0.88
1	2	28	0.9	0.2	215	50	28	8	4	1
1	1	54	22.6	11.4	558	30	37	7.8	3.4	0.8
1	2	62	10.9	5.5	699	64	100	7.5	3.2	0.74
1	2	22	0.8	0.2	198	20	26	6.8	3.9	1.3
1	2	42	8.9	4.5	272	31	61	5.8	2	0.5
1	2	60	11	4.9	750	140	350	5.5	2.1	0.6
1	2	38	0.9	0.3	310	15	25	5.5	2.7	1
1	2	57	1.3	0.4	259	40	86	6.5	2.5	0.6
1	1	32	0.6	0.1	176	39	28	6	3	1
1	2	34	8.7	4	298	58	138	5.8	2.4	0.7
1	2	62	1.8	0.9	224	69	155	8.6	4	0.8
1	2	26	7.1	3.3	258	80	113	6.2	2.9	0.8
1	2	32	0.6	0.1	237	45	31	7.5	4.3	1.34
1	1	58	2.8	1.3	670	48	79	4.7	1.6	0.5
1	2	34	5.9	2.5	290	45	233	5.6	2.7	0.9
1	2	30	0.7	0.2	262	15	18	9.6	4.7	1.2
1	2	38	1.7	0.7	859	89	48	6	3	1
1	1	68	0.6	0.1	1620	95	127	4.6	2.1	0.8
1	2	75	2.8	1.3	250	23	29	2.7	0.9	0.5
1	2	75	0.9	0.2	206	44	33	6.2	2.9	0.8
1	1	31	1.1	0.3	190	26	15	7.9	3.8	0.9
1	2	29	0.7	0.2	165	55	87	7.5	4.6	1.58
1	2	39	3.8	1.5	298	102	630	7.1	3.3	0.8
1	1	28	0.9	0.2	316	25	23	8.5	5.5	1.8
1	2	31	0.6	0.1	175	48	34	6	3.7	1.6
1	2	26	7.1	3.3	258	80	113	6.2	2.9	0.8
1	2	32	18	8.2	298	1250	1050	5.4	2.6	0.9
1	2	32	18	8.2	298	1250	1050	5.4	2.6	0.9
1	2	62	1.8	0.9	224	69	155	8.6	4	0.8
1	2	74	0.6	0.1	272	24	98	5	2	0.6
1	2	49	0.6	0.1	218	50	53	5	2.4	0.9
1	2	52	0.6	0.1	171	22	16	6.6	3.6	1.2
1	2	18	0.9	0.3	300	30	48	8	4	1

1	1	13	0.7	0.1	182	24	19	8.9	4.9	1.2
1	1	74	0.9	0.3	234	16	19	7.9	4	1
1	1	32	0.7	0.1	240	12	15	7	3	0.7
1	2	40	1.2	0.6	204	23	27	7.6	4	1.1
1	1	40	0.9	0.2	285	32	27	7.7	3.5	0.8
1	2	46	0.8	0.2	160	31	40	7.3	3.8	1.1
1	2	39	3.8	1.5	298	102	630	7.1	3.3	0.8
1	2	75	0.9	0.2	162	25	20	6.9	3.7	1.1
1	2	22	0.8	0.2	198	20	26	6.8	3.9	1.3
1	2	64	1.4	0.5	298	31	83	7.2	2.6	0.5
1	2	32	0.6	0.1	237	45	31	7.5	4.3	1.34
1	1	50	1	0.5	239	16	39	7.5	3.7	0.9
1	2	75	2.8	1.3	250	23	29	2.7	0.9	0.5
1	2	47	2.7	1.3	275	123	73	6.2	3.3	1.1
1	2	45	2.3	1.3	282	132	368	7.3	4	1.2
1	2	64	3	1.4	248	46	40	6.5	3.2	0.9
1	2	67	2.2	1.1	198	42	39	7.2	3	0.7
1	2	73	1.8	0.9	220	20	43	6.5	3	0.8
1	2	65	0.8	0.2	162	30	90	3.8	1.4	0.5
1	2	24	1	0.2	189	52	31	8	4.8	1.5
1	2	55	14.1	7.6	750	35	63	5	1.6	0.47
1	2	60	6.3	3.2	314	118	114	6.6	3.7	1.27
1	2	55	3.3	1.5	214	54	152	5.1	1.8	0.5
1	2	72	0.7	0.1	196	20	35	5.8	2	0.5
1	1	22	6.7	3.2	850	154	248	6.2	2.8	0.8
1	1	47	3	1.5	292	64	67	5.6	1.8	0.47
1	2	60	2.6	1.2	171	42	37	5.4	2.7	1
1	1	40	0.9	0.2	285	32	27	7.7	3.5	0.8
1	1	46	4.7	2.2	310	62	90	6.4	2.5	0.6
1	1	42	2.3	1.1	292	29	39	4.1	1.8	0.7
1	2	16	2.6	1.2	236	131	90	5.4	2.6	0.9
1	1	22	6.7	3.2	850	154	248	6.2	2.8	0.8
1	2	72	2.7	1.3	260	31	56	7.4	3	0.6
1	2	48	0.7	0.2	326	29	17	8.7	5.5	1.7
1	1	35	1	0.3	805	133	103	7.9	3.3	0.7
1	2	54	0.8	0.2	218	20	19	6.3	2.5	0.6
1	2	47	2.7	1.3	275	123	73	6.2	3.3	1.1
1	1	50	1	0.5	239	16	39	7.5	3.7	0.9
1	2	42	0.8	0.2	127	29	30	4.9	2.7	1.2
1	2	50	0.9	0.3	901	23	17	6.2	3.5	1.2
1	1	31	1.1	0.3	190	26	15	7.9	3.8	0.9
1	2	30	0.8	0.2	174	21	47	4.6	2.3	1
1	2	62	7.3	4.1	490	60	68	7	3.3	0.89
1	2	72	1.7	0.8	200	28	37	6.2	3	0.93
1	2	40	0.6	0.1	98	35	31	6	3.2	1.1
1	2	60	2.6	1.2	171	42	37	5.4	2.7	1

1	2	60	0.8	0.2	286	21	27	7.1	4	1.2
1	2	45	2.5	1.2	163	28	22	7.6	4	1.1
1	2	68	1.8	0.5	151	18	22	6.5	4	1.6
1	2	55	3.6	1.6	349	40	70	7.2	2.9	0.6
1	2	40	1.9	1	231	16	55	4.3	1.6	0.6
1	1	45	1	0.3	250	48	44	8.6	4.3	1
1	2	18	1.8	0.7	178	35	36	6.8	3.6	1.1
1	2	24	1	0.2	189	52	31	8	4.8	1.5
1	2	31	0.8	0.2	198	43	31	7.3	4	1.2
1	2	48	0.7	0.1	1630	74	149	5.3	2	0.6
1	2	40	14.5	6.4	358	50	75	5.7	2.1	0.5
1	1	44	1.9	0.6	298	378	602	6.6	3.3	1
1	2	75	2.5	1.2	375	85	68	6.4	2.9	0.8
1	2	35	0.7	0.2	198	42	30	6.8	3.4	1
1	2	45	2.8	1.7	263	57	65	5.1	2.3	0.8
1	2	44	0.8	0.2	335	148	86	5.6	3	1.1
1	2	48	3.2	1.6	257	33	116	5.7	2.2	0.62
1	2	60	8.9	4	950	33	32	6.8	3.1	0.8
1	2	40	1.2	0.6	204	23	27	7.6	4	1.1
1	2	75	2.8	1.3	250	23	29	2.7	0.9	0.5
1	2	54	0.8	0.2	218	20	19	6.3	2.5	0.6
1	1	66	4.2	2.1	159	15	30	7.1	2.2	0.4
1	1	46	0.8	0.2	182	20	40	6	2.9	0.9
1	2	57	4.5	2.3	315	120	105	7	4	1.3
1	2	46	18.4	8.5	450	119	230	7.5	3.3	0.7
1	2	55	0.6	0.2	220	24	32	5.1	2.4	0.88
1	1	45	23.3	12.8	1550	425	511	7.7	3.5	0.8
1	2	57	1.3	0.4	259	40	86	6.5	2.5	0.6
1	2	40	3.6	1.8	285	50	60	7	2.9	0.7
1	2	50	0.7	0.2	188	12	14	7	3.4	0.9
1	2	45	2.2	0.8	209	25	20	8	4	1
1	2	32	30.5	17.1	218	39	79	5.5	2.7	0.9
1	2	66	1	0.3	190	30	54	5.3	2.1	0.6
1	2	60	2.9	1.3	230	32	44	5.6	2	0.5
1	2	32	15.9	7	280	1350	1600	5.6	2.8	1
1	1	58	2.4	1.1	915	60	142	4.7	1.8	0.6
1	2	56	17.7	8.8	239	43	185	5.6	2.4	0.7
1	2	55	75	3.6	332	40	66	6.2	2.5	0.6
1	2	64	1.1	0.4	201	18	19	6.9	4.1	1.4
1	2	72	3.9	2	195	27	59	7.3	2.4	0.4
1	1	50	1.7	0.6	430	28	32	6.8	3.5	1
1	2	58	1	0.5	158	37	43	7.2	3.6	1
1	1	74	1.1	0.4	214	22	30	8.1	4.1	1
1	2	60	22.8	12.6	962	53	41	6.9	3.3	0.9
1	2	50	0.6	0.2	137	15	16	4.8	2.6	1.1
1	2	75	2.9	1.3	218	33	37	3	1.5	1

1	2	48	3.2	1.6	257	33	116	5.7	2.2	0.62
1	1	35	1	0.3	805	133	103	7.9	3.3	0.7
1	2	90	1.1	0.3	215	46	134	6.9	3	0.7
1	2	75	1.4	0.4	215	50	30	5.9	2.6	0.7
1	2	18	0.9	0.3	300	30	48	8	4	1
1	2	57	0.7	0.2	208	35	97	5.1	2.1	0.7
1	1	42	0.8	0.2	168	25	18	6.2	3.1	1
1	2	37	0.8	0.2	125	41	39	6.4	3.4	1.1
1	1	36	0.8	0.2	650	70	138	6.6	3.1	0.8
1	2	49	1	0.3	230	48	58	8.4	4.2	1
1	2	57	0.6	0.1	210	51	59	5.9	2.7	0.8
1	2	45	2.3	1.3	282	132	368	7.3	4	1.2
1	1	54	22.6	11.4	558	30	37	7.8	3.4	0.8
1	2	61	0.7	0.2	145	53	41	5.8	2.7	0.87
1	2	66	0.8	0.2	165	22	32	4.4	2	0.8
1	2	62	7.3	4.1	490	60	68	7	3.3	0.89
1	2	60	0.9	0.3	168	16	24	6.7	3	0.8
1	2	57	4.5	2.3	315	120	105	7	4	1.3
1	2	60	8.6	4	298	412	850	7.4	3	0.6
1	2	75	0.9	0.2	162	25	20	6.9	3.7	1.1
1	2	49	0.6	0.1	218	50	53	5	2.4	0.9
1	2	55	1.8	9	272	22	79	6.1	2.7	0.7
1	2	32	30.5	17.1	218	39	79	5.5	2.7	0.9
1	2	47	0.9	0.2	192	38	24	7.3	4.3	1.4
1	2	35	26.3	12.1	108	168	630	9.2	2	0.3
1	2	72	0.7	0.1	196	20	35	5.8	2	0.5
1	2	60	5.8	2.7	204	220	400	7	3	0.7
1	2	42	8.9	4.5	272	31	61	5.8	2	0.5
1	2	46	9.4	5.2	268	21	63	6.4	2.8	0.8
1	2	13	1.5	0.5	575	29	24	7.9	3.9	0.9
1	2	60	3.2	1.8	750	79	145	7.8	3.2	0.69
1	2	55	75	3.6	332	40	66	6.2	2.5	0.6
1	2	50	0.9	0.3	901	23	17	6.2	3.5	1.2
1	2	32	25	13.7	560	41	88	7.9	2.5	2.5
1	1	28	0.9	0.2	316	25	23	8.5	5.5	1.8
1	2	37	0.8	0.2	147	27	46	5	2.5	1
1	2	60	1.5	0.6	360	230	298	4.5	2	0.8
1	2	70	0.6	0.1	862	76	180	6.3	2.7	0.75
1	2	33	0.9	0.8	680	37	40	5.9	2.6	0.8
1	2	50	0.9	0.2	202	20	26	7.2	4.5	1.66
1	2	40	3.9	1.7	350	950	1500	6.7	3.8	1.3
1	2	50	4.2	2.3	450	69	50	7	3	0.7
1	2	30	1.6	0.4	332	84	139	5.6	2.7	0.9
1	2	64	1.1	0.4	201	18	19	6.9	4.1	1.4
1	2	33	2	1	258	194	152	5.4	3	1.25
1	2	72	0.8	0.2	148	23	35	6	3	1

1	2	53	19.8	10.4	238	39	221	8.1	2.5	0.4
1	2	37	1.3	0.4	195	41	38	5.3	2.1	0.6
1	2	56	17.7	8.8	239	43	185	5.6	2.4	0.7
1	1	32	0.7	0.1	240	12	15	7	3	0.7
1	2	32	18	8.2	298	1250	1050	5.4	2.6	0.9
1	1	42	0.5	0.1	162	155	108	8.1	4	0.9
1	1	34	0.8	0.2	192	15	12	8.6	4.7	1.2
1	1	7	27.2	11.8	1420	790	1050	6.1	2	0.4
1	2	40	14.5	6.4	358	50	75	5.7	2.1	0.5
1	2	32	0.7	0.2	276	102	190	6	2.9	0.93
1	2	29	1	0.3	75	25	26	5.1	2.9	1.3
1	2	26	7.1	3.3	258	80	113	6.2	2.9	0.8
1	2	75	10.6	5	562	37	29	5.1	1.8	0.5
1	2	26	7.1	3.3	258	80	113	6.2	2.9	0.8
1	1	34	0.6	0.1	161	15	19	6.6	3.4	1
1	2	65	1.1	0.5	686	16	46	5.7	1.5	0.35
1	2	26	1.3	0.4	173	38	62	8	4	1
1	2	57	0.7	0.2	208	35	97	5.1	2.1	0.7
1	2	28	0.9	0.2	215	50	28	8	4	1
1	2	60	22.8	12.6	962	53	41	6.9	3.3	0.9
1	2	70	2.7	1.2	365	62	55	6	2.4	0.6
1	2	37	0.8	0.2	125	41	39	6.4	3.4	1.1
1	2	34	4.1	2	289	875	731	5	2.7	1.1
1	2	46	0.7	0.2	224	40	23	7.1	3	0.7
1	2	37	0.7	0.2	235	96	54	9.5	4.9	1
1	2	41	2.7	1.3	580	142	68	8	4	1
1	2	18	0.7	0.1	312	308	405	6.9	3.7	1.1
1	2	66	16.6	7.6	315	233	384	6.9	2	0.4
2	2	17	0.9	0.3	202	22	19	7.4	4.1	1.2
2	2	64	0.9	0.3	310	61	58	7	3.4	0.9
2	2	25	0.6	0.1	183	91	53	5.5	2.3	0.7
2	2	33	1.6	0.5	165	15	23	7.3	3.5	0.92
2	2	63	0.9	0.2	194	52	45	6	3.9	1.85
2	2	20	1.1	0.5	128	20	30	3.9	1.9	0.95
2	1	84	0.7	0.2	188	13	21	6	3.2	1.1
2	2	57	1	0.3	187	19	23	5.2	2.9	1.2
2	1	38	2.6	1.2	410	59	57	5.6	3	0.8
2	1	38	2.6	1.2	410	59	57	5.6	3	0.8
2	1	17	0.7	0.2	145	18	36	7.2	3.9	1.18
2	2	62	0.6	0.1	160	42	110	4.9	2.6	1.1
2	2	42	6.8	3.2	630	25	47	6.1	2.3	0.6
2	1	85	1	0.3	208	17	15	7	3.6	1
2	2	35	1.8	0.6	275	48	178	6.5	3.2	0.9
2	2	33	0.8	0.2	198	26	23	8	4	1
2	1	48	0.9	0.2	175	24	54	5.5	2.7	0.9
2	2	64	1.1	0.5	145	20	24	5.5	3.2	1.39

2	2	60	0.8	0.2	215	24	17	6.3	3	0.9
2	1	29	0.7	0.1	162	52	41	5.2	2.5	0.9
2	2	70	1.4	0.6	146	12	24	6.2	3.8	1.58
2	2	49	0.7	0.1	148	14	12	5.4	2.8	1
2	2	13	0.6	0.1	320	28	56	7.2	3.6	1
2	2	27	0.6	0.2	161	27	28	3.7	1.6	0.76
2	2	27	0.7	0.2	243	21	23	5.3	2.3	0.7
2	1	55	0.8	0.2	225	14	23	6.1	3.3	1.2
2	2	36	5.3	2.3	145	32	92	5.1	2.6	1
2	2	36	5.3	2.3	145	32	92	5.1	2.6	1
2	2	36	0.8	0.2	158	29	39	6	2.2	0.5
2	2	36	0.8	0.2	158	29	39	6	2.2	0.5
2	2	36	0.9	0.1	486	25	34	5.9	2.8	0.9
2	1	24	0.7	0.2	188	11	10	5.5	2.3	0.71
2	2	27	1.2	0.4	179	63	39	6.1	3.3	1.1
2	2	50	5.8	3	661	181	285	5.7	2.3	0.67
2	2	50	7.3	3.6	1580	88	64	5.6	2.3	0.6
2	2	58	1.7	0.8	188	60	84	5.9	3.5	1.4
2	2	28	0.6	0.1	177	36	29	6.9	4.1	1.4
2	2	60	1.8	0.5	201	45	25	3.9	1.7	0.7
2	1	70	0.7	0.2	237	18	28	5.8	2.5	0.75
2	1	18	0.8	0.2	199	34	31	6.5	3.5	1.16
2	2	60	0.6	0.1	186	20	21	6.2	3.3	1.1
2	2	65	0.8	0.2	201	18	22	5.4	2.9	1.1
2	2	56	1.1	0.5	180	30	42	6.9	3.8	1.2
2	2	52	0.6	0.1	178	26	27	6.5	3.6	1.2
2	2	65	1.9	0.8	170	36	43	3.8	1.4	0.58
2	2	38	1.5	0.4	298	60	103	6	3	1
2	1	48	0.8	0.2	218	32	28	5.2	2.5	0.9
2	2	49	1.3	0.4	206	30	25	6	3.1	1.06
2	2	49	2	0.6	209	48	32	5.7	3	1.1
2	2	41	0.9	0.2	169	22	18	6.1	3	0.9
2	1	38	0.8	0.2	145	19	23	6.1	3.1	1.03
2	2	21	1	0.3	142	27	21	6.4	3.5	1.2
2	2	21	0.7	0.2	135	27	26	6.4	3.3	1
2	2	22	2.7	1	160	82	127	5.5	3.1	1.2
2	2	66	0.6	0.2	100	17	148	5	3.3	1.9
2	2	55	0.9	0.2	116	36	16	6.2	3.2	1
2	2	6	0.6	0.1	289	38	30	4.8	2	0.7
2	2	50	1.1	0.3	175	20	19	7.1	4.5	1.7
2	1	65	1	0.3	202	26	13	5.3	2.6	0.9
2	2	61	1.5	0.6	196	61	85	6.7	3.8	1.3
2	2	22	0.8	0.2	300	57	40	7.9	3.8	0.9
2	1	35	0.9	0.2	190	40	35	7.3	4.7	1.8
2	2	48	0.7	0.2	165	32	30	8	4	1
2	2	65	1.1	0.3	258	48	40	7	3.9	1.2

2	1	35	0.6	0.2	180	12	15	5.2	2.7	
2	1	38	0.7	0.1	152	90	21	7.1	4.2	1.4
2	2	36	0.8	0.2	182	31	34	6.4	3.8	1.4
2	2	38	0.8	0.2	247	55	92	7.4	4.3	1.38
2	2	4	0.9	0.2	348	30	34	8	4	1
2	2	4	0.8	0.2	460	152	231	6.5	3.2	0.9
2	2	26	1.9	0.8	180	22	19	8.2	4.1	1
2	2	35	0.9	0.2	190	25	20	6.4	3.6	1.2
2	2	50	0.7	0.2	192	18	15	7.4	4.2	1.3
2	2	18	1.3	0.7	316	10	21	6	2.1	0.5
2	2	43	1.3	0.6	155	15	20	8	4	1
2	2	60	0.7	0.2	174	32	14	7.8	4.2	1.1
2	2	23	1.1	0.5	191	37	41	7.7	4.3	1.2
2	1	25	0.9	0.3	159	24	25	6.9	4.4	1.7
2	1	24	0.9	0.2	195	40	35	7.4	4.1	1.2
2	2	58	0.8	0.2	180	32	25	8.2	4.4	1.1
2	2	50	0.7	0.2	206	18	17	8.4	4.2	1
2	1	54	1.4	0.7	195	36	16	7.9	3.7	0.9
2	2	27	1.3	0.6	106	25	54	8.5	4.8	
2	1	30	0.8	0.2	158	25	22	7.9	4.5	1.3
2	2	22	0.9	0.3	179	18	21	6.7	3.7	1.2
2	2	44	0.9	0.2	182	29	82	7.1	3.7	1
2	2	14	0.9	0.3	310	21	16	8.1	4.2	1
2	2	12	0.8	0.2	302	47	67	6.7	3.5	1.1
2	2	42	0.8	0.2	158	27	23	6.7	3.1	0.8
2	1	36	1.2	0.4	358	160	90	8.3	4.4	1.1
2	2	24	3.3	1.6	174	11	33	7.6	3.9	1
2	2	43	0.8	0.2	192	29	20	6	2.9	0.9
2	2	21	0.7	0.2	211	14	23	7.3	4.1	1.2
2	1	36	0.7	0.2	152	21	25	5.9	3.1	1.1
2	2	35	0.8	0.2	198	36	32	7	4	1.3
2	2	37	0.8	0.2	195	60	40	8.2	5	1.5
2	1	49	0.8	0.2	158	19	15	6.6	3.6	1.2
2	2	19	1.4	0.8	178	13	26	8	4.6	1.3
2	1	69	0.8	0.2	146	42	70	8.4	4.9	1.4
2	1	65	0.7	0.2	182	23	28	6.8	2.9	0.7
2	2	55	1.1	0.3	215	21	15	6.2	2.9	0.8
2	1	42	0.9	0.2	165	26	29	8.5	4.4	1
2	2	21	0.8	0.2	183	33	57	6.8	3.5	1
2	2	40	0.7	0.2	176	28	43	5.3	2.4	0.8
2	2	16	0.7	0.2	418	28	35	7.2	4.1	1.3
2	2	60	2.2	1	271	45	52	6.1	2.9	0.9
2	2	33	0.8	0.2	135	30	29	7.2	4.4	1.5
2	1	25	0.7	0.1	140	32	25	7.6	4.3	1.3
2	1	56	0.7	0.1	145	26	23	7	4	1.3
2	1	20	0.6	0.2	202	12	13	6.1	3	0.9

2	2	72	0.7	0.2	185	16	22	7.3	3.7	1
2	1	60	1.4	0.7	159	10	12	4.9	2.5	1
2	2	38	2.7	1.4	105	25	21	7.5	4.2	1.2
2	2	45	0.8	0.2	140	24	20	6.3	3.2	1
2	1	66	0.7	0.2	162	24	20	6.4	3.2	1
2	2	65	0.7	0.2	199	19	22	6.3	3.6	1.3
2	2	45	0.7	0.2	180	18	58	6.7	3.7	1.2
2	1	23	2.3	0.8	509	28	44	6.9	2.9	0.7
2	2	48	0.7	0.2	208	15	30	4.6	2.1	0.8
2	2	65	1.4	0.6	260	28	24	5.2	2.2	0.7
2	2	11	0.7	0.1	592	26	29	7.1	4.2	1.4
2	2	26	1	0.3	163	48	71	7.1	3.7	1
2	2	53	1.6	0.9	178	44	59	6.5	3.9	1.5
2	2	45	1.3	0.6	166	49	42	5.6	2.5	0.8
2	1	52	0.6	0.1	194	10	12	6.9	3.3	0.9
2	1	47	0.8	0.2	236	10	13	6.7	2.9	0.76
2	1	41	0.9	0.2	201	31	24	7.6	3.8	1
2	1	30	0.7	0.2	194	32	36	7.5	3.6	0.92
2	1	17	0.5	0.1	206	28	21	7.1	4.5	1.7
2	2	61	0.8	0.2	163	18	19	6.3	2.8	0.8
2	2	17	0.9	0.2	279	40	46	7.3	4	1.2
2	2	28	0.6	0.2	159	15	16	7	3.5	1
2	2	32	0.7	0.2	189	22	43	7.4	3.1	0.7
2	2	61	0.8	0.2	192	28	35	6.9	3.4	0.9
2	1	45	0.7	0.2	164	21	53	4.5	1.4	0.45
2	1	45	0.6	0.1	270	23	42	5.1	2	0.5
2	1	28	0.6	0.1	137	22	16	4.9	1.9	0.6
2	1	28	1	0.3	90	18	108	6.8	3.1	0.8
2	1	49	0.6	0.1	185	17	26	6.6	2.9	0.7
2	2	42	0.7	0.2	197	64	33	5.8	2.4	0.7
2	2	42	1	0.3	154	38	21	6.8	3.9	1.3
2	2	35	2	1.1	226	33	135	6	2.7	0.8
2	2	38	2.2	1	310	119	42	7.9	4.1	1
2	2	7	0.5	0.1	352	28	51	7.9	4.2	1.1
2	2	30	0.8	0.2	182	46	57	7.8	4.3	1.2
2	2	60	0.7	0.2	171	31	26	7	3.5	1
2	2	65	0.8	0.1	146	17	29	5.9	3.2	1.18
2	2	27	1	0.3	180	56	111	6.8	3.9	1.85
2	2	65	0.7	0.2	265	30	28	5.2	1.8	0.52
2	2	32	0.7	0.2	165	31	29	6.1	3	0.96
2	1	37	0.8	0.2	205	31	36	9.2	4.6	1
2	2	56	1	0.3	195	22	28	5.8	2.6	0.8
2	2	29	0.8	0.2	156	12	15	6.8	3.7	1.1
2	1	53	0.9	0.2	210	35	32	8	3.9	0.9
2	1	22	1.1	0.3	138	14	21	7	3.8	1.1
2	2	62	0.7	0.2	162	12	17	8.2	3.2	0.6

2	2	39	1.6	0.8	230	88	74	8	4	1
2	1	65	0.7	0.2	406	24	45	7.2	3.5	0.9
2	2	42	0.8	0.2	114	21	23	7	3	0.7
2	2	42	0.8	0.2	198	29	19	6.6	3	0.8
2	2	62	0.7	0.2	173	46	47	7.3	4.1	1.2
2	1	45	0.7	0.2	153	41	42	4.5	2.2	0.9
2	2	29	1.2	0.4	160	20	22	6.2	3	0.9
2	1	38	0.6	0.1	165	22	34	5.9	2.9	0.9
2	1	50	1	0.3	191	22	31	7.8	4	1
2	2	60	0.5	0.1	500	20	34	5.9	1.6	0.37
2	2	38	1	0.3	216	21	24	7.3	4.4	1.5
2	2	36	0.9	0.3	199	34	30	6.8	4	1.47
2	2	24	0.9	0.3	201	26	23	7.2	4.1	1.29
2	2	55	1.1	0.4	266	47	47	7.1	3.4	0.91
2	2	52	1	0.4	259	49	50	6.1	3	0.91
2	2	45	0.8	0.2	248	76	56	6.3	2.9	0.8
2	2	49	0.8	0.2	190	66	48	5.8	3.3	1.44
2	2	33	1.6	0.5	165	15	23	7.3	3.5	0.92
2	2	29	1.4	0.5	152	17	25	6.1	3	0.93
2	2	60	0.9	0.2	193	55	46	6	3.8	1.77
2	2	63	0.9	0.2	194	52	45	6	3.9	1.85
2	2	25	1.1	0.5	148	25	33	4.2	2.1	0.94
2	2	22	1.2	0.5	134	19	29	4.5	2.2	0.94
2	1	73	0.7	0.2	187	27	27	5.9	3	1.03
2	1	77	0.8	0.2	229	29	33	6.3	3.3	1.03
2	2	63	0.9	0.3	284	52	51	6.6	3.3	0.96
2	2	31	0.7	0.1	184	77	47	5.4	2.4	0.8
2	1	38	2.6	1.2	400	57	56	5.7	3	0.8
2	1	41	2.4	1.1	383	58	55	5.7	3.1	0.93
2	2	55	1.5	0.6	345	60	58	6.5	3.3	0.86
2	2	27	1.7	0.8	241	36	41	4.6	2.3	0.89
2	2	24	0.6	0.1	180	85	52	5.6	2.4	0.74
2	1	18	0.7	0.2	149	25	38	7	3.7	1.13
2	2	64	0.9	0.3	301	60	61	6.9	3.4	0.91
2	2	49	0.6	0.1	168	59	91	5.1	2.5	0.96
2	2	44	6.1	2.9	594	29	48	6.2	2.4	0.63
2	2	58	2.4	1.1	393	52	55	6.8	3.1	0.82
2	2	71	0.9	0.2	199	40	35	6.3	3.8	1.56
2	1	80	1	0.3	205	24	21	6.8	3.7	1.17
2	2	29	1.1	0.3	224	72	108	5.9	2.7	0.79
2	2	34	1.7	0.6	268	51	169	6.4	3.1	0.89
2	2	54	0.9	0.2	195	45	39	6.6	3.9	1.61
2	2	29	0.7	0.1	190	61	39	6.7	3.1	0.84
2	1	49	0.9	0.2	177	26	53	5.5	2.8	0.98
2	2	35	1.5	0.5	166	16	27	7.1	3.4	0.92
2	2	64	1.1	0.5	174	27	30	5.8	3.2	1.31

2	2	63	0.9	0.2	193	52	45	6	3.9	1.84
2	2	42	0.7	0.1	198	59	36	5.9	2.6	0.8
2	2	62	0.9	0.3	265	43	39	6.7	3.2	0.9
2	1	32	0.7	0.1	176	53	43	5.4	2.6	0.9
2	2	31	1.3	0.3	164	29	30	6.5	3.1	0.91
2	2	68	1.3	0.5	188	24	33	6.4	3.7	1.41
2	2	70	1.4	0.6	149	14	25	6.2	3.8	1.6
2	2	46	0.9	0.2	151	14	14	5.8	2.9	0.98
2	2	46	0.7	0.1	152	24	17	5.4	2.7	0.96
2	2	37	0.7	0.1	260	39	51	6.6	3.7	1.4
2	2	40	0.8	0.2	251	41	50	6.5	3.8	1.46
2	2	42	0.7	0.2	220	41	40	5	2.3	0.82
2	2	23	0.9	0.4	143	23	29	3.8	1.8	0.87
2	2	27	0.8	0.2	238	21	23	5.4	2.4	0.71
2	2	26	0.8	0.2	228	21	24	5.1	2.2	0.73
2	2	24	1.1	0.5	139	19	29	4.1	2.1	0.98
2	2	40	1.4	0.4	183	15	23	6.9	3.4	1
2	2	35	5.1	2.2	144	31	89	5	2.6	1
2	2	26	1	0.3	180	86	56	5.5	2.3	0.72
2	2	41	4.6	2	173	37	86	5.4	2.7	0.98
2	2	38	5	2.2	156	34	90	5.2	2.7	0.99
2	2	48	0.8	0.2	224	43	47	6.4	2.7	0.67
2	2	36	0.9	0.2	158	28	38	6.1	2.3	0.53
2	2	33	1.5	0.5	165	16	24	7.2	3.4	0.89
2	2	28	1	0.4	142	24	34	4.9	2	0.73
2	2	33	0.8	0.1	409	42	39	5.8	2.7	0.85
2	2	44	0.9	0.1	405	33	37	5.9	3.1	1.16
2	1	24	0.7	0.2	186	46	29	5.5	2.3	0.71
2	1	26	0.9	0.3	183	12	13	5.9	2.5	0.75
2	2	59	0.9	0.2	192	53	44	6	3.8	1.77
2	2	25	1.2	0.4	161	48	36	5.3	2.8	1.05
2	2	39	3	1.3	328	70	109	6.8	3.1	0.84
2	2	43	4.3	2.2	525	155	219	5.6	2.3	0.68
2	2	61	1.8	0.7	390	57	48	5.9	3.7	1.67
2	2	45	5.9	2.9	1296	89	62	5.6	2.3	0.62
2	2	45	1.5	0.7	167	46	65	5.2	2.9	1.24
2	2	64	0.9	0.3	309	61	58	7	3.4	0.9
2	2	28	0.7	0.1	176	34	29	6.9	4.1	1.36
2	2	26	0.6	0.1	182	81	49	5.8	2.6	0.83
2	2	60	1.8	0.5	201	45	25	3.9	1.7	0.7
2	2	60	1.8	0.5	200	45	25	3.9	1.7	0.7
2	2	37	0.6	0.1	197	72	47	5.6	2.4	0.71
2	2	65	0.8	0.2	209	40	39	5.9	3.4	1.46
2	2	56	0.9	0.3	291	56	53	6.9	3.4	0.94
2	1	18	0.8	0.2	193	33	31	6.3	3.4	1.14
2	2	61	0.7	0.1	189	33	31	6.1	3.6	1.41

2	2	60	0.6	0.1	187	22	23	6.2	3.3	1.15
2	2	64	0.9	0.3	275	47	47	6.5	3.2	0.96
2	2	64	0.9	0.2	196	43	39	5.8	3.6	1.66
2	2	25	0.6	0.1	183	91	53	5.5	2.3	0.7
2	2	47	1.3	0.5	174	24	34	7.1	3.7	1.09
2	2	58	0.8	0.2	187	41	37	6.2	3.8	1.58
2	2	29	0.6	0.1	182	82	50	5.6	2.5	0.77
2	2	35	1.6	0.5	165	17	25	7	3.3	0.89
2	2	45	1.7	0.6	167	23	30	6	2.7	0.8
2	2	44	1.4	0.4	301	60	93	6.2	3.1	0.98
2	2	57	1.1	0.3	307	61	70	6.7	3.3	0.93
2	1	50	0.8	0.2	215	35	30	5.3	2.7	1.02
2	2	57	0.9	0.3	270	48	45	6.2	3	0.9
2	2	40	1.2	0.4	182	27	27	5.3	2.7	1.03
2	2	59	1	0.3	198	45	39	6	3.6	1.6
2	2	30	1.4	0.5	155	29	31	4.5	2.3	1
2	2	51	1.8	0.6	223	50	36	5.9	3.1	1.07
2	2	33	0.8	0.2	176	55	35	5.8	2.7	0.8
2	2	28	1	0.4	144	21	25	4.7	2.3	0.93
2	2	26	1	0.4	134	20	28	4.6	2.3	0.98
2	1	43	0.8	0.2	177	27	30	6.3	3.2	1
2	2	42	1	0.3	223	43	39	6.7	3.5	1.06
2	2	26	1.2	0.4	151	22	22	6.8	3.5	1.08
2	2	21	0.7	0.2	135	27	26	6.4	3.3	1
2	2	44	0.8	0.3	229	45	43	6.7	3.4	0.95
2	2	23	1.8	0.6	170	86	94	5.5	2.7	0.98
2	2	57	1.2	0.4	285	65	70	6.7	3.3	0.95
2	2	63	0.9	0.2	193	52	46	6	3.9	1.85
2	2	35	0.6	0.1	162	73	77	5.4	2.5	1
2	2	64	0.9	0.3	304	60	57	7	3.4	0.9
2	2	62	0.9	0.2	181	49	40	6	3.8	1.7
2	2	12	0.8	0.3	224	31	30	4.4	2	0.8
2	2	25	0.6	0.1	186	90	52	5.5	2.3	0.7
2	2	53	1.1	0.3	179	27	25	6.9	4.4	1.73
2	2	35	1.5	0.5	166	16	23	7.3	3.6	1.02
2	2	26	1.1	0.5	138	21	28	4.1	2	0.94
2	1	58	0.9	0.3	199	38	20	5.3	2.5	0.86
2	2	63	1.1	0.4	268	61	68	6.9	3.5	1.05
2	2	64	1	0.3	296	61	61	7	3.4	0.95
2	2	25	0.6	0.1	195	88	52	5.7	2.4	0.72
2	2	24	0.7	0.1	237	75	47	6.6	3	0.79
2	2	27	1	0.4	158	30	32	5.5	3.2	1.36
2	2	54	0.9	0.3	270	54	50	7.1	3.8	1.2
2	2	41	1.1	0.3	165	24	27	7.7	3.8	0.96
2	2	52	0.8	0.2	173	37	34	7.5	4	1.22
2	2	46	0.9	0.2	223	68	46	6.3	3.2	0.97

2	2	64	1	0.3	288	55	50	7	3.6	1.03
2	1	32	0.6	0.2	181	32	25	5.3	2.6	
2	2	54	0.8	0.3	267	45	44	6.4	3.2	
2	1	39	0.7	0.1	154	88	22	7	4.2	1.42
2	1	37	0.8	0.2	154	79	21	7.1	4.1	1.33
2	2	31	0.7	0.2	182	61	43	6	3.1	1.05
2	2	34	1.2	0.4	173	23	28	6.9	3.6	1.15
2	2	41	0.8	0.2	241	55	87	7.2	4.3	1.43
2	2	55	0.9	0.2	210	53	59	6.4	4	1.71
2	2	4	0.9	0.2	347	30	34	8	4	1
2	2	11	0.8	0.2	296	49	40	7.2	3.5	0.9
2	2	23	1.3	0.4	266	62	94	7	3.4	0.91
2	2	18	1.1	0.5	164	34	52	4.2	2	0.94
2	2	25	0.7	0.2	183	85	50	5.7	2.5	0.73
2	2	25	1.8	0.7	170	22	21	7.4	3.7	0.99
2	2	50	0.9	0.2	192	40	34	6.2	3.8	1.55
2	2	34	1.4	0.4	173	18	22	7	3.5	1.01
2	2	47	0.7	0.2	191	28	20	7.1	3.9	1.22
2	2	60	0.8	0.3	273	47	44	7.1	3.7	1.03
2	2	30	1.5	0.5	196	14	23	7	3.2	0.83
2	2	20	1.3	0.7	310	12	22	6	2.2	0.57
2	2	30	1.2	0.5	139	18	26	5.6	2.8	0.97
2	2	48	1.2	0.5	164	23	26	7.5	4	1.19
2	2	64	0.9	0.3	302	59	55	7	3.4	0.91
2	2	38	0.9	0.4	149	25	23	5.7	2.9	1.02
2	2	20	1.1	0.5	129	20	30	3.9	1.9	0.95
2	2	54	0.9	0.3	193	48	44	6.4	4	1.7
2	1	28	1.2	0.4	161	20	24	7.1	4	1.38
2	1	23	1	0.4	144	22	27	5.4	3.2	1.33
2	2	52	0.9	0.3	276	55	51	7.1	3.6	0.99
2	1	29	0.9	0.2	209	43	38	7.4	4	1.16
2	2	21	1.1	0.5	130	20	30	4	2	0.95
2	2	25	0.6	0.1	183	91	53	5.5	2.3	0.7
2	2	52	0.7	0.2	223	25	24	8.2	4.1	0.98
2	2	61	0.9	0.2	196	47	41	6.4	3.9	1.72
2	1	48	1.5	0.6	186	30	18	7.7	3.6	0.91
2	2	60	1.1	0.5	260	50	40	7.4	3.5	0.9
2	2	28	1.4	0.6	121	23	46	8.2	4.5	
2	2	26	1	0.4	136	50	54	7.3	3.8	
2	1	31	1	0.3	160	23	22	7.8	4.3	1.21
2	1	26	0.9	0.3	147	23	25	6.4	3.5	1.17
2	2	40	0.9	0.3	235	36	37	6.8	3.6	1.07
2	2	27	0.9	0.3	181	22	24	6.6	3.7	1.28
2	2	61	0.9	0.2	193	50	49	6.1	3.9	1.77
2	2	43	0.9	0.2	181	28	79	7.1	3.7	1
2	2	18	1	0.4	193	20	25	5.4	2.7	0.97

2	2	15	0.9	0.3	293	30	21	7.8	4	0.96
2	2	21	0.8	0.2	303	49	65	6.8	3.5	1.07
2	2	19	1	0.3	260	37	53	6.9	3.5	1.04
2	2	27	0.6	0.1	180	82	49	5.7	2.4	0.71
2	2	26	1	0.4	136	22	28	4.6	2.2	0.91
2	2	28	1.1	0.5	237	86	58	6	3.1	1.02
2	2	28	0.8	0.2	232	110	63	6.3	2.9	0.81
2	2	36	2.6	1.2	214	26	40	7.4	3.8	0.97
2	2	56	1.3	0.5	190	45	43	6.3	3.9	1.7
2	2	45	0.8	0.2	205	32	24	6.1	3	0.9
2	2	39	0.8	0.2	190	44	28	5.9	2.8	0.85
2	2	39	0.8	0.2	253	34	38	7.2	3.8	1.07
2	2	21	0.8	0.3	192	15	25	6.5	3.6	1.14
2	2	59	0.9	0.2	187	47	42	6	3.8	1.73
2	2	58	0.9	0.3	277	53	51	6.8	3.3	0.94
2	2	34	1.3	0.4	175	22	26	7.2	3.7	1.04
2	2	52	0.9	0.3	264	51	47	7	3.6	1.06
2	2	54	0.9	0.2	194	55	43	6.7	4.3	1.74
2	2	36	1.1	0.3	185	45	34	7.9	4.5	1.3
2	1	44	1	0.3	160	18	17	6.8	3.6	1.12
2	1	54	0.8	0.2	209	33	29	6.7	3.5	1.1
2	2	26	1.5	0.7	172	14	25	7.7	4.1	1.12
2	2	19	1.3	0.7	161	15	27	6.6	3.7	1.18
2	1	68	0.8	0.2	157	44	64	7.8	4.7	1.5
2	2	39	0.7	0.1	171	76	58	6.4	3.1	0.92
2	2	26	0.6	0.1	183	90	53	5.5	2.3	0.7
2	1	65	0.8	0.2	214	33	36	6.9	3	0.75
2	2	21	1.1	0.5	131	20	30	4	1.9	0.95
2	2	62	0.9	0.2	197	47	41	6	3.8	1.7
2	1	42	0.9	0.2	164	26	29	8.4	4.4	1
2	2	55	0.9	0.2	183	43	39	6.9	4.1	1.54
2	2	23	0.7	0.2	183	56	55	6.3	3	0.88
2	2	20	1	0.4	141	23	36	4.6	2.3	0.96
2	2	39	0.7	0.2	176	30	43	5.3	2.4	0.8
2	2	37	1	0.3	172	23	36	6	2.8	0.84
2	2	19	1	0.4	219	23	32	4.9	2.6	1.06
2	2	55	0.9	0.2	232	48	43	6.2	3.9	1.76
2	2	61	1.8	0.7	245	47	50	6.1	3.2	1.22
2	2	64	1	0.4	307	60	57	6.9	3.4	0.9
2	2	29	0.7	0.2	158	59	40	6.4	3.4	1.12
2	2	36	0.8	0.2	141	32	31	7.1	4.3	1.54
2	2	22	1	0.4	132	24	28	5.1	2.7	1.06
2	1	25	0.7	0.1	140	32	25	7.5	4.2	1.29
2	1	57	0.7	0.1	165	30	27	7	3.9	1.25
2	1	58	0.7	0.1	156	32	28	6.8	4	1.42
2	1	28	0.7	0.2	201	19	19	6.1	3.2	1.07

2	1	21	0.6	0.2	200	20	17	6	2.9	0.88
2	2	54	1.1	0.3	176	16	22	7.3	3.6	0.96
2	2	71	0.7	0.2	203	22	27	7.3	3.7	0.99
2	1	51	1.2	0.5	165	31	23	5.1	2.4	0.92
2	2	45	1.5	0.6	162	13	18	6.3	3.1	0.95
2	2	33	2.3	1.1	111	24	24	6.5	3.6	1.13
2	2	59	1.2	0.5	271	54	51	7.1	3.6	0.96
2	2	51	0.8	0.2	189	35	31	6.5	3.3	0.97
2	2	29	0.6	0.1	174	77	46	5.7	2.5	0.76
2	2	63	0.9	0.2	190	49	42	6	3.8	1.75
2	2	43	0.6	0.1	174	62	39	5.9	2.7	0.83
2	2	61	0.8	0.2	194	18	22	6.4	3.6	1.25
2	2	65	0.8	0.2	229	30	32	6.5	3.5	1.19
2	2	42	0.9	0.3	176	17	49	6.9	3.6	1.13
2	2	53	0.8	0.2	186	33	52	6.4	3.8	1.48
2	2	32	1.7	0.5	206	17	26	7.3	3.4	0.89
2	2	44	1.6	0.5	408	45	51	7	3.2	0.8
2	2	42	1.1	0.3	191	15	27	5.7	2.7	0.85
2	2	51	0.7	0.2	205	23	33	4.9	2.5	1.02
2	2	42	1.2	0.5	192	24	27	4.5	2	0.83
2	2	63	0.9	0.2	197	51	44	6	3.8	1.79
2	2	19	1.1	0.5	176	21	30	4.2	2.1	1
2	2	30	0.8	0.1	449	35	35	6.7	4.1	1.56
2	2	29	1.3	0.4	164	34	50	7.2	3.6	0.97
2	2	31	1	0.3	167	49	68	7	3.7	1.11
2	2	42	1.6	0.7	171	28	39	6.9	3.7	1.18
2	2	47	1.6	0.8	174	36	49	6.7	3.8	1.34
2	2	43	1.3	0.6	166	43	39	5.9	2.7	0.82
2	2	22	1.1	0.5	131	22	31	4	2	0.94
2	1	47	0.8	0.2	187	11	15	7	3.3	0.9
2	2	62	0.8	0.3	286	50	48	7	3.4	0.9
2	2	37	1.4	0.4	183	14	20	7.1	3.3	0.88
2	2	28	0.6	0.1	190	80	48	5.7	2.4	0.71
2	1	32	1	0.3	170	26	27	6	3	0.98
2	1	34	1	0.3	175	27	26	6.3	3.1	0.98
2	1	26	0.9	0.3	167	27	34	6	2.9	0.93
2	1	34	0.7	0.2	194	34	37	7.3	3.6	1.02
2	1	28	0.6	0.1	231	36	30	7.1	4.2	1.51
2	1	17	0.6	0.2	193	27	22	6.6	4.1	1.58
2	2	61	0.8	0.2	166	22	22	6.3	2.9	0.91
2	2	57	0.8	0.2	160	18	20	6.1	2.7	0.81
2	2	18	0.9	0.2	269	45	47	7.1	3.8	1.15
2	2	17	0.9	0.2	273	39	45	7.2	3.9	1.19
2	2	32	0.6	0.2	163	19	19	6.9	3.5	1.09
2	2	25	0.6	0.1	181	86	51	5.6	2.4	0.72
2	2	42	0.8	0.2	191	31	44	7	3.4	1.06

2	2	60	0.9	0.3	296	56	56	7	3.4	0.88
2	2	63	0.9	0.2	194	46	43	6.2	3.8	1.62
2	2	61	0.8	0.2	202	31	37	6.9	3.4	0.9
2	1	41	1	0.3	164	19	43	5.5	2.1	0.61
2	2	34	1.5	0.5	165	15	25	7.1	3.4	0.89
2	2	60	0.8	0.2	207	47	44	5.8	3.6	1.61
2	1	37	0.6	0.1	235	50	46	5.3	2.1	0.58
2	2	26	0.6	0.1	163	61	37	5.2	2.1	0.66
2	2	57	0.8	0.3	276	53	50	6.6	3.1	0.84
2	1	44	1	0.3	136	33	80	6.4	3.5	1.27
2	1	25	1	0.4	103	19	81	5.8	2.7	0.85
2	1	44	0.9	0.2	179	16	25	6.8	3.1	0.77
2	2	56	0.8	0.2	190	35	36	6.3	3.4	1.28
2	2	34	0.8	0.3	172	48	32	5.1	2.2	0.79
2	2	32	0.6	0.1	189	80	45	5.6	2.3	0.7
2	2	26	1.1	0.4	135	25	28	4.7	2.4	1.04
2	2	38	1	0.3	149	35	23	6.3	3.5	1.24
2	2	29	1.2	0.5	201	67	87	5.7	2.5	0.74
2	2	29	1.1	0.5	199	70	83	5.7	2.4	0.74
2	2	23	1.3	0.6	162	39	32	4.7	2.3	0.96
2	2	45	1.8	0.8	276	100	43	7.3	4	1.25
2	2	12	0.7	0.3	264	25	43	6.3	3.3	1.04
2	2	20	1.1	0.3	258	21	37	7.6	3.8	1.01
2	2	31	1.1	0.3	177	36	46	7.6	4	1.11
2	2	21	1.1	0.5	133	22	33	4.3	2.1	0.97
2	2	40	0.6	0.1	178	66	42	6.1	2.8	0.82
2	2	40	1.4	0.4	166	19	24	7.2	3.5	0.94
2	2	63	0.9	0.2	185	45	42	6	3.8	1.72
2	2	37	1	0.3	135	19	30	4.7	2.4	1.04
2	2	26	0.7	0.2	182	78	74	6	2.9	1.11
2	2	29	1.2	0.4	174	39	75	7	3.7	1.47
2	2	64	0.8	0.3	298	53	50	6.5	3	0.8
2	2	65	0.7	0.2	261	31	29	5.2	1.9	0.59
2	2	33	1.6	0.5	165	15	23	7.3	3.5	0.92
2	2	27	0.9	0.3	150	26	29	5.2	2.5	0.96
2	2	34	1.4	0.4	174	19	26	7.7	3.8	0.94
2	2	24	1	0.4	146	23	31	5.1	2.5	0.96
2	2	40	0.8	0.2	189	58	41	5.6	2.4	0.75
2	2	24	1.1	0.5	135	20	30	4.1	2	0.94
2	2	56	0.9	0.3	273	49	48	7	3.5	0.95
2	2	29	0.9	0.2	157	12	16	6.8	3.7	1.09
2	2	35	1.5	0.5	170	17	24	7.4	3.5	0.92
2	1	58	0.9	0.2	202	43	38	7	3.9	1.36
2	1	25	1.2	0.4	146	14	22	7.1	3.7	1.05
2	1	23	1	0.3	149	32	29	6.6	3.4	1.01
2	2	50	1.1	0.3	163	13	19	7.8	3.3	0.73

2	2	57	0.9	0.3	163	13	18	8	3.3	0.65
2	2	34	1.5	0.7	201	69	61	6.8	3.4	0.99
2	2	38	1.6	0.7	215	71	62	7.8	3.9	0.98
2	1	61	0.7	0.2	379	24	44	6.9	3.3	0.9
2	2	63	0.9	0.2	209	50	45	6.1	3.9	1.78
2	2	25	1	0.4	125	20	28	4.6	2.2	0.89
2	2	28	0.6	0.1	171	79	48	5.8	2.4	0.7
2	2	61	0.9	0.3	296	57	53	7	3.4	0.89
2	2	41	0.8	0.2	196	29	19	6.5	3	0.8
2	2	60	0.7	0.2	171	45	46	7.1	4	1.19
2	2	63	0.9	0.2	189	51	45	6.3	3.9	1.71
2	1	41	0.7	0.2	158	50	44	4.7	2.2	0.86
2	2	34	1.5	0.5	164	17	25	7.1	3.4	0.92
2	2	34	1.2	0.4	165	25	25	6.2	3.1	1.03
2	2	62	0.9	0.3	299	58	55	6.9	3.4	0.9
2	2	24	1	0.4	137	20	31	4.4	2.1	0.94
2	1	42	0.6	0.1	188	28	38	6.1	3	0.9
2	2	24	1.1	0.5	136	20	30	4.4	2.2	0.96

BIBLIOGRAPHY

1. Multivariate data analysis (Fifth Edition) --- Joseph F.Hair, RolphE.Anderson, Ronald l Tatham and William C.Black

2. Data Mining- Theories, Algorithms, and Examples – NoNG YE

3. A Practical Guide to Data Mining for Business and Industry -- AnProfea Ahlemeyer-Stubbe, Shirley Coleman

4. Data Mining and Predictive Analytics – Daniel T. Larose, Chantal D.Lorse

5. machine_learning_mastery_with_r. – Jason Brownlee

6. master_machine_learning_algorithms -- Jason Brownlee

7. statistical_methods_for_machine_learning - Jason Brownlee

8. Machine Learning Using R -- KarthikRamasubramanian ,Abhishek Singh

9. Data Science for Business - Forster Provost & Tom Fawcett

10. Deep learning with Deep learning R by François Chollet