A PROJECT REPORT

ON

STATISTICAL MODELLING FOR CLASSIFICATION OF MUSHROOM

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

Roll No.: 100717507001	**Name: Param Meghana**
Roll No.: 100717507002	**Name: P. Kavitha**
Roll No.: 100717507006	**Name: J. Rajeshwari**
Roll No.: 100717507007	**Name: K M Krishna**
Roll No.: 100717507044	**Name: Mohammed Kadhim Shanshool**
Roll No.: 100717507003	**Name: A. Akshitha**
Roll No.: 100717507022	**Name: Mavilla Ramesh**
Roll No.: 100717507021	**Name: G. Sowjanya**

Under the Supervision of

Dr. M. VENUGOPALA RAO

2018

A PROJECT REPORT

ON

STATISTICAL MODELLING FOR CLASSIFICATION OF MUSHROOM

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

Roll No.: 100717507001	Name: Param Meghana
Roll No.: 100717507002	Name: P. Kavitha
Roll No.: 100717507006	Name: J. Rajeshwari
Roll No.: 100717507007	Name: K M Krishna
Roll No.: 100717507044	Name: Mohammed Kadhim Shanshool
Roll No.: 100717507003	Name: A. Akshitha
Roll No.: 100717507022	Name: Mavilla Ramesh
Roll No.: 100717507021	Name: G. Sowjanya

Under the Supervision of

Dr. M. VENUGOPALA RAO

2018

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.

Name: Param Meghana Sign:

Name: P. Kavitha Sign:

Name: J. Rajeshwari Sign:

Name: K M Krishna Sign:

Name: Mohammed Kadhim Shanshool Sign:

Name: A. Akshitha Sign:

Name: Mavilla Ramesh Sign:

Name: G. Sowjanya Sign:

Department of Statistics
Osmania University
Hyderabad – 500 007, T.S.
INDIA

CERTIFICATE

This is to certify that

Ms. Param Meghana (100717507001)

Ms. P. Kavitha (100717507002)

Ms. J. Rajeshwari (100717507006)

Mr. K M Krishna (100717507007)

Mr. Mohammed Kadhim Shanshool (100717507044)

Ms. A. Akshitha (100717507003)

Mr. Mavilla Ramesh (100717507022)

Ms. G. Sowjanya (100717507021)

have submitted the project titled **"Statistical Modelling for Classification of Mushroom"** in partial fulfilment for the degree of Master of Science in Statistics.

Head

Department of Statistics

Internal Examiner

External Examiner

ACKNOWLEDGEMENTS

I deem it a great pleasure to express my deep sense of gratitude and indebtedness to my research supervisor **Dr. M. VENUGOPALA RAO,** Statistics department, University College of Science, Osmania University for his 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 **Dr. 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 **Dr. N. Ch. BHATRACHARYULU**, **Dr. K. VANI, Dr. S. A. JYOTHI RANI, Dr. G. SIRISHA, Mrs. J. L. PADMA SHREE,** for their encouragement and constant help during the research.

I would like to express my deepest gratitude to **T. SANDHYA, 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

CHAPTER 1
INTRODUCTION & SCOPE OF THE PROBLEM

INTRODUCTION & SCOPE OF THE PROBLEM

1.1. Scope of the Problem

1. Problem: The problem concerns whether the mushroom's nature is fit for human consumption. There are 22 diverse characteristics defined for each mushroom in the dataset. The interest here is to find different combinations of aspects of their features and ecospheres that mostly influence their kind.
2. Objective: To assess the class/type of a mushroom entrenched on their varied environmental surroundings and physical appearance.
3. Goals:

- ✓ Identifying attributes that strongly affect a mushroom's type.
- ✓ Developing an algorithm, by which we can classify the mushrooms based on their physiognomies.
- ✓ Given a set of characteristics, predicting in which group of class a new mushroom falls.

4. Data Source: The dataset has been acquired from a secondary source. It was downloaded from UCI repository. (Source: https://archive.ics.uci.edu/ml/datasets/mushroom).

1.2. Data Description

The original dataset consists of 8124 observations taken over 22 features and the target variable being the type of mushroom. All the variables considered are categorical type. The variables along with their names and levels are concisely presented in the table below.

Table 1.2.: Mushroom – Attributes & Levels

Attribute	Levels	
Class	0	Edible
	1	Poisonous
Cap-Shape	1	Bell
	2	Conical
	3	Flat
	4	Knobbed
	5	Sunken
	6	Convex
Cap-Surface	1	Fibrous
	2	Grooves
	3	Smooth
	4	Scaly

Cap-Colour	1	Buff
	2	Cinnamon
	3	Red
	4	Grey
	5	Brown
	6	Pink
	7	Green
	8	Purple
	9	White
	10	Yellow
Bruises	1	False
	2	True
Odour	1	Almond
	2	Creosote
	3	Foul
	4	Anise
	5	Musty
	6	None
	7	Pungent
	8	Spicy
	9	Fishy
Gill-Attachment	1	Attached
	2	Free
	3	Descending
	4	Notched
Gill-Spacing	1	Close
	2	Crowded
	3	Distant

Gill-Size	1	Broad
	2	Narrow
Gill-Colour	1	Buff
	2	Red
	3	Grey
	4	Chocolate
	5	Black
	6	Brown
	7	Orange
	8	Pink
	9	Green
	10	Purple
	11	White
	12	Yellow
Stalk-Shape	1	Enlarging
	2	Tapering
Stalk-Root	1	Bulbous
	2	Club
	3	Equal
	4	Rooted
	5	Cup
	6	Rhizomorphs
Stalk-Surface-Above-Ring	1	Fibrous
	2	Silky
	3	Smooth
	4	Scaly
Stalk-Surface-Below-Ring	1	Fibrous
	2	Silky
	3	Smooth
	4	Scaly

Stalk-Colour-Above-Ring	1	Buff
	2	Cinnamon
	3	Red
	4	Grey
	5	Brown
	6	Orange
	7	Pink
	8	White
	9	Yellow
Stalk-Colour-Below-Ring	1	Buff
	2	Cinnamon
	3	Red
	4	Grey
	5	Brown
	6	Orange
	7	Pink
	8	White
	9	Yellow
Veil-Type	1	Partial
	2	Universal
Veil-Colour	1	Brown
	2	Orange
	3	White
	4	Yellow
Ring-Number	1	None
	2	One
	3	Two

Ring-Type	1	Evanescent
	2	Flaring
	3	Large
	4	None
	5	Pendant
	6	Sheathing
	7	Zone
	8	Cobwebby
Spore-Print-Colour	1	Buff
	2	Chocolate
	3	Brown
	4	Black
	5	Orange
	6	Green
	7	Purple
	8	White
	9	Yellow
Population	1	Abundant
	2	Clustered
	3	Numerous
	4	Scattered
	5	Several
	6	Solitary
Habitat	1	Woods
	2	Grasses
	3	Meadows
	4	Leaves
	5	Paths
	6	Urban
	7	Waste

1.3. Review of the Following Chapters

- Chapter 2 gives a brief outline of Machine Learning. It chiefly discusses/answers the following areas
  1. Why did ML come into existence?
  2. ML Process
  3. ML Techniques/Algorithms
  4. Which techniques to use where?
  5. Improving ML Algorithms for enhanced precision
- Chapter 3 includes a detailed procedure for solving a classification problem. This chapter starts with refining the data, followed by fitting various classification algorithms: k-NN, SVM, DT, RF, GLM. Thereafter, results obtained for the various algorithms are compared to see which best describes the data. Sequentially, you will realize that RF has given a better accuracy and as such in the next stage, hyper parameter tuning and variable importance have been carried out. Finally, a Logistic Regression model is fitted using significant variables that explain the data well. To conclude, Logistic Regression model was run on train and test data to observe if it performs satisfactorily. Outputs/results of all the algorithms, validation tables and performance matrices are displayed as when required.
- Chapter 4 comprises of quick overall summary and findings made in the previous chapter which ends in a conclusion.
- Appendix is attached after chapter 4. Appendix consists of R Code and Sampled Mushroom Data that were used in the project. Last part of Appendix is the Bibliography.

CHAPTER 2
OVERVIEW OF MACHINE LEARNING

OVERVIEW OF MACHINE LEARNING

2.1. 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 analysing 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-driven 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.

2.2. 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
- Modelling
- Evaluation
- Deployment

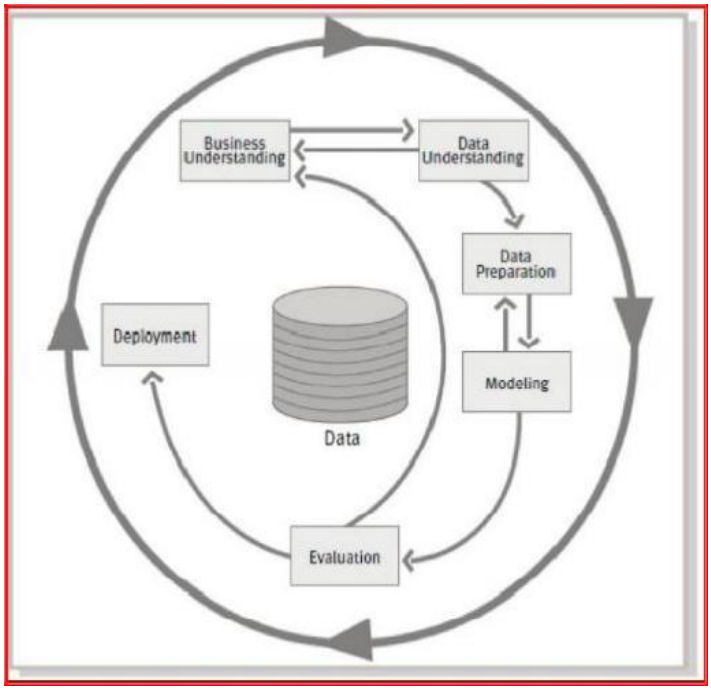


Figure 2.2.: Phases of CRISP-DM Process

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

2.2.1. 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

2.2.2. 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

2.2.3. 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

2.2.4. Modelling

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 modelling. 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 modelling 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.

2.2.5. 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

2.2.6. 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

2.3. Types of Machine Learning

Broadly, the Machine Learning Algorithms are classified into 3 types.

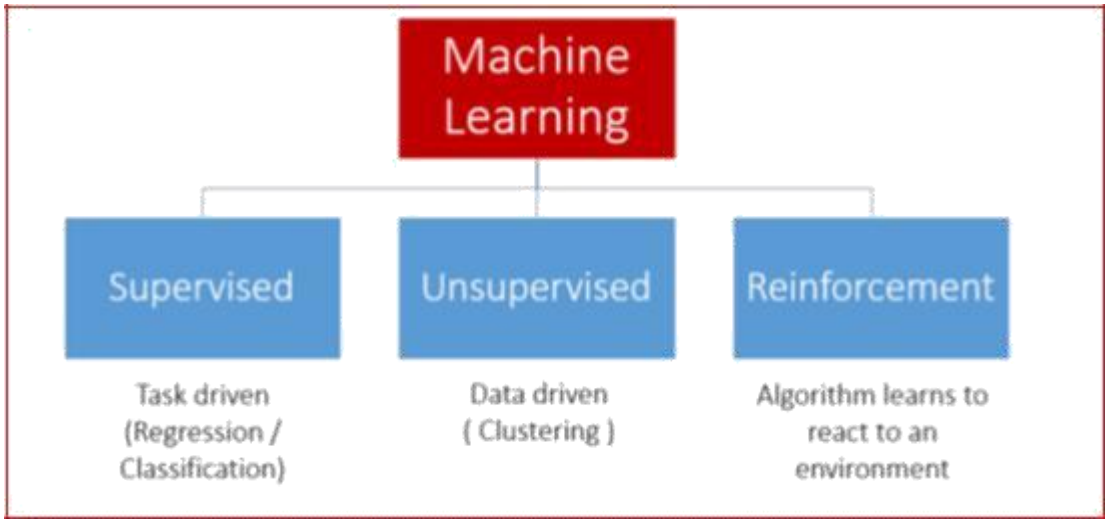


Figure 2.3.: Types of Machine Learning

2.3.1. 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:

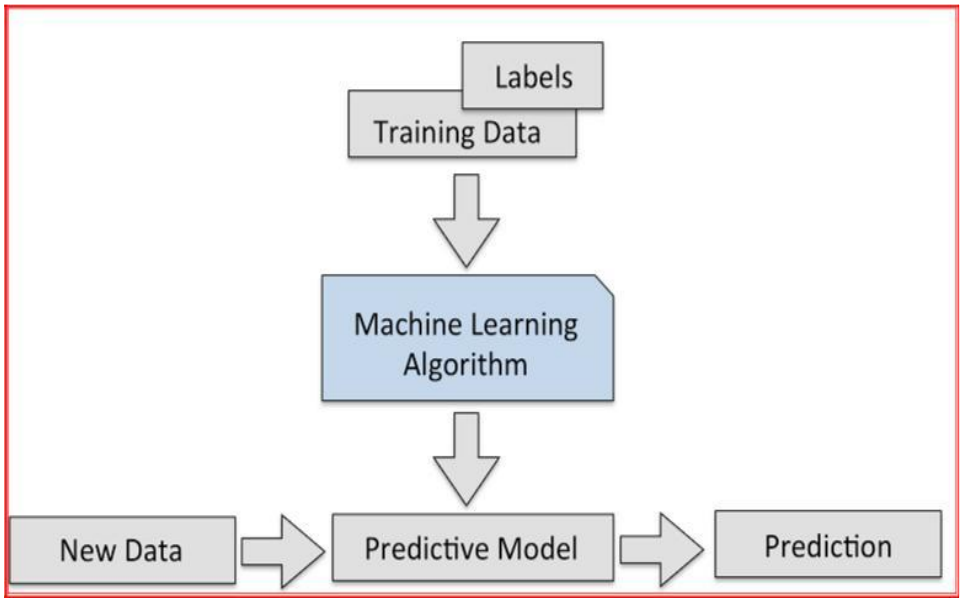


Figure 2.3.1. (a): Supervised Learning Process

Examples of Supervised Learning: Regression, Decision Tree, Random Forest, k–NN, Logistic Regression, etc.

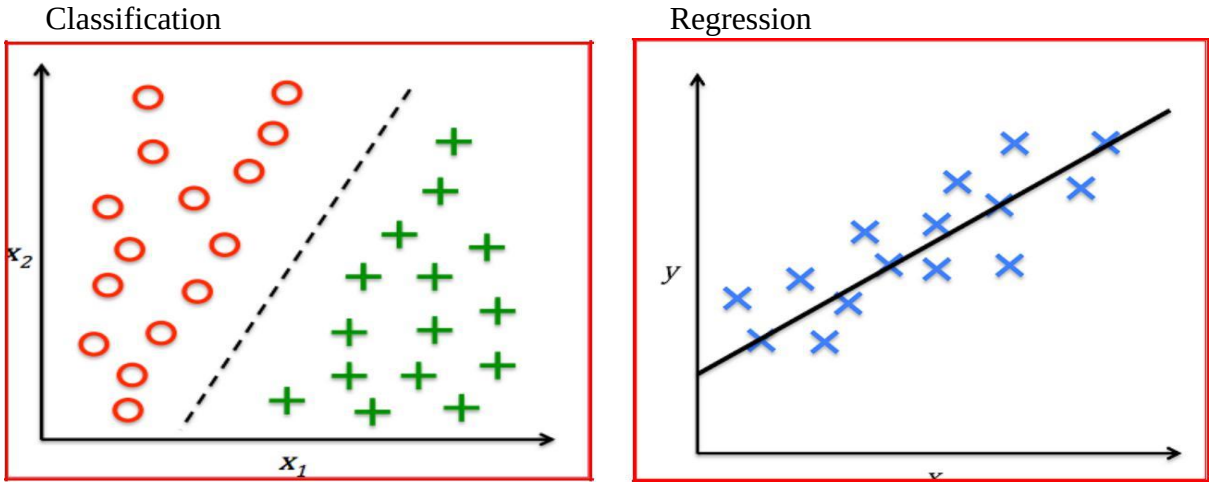


Figure 2.3.1. (b).: Examples of Supervised Learning

2.3.2. 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)

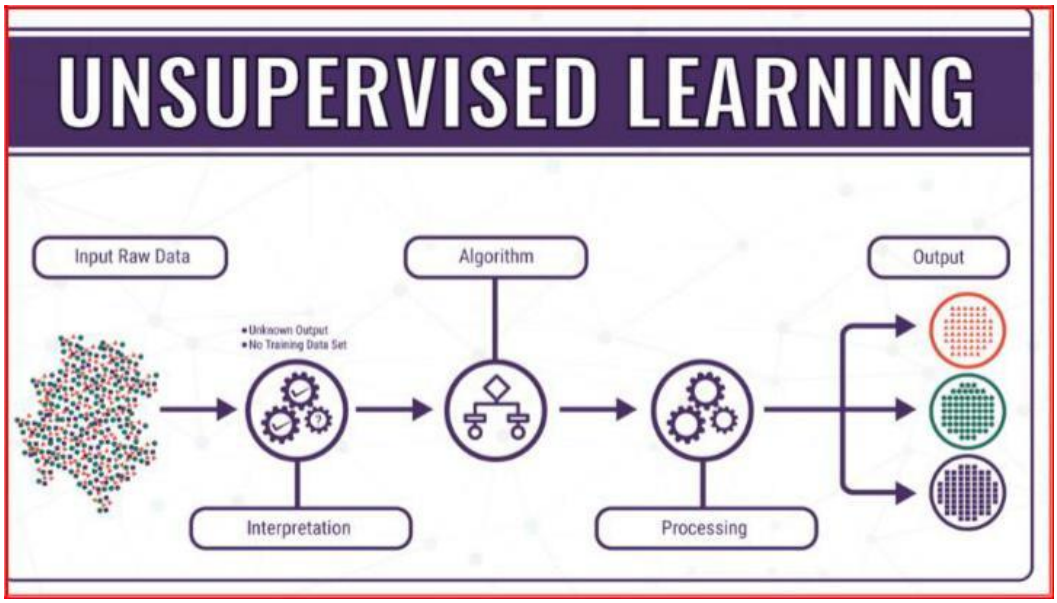


Figure 2.3.2. (a): Unsupervised Learning Process

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

Cluster Analysis Data Reduction Techniques

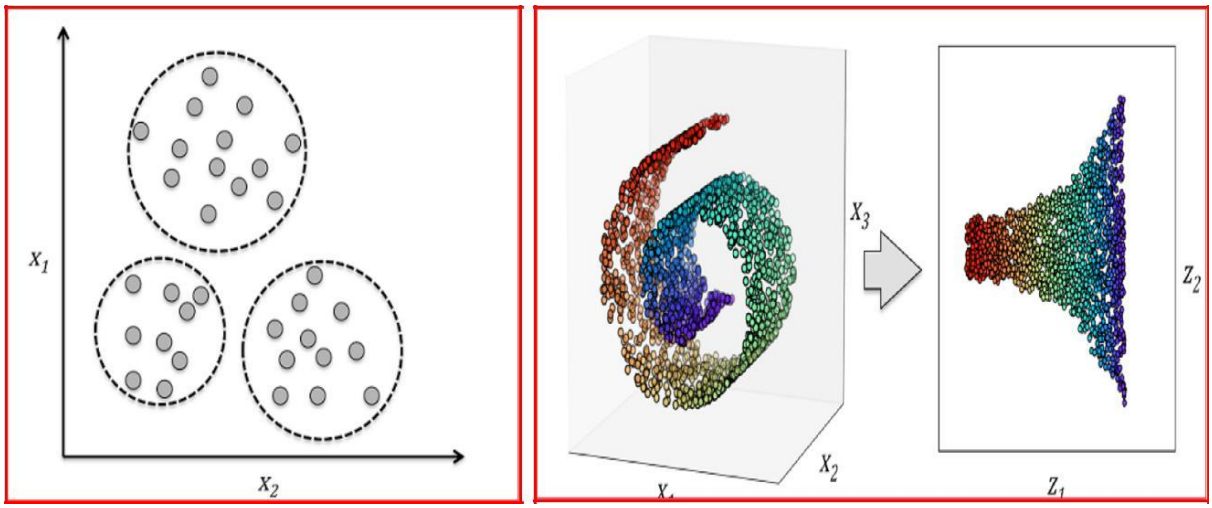


Figure 2.3.2. (b): Examples of Unsupervised Learning

2.3.3. 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:

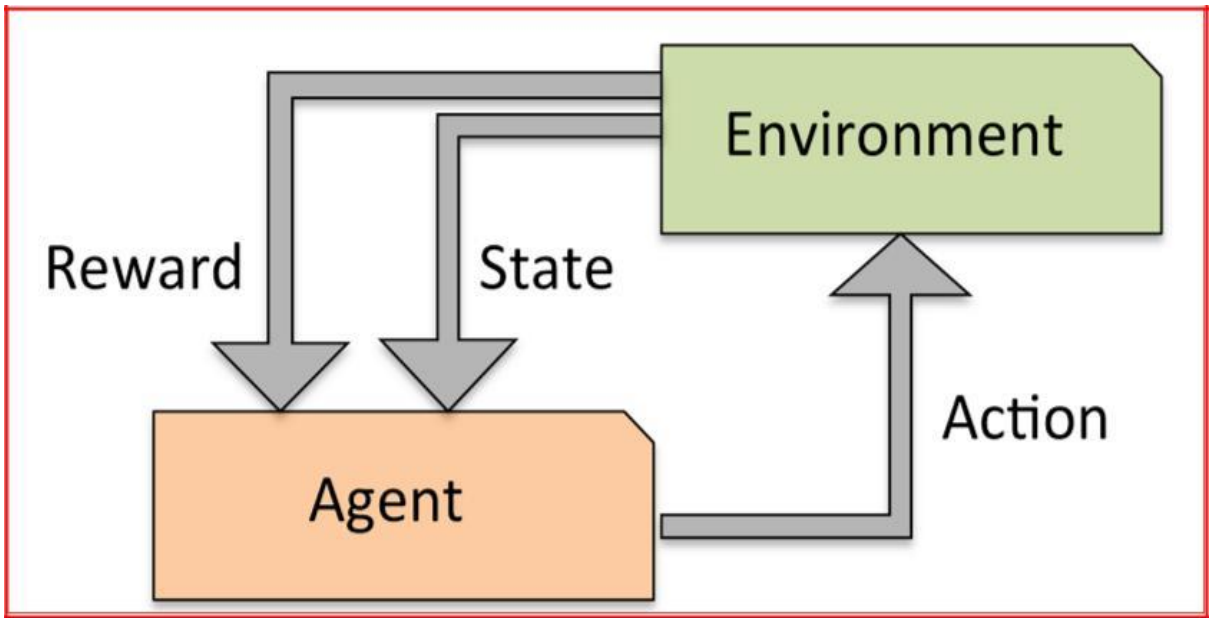


Figure 2.3.3.: Reinforcement Learning Process

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

2.4. Choosing an 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:

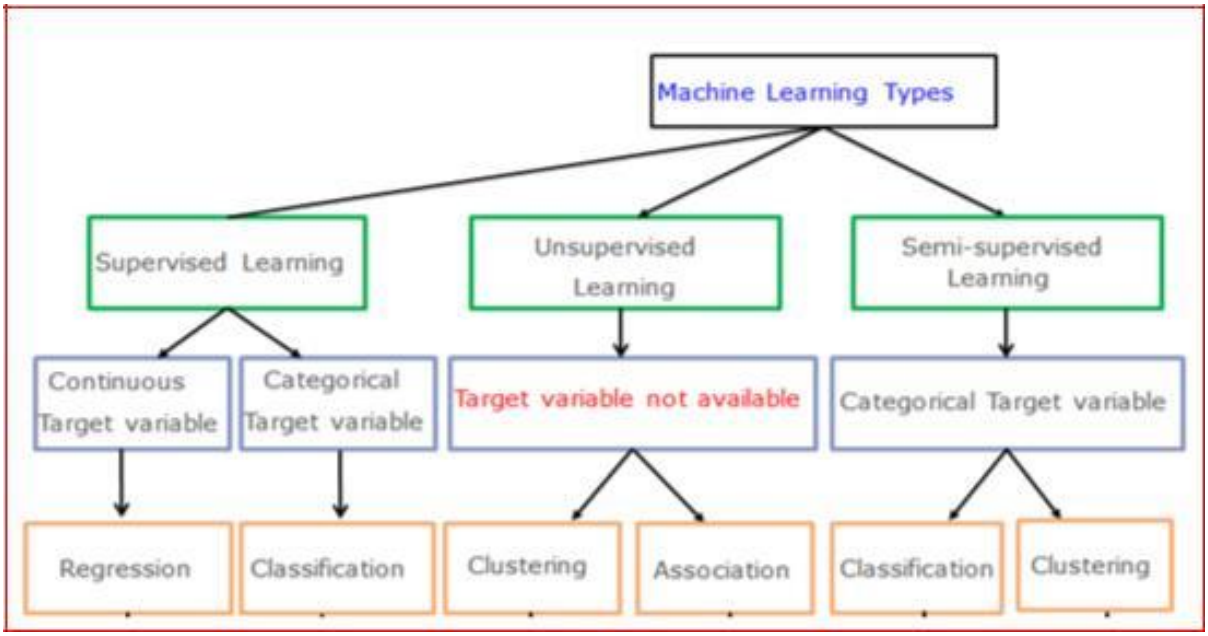


Figure:2.4.: Selecting an Algorithm

2.4.1. 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:

1. **Simple Linear Regression:** In simple linear regression, we predict scores on one 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.
2. **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.
3. **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.
4. **Support Vector Machines:** Support Vector Machines can be applied to regression problems as well as Classification. It contains all the features that characterises 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.
5. **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.
6. **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 Net regression which are used to select the key parameters and these is also Bayesian regression which works with the Bayes theorem.

2.4.2. 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:

1. **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”.
2. **K -Nearest Neighbours:** k-NN algorithm is one of the most straightforward algorithms in classification, and it is one of the most used ML algorithms. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours. 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 neighbours.

3. **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.
4. **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.
5. **Support Vector Machines:** A Support Vector Machine is a type of Classifier, in which a discriminative classifier is formally defined by a separating hyper plane. The algorithm outputs an optimal hyper plane which categorises new examples. In two dimensional space, this hyper plane is a line dividing a plane in two parts where in each class lay in either side.
6. **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.

2.4.3. Types of Unsupervised Learning

Clustering is the type of unsupervised learning in which an unlabelled data is used to draw 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.

2.4.3.1. 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:

1. **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.

2. **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.

2.4.3.2. 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.

1. **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.

2.5. 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.

2.5.1. 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.

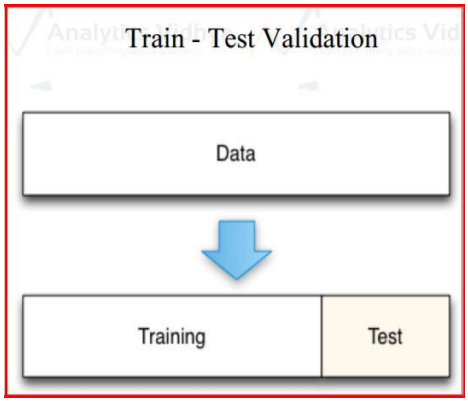


Figure 2.5.1. (a): Dividing data as Train & Test

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:

1. Randomly split your entire dataset into k "folds"
2. 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 kth fold.
3. Record the error you see on each of the predictions.
4. Repeat this until each of the k-folds has served as the test set.
5. 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.

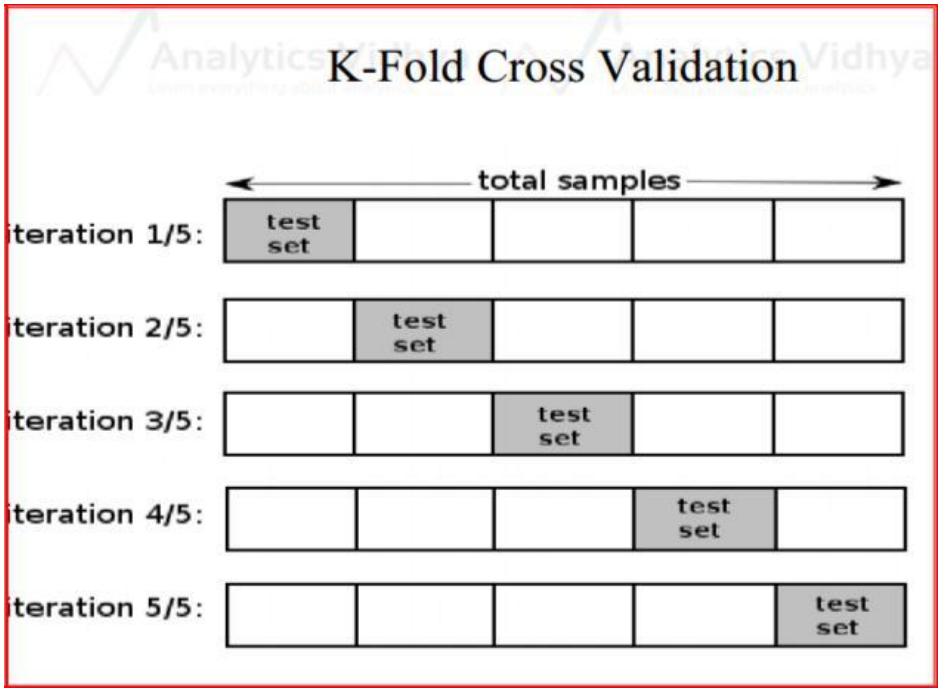


Figure 2.5.1. (b): Performing a 5-Fold CV

How to choose K:

1. Smaller dataset: 10-fold cross validation is better
2. Moderate dataset: 5 or 6-fold cross validation works mostly
3. 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.

2.6. Model Diagnosis with Over-fitting and Under-fitting

2.6.1. 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 drops drastically 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 Under fitting and over fitting like mentioned in the below picture.

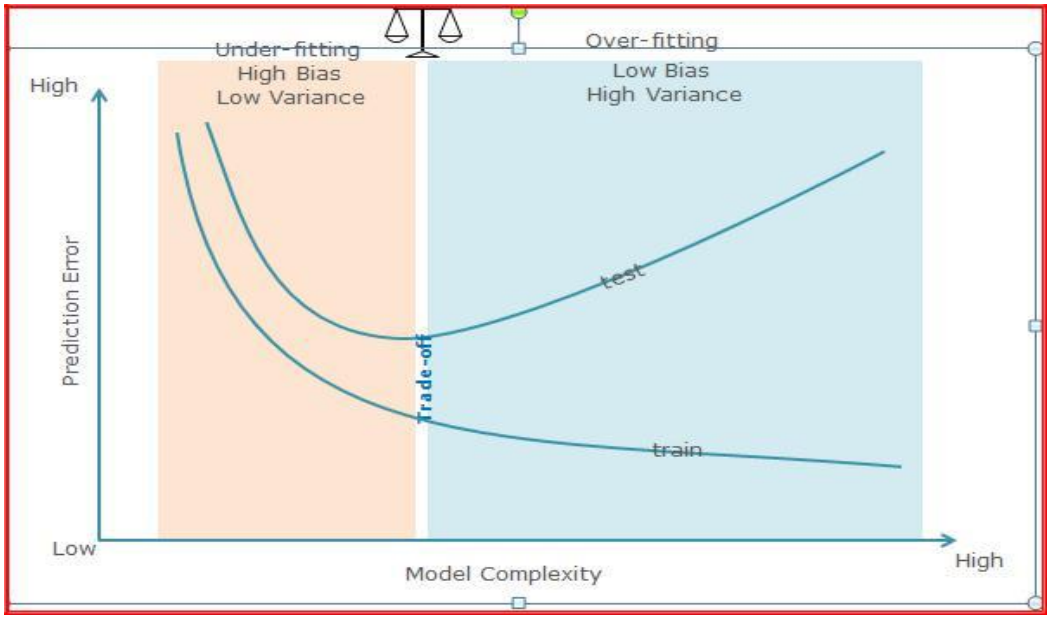


Figure 2.6.1.: Trade-off between Over-fitting & Under-fitting

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

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

1. **Grid search:** For a given model, you define a set of parameter values that you would like to try. Then using GridSearchCV function, models are built for all possible combinations of a present list of values of Hyper parameter provided by you, and the best combination is chosen based on the cross-validation score.
   Disadvantages: Computationally expensive.

2. **Random search:** Random search algorithm tries random combinations of range of values of given parameters. The numerical parameters can be specified as a range you can control the no of iterations of random searches that you would like to perform. It is known to find a very good combination in a lot less time compared to GridSearch. However, you have to carefully choose the range of parameters and the number of random search iteration as it can miss the best parameter combination with lesser iterations or smaller ranges.

3. **Manual Tuning:** In this approach we need to mention the parameters which we want to tune manually.

2.6.2. 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.

2.6.2.1. 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 2.6.2.1. (a): Structure of a Confusion Matrix

		Predicted classed		
		Positive (C_0)	Negative (C_1)	
Actual class	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		
		Positive (C_0)	Negative (C_1)	
Actual class	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.

Table 2.6.2.1. (b): Measures in Confusion Matrix with their Description & Formula

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))

2.6.2.2. 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:

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

Formula 2.6.2.2.: Root Mean Square Error

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

2.6.2.3. Evaluating Unsupervised Models

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

2.7. Overall Process of Machine Learning

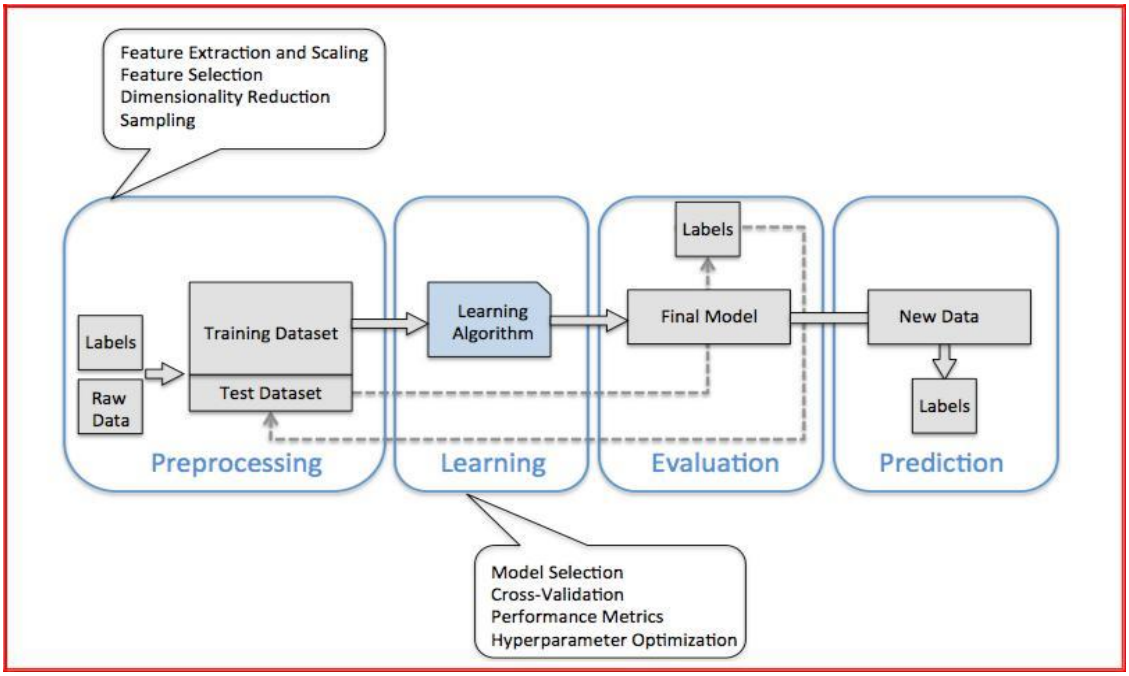


Figure 2.7.: Overall Machine Learning Process

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

CHAPTER 3
MACHINE LEARNING AT WORK

MACHINE LEARNING AT WORK

3.1. Approach to the Problem

The dataset consists of 8124 observations, each representing a single mushroom. The first column is the target variable containing the class labels, identifying whether the mushroom is poisonous or edible. The remaining columns are 22 discrete features that describe the mushroom in some observable way.

In order to carry out the analysis, we have extracted a random sample of 500 records from the 8124 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 algorithms in order to obtain the solution for the problem mentioned in chapter 1.

3.2. ML Process

3.2.1 Data Preparation

The foremost and an essential phase of any ML Algorithm is cleaning and preparing data for modelling. The process of data cleaning starts with reading the dataset into R. Next, we look at basic description of the data: datatypes, observation, attributes, missing values, and summary. That is, at this point we seek for any inconsistencies in the data and attempt to get a grip on what the data actually is. Simply, we try to understand the data.

```
> # Reading the data file into R
> # File name: Mushroom
> M1=read.csv("Mushroom.csv")
> # Looking at basic framework of the dataset - observations, attributes, datatypes
> dim(M1)
[1] 500  23
> class(M1)
[1] "data.frame"
> names(M1)
 [1] "y"   "x1"  "x2"  "x3"  "x4"  "x5"  "x6"  "x7"  "x8"  "x9"  "x10" "x11" "x12" "x13"
 "x14"
[16] "x15" "x16" "x17" "x18" "x19" "x20" "x21" "x22"
```

Output 3.2.1. (a): Reading Mushroom data file into R and looking at dimensions, class, columns

.

```
> summary(M1)
 y       x1      x2       x3      x4      x5      x6      x7      x8       x9      x10     x11     x12     x13     x14     x15
 0:360   1:116   1:106   3 :  1   1:104   1:173   1:  7   1:404   1:365   3 : 64   1:392   1: 50   1: 27   1: 30   4:  7   4:  2
 1:140   3:108   3:188   4 : 49   2:396   4:160   2:493   2: 96   2:135   4 : 11   2:108   2:220   3:473   3:395   7:  3   7:  5
         5: 14   4:206   5 :116           6:116                           5 :101           3:155           4: 75   8:490   8:493
         6:262           9 :168           7: 51                           6 :140           4: 75
                        10:166                                           8 : 78
                                                                        10:  2
                                                                        11:104

 x16     x17     x18     x19     x20     x21     x22
 1:500   2:  6   2:495   1: 58   3:223   1: 31   1: 50
         3:494   3:  5   5:442   4:256   3:100   2:238
                         7: 21   4:209   4:103
                                 5: 87   5: 39
                                 6: 73   6: 70
```

Output 3.2.1. (b): Counts of each level on every variable in the data set. (Categorical variables)

```
> str(M1)
'data.frame':   500 obs. of  23 variables:
 $ y  : int  1 0 0 0 1 0 0 0 0 1 ...
 $ x1 : int  6 1 6 1 6 1 6 6 1 6 ...
 $ x2 : int  3 3 3 3 4 3 4 4 3 4 ...
 $ x3 : int  5 9 4 9 9 10 10 10 10 9 ...
 $ x4 : int  2 2 1 2 2 2 2 2 2 2 ...
 $ x5 : int  7 4 6 1 7 1 4 1 1 7 ...
 $ x6 : int  2 2 1 1 1 1 1 1 1 2 ...
 $ x7 : int  1 1 2 1 1 1 1 1 1 1 ...
 $ x8 : int  2 1 1 1 2 1 1 1 1 2 ...
 $ x9 : int  5 6 5 3 8 3 3 6 11 5 ...
 $ x10: int  1 1 2 1 1 1 1 1 1 1 ...
 $ x11: int  3 2 3 2 3 2 2 2 2 3 ...
 $ x12: int  3 3 3 3 3 3 3 3 3 3 ...
 $ x13: int  3 3 3 3 3 3 3 3 3 3 ...
 $ x14: int  8 8 8 8 8 8 8 8 8 8 ...
 $ x15: int  8 8 8 8 8 8 8 8 8 8 ...
 $ x16: int  1 1 1 1 1 1 1 1 1 1 ...
 $ x17: int  3 3 3 3 3 3 3 3 2 2 ...
 $ x18: int  2 2 2 2 2 2 2 2 2 2 ...
 $ x19: int  5 5 1 5 5 5 5 5 5 5 ...
 $ x20: int  3 4 4 3 3 3 4 3 4 4 ...
 $ x21: int  4 3 1 3 5 4 3 4 4 5 ...
 $ x22: int  6 4 2 4 2 4 2 4 2 6 ...
```

Output 3.2.1. (c): Structure of Mushroom data – all categorical variables displayed as integers

Here, we observe that all the variables are in integer type, and they do not agree with the problem. R has, by default, understood the different levels of variables as numbers. As such, before we can carry out any analyses, these variables must be converted to factors.

```
> str(M1)
'data.frame':   500 obs. of  23 variables:
 $ y  : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 1 2 ...
 $ x1 : Factor w/ 4 levels "1","3","5","6": 4 1 4 1 4 1 4 4 1 4 ...
 $ x2 : Factor w/ 3 levels "1","3","4": 2 2 2 2 3 2 3 3 2 3 ...
 $ x3 : Factor w/ 5 levels "3","4","5","9",..: 3 4 2 4 4 5 5 5 5 4 ...
 $ x4 : Factor w/ 2 levels "1","2": 2 2 1 2 2 2 2 2 2 2 ...
 $ x5 : Factor w/ 4 levels "1","4","6","7": 4 2 3 1 4 1 2 1 1 4 ...
 $ x6 : Factor w/ 2 levels "1","2": 2 2 1 1 1 1 1 1 1 2 ...
 $ x7 : Factor w/ 2 levels "1","2": 1 1 2 1 1 1 1 1 1 1 ...
 $ x8 : Factor w/ 2 levels "1","2": 2 1 1 1 2 1 1 1 1 2 ...
 $ x9 : Factor w/ 7 levels "3","4","5","6",..: 3 4 3 1 5 1 1 4 7 3 ...
 $ x10: Factor w/ 2 levels "1","2": 1 1 2 1 1 1 1 1 1 1 ...
 $ x11: Factor w/ 4 levels "1","2","3","4": 3 2 3 2 3 2 2 2 2 3 ...
 $ x12: Factor w/ 2 levels "1","3": 2 2 2 2 2 2 2 2 2 2 ...
 $ x13: Factor w/ 3 levels "1","3","4": 2 2 2 2 2 2 2 2 2 2 ...
 $ x14: Factor w/ 3 levels "4","7","8": 3 3 3 3 3 3 3 3 3 3 ...
 $ x15: Factor w/ 3 levels "4","7","8": 3 3 3 3 3 3 3 3 3 3 ...
 $ x16: Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
 $ x17: Factor w/ 2 levels "2","3": 2 2 2 2 2 2 2 2 1 1 ...
 $ x18: Factor w/ 2 levels "2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ x19: Factor w/ 2 levels "1","5": 2 2 1 2 2 2 2 2 2 2 ...
 $ x20: Factor w/ 3 levels "3","4","7": 1 2 2 1 1 1 2 1 2 2 ...
 $ x21: Factor w/ 5 levels "1","3","4","5",..: 3 2 1 2 4 3 2 3 3 4 ...
 $ x22: Factor w/ 5 levels "1","2","4","5",..: 5 3 2 3 2 3 2 3 2 5 ...
```

Output 3.2.1. (d): Structure of Mushroom data – converting into factor variables

As we can see, variables are now changed to factors and suit the data captured.

Moreover, from above structure of data, we can notice that there is only one level for variable x16. This variable can be removed since it is not of much use in defining the type of mushroom. Also, it may cause troubles at later stages while running the models like: it can trigger error:

`contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels

Removing the x16 variable

```
> # Removing x16 variable as it has only one level and irrelevant for data analysis
> M2=M1[,c(1:16,18:23)]
> names(M2)
 [1] "y"   "x1"  "x2"  "x3"  "x4"  "x5"  "x6"  "x7"  "x8"  "x9"  "x10" "x11" "x12" "x13" "x14" "x15" "x17" "x18" "x19" "x20" "x21"
[22] "x22"
> summary(M2)
 y       x1      x2       x3      x4      x5      x6      x7      x8       x9      x10     x11     x12     x13     x14     x15
 0:360   1:116   1:106   3 :  1   1:104   1:173   1:  7   1:404   1:365   3 : 64   1:392   1: 50   1: 27   1: 30   4:  7   4:  2
 1:140   3:108   3:188   4 : 49   2:396   4:160   2:493   2: 96   2:135   4 : 11   2:108   2:220   3:473   3:395   7:  3   7:  5
         5: 14   4:206   5 :116           6:116                           5 :101           3:155           4: 75   8:490   8:493
         6:262           9 :168           7: 51                           6 :140           4: 75
                         10:166                                           8 : 78
                                                                          10:  2
                                                                          11:104

 x17     x18     x19     x20     x21     x22
 2:  6   2:495   1: 58   3:223   1: 31   1: 50
 3:494   3:  5   5:442   4:256   3:100   2:238
                         7: 21   4:209   4:103
                                 5: 87   5: 39
                                 6: 73   6: 70
```

Output 3.2.1. (e): Deleting x16 variable and again looking at the counts

Lastly, we search for missing values.

```
> #Checking for missing value
> print(all(!is.na(M2)))
[1] TRUE
```

Output 3.2.1. (f): Finding Missing Values

There are no missing values and need no attention.

The irregularities in the data are now taken care of.

3.2.2. Understanding Relationships Between Data

At the second phase, we try to discover certain relationships: among the variables, between variables and the target. Furthermore, we make an effort to represent these relations using plots.

1. **Contingency Tables:** Contingence tables are valuable for revealing how edible/poisonous mushrooms are segmented across their features.

For all variables x1 through x22, the following holds

- prop.table(m,1) – value of each cell divided by the sum of the rows cells
- prop.table(m,2) – value of each cell divided by the sum of the column cells

where ‘m’ represents an object in R, here it is a table.

```
> prop.table(table(M2$x1,M2$y),2)

            0          1
  1 0.25000000 0.18571429
  3 0.22500000 0.19285714
  5 0.02500000 0.03571429
  6 0.50000000 0.58571429
> prop.table(table(M2$x1,M2$y),1)

           0         1
  1 0.7758621 0.2241379
  3 0.7500000 0.2500000
  5 0.6428571 0.3571429
  6 0.6870229 0.3129771
```

Output 3.2.2. (1a): Proportionate table – x1 v/s y

```
> prop.table(table(M2$x2,M2$y),2)

            0         1
  1 0.2416667 0.1357143
  3 0.3750000 0.3785714
  4 0.3833333 0.4857143
> prop.table(table(M2$x2,M2$y),1)

           0         1
  1 0.8207547 0.1792453
  3 0.7180851 0.2819149
  4 0.6699029 0.3300971
```

Output 3.2.2. (1b): Proportionate table – x2 v/s y

```
> prop.table(table(M2$x3,M2$y),2)

               0           1
  3  0.002777778 0.000000000
  4  0.108333333 0.071428571
  5  0.188888889 0.342857143
  9  0.338888889 0.328571429
  10 0.361111111 0.257142857
> prop.table(table(M2$x3,M2$y),1)

            0         1
  3  1.0000000 0.0000000
  4  0.7959184 0.2040816
  5  0.5862069 0.4137931
  9  0.7261905 0.2738095
  10 0.7831325 0.2168675
```

Output 3.2.2. (1c): Proportionate table – x3 v/s y

```
> prop.table(table(M2$x4,M2$y),2)

            0         1
  1 0.2388889 0.1285714
  2 0.7611111 0.8714286
> prop.table(table(M2$x4,M2$y),1)

            0         1
  1 0.8269231 0.1730769
  2 0.6919192 0.3080808
```

Output 3.2.2. (1d): Proportionate table – x4 v/s y

```
> prop.table(table(M2$x5,M2$y),2)

           0         1
  1 0.3888889 0.2357143
  4 0.3444444 0.2571429
  6 0.2666667 0.1428571
  7 0.0000000 0.3642857
> prop.table(table(M2$x5,M2$y),1)

           0         1
  1 0.8092486 0.1907514
  4 0.7750000 0.2250000
  6 0.8275862 0.1724138
  7 0.0000000 1.0000000
```

Output 3.2.2. (1e): Proportionate table – x5 v/s y

```
> prop.table(table(M2$x6,M2$y),2)

             0           1
  1 0.016666667 0.007142857
  2 0.983333333 0.992857143
> prop.table(table(M2$x6,M2$y),1)

           0         1
  1 0.8571429 0.1428571
  2 0.7180527 0.2819473
```

Output 3.2.2. (1f): Proportionate table – x6 v/s y

```
> prop.table(table(M2$x7,M2$y),2)

           0         1
  1 0.7722222 0.9000000
  2 0.2277778 0.1000000
> prop.table(table(M2$x7,M2$y),1)

           0         1
  1 0.6881188 0.3118812
  2 0.8541667 0.1458333
```

Output 3.2.2. (1g): Proportionate table – x7v/s y

```
> prop.table(table(M2$x8,M2$y),2)

           0         1
  1 0.8111111 0.5214286
  2 0.1888889 0.4785714
> prop.table(table(M2$x8,M2$y),1)

           0         1
  1 0.8000000 0.2000000
  2 0.5037037 0.4962963
```

Output 3.2.2. (1h): Proportionate table – x8 v/s y

```
> prop.table(table(M2$x9,M2$y),2)

                0           1
  3  0.136111111 0.107142857
  4  0.027777778 0.007142857
  5  0.188888889 0.235714286
  6  0.294444444 0.242857143
  8  0.152777778 0.164285714
  10 0.005555556 0.000000000
  11 0.194444444 0.242857143
> prop.table(table(M2$x9,M2$y),1)

               0          1
  3  0.76562500 0.23437500
  4  0.90909091 0.09090909
  5  0.67326733 0.32673267
  6  0.75714286 0.24285714
  8  0.70512821 0.29487179
  10 1.00000000 0.00000000
  11 0.67307692 0.32692308
```

Output 3.2.2. (1i): Proportionate table – x9 v/s y

```
> prop.table(table(M2$x10,M2$y),2)

           0         1
  1 0.7444444 0.8857143
  2 0.2555556 0.1142857
> prop.table(table(M2$x10,M2$y),1)

           0         1
  1 0.6836735 0.3163265
  2 0.8518519 0.1481481
```

Output 3.2.2. (1j): Proportionate table – x10 v/s y

```
prop.table(table(M2$x11,M2$y),2)

           0          1
1 0.11666667 0.05714286
2 0.48888889 0.31428571
3 0.23888889 0.49285714
4 0.15555556 0.13571429
prop.table(table(M2$x11,M2$y),1)

          0         1
1 0.8400000 0.1600000
2 0.8000000 0.2000000
3 0.5548387 0.4451613
4 0.7466667 0.2533333
```

Output 3.2.2. (1k): Proportionate table – x11 v/s y

```
prop.table(table(M2$x12,M2$y),2)

           0          1
1 0.06944444 0.01428571
3 0.93055556 0.98571429
prop.table(table(M2$x12,M2$y),1)

           0          1
1 0.92592593 0.07407407
3 0.70824524 0.29175476
```

Output 3.2.2. (1l): Proportionate table – x12 v/s y

```
prop.table(table(M2$x13,M2$y),2)

           0          1
1 0.06944444 0.03571429
3 0.77500000 0.82857143
4 0.15555556 0.13571429
prop.table(table(M2$x13,M2$y),1)

          0         1
1 0.8333333 0.1666667
3 0.7063291 0.2936709
4 0.7466667 0.2533333
```

Output 3.2.2. (1m): Proportionate table – x13 v/s y

```
prop.table(table(M2$x14,M2$y),2)

             0           1
4 0.019444444 0.000000000
7 0.005555556 0.007142857
8 0.975000000 0.992857143
prop.table(table(M2$x14,M2$y),1)

          0         1
4 1.0000000 0.0000000
7 0.6666667 0.3333333
8 0.7163265 0.2836735
```

Output 3.2.2. (1n): Proportionate table – x14 v/s y

```
prop.table(table(M2$x15,M2$y),2)

             0           1
4 0.002777778 0.007142857
7 0.013888889 0.000000000
8 0.983333333 0.992857143
prop.table(table(M2$x15,M2$y),1)

          0         1
4 0.5000000 0.5000000
7 1.0000000 0.0000000
8 0.7180527 0.2819473
```

Output 3.2.2. (1o): Proportionate table – x15 v/s y

```
prop.table(table(M2$x17,M2$y),2)

             0           1
2 0.008333333 0.021428571
3 0.991666667 0.978571429
prop.table(table(M2$x17,M2$y),1)

          0         1
2 0.5000000 0.5000000
3 0.7226721 0.2773279
```

Output 3.2.2. (1p): Proportionate table – x17 v/s y

```
prop.table(table(M2$x18,M2$y),2)

             0           1
2 0.991666667 0.985714286
3 0.008333333 0.014285714
prop.table(table(M2$x18,M2$y),1)

          0         1
2 0.7212121 0.2787879
3 0.6000000 0.4000000
```

Output 3.2.2. (1q): Proportionate table – x18 v/s y

```
prop.table(table(M2$x19,M2$y),2)

             0          1
1 0.13888889 0.05714286
5 0.86111111 0.94285714
prop.table(table(M2$x19,M2$y),1)

          0         1
1 0.8620690 0.1379310
5 0.7013575 0.2986425
```

Output 3.2.2. (1r): Proportionate table – x19 v/s y

```
prop.table(table(M2$x20,M2$y),2)

             0          1
3 0.45555556 0.42142857
4 0.50277778 0.53571429
7 0.04166667 0.04285714
prop.table(table(M2$x20,M2$y),1)

          0         1
3 0.7354260 0.2645740
4 0.7070312 0.2929688
7 0.7142857 0.2857143
```

Output 3.2.2. (1s): Proportionate table – x20 v/s y

```
prop.table(table(M2$x21,M2$y),2)

             0          1
1 0.06944444 0.04285714
3 0.23055556 0.12142857
4 0.39444444 0.47857143
5 0.14722222 0.24285714
6 0.15833333 0.11428571
prop.table(table(M2$x21,M2$y),1)

          0         1
1 0.8064516 0.1935484
3 0.8300000 0.1700000
4 0.6794258 0.3205742
5 0.6091954 0.3908046
6 0.7808219 0.2191781
```

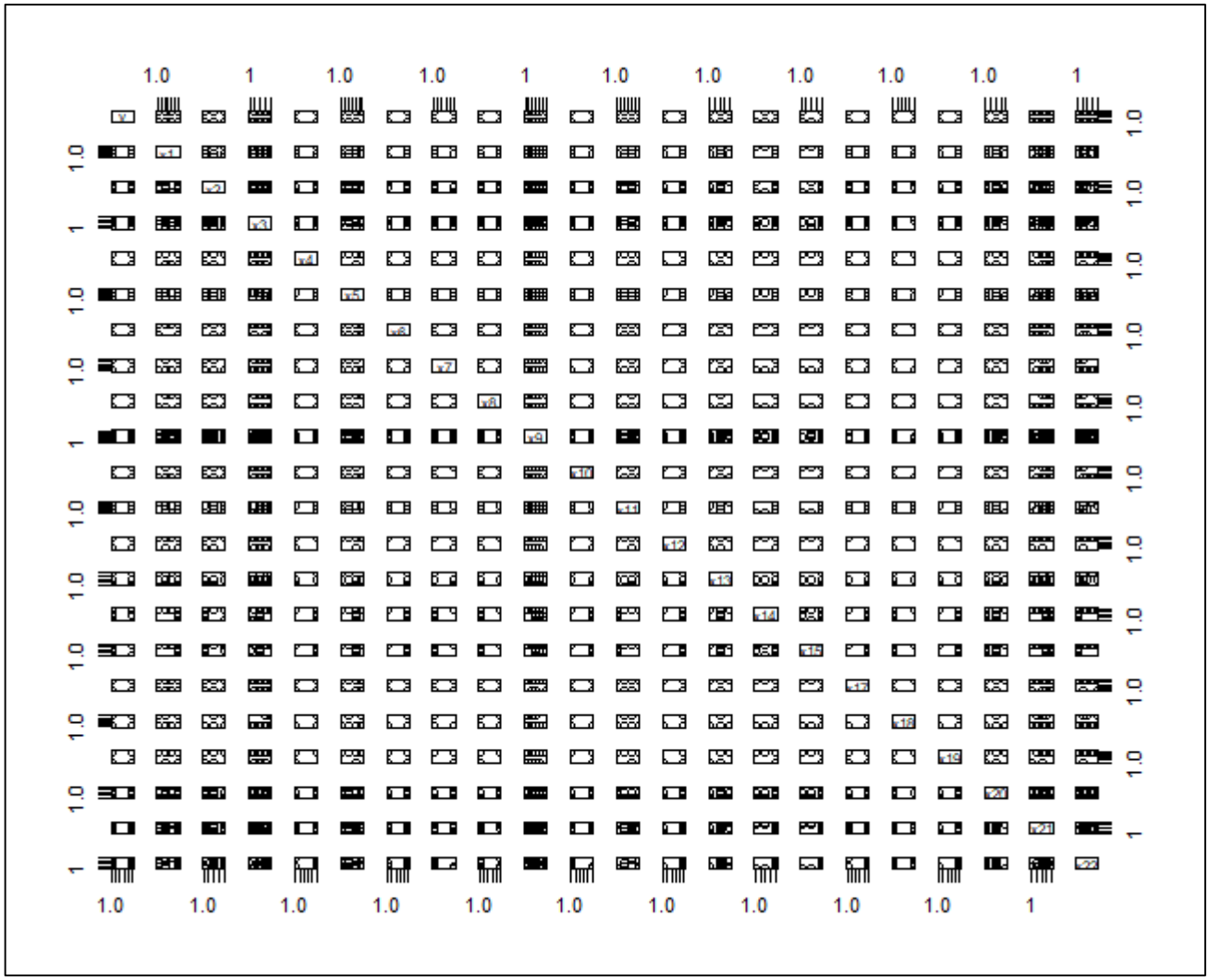
Output 3.2.2. (1t): Proportionate table – x21 v/s y

```
prop.table(table(M2$x22,M2$y),2)

           0          1
1 0.11666667 0.05714286
2 0.45277778 0.53571429
4 0.23888889 0.12142857
5 0.09166667 0.04285714
6 0.10000000 0.24285714
prop.table(table(M2$x22,M2$y),1)

          0         1
1 0.8400000 0.1600000
2 0.6848739 0.3151261
4 0.8349515 0.1650485
5 0.8461538 0.1538462
6 0.5142857 0.4857143
```

Output 3.2.2. (1u): Proportionate table – x22 v/s y

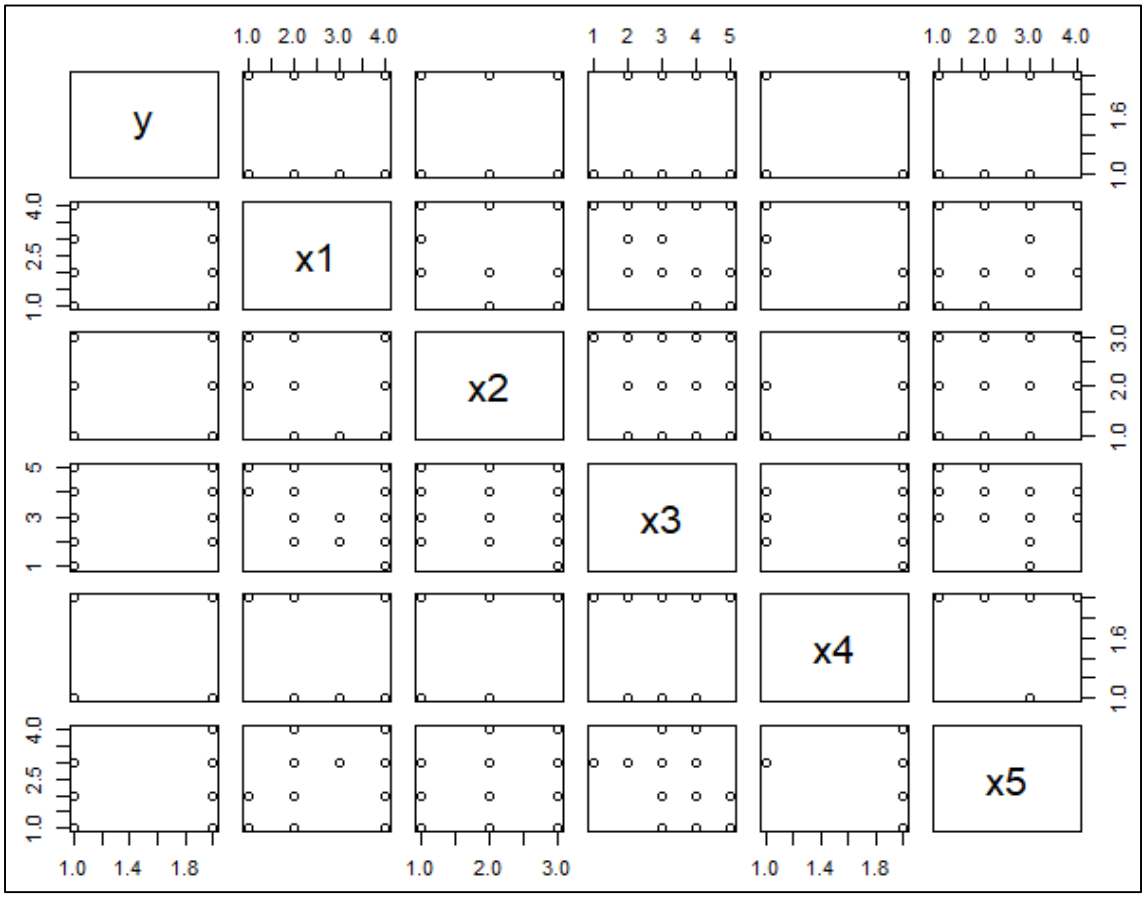
All of these tables specify the proportions of each level of a variable as either edible or poisonous. Some levels of these variables have high proportions of being edible and others have low. While in other variables, the levels show almost equal proportions of poisonous and edible types, in which case, it implies that a mushroom chosen at random based on these features has an equal chance of being edible or poisonous.

2. **Visual representation of relations among features and target variable:** The below plot shows the associations: all variables v/s all other variables

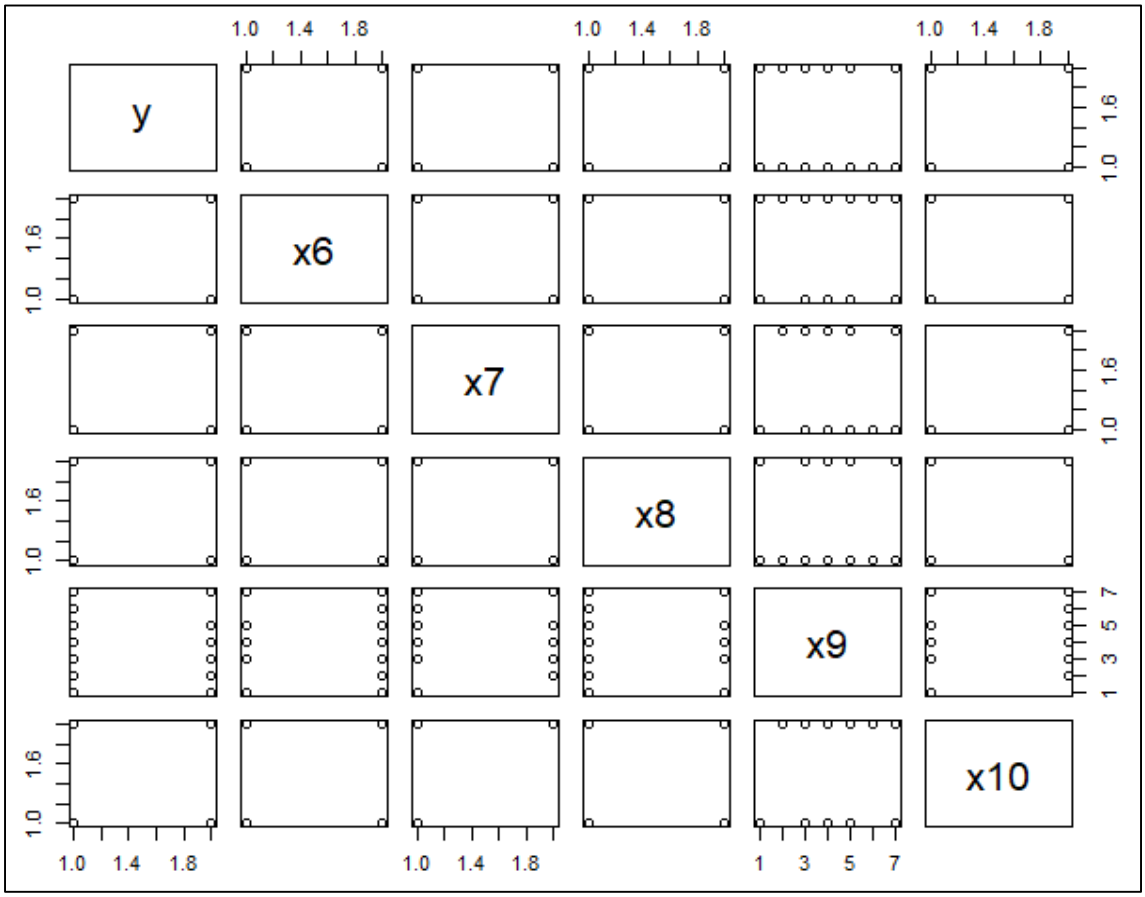


Output 3.2.2. (2a): Scatter plot – shows relation between all variables with every other variable

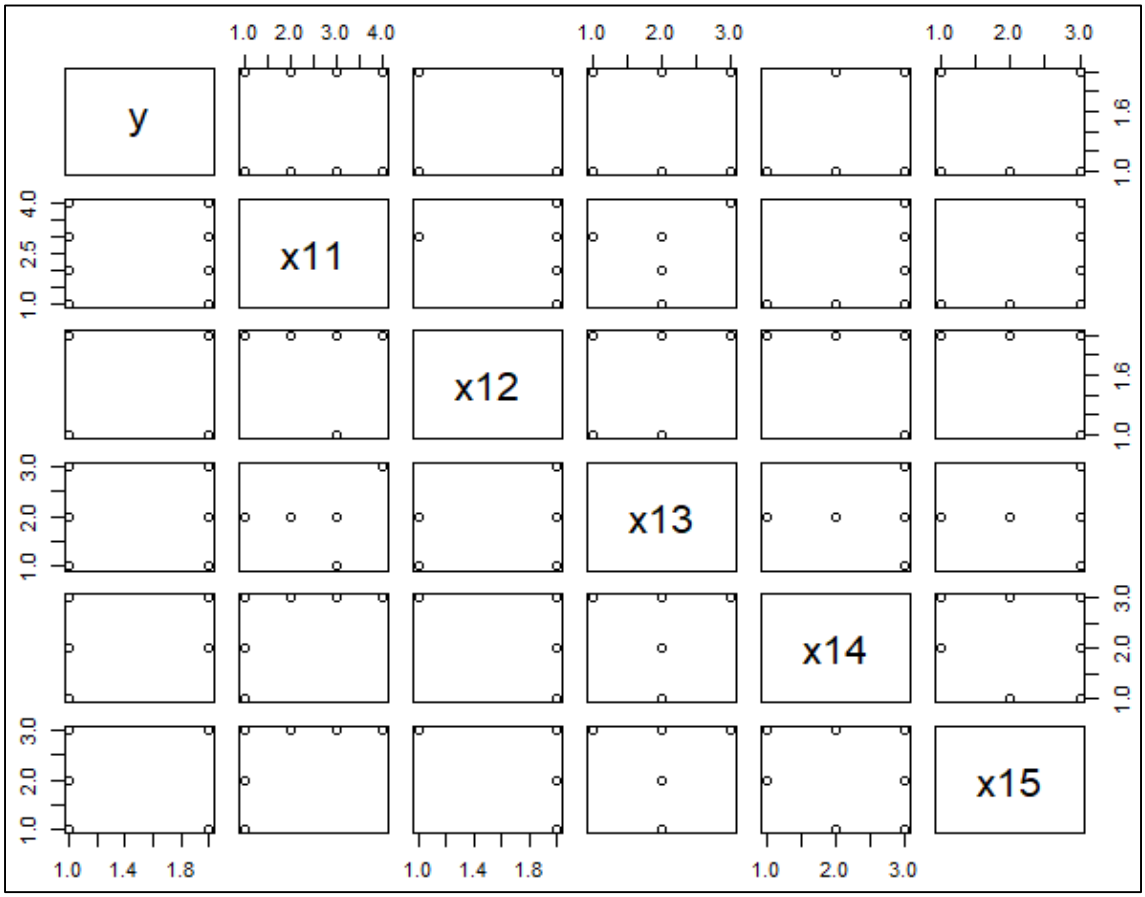
Because the dataset considered consists of several features, this is unpleasant and is not helping the purpose. We have, therefore split the above diagrams as smaller subsets of plots, where the relationships can easily be identified. Below are the plots taken with 5 – 6 variables at a time along with target variable.



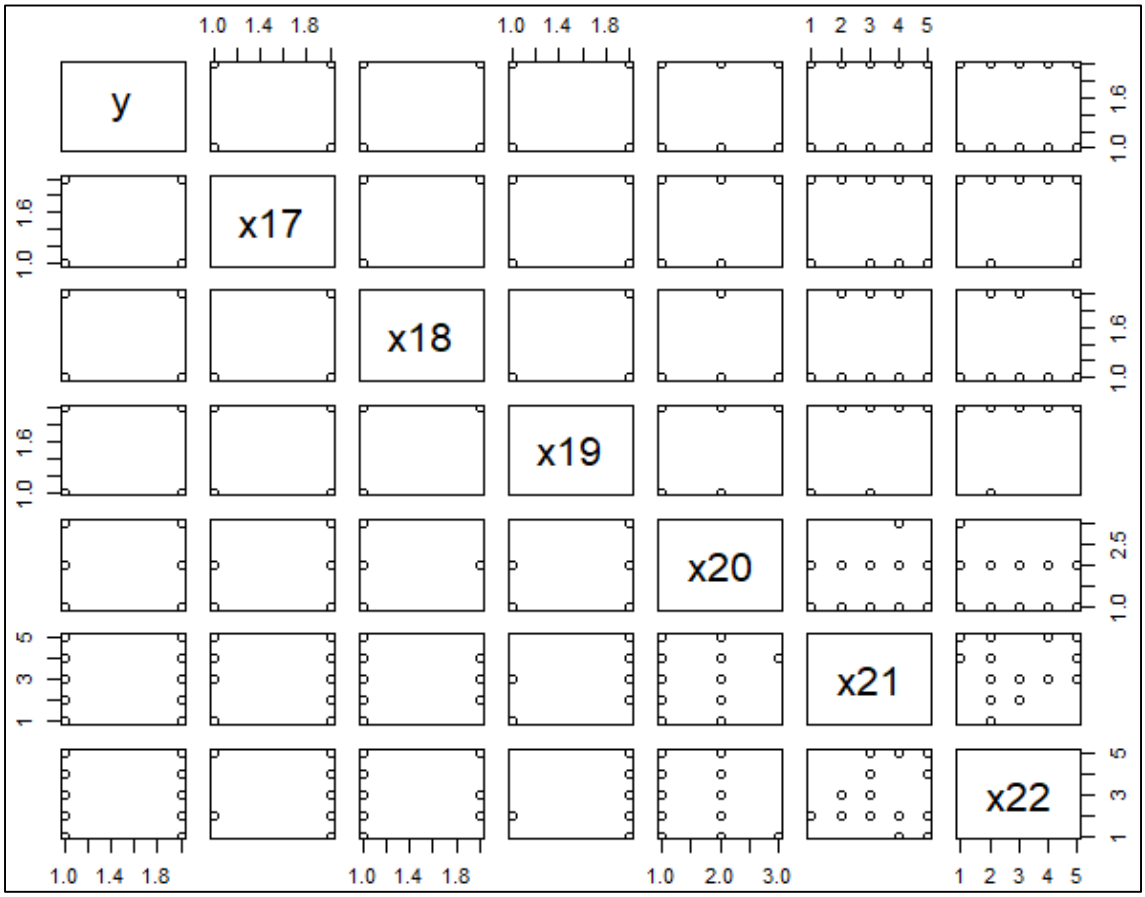
Output 3.2.2. (2b): Scatter plot – shows pairwise relations among variables x1 to x5 with y



Output 3.2.2. (2c): Scatter plot – shows pairwise relations among variables x6 to x10 with y



Output 3.2.2. (2d): Scatter plot – shows pairwise relations among variables x11 to x15 with y



Output 3.2.2. (2e): Scatter plot – shows pairwise relations among variables x17 to x22 with y

3. **Variable significance:** At this phase of modelling, we even find the significance of variables. We are using the chi-squared test of significance.

```
> # Finding if variables are significant by performing chi-squared test
> chisq.test(M2$y,M2$x1)

        Pearson's Chi-squared test

data:  M2$y and M2$x1
X-squared = 4.1043, df = 3, p-value = 0.2504

Warning message:
In chisq.test(M2$y, M2$x1) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3a): χ^2 test – y v/s x1

```
> chisq.test(M2$y,M2$x2)

        Pearson's Chi-squared test

data:  M2$y and M2$x2
X-squared = 7.9055, df = 2, p-value = 0.0192
```

Output 3.2.2. (3b): χ^2 test – y v/s x2

```
> chisq.test(M2$y,M2$x3)

        Pearson's Chi-squared test

data:  M2$y and M2$x3
X-squared = 15.404, df = 4, p-value = 0.003933

Warning message:
In chisq.test(M2$y, M2$x3) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3c): χ^2 test – y v/s x3

```
> chisq.test(M2$y,M2$x4)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x4
X-squared = 6.792, df = 1, p-value = 0.009157
```

Output 3.2.2. (3d): χ^2 test – y v/s x4

```
> chisq.test(M2$y,M2$x5)

        Pearson's Chi-squared test

data:  M2$y and M2$x5
X-squared = 147.04, df = 3, p-value < 2.2e-16
```

Output 3.2.2. (3e): χ^2 test – y v/s x5

```
> chisq.test(M2$y,M2$x6)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x6
X-squared = 0.15207, df = 1, p-value = 0.6966

Warning message:
In chisq.test(M2$y, M2$x6) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3f): χ^2 test – y v/s x6

```
> chisq.test(M2$y,M2$x7)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x7
X-squared = 9.8009, df = 1, p-value = 0.001744
```

Output 3.2.2. (3g): χ^2 test – y v/s x7

```
> chisq.test(M2$y,M2$x8)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x8
X-squared = 41.459, df = 1, p-value = 1.204e-10
```

Output 3.2.2. (3h): χ^2 test – y v/s x8

```
> chisq.test(M2$y,M2$x9)

        Pearson's Chi-squared test

data:  M2$y and M2$x9
X-squared = 6.6632, df = 6, p-value = 0.3531

Warning message:
In chisq.test(M2$y, M2$x9) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3i): χ^2 test – y v/s x9

```
> chisq.test(M2$y,M2$x10)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x10
X-squared = 11.06, df = 1, p-value = 0.0008822
```

Output 3.2.2. (3j): χ^2 test – y v/s x10

```
> chisq.test(M2$y,M2$x11)

        Pearson's Chi-squared test

data:  M2$y and M2$x11
X-squared = 31.793, df = 3, p-value = 5.787e-07
```

Output 3.2.2. (3k): χ^2 test – y v/s x11

```
> chisq.test(M2$y,M2$x12)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x12
X-squared = 4.9723, df = 1, p-value = 0.02576
```

Output 3.2.2. (3l): χ^2 test – y v/s x12

```
> chisq.test(M2$y,M2$x13)

        Pearson's Chi-squared test

data:  M2$y and M2$x13
X-squared = 2.5421, df = 2, p-value = 0.2805
```

Output 3.2.2. (3m): χ^2 test – y v/s x13

```
> chisq.test(M2$y,M2$x14)

        Pearson's Chi-squared test

data:  M2$y and M2$x14
X-squared = 2.7973, df = 2, p-value = 0.2469

Warning message:
In chisq.test(M2$y, M2$x14) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3n): χ^2 test – y v/s x14

```
> chisq.test(M2$y,M2$x15)

        Pearson's Chi-squared test

data:  M2$y and M2$x15
X-squared = 2.4339, df = 2, p-value = 0.2961

Warning message:
In chisq.test(M2$y, M2$x15) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3o): χ^2 test – y v/s x15

```
> chisq.test(M2$y,M2$x17)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x17
X-squared = 0.56264, df = 1, p-value = 0.4532

Warning message:
In chisq.test(M2$y, M2$x17) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3p): χ^2 test – y v/s x17

```
> chisq.test(M2$y,M2$x18)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x18
X-squared = 0.010021, df = 1, p-value = 0.9203

Warning message:
In chisq.test(M2$y, M2$x18) : Chi-squared approximation may be incorrect
```

Output 3.2.2. (3q): χ^2 test – y v/s x18

```
> chisq.test(M2$y,M2$x19)

        Pearson's Chi-squared test with Yates' continuity correction

data:  M2$y and M2$x19
X-squared = 5.7958, df = 1, p-value = 0.01606
```

Output 3.2.2. (3r): χ^2 test – y v/s x19

```
> chisq.test(M2$y,M2$x20)

        Pearson's Chi-squared test

data:  M2$y and M2$x20
X-squared = 0.4802, df = 2, p-value = 0.7866
```

Output 3.2.2. (3s): χ^2 test – y v/s x20

```
> chisq.test(M2$y,M2$x21)

        Pearson's Chi-squared test

data:  M2$y and M2$x21
X-squared = 15.496, df = 4, p-value = 0.003776
```

Output 3.2.2. (3t): χ^2 test – y v/s x21

```
> chisq.test(M2$y,M2$x22)

        Pearson's Chi-squared test

data:  M2$y and M2$x22
X-squared = 29.552, df = 4, p-value = 6.038e-06
```

Output 3.2.2. (3u): χ^2 test – y v/s x22

As p – values (probability values) of x1, x6, x9, x13, x14, x15, x17, x18, x20 are greater than α, these variables may be insignificant in determining the dependent variable and all other variables have lesser p – values, which indicates that they significantly determine the dependent variable.

3.2.3. Modelling

The third phase of ML is to fit an appropriate model to the data that takes few variables but explains most of the substantial differences adequately. In our case, we are trying to locate key features of mushroom that will classify it as edible or poisonous.

Initially, dividing data as train and test data. Then, we performed a 5 – fold cross validation on the train data. Later, we fitted different classification algorithms such as k-NN, SVM, Random Forest, Decision Trees, and GLM on this train data.

1. **Splitting the data:** Here, we split our dataset into two: Train & Test.

```
> dim(M2)
[1] 500  22
> dim(Train)
[1] 400  22
> dim(Test)
[1] 100  22
```

Output 3.2.3. (1a): Splitting – Train & Test

2. **Deciding to do a 5 – fold Cross Validation:** The train data is divided into 5 samples of equal size. After division of the data 4 samples are trained to fit the model and 1 sample is tested using the fitted model. We have done this using 3 repeats.

```
> # Run algorithms using 5-fold cross validation
> library(caret)
> control = trainControl(method="repeatedcv", number=5, repeats=3)
```

Output 3.2.3. (2a): Running a 5-Fold CV

3. **Running model pipeline:** Fitting different classification algorithms and checking for their accuracies.

a. k-NN

```
k-Nearest Neighbors

400 samples
 21 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 321, 320, 320, 319, 320, 321, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  1  0.7399514  0.3254608
  3  0.7740888  0.3611547
  5  0.8091958  0.4270617
  7  0.8183216  0.4409094

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
```

Output 3.2.3. (3a): Employing k-NN Algorithm

b. DT

```
CART

400 samples
 21 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 321, 320, 320, 319, 320, 321, ...
Resampling results across tuning parameters:

  cp    Accuracy   Kappa
  0.01  0.8174885  0.4378799
  0.05  0.8225096  0.4453328
  0.10  0.8225096  0.4453328

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.1.
```

Output 3.2.3. (3b): Employing DT Algorithm

c. GLM

```
Generalized Linear Model

400 samples
 21 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 321, 320, 320, 319, 320, 321, ...
Resampling results:

  Accuracy   Kappa
  0.7723828  0.3490256
```

Output 3.2.3. (3c): Employing GLM Algorithm

d. Random Forest

```
Random Forest

400 samples
 21 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 320, 319, 320, 320, 321, 321, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8191282  0.4322720
  24    0.7775117  0.3564410
  46    0.7566879  0.3159725

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Output 3.2.3. (3d): Employing RF Algorithm

e. SVM

```
Support Vector Machines with Radial Basis Function Kernel

400 samples
 21 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 321, 320, 320, 319, 320, 321, ...
Resampling results across tuning parameters:

  sigma  Accuracy   Kappa
  0.01   0.8200199  0.4373778
  0.05   0.8066855  0.3855275
  0.10   0.8024978  0.3810492

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 = 0.01 and C = 1.
```

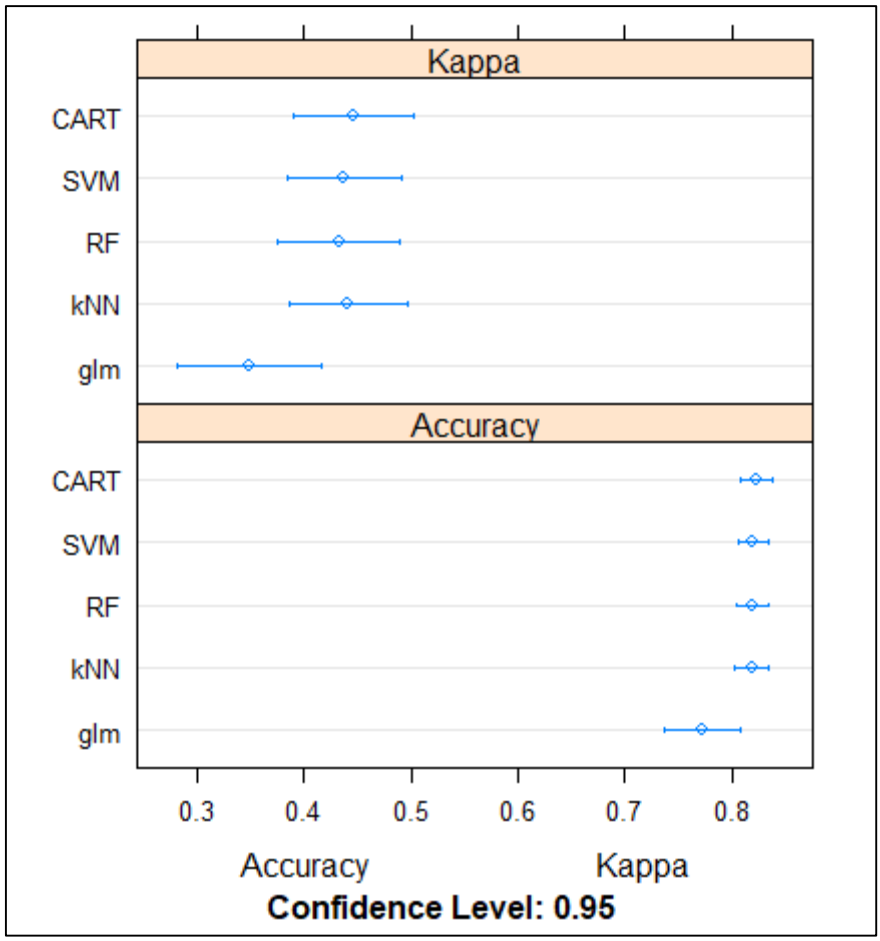
Output 3.2.3. (3e): Employing SVM Algorithm

4**. Inspecting accuracy:** After running these classification algorithms, we compare their accuracies. That is, how well are these models explaining the data. Below table displays this comparison.

```
Call:
summary.resamples(object = M3)

Models: SVM, CART, kNN, glm, RF
Number of resamples: 15

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SVM  0.7750000 0.8113133 0.8125000 0.8200199 0.8364715 0.8641975    0
CART 0.7750000 0.8113133 0.8250000 0.8225096 0.8364715 0.8765432    0
kNN  0.7777778 0.8037975 0.8227848 0.8183216 0.8312500 0.8888889    0
glm  0.5750000 0.7609968 0.7875000 0.7723828 0.8049842 0.8641975    0
RF   0.7750000 0.8062500 0.8250000 0.8191282 0.8395062 0.8500000    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SVM  0.2436975 0.4029996 0.4241843 0.4373778 0.5003581 0.6097240    0
CART 0.2436975 0.4029996 0.4531250 0.4453328 0.5003581 0.6505608    0
kNN  0.2845927 0.3886079 0.4519326 0.4409094 0.4863289 0.6901827    0
glm  0.1099476 0.2800411 0.3474088 0.3490256 0.4094101 0.6213345    0
RF   0.2436975 0.3779026 0.4531250 0.4322720 0.5241753 0.5471698    0
```

Output 3.2.3. (4a): Comparing results of various algorithms

The visual representation of these accuracies is presented in the dot plot below



Output 3.2.3. (4b): Pictorial representation of results of various algorithms

5**. Inference made based on accuracy:** As we can notice, Random Forest has the best accuracy of all. We can do two things at this stage: Parameter Tuning and Variable Importance.

a**. Tuning the parameters in RF:** Hyper parameters in Random Forest are tuned to extract the best parameters for final model. Here,

i. Setting the number of trees as 100 or 200 or 300
ii. Number of variables in each tree would range from 1 to 6

```
400 samples
 21 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 321, 320, 320, 319, 320, 321, ...
Resampling results across tuning parameters:

  mtry  ntree  Accuracy   Kappa
  1     100    0.7225137  0.0000000
  1     200    0.7225137  0.0000000
  1     300    0.7225137  0.0000000
  2     100    0.8199991  0.4355296
  2     200    0.8208430  0.4389490
  2     300    0.8216763  0.4420297
  3     100    0.8225096  0.4453328
  3     200    0.8225096  0.4453328
  3     300    0.8225096  0.4453328
  4     100    0.8225096  0.4453328
  4     200    0.8225096  0.4453328
  4     300    0.8225096  0.4453328
  5     100    0.8216763  0.4433820
  5     200    0.8225096  0.4453328
  5     300    0.8225096  0.4453328
  6     100    0.8225096  0.4453328
  6     200    0.8225096  0.4453328
  6     300    0.8216763  0.4433820

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were mtry = 3 and ntree = 100.
```

Output 3.2.3. (5a): Tuning parameters in RF

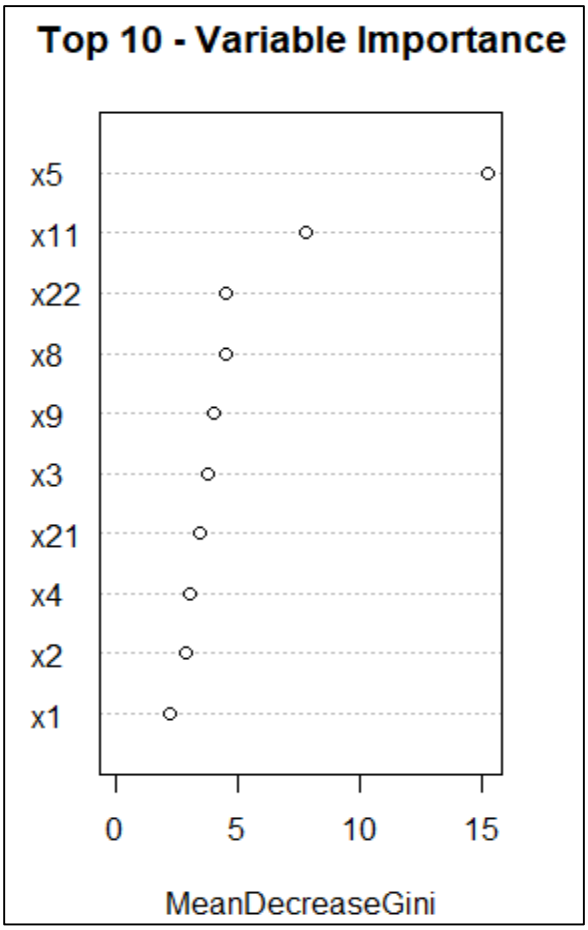
b**. Variable Importance:** Here, we identify the key features that establish whether a mushroom is edible or not.

From the following table, we observe that variables x5, x8, x11, x22 mostly influence the type of mushroom.

	MeanDecreaseGini
x1	2.1971011
x2	2.9113411
x3	3.7505024
x4	3.0454743
x5	15.2631907
x6	0.1932431
x7	1.1393105
x8	4.5285122
x9	4.0040750
x10	1.7429771
x11	7.8514497
x12	0.4340706
x13	1.5383482
x14	0.3569888
x15	0.4555700
x17	0.3381016
x18	0.1105427
x19	0.7001473
x20	1.5975524
x21	3.4782738
x22	4.5339521

Output 3.2.3. (5b): Feature Selection in RF

The plot showing variable importance is as follows



Output 3.2.3. (5c): Graphical representation of Feature Selection in RF

This feature selection plot gives the top 10 variables that prominently define a mushroom into its type.

6. Final Model: The final step under this phase of modelling is to fit a model with 4 key features x5, x8, x11, x22. We fit a Logistic Regression due to its simplicity to the end-user.

```
Call:
NULL

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.09394  -0.64743  -0.58114   0.00013   2.08643

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.260e+00  2.015e+03  -0.001   0.9991
x54          3.932e-01  3.195e-01   1.231   0.2184
x56          8.735e-01  2.015e+03   0.000   0.9997
x57          2.093e+01  1.459e+03   0.014   0.9886
x112         5.667e-01  8.408e-01   0.674   0.5003
x113        -3.054e-01  2.015e+03   0.000   0.9999
x114         1.667e+00  8.799e-01   1.895   0.0582 .
x222        -1.204e-07  2.015e+03   0.000   1.0000
x224        -1.562e-01  2.015e+03   0.000   0.9999
x225        -1.156e+00  2.015e+03  -0.001   0.9995
x226                NA         NA      NA       NA
x82          2.036e-01  2.015e+03   0.000   0.9999
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 470.54  on 399  degrees of freedom
Residual deviance: 338.01  on 389  degrees of freedom
AIC: 360.01

Number of Fisher Scoring iterations: 17
```

Output 3.2.3. (6a): Decision to apply Logistic Regression – calculating required parameters

We have fitted the model using the 4 important features.
The following is the accuracy obtained from Logistic Regression model.

```
Generalized Linear Model

400 samples
  4 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 320, 320, 320, 320, 320, 320, ...
Resampling results:

  Accuracy   Kappa
  0.8258333  0.4597917
```

Output 3.2.3. (6b): Employing Logistic Regression as the final model

3.2.4. Validation & Prediction

The last phase is the evaluation of the model. This is a crucial step of the machine learning process. This is where we can find if the algorithms perform well at any circumstances. That is, determining how well do they perform on unseen data.

1. **Validating the accuracy of the trained data**

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 290  68
         1   0  42

               Accuracy : 0.83
                 95% CI : (0.7895, 0.8655)
    No Information Rate : 0.725
    P-Value [Acc > NIR] : 5.665e-07

                  Kappa : 0.4725
 Mcnemar's Test P-Value : 4.476e-16

            Sensitivity : 1.0000
            Specificity : 0.3818
         Pos Pred Value : 0.8101
         Neg Pred Value : 1.0000
             Prevalence : 0.7250
         Detection Rate : 0.7250
   Detection Prevalence : 0.8950
      Balanced Accuracy : 0.6909

       'Positive' Class : 0
```

Output 3.2.4. (a): Confusion Matrix – Train data

2**. Validating the accuracy of the test data**

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 70 21
         1  0  9

               Accuracy : 0.79
                 95% CI : (0.6971, 0.8651)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.02883

                  Kappa : 0.375
 Mcnemar's Test P-Value : 1.275e-05

            Sensitivity : 1.0000
            Specificity : 0.3000
         Pos Pred Value : 0.7692
         Neg Pred Value : 1.0000
             Prevalence : 0.7000
         Detection Rate : 0.7000
   Detection Prevalence : 0.9100
      Balanced Accuracy : 0.6500

       'Positive' Class : 0
```

Output 3.2.4. (b): Confusion Matrix – Test data

From the above confusion matrices, we can see that both train and test data show almost nearing accuracies. We can therefore conclude that Logistic regression with 4 variables: Odour, Habitat, Gill size, Stalk root classify the mushroom correctly 80% of the time.

CHAPTER 4
SUMMARY, INFERENCES & CONCLUSION

SUMMARY, INFERENCES & CONCLUSION

4.1. Summary & Inferences

Classification systems play a major role in decision-making tasks by categorizing the available information based on some principles.

In this project, we focused on prediction accuracy. Our intent was to learn a model that has a good generalization performance on the Mushroom dataset. We recognized the characteristics of the mushroom that are best-suited for its classification. In a way, we compared and evaluated the relative performance of some well-known classification models: k-NN, Decision Trees, SVM, Random Forest, GLM, for the problem at hand.

Over all the algorithms, Random Forest gave a pre-eminent accuracy. We have then tried to tune the parameter of Random Forest Technique to obtain superior performance.

The summary of accuracies of various algorithms are as follows.

Table 4.1.: Accuracy obtained from different algorithms

k-NN	SVM	Decision Trees	Random Forest	GLM
81%	80%	82%	82%	77%

After this, we have identified 4 key variables that mostly categorize the mushroom as poisonous or edible. These variables are: Odour, Gill Size, Stalk Root, Habitat.

Using these, we have fitted logistic regression and obtained the final model. The accuracy of logistic regression was 83% on train data and around 80% on test data. The performance of the model is considered reasonable and we accept it.

4.2. Conclusion

A decision to choose Logistic regression model with 4 variables: Odour, Gill Size, Stalk Root, Habitat has been accepted to group the mushroom as edible or poisonous.

APPENDIX

- **Appendix A: R Code**
- **Appendix B: Sampled Mushroom Data**
- **Bibliography**

Appendix A: R Code

```
# Checking and setting the path
getwd()
setwd("C:/Users/mghnp/Desktop/RProject")
getwd()

# Loading required packages
install.packages("mice")
install.packages("randomForest")
install.packages("ggplot2")
install.packages("glmnet")
install.packages("Hmisc")
install.packages("e1071")
install.packages("caret")
library(mice)
library(randomForest)
library(ggplot2)
library(glmnet)
library(Hmisc)
library(e1071)
library(caret)

# Reading the data file into R
# File name: Mushroom
M1=read.csv("Mushroom.csv")

# Looking at basic framework of the dataset – observations, attributes, datatypes
dim(M1)
class(M1)
names(M1)
str(M1)
cols=c("y","x1","x2",  "x3","x4","x5",  "x6","x7","x8","x9",  "x10",  "x11",  "x12",  "x13",  "x14",
"x15","x16", "x17", "x18", "x19", "x20", "x21", "x22")
M1[cols]=lapply(M1[cols], factor)
str(M1)
summary(M1)
View(M1)

# Removing x16 variable as it has only one level and irrelevant for data analysis
M2=M1[,c(1:16,18:23)]
str(M2)
summary(M2)

#Checking for missing value
print(all(!is.na(M2)))

# Numerical relationships between variables
prop.table(table(M2$x1,M2$y),2)
prop.table(table(M2$x1,M2$y),1)
prop.table(table(M2$x2,M2$y),2)
prop.table(table(M2$x2,M2$y),1)
```

```
prop.table(table(M2$x3,M2$y),2)
prop.table(table(M2$x3,M2$y),1)
prop.table(table(M2$x4,M2$y),2)
prop.table(table(M2$x4,M2$y),1)
prop.table(table(M2$x5,M2$y),2)
prop.table(table(M2$x5,M2$y),1)
prop.table(table(M2$x6,M2$y),2)
prop.table(table(M2$x6,M2$y),1)
prop.table(table(M2$x7,M2$y),2)
prop.table(table(M2$x7,M2$y),1)
prop.table(table(M2$x8,M2$y),2)
prop.table(table(M2$x8,M2$y),1)
prop.table(table(M2$x9,M2$y),2)
prop.table(table(M2$x9,M2$y),1)
prop.table(table(M2$x10,M2$y),2)
prop.table(table(M2$x10,M2$y),1)
prop.table(table(M2$x11,M2$y),2)
prop.table(table(M2$x11,M2$y),1)
prop.table(table(M2$x12,M2$y),2)
prop.table(table(M2$x12,M2$y),1)
prop.table(table(M2$x13,M2$y),2)
prop.table(table(M2$x13,M2$y),1)
prop.table(table(M2$x14,M2$y),2)
prop.table(table(M2$x14,M2$y),1)
prop.table(table(M2$x15,M2$y),2)
prop.table(table(M2$x15,M2$y),1)
prop.table(table(M2$x17,M2$y),2)
prop.table(table(M2$x17,M2$y),1)
prop.table(table(M2$x18,M2$y),2)
prop.table(table(M2$x18,M2$y),1)
prop.table(table(M2$x19,M2$y),2)
prop.table(table(M2$x19,M2$y),1)
prop.table(table(M2$x20,M2$y),2)
prop.table(table(M2$x20,M2$y),1)
prop.table(table(M2$x21,M2$y),2)
prop.table(table(M2$x21,M2$y),1)
prop.table(table(M2$x22,M2$y),2)
prop.table(table(M2$x22,M2$y),1)

# Visual relationships between variables
pairs(M2)

#Dividing dataset into smaller dataset for clear picturing
M_a=M2[,c(1,2:6)]
M_b=M2[,c(1,7:11)]
M_c=M2[,c(1,12:16)]
M_d=M2[,c(1,17:22)]
pairs(M_a)
pairs(M_b)
pairs(M_c)
pairs(M_d)
```

```
# Finding if variables are significant by performing chi-squared test
chisq.test(M2$y,M2$x1)
chisq.test(M2$y,M2$x2)
chisq.test(M2$y,M2$x3)
chisq.test(M2$y,M2$x4)
chisq.test(M2$y,M2$x5)
chisq.test(M2$y,M2$x6)
chisq.test(M2$y,M2$x7)
chisq.test(M2$y,M2$x8)
chisq.test(M2$y,M2$x9)
chisq.test(M2$y,M2$x10)
chisq.test(M2$y,M2$x11)
chisq.test(M2$y,M2$x12)
chisq.test(M2$y,M2$x13)
chisq.test(M2$y,M2$x14)
chisq.test(M2$y,M2$x15)
chisq.test(M2$y,M2$x17)
chisq.test(M2$y,M2$x18)
chisq.test(M2$y,M2$x19)
chisq.test(M2$y,M2$x20)
chisq.test(M2$y,M2$x21)
chisq.test(M2$y,M2$x22)

# Splitting dataset as Train and Test data
Train_Rows=sample(1:nrow(M2), size=0.8*nrow(M2))
Train_Rows
Train=M2[Train_Rows, ]
Test=M2[-Train_Rows, ]
dim(M2)
dim(Train)
dim(Test)

# Model Pipeline
# Run algorithms using 5-fold cross validation
library(caret)
control = trainControl(method="repeatedcv", number=5, repeats=3)

# GLM
set.seed(9)
fit.glm = train(y~., data=Train, method="glm", metric="Accuracy", trControl=control)
print(fit.glm)

# CART
set.seed(9)
grid = expand.grid(.cp=c(0.01,0.05,0.1))
fit.cart = train(y~., data=Train, method="rpart", metric="Accuracy", tuneGrid=grid,
trControl=control)
print(fit.cart)
```

```
# SVM
set.seed(9)
grid = expand.grid(.sigma=c(0.01,0.05,0.1), .C=c(1))
fit.svm = train(y~., data=Train, method="svmRadial", metric="Accuracy", tuneGrid=grid, trControl=control)
print(fit.svm)

# kNN
set.seed(9)
grid = expand.grid(.k=c(1,3,5,7))
fit.knn = train(y~., data=Train, method="knn", metric="Accuracy", tuneGrid=grid, trControl=control)
print(fit.knn)

#Random Forest
fit.rf = train(y~., data=Train, method="rf", metric="Accuracy", trControl=control)
print(fit.rf)

# Comparing algorithms
M3=resamples(list(SVM=fit.svm, CART=fit.cart, kNN=fit.knn, glm=fit.glm, RF=fit.rf))
summary(M3)
dotplot(M3)

#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

# Choosing desired parameter values
library(caret)
library(randomForest)
control = trainControl(method="repeatedcv", number=5, repeats=3)
tunegrid = expand.grid(.mtry=c(1:6), .ntree=c(100, 200, 300))
set.seed(9)
M4= train(y~., data=Train,method=customRF, tuneGrid=tunegrid, trControl=control)
print(M4)

# Feature Selection
rf_model3 = randomForest(y ~ ., data =Train, ntree=100, mtry=2)
varImpPlot(rf_model3, sort = T, n.var = 10, main = "Top 10 - Variable Importance")
importance(rf_model3)
```

```
#Running final model using Logistic Regression
dim(Train)
trainControl = trainControl(method="repeatedcv", number=5, repeats=3)
final.glm =train (y~ x5 + x11 + x9 + x8 + x21, data=Train, method="glm", trControl=trainControl)
summary(final.glm)
print (final.glm)

# Finding accuracy of Logistic regression on Train data
predslog = predict(final.glm, data=Train, type="raw")
tabtrain = table(Predicted = predslog, Actual = Train$y)
caret::confusionMatrix(predslog,Train$y)

# Running and finding accuracy of the final model on Test data
dim(Test)
names(Test)
M5 = predict(final.glm,newdata=Test,type="raw")
tabTest = table(Predicted = M5, Actual = Test$y)
caret::confusionMatrix(M5,Test$y)
```

Appendix B: Sampled Mushroom Data

Table 1: Variable Representations

Nature	y
Cap_Shape	x1
Cap_Surface	x2
Cap_Colour	x3
Bruises	x4
Odour	x5
Gill_Attachment	x6
Gill_Spacing	x7
Gill_Size	x8
Gill_Colour	x9
Stalk_Shape	x10
Stalk_Root	x11
Stalk_Surface_Above_Ring	x12
Stalk_Surface_Below_Ring	x13
Stalk_Colour_Above_Ring	x14
Stalk_Colour_Below_Ring	x15
Veil_Type	x16
Veil_Colour	x17
Ring_Number	x18
Ring_Type	x19
Spore_Print_Colour	x20
Population	x21
Habitat	x22

Table 2: Sampled Data – 500 Observations

y	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22
1	6	3	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	4	6
0	1	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	3	4	1	6	1	2	1	5	2	3	3	3	8	8	1	3	2	1	4	1	2
0	1	3	9	2	1	1	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	4
1	6	4	9	2	7	1	1	2	8	1	3	3	3	8	8	1	3	2	5	3	5	2
0	1	3	10	2	1	1	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	4	10	2	4	1	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	4	10	2	1	1	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	3	10	2	1	1	1	1	11	1	2	3	3	8	8	1	2	2	5	4	4	2
1	6	4	9	2	7	2	1	2	5	1	3	3	3	8	8	1	2	2	5	4	5	6
0	5	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	2	2	5	4	6	6
0	3	1	9	1	6	2	2	1	5	2	3	3	3	8	8	1	2	2	1	4	1	2
1	6	3	5	2	7	2	1	2	6	1	3	3	3	8	8	1	2	2	5	3	4	2
1	6	3	5	2	7	2	1	2	5	1	3	3	3	8	8	1	2	2	5	4	4	6
1	6	4	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	5	2
0	1	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	3	5	4	4	4
1	3	3	9	2	7	2	1	2	6	1	3	3	3	8	8	1	3	3	5	4	5	2
0	1	3	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	3	5	4	3	4
1	6	4	9	2	7	2	1	2	5	1	3	3	3	8	8	1	3	3	5	4	4	6
0	6	4	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	3	5	4	3	4
0	1	4	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
1	6	4	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	4	6
0	1	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4

1	6	4	9	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	5	2
0	6	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2
0	1	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
1	6	4	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	6
0	3	1	4	1	6	2	2	1	6	2	3	3	3	8	8	1	3	2	1	4	1	2
0	6	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	4	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	1	4	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	5	6
0	3	1	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	1	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	4
0	3	1	10	2	4	2	2	2	11	2	1	3	3	8	8	1	3	2	5	4	5	1
0	6	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	5
0	3	3	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	3	4	5	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	5
1	6	4	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	3	5	1	6	2	2	1	5	2	3	1	3	8	8	1	3	2	1	4	4	2
1	6	4	9	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	4	2
0	3	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	6	6
0	1	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	3	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	2
0	1	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2

0	6	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	4	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
0	3	4	5	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	2
1	6	4	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	2
1	1	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	2
1	1	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
1	1	4	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
1	6	1	9	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1
1	6	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	4
1	6	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	2
1	5	1	4	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	6	6
1	6	4	10	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	2
1	6	3	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
1	5	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	6	6
1	6	3	9	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	5	2
1	3	3	4	1	6	2	2	1	5	2	3	3	3	8	8	1	3	2	1	4	1	2
1	3	1	4	1	6	2	2	1	4	2	3	3	3	8	8	1	3	2	1	4	1	2
1	1	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
1	1	3	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
1	3	3	9	2	4	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1
1	6	4	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
1	6	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
1	6	3	5	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	5	2
1	1	3	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2

1	3	4	5	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	5
1	6	4	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
1	1	4	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	4
1	6	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
1	5	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	5	6
1	6	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
1	1	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
1	1	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	2
1	1	3	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
1	1	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	2
1	6	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	6	6
1	3	4	5	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	6	2
1	6	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	2
1	3	4	10	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	5
1	1	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
1	1	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	2
1	6	1	10	2	4	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
1	1	4	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
1	3	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	2
1	1	3	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
1	5	1	5	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	4	6	6
1	3	1	5	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	3	5	6
1	1	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	4
1	3	1	4	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	3	5	6
1	6	4	5	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	5

1	6	1	9	1	6	2	2	1	8	2	3	3	1	8	8	1	3	2	1	3	4	2
1	1	3	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
1	6	3	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
1	1	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
1	6	4	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
0	3	3	9	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	4	5	1
0	6	3	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	1	9	1	6	2	2	1	4	2	3	1	3	8	8	1	3	2	1	3	4	2
0	1	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	4	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	3	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	1	10	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	4	5	1
0	6	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	6	6
0	6	4	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	4	10	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	4	2
0	6	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	4
0	1	4	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	6	6
1	6	4	9	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	4	6
0	1	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	5
1	6	3	9	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	5	6
0	3	1	9	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
0	6	3	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2

0	3	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	2
0	1	4	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
0	1	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	3	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	4
0	1	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	3	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	4	2
0	6	3	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	3	4	1	6	2	2	1	8	2	3	3	3	8	8	1	3	2	1	3	1	2
0	6	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	4
1	3	3	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	5	6
0	6	4	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	4	5	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	2
0	6	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	1	5	1	6	2	2	1	4	2	3	3	3	8	8	1	3	2	1	3	4	2
1	6	4	9	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	4	2
0	1	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	4	5	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	6	5
0	3	1	4	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	6	6
0	6	1	4	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	3	6	6
0	1	4	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	1	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	3	10	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1

1	6	4	9	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	3	4	2
0	6	1	9	1	6	2	2	1	5	2	3	1	3	8	8	1	3	2	1	3	1	2
0	1	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	3	10	2	4	2	2	2	11	2	1	3	3	8	8	1	3	2	5	4	5	1
0	1	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	1	5	2	6	2	1	1	8	2	1	3	3	4	7	1	3	2	5	4	6	1
0	1	4	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	4
0	3	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	2
0	6	3	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	1	9	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	1	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
1	6	4	9	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	5	6
0	6	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	4
0	3	1	10	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1
0	6	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	4	5	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	5
0	1	4	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
0	1	3	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	2
0	1	4	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	1	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	4	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	4
0	1	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2

0	6	1	4	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	4	5	6
0	1	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
0	3	1	10	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	1	4	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	1	10	2	4	2	2	2	6	2	1	3	3	8	8	1	3	2	5	4	5	1
0	6	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	1	5	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	4	6	6
0	1	3	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	4	10	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	5
0	5	1	4	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	4	5	6
0	3	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	5
0	6	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	5
0	6	1	5	1	6	2	2	1	4	2	3	3	1	8	8	1	3	2	1	4	4	2
0	6	1	5	1	6	2	2	1	8	2	3	1	1	8	8	1	3	2	1	3	1	2
0	3	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	4	5
0	6	1	10	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
0	1	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	1	4	1	6	2	2	1	6	2	3	3	3	8	8	1	3	2	1	3	4	2
0	3	4	5	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	6	2
0	1	4	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	4	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	3	9	1	6	2	2	1	5	2	3	1	3	8	8	1	3	2	1	4	4	2
0	6	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	4

0	1	3	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	1	5	2	6	2	1	1	6	2	1	3	3	4	8	1	3	2	5	4	6	1
1	6	3	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	4	6
0	6	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	5
0	1	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	6
0	3	4	5	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	2
0	6	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
0	1	4	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
0	1	4	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
1	6	3	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	4	6
0	6	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
0	3	1	4	1	6	2	2	1	4	2	3	1	1	8	8	1	3	2	1	4	1	2
1	6	4	5	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	4	6
0	3	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	2
0	6	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	6	5
0	3	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	4	10	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	6	2
0	1	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	1	10	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
0	6	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	5

0	6	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	4	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	3	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	1	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	1	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	4
0	1	3	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	2
0	3	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	2
0	1	4	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	5	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	6	6
0	3	3	10	2	4	2	2	2	6	2	1	3	3	8	8	1	3	2	5	4	5	1
0	3	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	6	6
0	6	1	5	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	6	6
0	1	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	4	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
0	3	3	9	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
0	3	1	4	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	3	6	6
0	3	3	9	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	4	5	1
0	3	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	2
0	6	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	4	5
0	3	1	10	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
1	6	3	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	2

0	6	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	3	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	4
0	1	3	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	4	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
1	6	3	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	5	2
0	3	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	6	6
0	6	1	9	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
0	5	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	4	5	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	4	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	3	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	4
0	1	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	3	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	5
0	1	3	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	1	5	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	6	6
0	3	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	5
0	6	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	6	5
0	6	3	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	3	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	5
1	3	3	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	4	2
0	1	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4

0	3	1	4	1	6	2	2	1	8	2	3	3	1	8	8	1	3	2	1	3	4	2
0	3	1	5	1	6	2	2	1	6	2	3	3	3	8	8	1	3	2	1	3	1	2
0	6	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	4	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	1	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
0	6	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	5
0	3	3	9	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	6	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2
0	3	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	4
0	1	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	3	10	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1
0	1	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	3	4	1	6	2	2	1	5	2	3	3	1	8	8	1	3	2	1	3	4	2
1	6	3	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	4	2
0	6	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	4	5	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	2
0	6	1	5	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	4	10	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	6	2
1	6	3	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	4	2
0	6	3	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4

0	3	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	2
0	6	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	4
0	5	1	4	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	3	5	6
0	3	1	5	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	6	6
0	5	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	5	6
0	6	3	4	1	6	2	2	1	4	2	3	3	1	8	8	1	3	2	1	3	4	2
1	6	4	9	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	4	2
0	1	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
1	6	3	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	4	6
1	6	3	4	1	6	2	2	1	6	2	3	3	1	8	8	1	3	2	1	4	1	2
1	3	1	9	1	6	2	2	1	5	2	3	1	1	8	8	1	3	2	1	4	1	2
1	6	3	4	1	6	2	2	1	6	2	3	3	1	8	8	1	3	2	1	4	4	2
1	3	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	6	2
1	3	3	9	1	6	2	2	1	8	2	3	3	1	8	8	1	3	2	1	3	1	2
1	6	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	2
1	1	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	2
1	5	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	6	6
1	3	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	2
1	6	1	9	2	4	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
1	3	1	4	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	3	6	6
1	6	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	4	4
1	1	4	9	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
1	1	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
1	1	4	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
1	6	1	5	2	6	2	1	1	6	2	1	3	3	8	4	1	3	2	5	3	5	1

1	6	4	5	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	4	2
1	6	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	2
1	1	4	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
1	3	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	2
1	3	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	5
1	3	4	10	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	4	5
1	6	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
1	1	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
1	3	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	2
1	6	4	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	4	2
1	6	3	10	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
1	3	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	6	6
1	6	4	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	4	2
1	6	3	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	4	2
1	3	4	5	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	4	2
1	6	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	2
1	3	4	10	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	6	5
1	6	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
1	6	3	10	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2
1	6	4	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	4
1	6	1	10	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1
1	1	3	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	2
1	6	1	5	2	6	2	1	1	8	2	1	3	3	7	8	1	3	2	5	4	5	1
1	6	4	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	4
1	6	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2

1	1	4	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2
1	3	3	4	1	6	2	2	1	5	2	3	1	3	8	8	1	3	2	1	4	1	2
1	6	4	5	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	4	2
1	6	4	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	5	2
0	3	3	5	1	6	2	2	1	6	2	3	1	3	8	8	1	3	2	1	4	1	2
1	6	3	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	4	2
0	6	4	10	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	4	5
0	1	4	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
0	3	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	5
0	1	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	4	5
0	3	3	5	1	6	2	2	1	8	2	3	1	1	8	8	1	3	2	1	4	1	2
0	6	3	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
0	1	4	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
0	1	4	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	2
0	1	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	4	10	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	3	4	5
0	6	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	4	2
0	5	1	5	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	3	5	6
0	3	4	5	2	1	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	4	2
0	6	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	3	5	1	6	2	2	1	8	2	3	3	3	8	8	1	3	2	1	4	1	2
0	3	4	10	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	2
0	6	1	5	1	6	2	2	1	6	2	3	1	3	8	8	1	3	2	1	4	4	2

0	6	4	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	1	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	1	9	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	4	5	1
0	6	1	4	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	3	5	6
0	6	1	5	1	6	2	1	2	8	1	3	3	3	8	8	1	3	2	5	3	5	6
0	6	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	4	2
0	1	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	4	10	2	4	2	1	1	6	1	4	3	4	8	8	1	3	2	5	4	4	5
0	1	4	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
0	1	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	4
0	3	3	10	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
0	1	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	3	9	2	1	2	2	2	6	2	1	3	3	8	8	1	3	2	5	7	5	1
0	1	4	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
0	1	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	4	5	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	2
0	6	1	5	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	4	5	6
0	3	1	4	1	6	2	2	1	5	2	3	3	1	8	8	1	3	2	1	3	1	2
0	3	1	5	1	6	2	1	2	3	1	3	3	3	8	8	1	3	2	5	4	6	6
0	1	4	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	4
0	1	3	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	3	10	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
1	6	4	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	4	6
0	1	4	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	4

0	6	3	9	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	3	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	2
1	6	4	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	4	2
0	3	4	5	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
1	6	4	5	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	5	2
1	6	4	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	4	6
0	6	3	9	1	6	2	2	1	4	2	3	1	3	8	8	1	3	2	1	3	4	2
1	6	3	9	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	2
0	6	4	5	2	4	2	1	1	11	1	4	3	4	8	8	1	3	2	5	4	4	5
1	6	3	5	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	4	5	6
0	6	4	10	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	2
0	1	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	3	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	4	5	1
0	1	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	3	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
0	1	4	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	3	5	6
0	1	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	3	9	2	4	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	2
0	3	3	10	2	4	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
1	6	3	5	2	7	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	5	2
0	6	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
1	6	4	9	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	4	6
0	6	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2

0	1	3	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	5	1	4	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	6	6
0	6	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	1	4	1	6	2	2	1	5	2	3	3	1	8	8	1	3	2	1	4	4	2
0	5	1	5	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	5	6
0	1	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	3	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	4	10	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	4	4
0	3	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	6	5
0	6	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	4	5	2	1	2	1	1	8	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	1	5	2	6	2	1	1	6	2	1	3	3	4	8	1	3	2	5	3	5	1
0	6	3	9	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	4	10	2	4	2	1	1	8	1	4	3	4	8	8	1	3	2	5	4	4	5
0	6	4	9	2	1	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	3	4
0	6	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	2
0	1	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	2
0	3	3	4	1	6	2	2	1	6	2	3	3	1	8	8	1	3	2	1	4	4	2
0	6	4	9	2	1	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	3	2
0	6	3	9	2	1	2	2	2	8	2	1	3	3	8	8	1	3	2	5	7	5	1
0	6	3	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	2
0	6	4	10	2	1	2	1	1	11	1	4	3	4	8	8	1	3	2	5	3	6	5
0	6	1	5	2	6	2	1	1	10	2	1	3	3	4	7	1	3	2	5	4	6	1
0	1	3	10	2	1	2	1	1	3	1	2	3	3	8	8	1	3	2	5	4	4	4
0	6	3	10	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	4	4	4

0	6	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	3	4
0	6	4	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
0	6	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	6	1	9	1	6	2	2	1	6	2	3	1	1	8	8	1	3	2	1	4	1	2
0	6	3	9	1	6	2	2	1	4	2	3	1	3	8	8	1	3	2	1	4	4	2
0	6	3	10	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	4	3	4
0	3	1	9	1	6	2	2	1	8	2	3	1	3	8	8	1	3	2	1	3	4	2
1	3	4	5	2	7	2	1	2	8	1	3	3	3	8	8	1	3	2	5	4	5	6
1	6	3	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	5	2
0	3	1	4	1	6	2	2	1	6	2	3	3	3	8	8	1	3	2	1	3	1	2
0	6	4	4	2	6	2	1	1	6	2	1	3	3	4	8	1	3	2	5	4	6	1
0	3	1	5	1	6	2	2	1	5	2	3	1	1	8	8	1	3	2	1	4	1	2
0	6	3	9	1	6	2	2	1	4	2	3	3	1	8	8	1	3	2	1	3	4	2
0	6	3	5	1	6	2	2	1	8	2	3	1	1	8	8	1	3	2	1	4	1	2
0	6	3	10	2	4	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1
0	3	1	9	1	6	2	2	1	5	2	3	1	3	8	8	1	3	2	1	4	4	2
0	6	3	5	1	6	2	2	1	4	2	3	1	3	8	8	1	3	2	1	4	1	2
0	6	3	4	1	6	2	2	1	5	2	3	1	3	8	8	1	3	2	1	3	1	2
0	6	1	4	1	6	2	1	2	5	1	3	3	3	8	8	1	3	2	5	4	6	6
0	6	1	5	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	6	6
0	3	3	9	1	6	2	2	1	8	2	3	3	1	8	8	1	3	2	1	3	4	2
0	6	4	3	2	6	2	1	1	10	2	1	3	3	4	7	1	3	2	5	4	6	1
0	1	4	10	2	1	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	3	2
0	6	1	9	1	6	2	2	1	6	2	3	1	1	8	8	1	3	2	1	3	4	2
1	6	3	9	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	4	2

0	3	1	9	1	6	2	2	1	6	2	3	1	1	8	8	1	3	2	1	3	4	2
0	1	4	9	2	4	2	1	1	11	1	2	3	3	8	8	1	3	2	5	3	4	2
1	3	4	5	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	4	5	6
0	1	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	4	2
0	1	3	9	2	4	2	1	1	5	1	2	3	3	8	8	1	3	2	5	3	4	4
1	6	3	9	2	7	2	1	2	11	1	3	3	3	8	8	1	3	2	5	3	5	6
0	6	1	5	1	6	2	2	1	6	2	3	1	1	8	8	1	3	2	1	3	1	2
0	3	1	9	1	6	2	2	1	4	2	3	1	3	8	8	1	3	2	1	3	1	2
0	6	1	5	2	6	2	1	1	6	2	1	3	3	4	7	1	3	2	5	3	5	1
0	6	1	5	2	6	2	1	1	6	2	1	3	3	7	4	1	3	2	5	4	6	1
0	3	3	4	1	6	2	2	1	6	2	3	3	3	8	8	1	3	2	1	3	4	2
0	6	1	5	2	6	2	1	1	6	2	1	3	3	8	7	1	3	2	5	3	6	1
0	6	1	9	1	6	2	2	1	5	2	3	3	1	8	8	1	3	2	1	3	4	2
0	6	1	5	2	6	2	1	1	6	2	1	3	3	7	8	1	3	2	5	3	5	1
0	6	1	4	1	6	2	2	1	6	2	3	3	1	8	8	1	3	2	1	3	4	2
0	3	1	4	1	6	2	2	1	8	2	3	1	1	8	8	1	3	2	1	3	1	2
0	6	1	5	1	6	2	2	1	5	2	3	3	3	8	8	1	3	2	1	4	4	2
1	6	3	9	2	7	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	4	6
0	3	1	9	1	6	2	2	1	6	2	3	1	3	8	8	1	3	2	1	3	1	2
0	6	3	9	1	6	2	2	1	6	2	3	3	1	8	8	1	3	2	1	4	1	2
0	6	3	4	1	6	2	2	1	5	2	3	3	1	8	8	1	3	2	1	3	1	2
0	6	3	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	3	3	4
0	3	1	5	1	6	2	1	2	6	1	3	3	3	8	8	1	3	2	5	3	5	6
0	3	3	9	1	6	2	2	1	5	2	3	3	1	8	8	1	3	2	1	3	4	2
0	3	3	9	1	6	2	2	1	6	2	3	1	1	8	8	1	3	2	1	3	1	2

0	6	4	10	2	4	2	1	1	6	1	2	3	3	8	8	1	3	2	5	4	4	4
0	3	3	9	2	1	2	2	2	11	2	1	3	3	8	8	1	3	2	5	7	5	1

BIBLIOGRAPHY

1. Multivariate Data Analysis (Fifth Edition) – Joseph F.Hair, Rolph E.Anderson, Ronald 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 – Andrea 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 – Karthik Ramasubramanian, Abhishek Singh

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

10. Deep Learning with R – François Chollet