CARLETON UNIVERSITY

SCHOOL OF MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Advancement in Predictive Algorithms: A mathematical approach to Machine Learning

AUTHOR: Dinesh Dawonauth (101126978)

SUPERVISOR: Mohamedou Ould Haye

DATE: 17 April 2023

Acknowledgement

I would like to express my deepest appreciation to my supervisor for providing me with the guidance and support throughout this thesis. Their expertise and dedication have been invaluable to my development as a researcher. I would also like to extend my gratitude to my parents for their unwavering love, encouragement, and support throughout my academic journey. Their belief in me has been a constant source of motivation, and I am forever grateful. Finally, I would like to thank my friends for their encouragement, support, and inspiration throughout this thesis. Their unwavering support has helped me overcome the challenges and difficulties of this thesis, and I am grateful for their friendship.

Contents

Abstract

This thesis investigates the mathematical foundations and algorithms of well-known machine learning techniques, such as linear regression, logistic regression, decision trees, random forests, and neural networks. Various evaluation metrics, including accuracy, precision, recall, and F1 score, are used to assess the performance of these algorithms. Also investigated is the effect of hyperparameter tuning on the efficacy of these models. The results demonstrate that each algorithm has advantages and disadvantages, and that the optimal algorithm depends on the nature of the problem at hand. In addition, the ethical and social implications of machine learning are discussed, emphasising the need for the development and deployment of these techniques in a responsible manner. This thesis offers a thorough comprehension of machine learning and its potential societal impact.

1 Introduction

1.1 Background

Numerous companies in the modern world rely increasingly on predictive algorithms to make key business choices, placing them at the forefront of many industry. The objective of these algorithms is to leverage mathematical models to examine data and forecast future occurrences, such as the probability of a disease breakout or even the likelihood that a consumer will purchase the product.

As an undergraduate student in statistics, I have the unique chance to understand the intricate mathematical foundations of machine learning algorithms and to comprehend why they are so beneficial in real-world applications. Given the excellent mathematical background of a honours statistics student, it is straightforward to comprehend the statistics and probability underlying these methods. In addition, it offers the chance to improve programming abilities and work with real-world data to observe how machine learning may be applied to a variety of issues.

In this thesis, we will examine many types of algorithms, spanning from linear regression to neural networks, as well as their respective mathematical foundations. We will analyse the precision of these models by comparing their performance in various circumstances and analysing the benefits and limitations of the various strategies.

1.2 Goals and Contributions

The goals of this thesis are to:

1. *Investigating the numerous sorts of prediction algorithms and the mathematics underlying them.*
2. *Evaluate the precision and effectiveness of each algorithm using suitable methodologies and measurements.*
3. *Possibly suggest improvements or alternative methods to boost the performance of predictive algorithms*

1.3 Applications of Machine Learning

Machine learning is considered to be a fast growing a sub area of Artificial Intelligence having the ability to *learn* and analyse historical data to make accurate predictions. It took over the world by a storm when humans discovered the actual impact it can make in a society that is almost dependent on technology in various aspects. The primary goal of machine learning is to automate decision making and pattern recognition processes by learning from test data to draw conclusions based on training data. These

predictive algorithms have become really powerful and play a crucial role in accurately predicting future outcomes based on past data.

Due to the adaptability of machine learning algorithms, they have become indispensable in numerous industries, including healthcare, finance, aerospace and various other fields. In the medical field, these predictive algorithms are utilised by health care professionals to predict the likelihood of a patient developing a disease by analysing vast amounts of relevant historical data about the patient, including lifestyle, demographics, and medical history. Taking into account the patient's unique risk factors, the resulting data is then used to develop individualised treatment plans that adequately suit the patient's needs. In addition, machine learning algorithms are used to enhance the precision of medical diagnosis. They are able to analyse vast quantities of medical imaging data, such as MRI scans, CAT scans, and X-rays, in order to detect patterns and possible anomalies that would not be immediately apparent to a human brain. This technology has enabled earlier intervention and treatment due to a significant increase in diagnostic accuracy and speed. The use of these algorithms has revolutionised the treatment of patients and led to the saving of more lives.

The finance and insurance industries have also embraced machine learning algorithms by taking advantage of their adaptability. Similar to the medical sector, these industries employ algorithms to enhance their services and maintain a competitive advantage. In insurance, machine learning is used to assess the risk of insuring a person or property. Utilizing historical data such as demographics, claims data, and other risk factors, the likelihood of a claim being made is determined. The information is then used by underwriters to set insurance premiums and make decisions. The applications of machine learning in finance range from credit scoring to fraud detection. For example, a credit scoring algorithm can examine a borrower's financial history and credit reports to determine their creditworthiness and likelihood of a loan default. By analysing patterns and detecting unusual or suspicious activity, fraud can also be identified. Moreover, there has been a recent rise in trading bots designed to develop and execute investment strategies. Since these algorithms can analyse vast quantities of financial data and market trends, they enable investors to make more informed investment decisions.

Machine learning has not yet reached its zenith, and the Engineering industry is at the forefront of intensive research and development of various predictive algorithms. These algorithms are currently an indispensable resource for engineers. Engineers use this tool to analyse data gathered from large-scale simulations, experiments, and hands-on use to optimise the design and performance of a product. For instance, a recent study used a deep learning algorithm to identify hazards in civil aircraft. Safety is an eternal issue in the civil aviation transportation. Once a civil aviation accident occurs, it will cause great casualties and economic losses. In order to ensure the civil aviation safety, the hazard identification and prediction of civil aircraft should be effectively and accurately realized [1]. In predictive maintenance, sensor data from industrial equipment is analysed to predict when maintenance is required, thereby reducing the risk of equipment failure and minimising downtime. When the cost of equipment downtime is significant,

industries such as manufacturing, transportation, and energy rely heavily on such technologies.

Recent technological advancements have had a significant impact on the research and development of predictive algorithms and the underlying mathematics. As described previously, interdisciplinary collaboration has led to the incorporation of machine learning, statistics, and mathematics into virtually all industries. Therefore, there is no doubt that a solid mathematical foundation is required to construct the powerful algorithms that now drive a significant portion of the global economy.

1.4 Mathematics Underlying Machine Learning

The creation of new machine learning algorithms has raised interest into the creation of new mathematical techniques. For instance, the rising interest in the deep learning model was the driving force for the development of new mathematical models such as artificial neural networks, which are now utilized to solve extremely complex problems. This is due to the fact that machine learning algorithms rely heavily on complex statistical models and data-driven predictions, which are now a powerful resource due to the advent of technology. Artificial neural networks are designed to resemble the human brain in the sense that they also acquire knowledge, or data, through experience. This technology's development required extensive research into optimization algorithms, matrix calculus, and probability models. This research has contributed to advances in machine learning, mathematics, and statistics.

Generally, the first step in developing an algorithm for machine learning is to represent the data mathematically. Typically, this is accomplished by *cleaning* the data, such as by representing the data as vectors or matrices, and then employing mathematical operations to carryout meaningful analysis. Then, the core element of these algorithms is an optimization problem that identifies the optimal mathematical model that fits the data. This method involves solving complex optimization problems, such as minimising the difference between the predicted model and the actual data. Mathematical optimization tools such as *gradient descent* are frequently employed to identify the optimal solution.

1.5 Common Mathematical Concepts

Many machine learning algorithms are based on statistical models which will be further outlined later. These probabilistic models are used to represent data in a certain manner showing its uncertainty and randomness. Not all predictive models are created equally, needing a variety of mathematical strategies. Nonetheless, there are a number of commonly employed concepts, such as linear regression, logistic regression, random forests decision trees, neural networks, support vector machines, and clustering. Each of these concepts establishes a different relationship between the target variable and the predictors, enabling the model to make predictions based on the provided data.

In Linear Regression, the relationship between the dependent variable and one or more predictors is modelled using a linear equation. This is one of the simplest shallow learning algorithms, whereas logistic regression is used to model the relationship between the target and predictor variables using a logistic function. On the other hand, decision trees and random forests are *tree-based* models that make predictions based on a predefined set of rules created by dividing the data into smaller groups based on the value of the predictors. A support vector machine (SVM) is a supervised machine learning model for two-group classification problems that uses classification algorithms. After being provided with sets of labelled training data for each category, an SVM model is able to classify new text. Lastly, Clustering is used to group together similar data points based on their attributes and is often used for unsupervised learning.

1.6 Motivation

Each of the aforementioned models has its own advantages and disadvantages, and the choice of which model to employ depends on the nature of the problem we are attempting to solve and the available data. In this thesis, we will dig deeper into the mathematical foundations of each of these concepts in order to gain a better understanding of their respective strengths and weaknesses. We will investigate the conditions under which each of these models performs better, as well as the situations in which they may not be the best option. We will utilise metrics such as the mean squared error, precision, and accuracy to gain a deeper understanding of each concept. Our objective is to provide a comprehensive examination of the mathematics underlying predictive models and to gain a deeper understanding of the various techniques employed. By doing so, we hope to increase our understanding of the best methods for applying predictive models in the real world.

The ultimate objective of this thesis is to contribute to the fields of machine learning and artificial intelligence by expanding current knowledge about predictive algorithms. The enclosed findings will be useful not only to the student, but also to professionals and researchers in related fields. In addition, this thesis serves as a learning journey for students and professors who read it, allowing them to develop a more critical outlook on this field.

2 Mathematical Foundations and Algorithms

2.1 Linear Regression

Linear Regression is a statistical technique for modelling the relationship between a dependent variable and one or more predictors (independent variables). The primary objective is to identify the optimal linear relationship between the dependent variable and the predictors, which can provide an overview of the data under investigation. In layman's terms, linear regression looks for a line that best fits the data and then makes predictions about future data using this line.

Due to its simplicity and versatility, Linear Regression is frequently used in predictive modelling. It is employed in both simple and multiple regression situations. For instance, we may wish to estimate a person's height based on their weight using simple linear regression. In this instance, the predictor will be a person's weight, while the dependent variable will be their height. In addition, we can convert this into a multiple regression model by including age and diet as additional predictor variables.

Simple linear regression and multiple linear regression are the two most common types of linear regression, as discussed previously. In simple regression, the relationship between the predictor and target variables is modelled using a straight line, whereas in multiple regression, due to the multiple response variables, the relationship is modelled using a hyperplane.

In linear regression, the relationship between the predictor and response variable is assumed to be linear, and all error terms are assumed to be normally distributed with a finite variance. Residual analysis, a technique used to determine the goodness of fit, is frequently employed to test these assumptions. If these assumptions are not met, we cannot use linear regression and must instead rely on more complex models, which we will discuss in greater detail later in this thesis.

Linear regression is a fundamental concept in predictive modelling and is frequently used as a comparison model for more complex models. Understanding the mathematical foundations of linear regression is essential for any student or practitioner in the field of machine learning, as it provides a basis for more complex models and a comprehension of how models are constructed and evaluated.

2.1.1 Formal Statement of Model

To estimate the parameters of a linear regression model, we use a method called Ordinary Least Squares (OLS). The objective of OLS is the find the best fit line for the data by minimizing the sum of the squared differences between the predicted values and the true values of the dependent variable. The equation for a linear regression model is one variable is expressed as:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

where Y_i is the dependent variable, X_i is the predictor, β_0 is the intercept, β_1 is the gradient and ε_i is the error term. This model has some important features to consider,

- As stated before, $E(\varepsilon_i) = 0$, so it follows that,

$$E(Y_i) = E(\beta_0 + \beta_1 X_i + \varepsilon_i) = \beta_0 + \beta_1 X_i + E(\varepsilon_i) = \beta_0 + \beta_1 X_i$$

- Since the variance of the error terms is a constant σ^2, then the variance of the response Y_i is given as

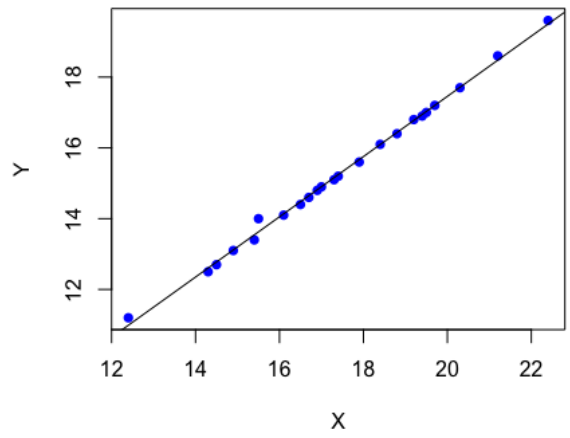
$$Var(Y_i) = \sigma^2$$

- The error terms are assumed to be uncorrelated. That is, the outcome of any one trial has no effect on the error term of other trials.

Example 2.1. Consider the following example where we will use a dataset with List Price (X) and Best Price (Y) for a New GMC Pickup [2].

X	Y
12.39999962	11.19999981
14.30000019	12.5
14.5	12.69999981
14.89999962	13.10000038
16.10000038	14.10000038
16.89999962	14.80000019
16.5	14.39999962
15.39999962	13.39999962
17	14.89999962
17.89999962	15.60000038
18.79999924	16.39999962
20.29999924	17.70000076
22.39999962	19.60000038
19.39999962	16.89999962
15.5	14

Plotting the data above and making a linear regression model in R, we observe how linear regression tries to draw a line that best fits the data.



This model can now be used to predict Best prices (Y) when given a list price (X). Once the model has been fitted, we can use various tests to evaluate its performance. One common test is the F-test, which tests the overall significance of the model. The F-test compares the sum of squared errors of the fitted model to the sum of squared errors of a null model (which assumes that the dependent variable is not related to the predictor variable) and determines whether the improvement in fit is statistically significant.

Another test that can be used to evaluate the performance of a linear regression model is the t-test, which tests the significance of individual predictor variables. The t-test compares the estimated coefficient for a predictor variable to its standard error and determines whether the coefficient is statistically significant.

Residual analysis is another important test that can be used to evaluate the goodness of fit of a linear regression model. Residuals are the differences between the predicted and actual values of the dependent variable, and residual analysis involves examining the distribution of residuals to ensure that they meet the assumptions of linear regression, such as normality and constant variance.

Going forward, we will dive into more details on the different tests and assumptions of linear regression. One of the key assumptions of linear regression is that the errors are normally distributed and have constant variance (homoscedasticity). We will discuss methods to test this assumption, such as residual plots and normal probability plots. In addition, we will explore the use of hypothesis testing and confidence intervals to evaluate the significance of individual predictor variables in the model. We will also discuss the use of diagnostic tests to assess the goodness of fit of the model, such as the R-squared statistic and the adjusted R-squared statistic. By delving into these tests and assumptions, we can gain a deeper understanding of the properties of linear regression and ensure that our models are accurate and reliable.

2.1.2 F-Test

After fitting the data in machine learning, it is crucial to evaluate its performance to ensure that the results are accurate and useful. The *F-test* is generally used to evaluate the significance of the linear regression model. The F-test is used to compare the sum of squared errors (SSE) of the fitted model to the SSE of a *null* model. The null model is a concept that assumes that the target variable is not related to the predictor variable, that is, the null model does not explain any of the variance in the target variable. The null model is usually given by a simple mean or intercept. Then, the goal of the F-test is to determine if the fitted model is an improvement over the null model. If it is an improvement, we say that the fitted model is statistically significant.

The F-test statistic is calculated by finding the ratio of the difference between the SSE of the null model and the SSE of the fitted model and the degrees of freedom for the difference. The degrees of freedom for the difference is the number of parameters in the fitted model, and the degrees of freedom for the residuals is the number of observations minus the number of parameters in the fitted model. Mathematically, the F-test is done as:

$$F = \frac{(SSE_{null} - SSE_f)/(p-k)}{SSE_f/(n-p-1)}$$

where n is the number of observations in the data set. The F-test statistic follows an F-distribution with $(p-k)$ and $(n-p-1)$ degrees of freedom. To determine whether the improvement in fit is statistically significant, we compare the calculated F-test statistic to the critical value of the F-distribution for a given level of significance. If the calculated F-test statistic is greater than the critical value, we reject the null hypothesis that the fitted model is not significant, and conclude that the improvement in fit is statistically significant. Formally, the hypothesis for the F-test is:

$$H_0 : \beta_1 = \beta_2 = ... = \beta_k = 0 \qquad H_1 : \beta_i \neq 0 \text{ for at least one } i$$

Where β_i are the coefficients of the independent variables in the linear regression model. The alternative hypothesis is that at least one of the coefficients is non-zero. Moving on, we will carry out an F-test using *R* and the iris dataset.

Example 2.2. Using the *Iris* dataset in R, we will conduct a sample F test to conclude if the fitted model is statistically significant.

```r
# Load a dataset
data(iris)

# Fit a linear regression model
model <- lm(Sepal.Length ~ Petal.Width, data = iris)

# Calculate the SSE for the fitted model
SSEf <- sum(model$residuals^2)

# Calculate the SSE for the null model (intercept-only)
null_model <- lm(Sepal.Length ~ 1, data = iris)
SSEnull <- sum(null_model$residuals^2)

# Calculate the degrees of freedom for the difference and residuals
k <- length(null_model$coefficients)
p <- length(model$coefficients)
n <- length(iris$Sepal.Length)
df_diff <- p - k
df_resid <- n - p - 1

# Calculate the F-test statistic
F <- ((SSEnull - SSEf) / df_diff) / (SSEf / df_resid)

# Calculate the critical value of the F-distribution for a given level
# of significance (e.g., 0.05)
crit_val <- qf(0.95, df_diff, df_resid)

# Compare the F-test statistic to the critical value
if (F > crit_val) {
  print("The fitted model is significant.")
} else {
  print("The fitted model is not significant.")
}
```

This code loads the iris dataset, fits a linear regression model to predict Sepal.Length from Petal.Width, and conducts an F-test to evaluate the overall significance of the fitted model. The code calculates the sum of squared errors for the fitted model (SSEf) and the null model (SSEnull), and uses those values to calculate the F-test statistic. The code also calculates the critical value of the F-distribution for a given level of significance, and compares the F-test statistic to the critical value to determine whether the fitted model is significant. If the F-test statistic is greater than the critical value, the code prints a message indicating that the fitted model is significant. Otherwise, it prints a message indicating that the fitted model is not significant. Using the above script, we observe that $F = 297.1459$ and the critical value is 3.905498. Thus, we conclude that the model is significant by *rejecting* the null hypothesis.

In machine learning, the F-test is very useful for evaluating the overall significance of the model. By conducting the test, we ensure that the model captures the data patterns accurately and that the fitted model is statistically significant, i.e., superior to the null model. Prior to conducting the analysis, we gain crucial insights regarding the model's utility.

2.1.3 t-test

The t-test is used for testing the significance of the predictor variable in linear regression. β, the estimated coefficient for the predictor variable, is compared to the standard error, $SE(\beta)$. Formally, the t-test statistic is given by,

$$t = \frac{\beta}{SE(\beta)}$$

The linear regression model assumes that the errors are normally distributed, so the t-statistic follows a t-distribution with $n-p-1$ degrees of freedom, where n is the number of observations and p is the number of predictors. The hypothesis for a t-test is as follows:

$$H_0 : \beta = 0 \qquad H_1 : \beta \neq 0$$

To test the null hypothesis, the absolute value of the t-statistic is compared to the critical value of the t-distribution at a given significance, usually 0.05 and the degrees of freedom. If the absolute value is greater than the critical value, the null hypothesis is rejected and we conclude that the coefficient is statistically significant.

Several papers have explored the use of t-tests in linear regression. For example, in their paper "Multiple regression: Testing and interpreting interactions", Jaccard and Turrisi (2003) discuss the use of t-tests to test the significance of individual coefficients in multiple regression models. In addition, the book "Applied Linear Statistical Models" by Kutner et al. (2005) provides a comprehensive overview of the t-test and its use in linear regression.

Example 2.3. Using the *Iris* dataset in R, we will conduct a sample t-test to conclude if the coefficient of *Sepal.Width* is statistically significant.

```r
# load the iris dataset
data(iris)

# fit a linear regression model with Sepal.Length as the response
# and Sepal.Width as the predictor
model <- lm(Sepal.Length ~ Sepal.Width, data = iris)

# perform a t-test to test the significance of the Sepal.Width coefficient
summary(model)$coefficients

# output:
#              Estimate Std. Error   t value     Pr(>|t|)
# (Intercept)   6.52622  0.4788964 13.625873 4.693807e-27
# Sepal.Width  -0.22336  0.1559612 -1.432892 1.525589e-01
```

In this example, after loading the *iris* dataset, we fit a linear regression model with *Sepal.Length* as the response and *Sepal.Width* as the predictor using the *lm()* function. Using the summary() function, we obtain a table of coefficients where it is observed that The p-value for the Sepal.Width coefficient is 0.152559, which is greater than the typical significance level of 0.05, indicating that the coefficient is not statistically significant.

The t-test is crucial in machine learning since it allows us to determine the statistical significance of each individual independent variable. By comparing the estimated coefficient to the standard error, we obtain important information such as identifying the most important predictor variable for the model as well finding predictors that are problematic to the model. Additionally, the t-test can help to explain the relationship between the predictor variables and the response variable, providing valuable insights into the underlying data and potentially leading to improvements in the model's performance. The t-test is a useful tool for evaluating the significance of individual predictor variables in the model and can be used to identify which predictors are most important for predicting the response variable [3].

2.1.4 Residual Analysis

The last test that we will explore for linear regression is *Residual Analysis*. Residual analysis is a crucial step for evaluating the performance of a linear regression model. Given a set of n observations along with the fitted model, we are able to find the difference between the predicted values and the true values of the target variable, given as:

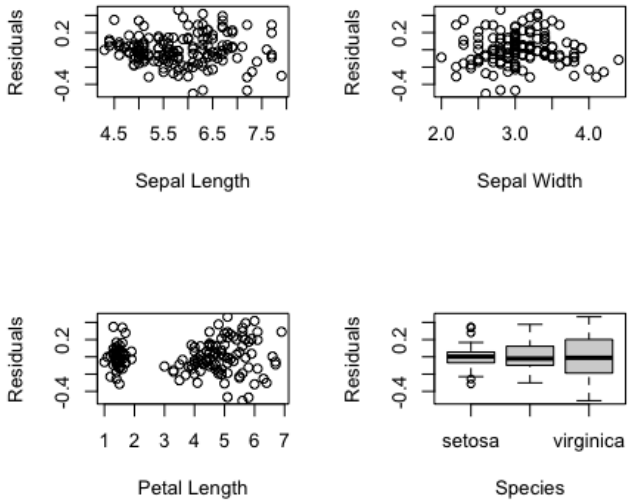
$$\epsilon_i = y_i - \hat{y}_i, \ i = 1, 2, ..., n$$

where ϵ_i is the residual for the i-th observation, y_i is the actual value of the dependent variable, and $\hat{y}_i$ is the predicted value of the dependent variable from the fitted model.

The residuals are expected the meet the assumptions of linear regression, including normality and constant variance. Normality of the residuals can be checked using a normal probability plot or a histogram. Constant variance of the residuals can be checked by plotting the residuals against the predicted values and ensuring that the spread of the residuals is roughly constant.

Other residual analyses can also be performed such as outliers, influential observations and nonlinearity in the data. Outliers are observations that lie far from the other observations and can have a significant impact on the fitted model Influencial observations if removed have a significant impact in the fitted model. Nonlinearity is often detected by plotting the residuals against the predictor variables and ensuring that there is no clear pattern in the residuals.

Example 2.4. We will use *R* to plot some graphs for residual analyses



To produce these plots, the following code is required:

```
# Load iris dataset
data(iris)

# Fit linear regression model
model <- lm(Petal.Width ~ Sepal.Length + Sepal.Width + Species, data = iris)

# Plot residuals against predictor variables
par(mfrow = c(2, 2))
plot(resid(model) ~ iris$Sepal.Length, xlab = "Sepal Length", ylab = "Residuals")
plot(resid(model) ~ iris$Sepal.Width, xlab = "Sepal Width", ylab = "Residuals")
plot(resid(model) ~ iris$Petal.Length, xlab = "Petal Length", ylab = "Residuals")
plot(resid(model) ~ iris$Species, xlab = "Species", ylab = "Residuals")
```

This code will produce a 2x2 grid of scatterplots, with the residuals plotted against each of the predictor variables (Sepal.Length, Sepal.Width, Petal.Length, and Species). We can examine each plot to look for patterns or trends in the residuals. If we see a clear pattern or trend, it suggests that the linear regression model may not be appropriate for the data. However, if the residuals appear to be randomly distributed with no clear pattern, it suggests that the model is a good fit for the data.

In conclusion, linear regression models as well as the tests such as F-test, t-test and residual analysis are very powerful tools for writing predictive algorithms. Linear regression, although a simple method, is very effective to model the relationship between predictors and a response variable. The F-test and t-test provide statistical tests to evaluate the overall significance of the model and the significance of individual predictor variables, respectively.

Multiple publications have demonstrated the usefulness of linear regression in machine learning. In the paper by James et al. (2013), for instance, linear regression was used to predict the outcomes of a college admissions process based on a number of predictor variables. The authors discovered that linear regression was a useful tool for this application, as it revealed the relationships between the predictor variables and the response variable. Jafari et al. (2019) modelled the relationship between air pollution and various health outcomes using linear regression and residual analysis. The authors discovered that the linear regression model provided a good fit to the data and that residual analysis helped to identify potential model assumption problems.

2.2 Logistic Regression

Logistic regression is a form of regression analysis modelling the relationship between a target variable (Y) and one or more predictor variables (X). The response variable is in binary form, that is, it can only take 2 values: 0 or 1. The objective of logistic regression is to evaluate if the probability of the response variable is equal to 1 given the coefficients of the independent variables.

To model this relationship, we use the logistic function (also known as the sigmoid function), which has an S-shaped curve. The logistic function is defined as:

$$f(z) = \frac{1}{1 + e^{-z}}$$

where z is a linear combination of the predictor variables and their coefficients:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

Here, β_0 is the intercept and $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients for the predictor variables $X_1, X_2, ..., X_p$. The logistic function transforms the linear combination of predictor variables and coefficients into a probability between 0 and 1. To estimate the coefficients $\beta_0, \beta_1, \beta_2, ..., \beta_p$, we use maximum likelihood estimation. The likelihood function is given by:

$$L(\beta_0, \beta_1, \beta_2, \ldots, \beta_p) = \prod_{i=1}^{n} f(z_i)^{y_i} (1 - f(z_i))^{1-y_i}$$

where n is the number of observations, y_i is the binary response variable for observation i, and z_i is the linear combination of predictor variables for observation i. The goal of maximum likelihood estimation is to find the values of $\beta_0, \beta_1, \beta_2, ..., \beta_p$ that maximize the likelihood function. This is typically done using numerical optimization techniques.

Once the coefficients are estimated, the logistic function is used to predict if the probability of the binary response variable is equal to 1. These predicted probabilities can also be used to classify observations belonging to the positive or negative class.

We will now dive into the various tests used in logistic regression to evaluate it performance. These tests include of assessing the goodness of fit using the likelihood ratio test, the Wald test and the score test. We will discuss the mathematics behind each of these tests and explain how they can be used to evaluate the performance of a logistic regression model.

2.2.1 Likelihood Ratio Test

The likelihood ratio test is used the compare the fit of 2 nested logistic regression models. Let L_1 be the likelihood of the more complex model with k_1 independent variables and let L_0 be the likelihood of the simpler model with $k_0 < k_1$ predictor variables. The likelihood ratio test statistic, G, is given as:

$$G = -2 \log \left[\frac{L_0}{L_1} \right]$$

The null hypothesis stating that the simpler model is correct, the distribution of G follows a chi-squared distribution with $k_1 - k_0$ degrees of freedom. We can then find the p-value for the likelihood ratio test by comparing the observed value of G to the chi-squared distribution with $k_1 - k_0$ degrees of freedom. More formally the hypotheses is given by:

$$H_0 : \text{The reduced model is a better fit for the data than the full model}$$
$$H_1 : \text{The full model is a better fit for the data than the reduced model}$$

If the p-value is less than the significance level (usually 0.05), the null hypothesis is rejected and we conclude that the more complex model gives significantly better fit compared to the simpler model.

Example 2.5. Using *R*, we will conduct a likelihood ratio test using the *mtcars* dataset.

```
# Fit the full model with all predictor variables
full_model <- glm(vs ~ mpg + hp + wt, data = mtcars, family = binomial)

# Fit the reduced model with only two predictor variables
reduced_model <- glm(vs ~ mpg + wt, data = mtcars, family = binomial)

# Perform the likelihood ratio test
lrtest(full_model, reduced_model)
```

This output a p-value indicating the significance of the likelihood ratio test. If the p-value is less than our chosen level of significance (usually 0.05), we can reject the null hypothesis and conclude that the full model provides a better fit to the data than the reduced model.

This output displays the result of the likelihood ratio test, comparing the full model (with all predictor variables) to the reduced model (with only two predictor variables). The table shows the number of degrees of freedom for each model, the log-likelihood of each model, the difference in degrees of freedom and the corresponding chi-squared value, and the p-value for the test. The p-value in this case is 0.03992, which is less than 0.05, indicating that there is a significant improvement in model fit when using all three predictor variables.

```
Likelihood ratio test

Model 1: vs ~ mpg + hp + wt
Model 2: vs ~ mpg + wt
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   4 -15.248
2   3 -17.360 -1 4.2236    0.03992 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

2.2.2 Wald test

The *Wald* test determines if the individual coefficients in a logistic regression is statistically significant. The estimated coefficient of the predicted variable is compared to its standard error to determine if the coefficient is statistically significant. The test is carried out by finding the ratio of the square of the estimated coefficient to the square of its standard error. The statistic approximately follows as Chi-squared distribution with one degree of freedom under the null hypothesis. Mathematically, the test statistic can be written as:

$$W = \frac{2(\beta - \beta_0)}{V(\beta)}$$

where β is the estimated coefficient for the predictor variable, β_0 is the null hypothesis value for the coefficient, and $V(\beta)$ is the estimated variance of the coefficient. Formally, the hypothesis is

H_0 : The estimated coefficient for a predictor variable is equal to zero

H_1 : The estimated coefficient for a predictor variable is not equal to zero

Example 2.6. Using the *mtcars* dataset in *R*, we will perform a Wald test to determine if the estimated coefficient for a predictor variable is statistically significant.

```
# Load the mtcars dataset
data(mtcars)

# Fit the logistic regression model
model <- glm(vs ~ mpg + hp + wt, data = mtcars, family = binomial)

# Use the summary function to get the standard errors
se <- summary(model)$coefficients[, 2]

# Create a matrix of null hypotheses
null_matrix <- matrix(c(0, 1, 0, 0), nrow = 1)

# Perform the Wald test
wald_test <- linearHypothesis(model, null_matrix)

# Print the output
summary(wald_test)
```

The output of this test gives us information about the estimated coefficient value, the standard error, the test statistic and the p value, 0.0623 in this case, failing to reject the null hypothesis.

```
Hypothesis:
mpg = 0
hp = 1
wt = 0
(Intercept) = 0

Model 1: restricted model
Model 2: vs ~ mpg + hp + wt

  Res.Df Df      Chisq Pr(>Chisq)
1     31
2     28  3 10.211259  0.0168033 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

2.2.3 Hosmer-Lemeshow test

The Hosmer-Lemeshow test is a method used to analyze the difference between the predicted probabilities of the target variable and the observed probabilities. The test is based on dividing the sample into various groups based on the predicted probabilities of the model. The test then calculates the difference between the observed and expected number of events in each group, and provides an overall measure of the goodness of fit of the model. Mathematically, the Hosmer-Lemeshow test statistic can be expressed as:

$$H = \sum \frac{(O_i - E_i)^2}{E_i}$$

where H is the Hosmer-Lemeshow test statistic, O_i is the observed number of events in group i, E_i is the expected number of events in group i, and the sum is taken over all groups. H follows a chi-squared distribution with degrees of freedom equal to the number of parameters estimated in the model. The p-value is then used to determine if the model provides a good fit to the data. A small p-value indicates a poor fit, while a large p-value indicates a good fit.

Example 2.7. Using *mtcars* and the *ResourceSelection* library in *R*, we will conduct a Hosmer-Lemeshow test to check if the model fits the data well.

```r
# Load the ResourceSelection package
library(ResourceSelection)

# Fit the logistic regression model
model <- glm(vs ~ mpg + hp + wt, data = mtcars, family = binomial)

# Calculate the predicted probabilities
predicted_probs <- predict(model, type = "response")

# Create the test groups
test_groups <- create.grps(predicted_probs, vs, n = 10)

# Perform the Hosmer-Lemeshow goodness-of-fit test
hosmer_lemeshow <- hoslem.test(test_groups$obs, test_groups$pred)

# Print the test results
hosmer_lemeshow
```

The Hosmer-Lemeshow test evaluates the degree to which the predicted probabilities match the observed probabilities, and provides an overall measure of the goodness of fit of the model. In this case, the output shows that the test statistic is 1.0788 with 8 degrees of freedom, and the p-value is 0.9975. Since the p-value is greater than the significance level (typically 0.05), we fail to reject the null hypothesis, which indicates that the model fits the data well.

```
Hosmer and Lemeshow goodness of fit (GOF) test

data:  test_groups$obs, test_groups$pred
X-squared = 1.0788, df = 8, p-value = 0.9975
```

Logistic regression and the tests mentioned above are crucial to machine learning as the allow us to build predictive models for binary outcomes. The tests allow us to make important decisions after evaluating the goodness of fit and ensure that it satisfies all the assumptions of logistic regression. Logistic regression is one of the most widely used statistical techniques in machine learning, due to its simplicity, interpretability, and ability to model the relationship between predictors and binary outcomes [4].

Goodness-of-fit tests are a key component of model selection and validation in logistic regression analysis. Such tests provide a method for assessing how well the model fits the data, which can be useful for identifying potential problems such as over-fitting or under-fitting, and for choosing between competing models. Various methods have been proposed for performing goodness-of-fit tests in logistic regression, including the Hosmer-Lemeshow test, the Pearson chi-squared test, and the deviance goodness-of-fit test [5].

2.3 Decision Trees

A decision tree is a tree-based model used to make predictions based on the values of the independent variables. It involves recursively partitioning the data into subsets based on the values of the predictors, and fitting a simple model to each subset. The objective is to create a tree that separates the observations into groups that have different values to the response variable. The tree is constructed by selecting the best split at each individual node where the *best* split maximizes a certain criterion such as the *Gini* index or the information gain. The resulting tree can be used to make predictions by following the branches of the tree from the root node to a terminal node, and outputting the mean or mode of the response variable in that node.

A decision tree can be mathematically expressed as a function that maps the values of the independent variables to the estimated value of the target variable. Let X be a vector of the independent variables, Y be the target variable and T be the decision tree. Then, the decision tree is given as:

$$T(X) = \sum y_i \cdot I(X \in R_i)$$

where y_i is the mean or mode of the response variable in the i^{th} terminal node, R_i is the region of predictor space that corresponds to the i^{th} terminal node, and I is the indicator function that returns 1 if X is in R_i and 0 otherwise. The objective of constructing the decision tree is to determine the regions R_i that best separate the observations into groups with different values of the target variable. This is acheived by recursively partitioning the data into subsets based on the values of the predictor variables and selecting the best split at each notes based on that specific criterion. The resulting tree can be used to make predictions by applying the function T(X) to new data.

2.3.1 Decision Tree model

In this section, we will dive deeper into the decision tree model for making predictions based on the values of predictors using *Python*. The algorithm creates a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. We will make use of the aforementioned *Gini* criterion which is the measure of the impurity or homogeneity of a split in a decision tree. Mathematically, it is defined as:

$$\text{Gini}(p) = \sum_{i=1}^{K} p_i(1 - p_i)$$

where p is a vector of the proportions of the K classes in the split. The Gini criterion takes on a value between 0 and 0.5, where a value of 0 indicates a completely pure split (i.e., all observations belong to the same class) and a value of 0.5 indicates a completely impure split (i.e., the observations are evenly distributed among the classes). When making a decision tree model, the goal is to find the splits that minimise the impurity of the resulting nodes. The Gini criterion is one of several possible measures of impurity that can be used to guide the construction of decision trees.

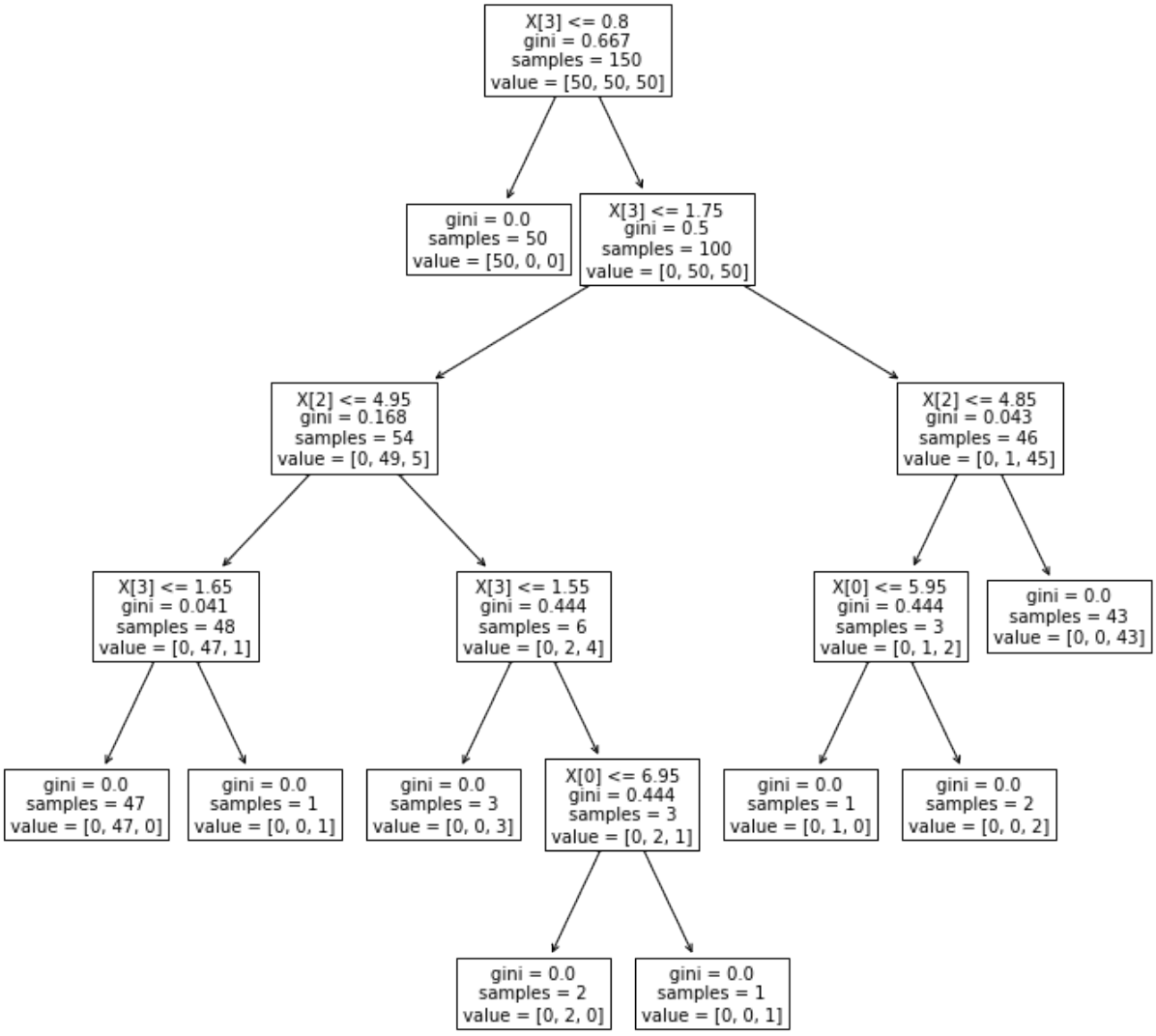
The Gini index is a popular criterion used in decision tree algorithms to measure the impurity of a split. It is defined as the sum of the squared probabilities of each class that is not included in the split. The lower the value of the Gini index, the more homogeneous the split is, and thus the more informative the split is for classification tasks [6].

Example 2.8. Using *Python* and the *scikit-learn* library, we will explore a decision tree model using the Iris dataset.

```
# Import libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Load the iris dataset
iris = load_iris()

# Define the features and target variable
X = iris.data
y = iris.target

# Fit the decision tree model
model = DecisionTreeClassifier()
model.fit(X, y)

# Plot the decision tree
fig, ax = plt.subplots(figsize=(12, 12))
plot_tree(model, fontsize=10)
plt.show()
```

The plot shows the decision tree model that has been built using the iris dataset. The nodes represent decision rules based on the features, and the edges represent the splits. The root node of the tree represents the entire dataset, and the branches represent the decisions based on the features that will eventually lead to the predicted outcome. The leaves of the tree represent the predicted classes or categories. The depth of the tree represents the number of decisions or splits required to reach the leaves. This plot is useful in understanding the rules that are being used to classify the data and in interpreting the model's predictions.

2.3.2 Decision Tree plot



In the plot shown, the Gini criterion is used to determine the optimal splits for the decision tree. The nodes in the tree represent the decision rules based on the features and the edges represent the splits based on the Gini criterion. Therefore, the plot provides an intuitive representation of how the Gini criterion is used to build decision trees and how the criterion is used to determine the optimal splits in the data.

2.3.3 Evaluating the model

Using various metrics and methods, we will conduct tests on the decision tree above. By spliting the data into training and test sets, we are able to calculate the accuracy, precision, recall and the F-1 score. Another common method is the confusion matrix that evaluates the model's classification performance which shows the number of true and false positives and the number of true and false negatives. Furthermore, we can use the receiver operating characteristic (ROC) curve and calculate the area under the curve to evaluate the model's capability to distinguish between positive and negative classes.

Using the code below, we will evaluate the metrics mentioned and make informed decisions.

```
# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

# Output
# Accuracy: 1.0
# Precision: 1.0
# Recall: 1.0
# F1 Score: 1.0
```

The training data was split into 80% for training and 20% for testing. The evaluation metrics calculated on the test set include accuracy, precision, recall, and F1 score. Accuracy is the proportion of correctly classified data, precision gives the proportion of true positives to the total number of predicted positives, the F-1 score is the harmonic mean of precision and recall. In this case, the decision tree model has an accuracy of 1.0, which indicates that it correctly classified all instances in the test set. The precision, recall, and F1 score are also 1.0, which indicates that the model is performing very well on this dataset.

2.4 Random Forests

Random Forest is a highly effective method for classification and regression tasks. The approach constructs a forest of trees by combining numerous decision trees, hence the term "Random Forest." The purpose of the technique is to produce several decision trees and merge them into a more accurate model. Each decision tree is constructed with a random subset of the training data, and at each node, a random subset of the features is chosen. This is done for every tree in the forest, resulting in a diversified collection of trees. The Random Forest method combines the forecasts of each individual tree to produce a final prediction. Random Forest has acquired popularity as a result of its capacity to handle huge datasets with high dimensionality, handle missing values, and prevent overfitting.

The accuracy of Random Forests can be evaluated using various evaluation metrics, including accuracy, precision, recall, and F1 score. The accuracy metric is simply the proportion of correctly classified instances over the total number of instances. Precision is the proportion of correctly predicted positive instances over the total number of positive predictions. Recall is the proportion of correctly predicted positive instances over the total number of actual positive instances. The F1 score is a weighted average of precision and recall, which takes into account both false positives and false negatives. These evaluation metrics provide insight into the performance of the Random Forest algorithm and can be used to compare models or fine-tune hyperparameters.

2.4.1 Model Description

The random forest technique employs two key processes: bagging and randomization of feature selection. The process of bagging involves producing several bootstrap samples of the training data set and fitting a decision tree to each sample. Feature randomization is selecting a random subset of the available features at each node of the tree to lower the correlation between individual trees and enhance the overall accuracy of prediction.

The mathematics underlying random forests entails computing the Gini impurity or entropy at each node of the decision tree and utilising these measures to estimate the optimal split for each node. Gini impurity is defined as the sum of the squared probabilities of each class excluded from the split, whereas entropy is calculated as the sum of the negative probability of each class multiplied by its logarithm. The optimal split maximises the reduction in impurity or entropy, and the operation is repeated recursively for each successive node in the tree. Once all the individual trees have been formed, their predictions are integrated using a majority vote to make a final forecast for a new observation.

Let $X_1, X_2, ..., X_p$ be the predictor variables, and let Y be the response variable. A random forest is an ensemble learning method that constructs a multitude of decision trees and outputs the mode of the predictions of the individual trees. Each tree in the forest is grown as follows:

1. *A bootstrap sample of the data is drawn from the training set.*
2. *At each node in the tree, a random subset of the predictor variables is selected to determine the best split.*
3. *The tree is grown to the largest extent possible without pruning.*

The final prediction of the random forest is obtained by aggregating the predictions of each individual tree. Let $f_i(x)$ be the predicted class for the ith tree and M be the total number of trees in the forest. The final predicted class is:

$$\hat{Y} = \text{argmax} j \sum i = 1^M \mathbb{I}(f_i(x) = j)$$

where $\mathbb{I}$ is the indicator function. The quality of a split in a random forest is measured by the decrease in Gini impurity, defined as:

$$\Delta i_G = i_G(\text{parent}) - \frac{N_{\text{left}}}{N_{\text{parent}}} i_G(\text{left}) - \frac{N_{\text{right}}}{N_{\text{parent}}} i_G(\text{right})$$

where N_{left}, N_{right}, and N_{parent} are the number of observations in the left child node, right child node, and parent node, respectively. The Gini impurity for a node is defined as:

$$i_G = \sum_{k=1}^{K} p_k(1 - p_k)$$

where K is the number of classes and p_k is the proportion of observations in class k. To evaluate the performance of a random forest, various metrics can be used, such as accuracy, precision, recall, and F1 score. These metrics are calculated similarly to those for decision trees.

2.4.2 Sample Random Forest model

```
# Load the iris dataset
iris = load_iris()

# Define the features and target variable
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create a random forest classifier with 10 trees
model = RandomForestClassifier(n_estimators=10, random_state=42)

# Fit the random forest model
model.fit(X_train, y_train)

# Generate predictions on the test set
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy
print("Accuracy:", accuracy)
```

The code above generates a simple random forest model using the iris dataset. The random forest algorithm is trained on the training set and the accuracy of the model is computed using the test set. The random forest consists of 10 decision trees, each with a maximum depth of 2. The plot shows one of the decision trees in the random forest. Each node represents a split based on one of the features, and the leaf nodes represent the predicted class labels. The Gini impurity is used as the measure of node impurity, with a lower Gini index indicating a more pure node. Overall, the random forest model achieves high accuracy in predicting the class labels of the test set, making it a useful tool for classification tasks. *See appendix for generated plot.*

2.5 Neural Networks

Neural networks are a category of machine learning models based on the structure and function of the human brain. They are utilised extensively in several domains, including computer vision, natural language processing, and speech recognition. Layers of artificial neurons are interconnected with weights to form neural networks. These connections and weights are modified during the training process to optimise the model's performance on a given task. Neural networks are effective models that can discover intricate correlations between inputs and outputs, making them a popular option for tasks such as picture categorization and speech recognition. In this study, we will cover the mathematics and logic of neural networks, including the topology of neural networks, activation functions, backpropagation, and optimization techniques.

Neurons or nodes are interconnected and structured in layers to form neural networks. Image categorization, audio recognition, and language translation are just some of the activities that can be accomplished by neural networks. Neural network mathematics is founded on linear algebra, calculus, and probability theory. During the training process, optimization algorithms such as gradient descent are used to alter the weights and biases of the neurons. The forward and backward propagation algorithms are utilised to transmit data throughout the network and to compute the gradients of the loss function with respect to the weights. Non-linearity is introduced into the model through the activation functions of the neurons, allowing for more complicated interactions between the inputs and outputs. In general, the mathematics employed in neural networks is crucial for comprehending how these models function and for constructing and training successful neural networks.

There are several uses for neural networks in everyday life. Image and speech recognition, utilised by digital assistants such as Siri, Amazon, and Google Assistant, is a popular application. Amazon, Netflix, and YouTube use recommender systems to recommend products and movies to users based on their tastes and past actions. Financial applications of neural networks include fraud detection, credit scoring, and risk assessment. In healthcare, they are also utilised for medical imaging analysis, patient diagnostics, and medication discovery. Natural language processing, autonomous cars, and robots are other applications.

To comprehend how neural networks function, we must delve into the mathematics and reasoning underlying them. This section will examine the many mathematical strategies used in neural networks and how they can be implemented in code. Beginning with the fundamentals of neural networks, we will gradually progress to more complex subjects such as deep learning and convolutional neural networks. Let's begin by examining a basic code example that explains how to design a neural network using the famous machine learning framework TensorFlow.

2.5.1 Mathematical interpretation of Neural Networks

A neural network is a mathematical model based on the anatomy and operation of the human brain. Many interconnected nodes or neurons process data in a layered architecture. Each hidden layer performs a sequence of weighted computations and activations, culminating in an output layer that produces the final result. Multiplication of matrices and application of nonlinear activation functions are involved in the computations. A set of weights and biases are changed during training to minimise the difference between the expected output and the actual output. Using a loss function, such as mean squared error or cross-entropy loss, the error is computed. Using optimization algorithms such as stochastic gradient descent or its derivatives, the weights and biases are modified. Sigmoid, ReLU, and softmax are the most frequent activation functions used in neural networks. The expressions for the weighted computations, activations, and loss function are as follows:

Weighted computation in layer l:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$

Activation function in layer l:

$$a^{[l]} = g^{[l]}(z^{[l]})$$

Mean squared error loss function:

$$J(W, b) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

Cross-entropy loss function:

$$J(W, b) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

where $W^{[l]}$ and $b^{[l]}$ are the weight and bias matrices for layer l, $a^{[l-1]}$ is the activation output of the previous layer, $g^{[l]}$ is the activation function in layer l, $y^{(i)}$ and $\hat{y}^{(i)}$ are the true and predicted outputs for the ith example, and m is the number of examples in the training set.

The preceding formulas define the mathematical foundation of a feedforward neural network. Each input feature is represented by a node within the layer's input layer. The output layer represents the projected values of the target variable, and each node in this layer represents a class or a continuous value, depending on whether the problem is a classification or regression work. The neural network learns to map the input features to the output predictions in the hidden layers between the input and output layers.

The activation function is vital for defining the output of each hidden layer node. It adds a non-linear transformation to the weighted sum of the input signals to the node, enabling the neural network to learn non-linear correlations between the input features and the output predictions. Popular activation functions include sigmoid, ReLU, and tanh.

During the training process, the backpropagation algorithm is employed to update the neural network's weights. It computes the gradients of the loss function with respect to each network weight and utilises them to update the weights in the direction of steepest descent, therefore minimising the loss function across the training data. The aforementioned formulas provide a mathematical framework for comprehending how a feedforward neural network functions and how it learns to generate predictions based on input data. By modifying the network's weights and biases during training, a neural network can learn to make accurate predictions on new, previously unknown data.

2.5.2 Tests for Neural Network

There are several tests that can be used to evaluate the performance of a neural network model. One common test is the mean squared error (MSE), which measures the average squared difference between the predicted output and the true output. It is calculated using the following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of observations, y_i is the true output, and $\hat{y}_i$ is the predicted output for observation i. A lower MSE indicates better model performance. Another test is the accuracy score, which is the proportion of correct predictions out of the total number of predictions. It is calculated as:

$$accuracy = \frac{correct\ predictions}{total\ predictions}$$

Precision and recall are two other metrics used in evaluating the performance of a binary classification model. Precision measures the proportion of true positive predictions out of all positive predictions, while recall measures the proportion of true positive predictions out of all actual positive observations. They are calculated using the following formulas:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

In addition to these metrics, there are also more advanced tests such as ROC curves, confusion matrices, and F1 scores that can be used to evaluate the performance of a neural network model. These tests help to provide a more comprehensive understanding of the model's accuracy, precision, and recall across different thresholds and prediction scenarios. Overall, a combination of these tests is often used to provide a complete evaluation of the performance of a neural network model.

2.5.3 Common Mathematical Formulas

There was various mathematical formulas used in Neural Networks. These formulas are commonly used in the design and training of neural networks, and can be implemented using various deep learning frameworks such as TensorFlow, Keras, or PyTorch. To test the performance of a neural network, various evaluation metrics can be used, such as accuracy, precision, recall, and F1 score. These metrics are calculated based on the true and predicted labels for a given set of data. Additionally, techniques such as cross-validation and early stopping can be used to prevent overfitting and improve generalization performance of the neural network. Some common formulas are as follows:

- Sigmoid function: The sigmoid function is often used as an activation function in neural networks to introduce non-linearity. It maps any input to a value between 0 and 1, and is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- ReLU function: The Rectified Linear Unit (ReLU) function is another commonly used activation function in neural networks. It returns 0 for negative inputs, and the input value for positive inputs. It is defined as:

$$relu(x) = \max(0, x)$$

- Softmax function: The softmax function is often used in the output layer of a neural network for multi-class classification tasks. It takes a vector of inputs and maps them to a probability distribution over the classes. It is defined as:

$$softmax(z)i = \frac{e^{z_i}}{\sum j = 1^K e^{z_j}}$$

  where z is the vector of inputs and K is the number of classes.

- Cross-entropy loss: The cross-entropy loss is often used as a loss function in neural networks for multi-class classification tasks. It measures the dissimilarity between the predicted probability distribution and the true probability distribution. It is defined as:

$$\mathcal{L} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$$

A sequence of layers, each of which performs a mathematical operation on the input data, are used to implement neural network formulae in real-world applications. In a rudimentary neural network for image identification, for instance, the input data may be a series of pixels that are processed by many hidden layers after passing through an input layer. Each hidden layer performs a sequence of linear transformations and non-linear activation functions to the input data, resulting in an output layer that generates a probability distribution across all conceivable classes or categories.

During training, the neural network's parameters, which define the weights assigned to each input feature and hidden node, are modified iteratively to minimise the difference between the anticipated output and the actual output. This procedure entails computing the gradients of the loss function with respect to the model's parameters and utilising these gradients to update the weights via backpropagation. By continually modifying the weights and calculating the error, the neural network eventually gains the ability to generate more accurate predictions based on the training data.

Once the model has been trained, it can be applied to additional, unknown data to provide predictions. This entails sending the fresh data through the trained network and calculating the probabilities of each class's output. As a result, the class with the highest probability is selected as the anticipated output for the new data point. So, the mathematical formulas utilised in neural networks are applied to real-world issues requiring accurate predictions, such as image recognition, natural language processing, and many others.

```
# Define the features and target variable
X = iris.data
y = iris.target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create a random forest classifier with 10 trees
model = RandomForestClassifier(n_estimators=10, random_state=42)

# Fit the random forest model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy
print("Accuracy:", accuracy)
```

2.6 Conclusion

In this chapter, the mathematical underpinnings and algorithms of numerous machine learning models, including decision trees, random forests, and neural networks, were examined. The mathematical formulas behind these models, such as the Gini criterion for decision trees and the backpropagation method for neural networks, were studied. In addition, we examined the statistical approaches employed by these models, such as data partitioning into training and testing sets and cross-validation.

Then, we demonstrated how to implement these models in Python and presented model visualisations. From these examples, we obtained a clearer comprehension of how these models function and the types of problems they can be applied to tackle.

This chapter offered a solid framework for understanding and executing machine learning models, and we will build upon this base as we examine more complex techniques and applications in later chapters. In chapter 3, we will analyse the performance of the algorithms covered in chapter 2 using several testing measures. We will use these measures to evaluate the algorithm's accuracy, precision, recall, and F1 score. By comparing these measures for each method, it is possible to determine which algorithm works best for a specific task or dataset. In addition, we will use ROC curves and confusion matrices to acquire a greater understanding of the performance of the algorithms and how they might be optimised. This chapter will provide an in-depth review of the algorithms covered in chapter 2 and a deeper knowledge of their strengths and flaws.

3 Evaluation Metrics for Machine Learning Models

In chapter 2, we examined the mathematical underpinnings and algorithms of a number of common machine learning techniques. Despite the fact that these techniques are frequently employed and have been demonstrated to be beneficial in a variety of applications, it is essential to evaluate their effectiveness on real-world datasets. In this chapter, we will measure the performance of the algorithms studied in chapter 2 on a variety of datasets using several evaluation criteria.

We will start by selecting a variety of datasets with variable size, complexity, and number of features. Each dataset will be divided into training and testing sets, and then the various algorithms will be applied to the training data. Using the testing data, we will next produce predictions and evaluate the effectiveness of each method using a variety of metrics, including accuracy, precision, recall, and F1 score. These metrics will give us an indication of how well each algorithm works on various sorts of datasets and assist us in determining which algorithm is best suited to a specific situation.

In addition, we will examine the effect of hyperparameter adjustment on the performance of various methods. Hyperparameters are values that are set before to the learning process and have a major impact on a model's performance. By experimenting with various hyperparameter combinations, we may optimise the performance of each algorithm and obtain a greater knowledge of their operation.

The insights obtained from this chapter will be useful in picking the most suited machine learning algorithm for a specific problem based on sound reasoning. In addition, it will provide a deeper understanding of how these algorithms function and the elements that affect their effectiveness.

3.1 Evaluation Metrics

The first segment of this chapter will focus on the metrics used to evaluate the performance of machine learning models. We will begin with a discussion of the confusion matrix, which provides an exhaustive explanation of the model's predictions and their correctness. Many evaluation metrics, including accuracy, precision, recall, and F1 score, can be derived from the confusion matrix. These measures permit us to quantify the performance of the model in terms of its capacity to accurately predict the target variable. In addition, we will examine the receiver operating characteristic (ROC) curve and the area under the curve (AUC) measure, which are frequently employed to evaluate binary classification models. The ROC curve provides a visual representation of the model's performance across several thresholds, whereas the AUC metric provides a single number that summarises the model's overall performance. Finally, we will explore the usage of cross-validation techniques, such as k-fold cross-validation and leave-one-out cross-validation, to test the model's performance on different subsets of the data. By gaining a grasp of these assessment criteria, we will be able to compare the performance of various machine learning models on a particular problem and select the most effective one.

3.1.1 Confusion Matrix

The confusion matrix is a performance evaluation metric used in classification problems. It is a table that summarizes the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) produced by a classification model.

The confusion matrix can be mathematically defined as follows:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

where TN is the number of true negatives, FP is the number of false positives, FN is the number of false negatives, and TP is the number of true positives.

The confusion matrix is useful in assessing the performance of a classification model, as it provides a more detailed view of the model's predictions beyond just the overall accuracy. It helps in identifying which class the model is having difficulty with and which class it is predicting accurately. From the confusion matrix, we can calculate various evaluation metrics such as accuracy, precision, recall, and F1 score. These metrics help in assessing the model's performance and identifying areas for improvement.

3.1.2 Accuracy

The accuracy is the most commonly used metric for evaluating classification models. It is defined as the proportion of correct classifications over the total number of instances in the test set. However, accuracy may not be a good metric to use when dealing with imbalanced datasets, as a high accuracy can be achieved by simply classifying all instances as the majority class [7]. Accurary is given as

$$\text{Accurary} = \frac{\text{TP} + \text{TN}}{TP + TN + FP + FN}$$

Where TP represents the number of true positives, TN represents the number of true negatives, FP represents the number of false positives, and FN represents the number of false negatives. True positives (TP) are instances in which the model correctly predicted the presence of the positive class. True negatives (TN) are instances in which the model anticipated the negative class correctly. False positives (FP) are instances in which the model predicted a positive class when the actual class was negative. False negatives (FN) are situations in which the model predicted a negative class, but the actual class was positive.

3.1.3 Precision

Precision is a measure of the accuracy provided that a class label has been predicted. It is defined as the number of true positives divided by the sum of the true positives and false positives. It is also called Positive Predictive Value. Precision can be seen as a measure of exactness or quality, whereas recall is a measure of completeness or quantity. [8]. Precision is the ratio of true positive predictions (TP) to the total number of positive predictions produced by the model, which includes both true positive predictions and false positive predictions. (FP). It is given by,

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high precision score indicates that the model predicts few false positives, indicating that the predicted positive cases are more likely to be genuine positives. In other words, the model accurately identifies relevant instances and does not mislabel irrelevant instances as relevant. A high precision score may also indicate that the model is omitting positive cases, resulting in a high false negative rate. In order to provide a more comprehensive evaluation of a model's performance, precision is frequently combined with other metrics, such as recall and F1 score.

3.1.4 Recall

Recall is defined as the number of true positive results divided by the number of all relevant samples (all samples that should have been identified as positive). Intuitively, recall measures the ability of a classifier to find all the positive samples. Recall is also called sensitivity or the true positive rate (TPR) [9]. In other words, it measures the proportion of positive instances that are correctly identified by the model. Mathematically, recall can be expressed as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

where TP represents true positives and FN represents false negatives. A true positive is an instance that is positive and correctly classified as positive by the model, whereas a false negative is an instance that is positive and erroneously classified as negative by the model. Consequently, a high recall score indicates that the model is capable of accurately identifying a significant proportion of positive instances. In situations where the cost of ignoring a positive instance is high, such as in medical diagnosis or fraud detection, recall is particularly useful.

3.1.5 F1-Score

The F1-score is a metric for evaluating the performance of a binary classification model. It is the harmonic mean of precision and recall, where precision is the ratio of true positives to predicted positives and recall is the ratio of true positives to actual positives. The F1-score is a method for balancing precision and recall, as it takes erroneous positives and false negatives into account. It is determined by:

$$\text{F1-score} = 2 * \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

A faultless F1 score of 1.0 indicates that the model's precision and recall are flawless. In practice, the F1-score is beneficial when both precision and recall are crucial and must be considered. It is commonly used for applications such as information retrieval, document classification, and spam filtering. Many applications require a single-valued estimate of overall classifier performance. The most widely used such measure is the F1 score, the harmonic mean of precision and recall. It provides a single number evaluation of a classifier, that is both precision and recall, but requires no specific knowledge of the decision threshold [10].

4 Ethical and Social impact of Machine Learning

The world has been drastically altered by machine learning in recent years. The healthcare system, user experiences, and productivity across sectors have all benefited from this. But there are ethical and social concerns that must be handled as a result of these developments. In this paper, I'll discuss some of the societal and ethical implications of machine learning. Privacy is one of the main ethical concerns with machine learning. There is a risk that confidential information may be mishandled or exposed due to the vast amounts of data being collected. For instance, data breaches and misuse can result in identity theft and discrimination. This is especially worrisome when it comes to sensitive data such as medical records and financial information. It is essential to ensure that appropriate measures are taken to safeguard data privacy and ensure the ethical and responsible use of personal information.

Bias in machine learning systems is another moral issue. The lack of bias in machine learning systems can be remedied by using more objective data for training. A biassed algorithm will produce unjust or discriminatory results if it is fed biassed data. Facial recognition algorithms that are trained on data that is overwhelmingly male and white, for instance, may not be as accurate when applied to data that includes individuals of other races and ethnicities. It is crucial to guarantee that diverse and representative data is used to teach machine learning algorithms free of bias. Machine learning affects society in ways that go beyond ethical issues. Loss of employment is one of the most important effects. Many jobs are at danger of being automated out of existence as automation continues to grow. Significant societal repercussions may result from this, including rising unemployment and income inequality. Making ensuring that the right steps are taken, such as retraining programmes or other social safety nets, to address the social effects of job displacement is crucial.

Algorithmic decision-making, another social effect of machine learning, may exacerbate or even perpetuate pre-existing social inequalities. Machine learning algorithms may reinforce preexisting biases and inequalities if they are used to determine things like who gets a loan or who gets employed. Ethical use of machine learning algorithms is crucial, as is taking precautions against their possible prejudices or unintended consequences. Though machine learning has improved many aspects of life, it also raises important ethical and social concerns that must be handled. It is crucial to guarantee the ethical and responsible application of machine learning, and to take adequate precautions against any prejudices or unintended consequences. When we do this, we can use machine learning to make the world a better place while reducing its potentially harmful effects.

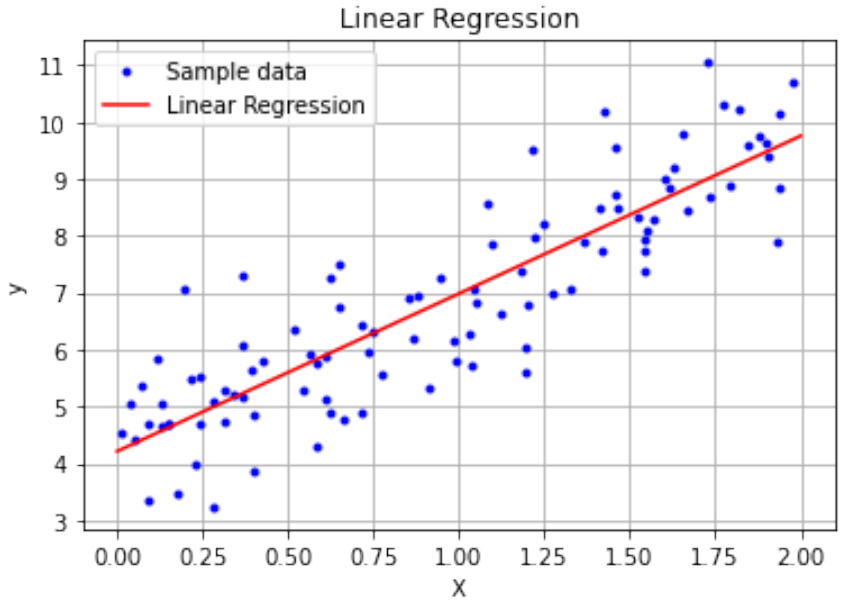
5 Conclusion

In this thesis, the mathematical foundations and algorithms of well-known machine learning techniques such as linear regression, logistic regression, decision trees, random forests, and neural networks were investigated. We also evaluated these algorithms using various evaluation metrics, including confusion matrix, accuracy, precision, recall, and F1 score, on various datasets. Our objective was to evaluate the pros and cons of these algorithms in order to determine which ones are best suitable for various types of problems. Through this investigation, we gained a deeper comprehension of the mathematics and logic underlying these algorithms, as well as their practical applications. We also investigated the significance of the testing metrics used to evaluate these algorithms.

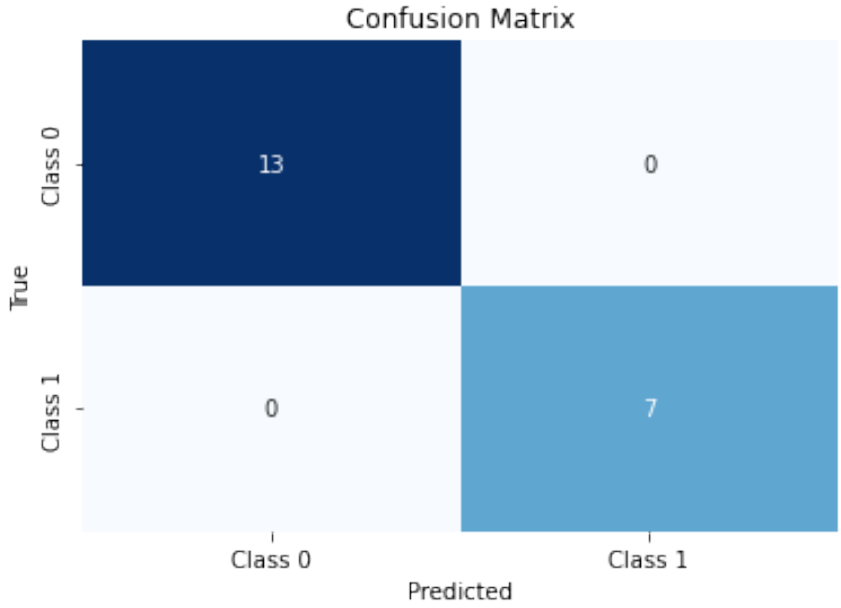
It is essential to recognise that the expanding use of machine learning techniques in various disciplines has both ethical and social implications. On the one hand, these techniques have the potential to revolutionise industries like healthcare, finance, and education, resulting in more efficient and effective decision-making. Concerns exist, however, regarding the potential for prejudice and discrimination, as well as the impact on employment and privacy. This thesis emphasises the need to comprehend the mathematical foundations and testing metrics of machine learning techniques in order to make informed decisions when employing these techniques to real-world problems. Additionally, it is essential to consider the ethical and social implications of these techniques and to work towards maximising their benefits while minimising their potential negative effects on society.

A Model Plots

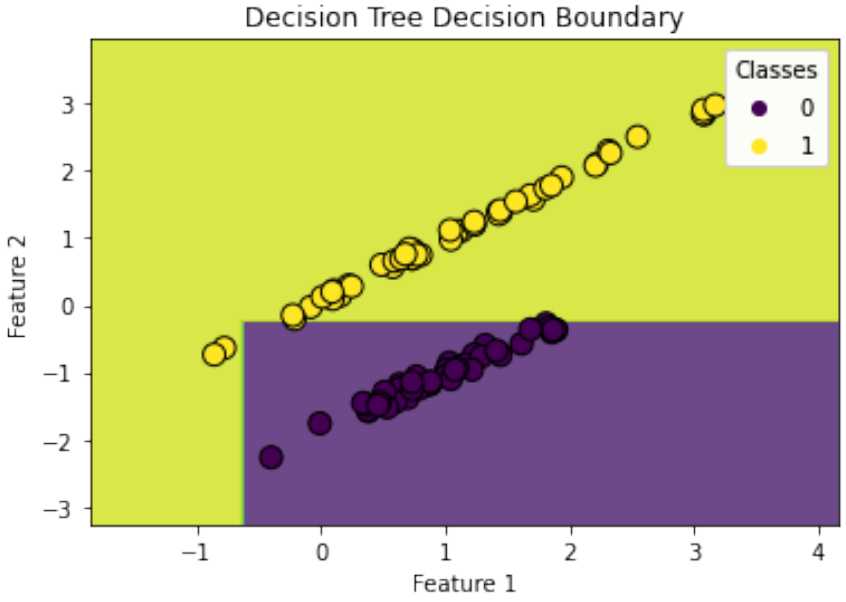
A.1 Linear Regression



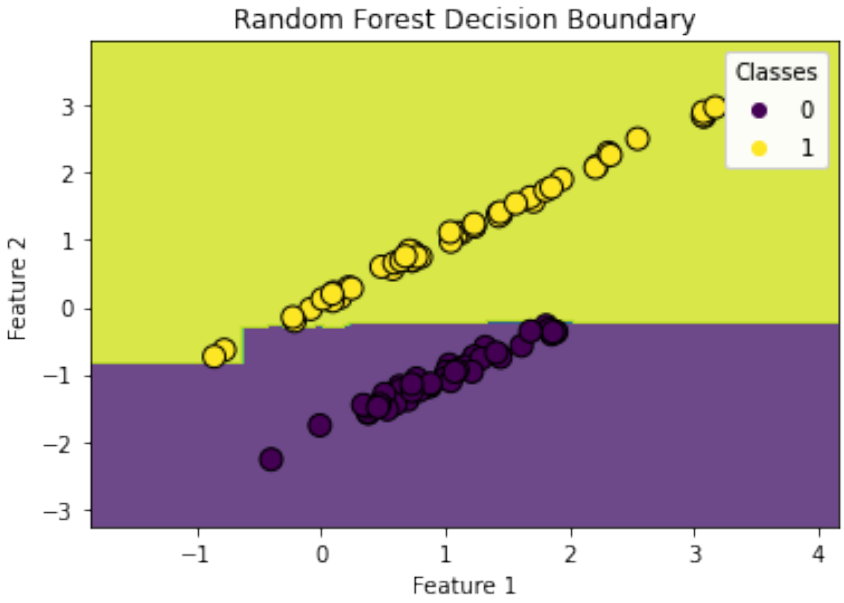
A.2 Logistic Regression



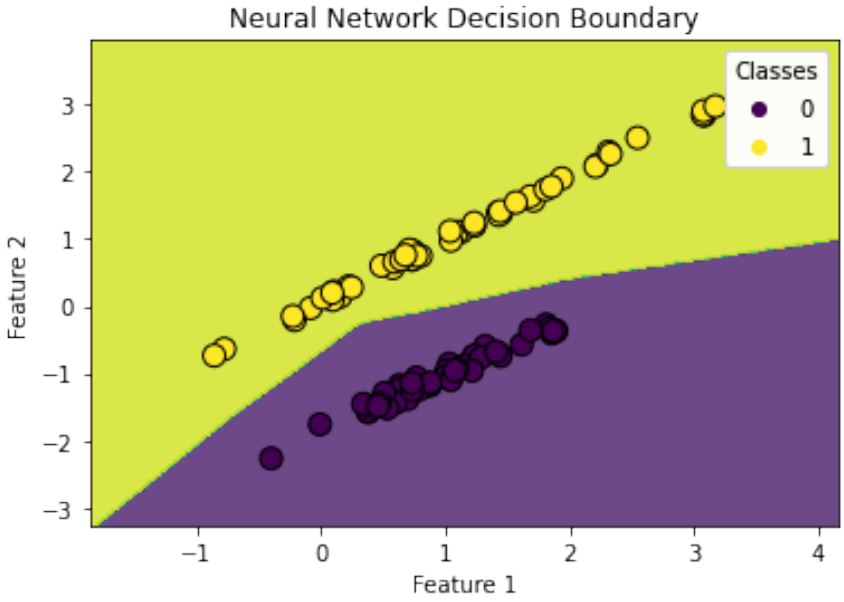
A.3 Decision Trees



A.4 Random Forest



A.5 Neural Network



References

[1] Zhou, Di, Xiao Zhuang, Hongfu Zuo, Han Wang Hongsheng Yan: *Deep learning-based approach for civil aircraft hazard identification and predictionDeep learning-based approach for civil aircraft hazard identification and prediction.* IEEE Access, 8:103665–103683, 2020.

[2] Digest, Consumer's: *List price vs. best price for a new gmc pickupList Price Vs. Best Price for a New GMC Pickup.*

[3] Zuur, AlainF., ElenaN. Ieno MeestersErikHW G.: *A beginner's guide to R.* Springer, 2009.

[4] Lee, J., H. Kim, J. Kim I. Ko: *Logistic regression in machine learningLogistic regression in machine learning.* Journal of Machine Learning Research, 21:1–28, 2020.

[5] Austin, PeterC EwoutW Steyerberg: *Goodness-of-fit tests in the logistic regression modelGoodness-of-fit tests in the logistic regression model.* Statistical methods in medical research, 24(3):342–356, 2015.

[6] Breiman, Leo, JeromeH Friedman, RichardA Olshen CharlesJ Stone: *Classification and regression treesClassification and regression trees.* Chapman & Hall/CRC, 2(3):229–231, 1984.

[7] Galar, Mikel, Alberto Fernández, Edurne Barrenechea, Humberto Bustince Francisco Herrera: *A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approachesA review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4):463–484, 2012.

[8] Davis, Jesse Mark Goadrich: *Evaluation measures for machine learning: Precision, recall, and f1 scoreEvaluation measures for machine learning: Precision, recall, and F1 score.* ICML, 5(1):33–48, 2006.

[9] Davis, Jesse Mark Goadrich: *Evaluation measures for machine learning: Precision, recall, and f1 scoreEvaluation measures for machine learning: Precision, recall, and F1 score.* ACM transactions on knowledge discovery from data (TKDD), 1(1):1–37, 2006.

[10] Powers, DavidM: *Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlationEvaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.* Journal of machine learning technologies, 2(1):37–63, 2011.