Analysis of Three Statistical Classification Methods on the Two Oceans Marathon Race (MRC Project)

Callum Pet

Matthew Parfitt

Report presented in partial fulfilment
of the requirements for the degree of
BComHons **(Statistics**)
at the University of Stellenbosch

Supervisor: Dr Justin Harvey

Degree of confidentiality: "[A, B or C]" 5 November 2021

PLAGIARISM DECLARATION

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. I also understand that direct translations are plagiarism.
4. Accordingly, all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. I declare that the work contained in this assignment, except otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

21613974 21679495	Callum Pet Matthew Parfitt
Student number	**Signature**
C.J Pet M.D Parfitt	4 November 2021
Initials and surname	**Date**

Acknowledgements

We would like to express our sincere gratitude to Dr Justin Harvey, our supervisor, for his patience and valuable guidance throughout the year in regards to this research project. Also we would like to thank the Department of Statistics and Actuarial Science at Stellenbosch University for providing us with a vast amount of statistical knowledge and encouragement over the past four years, and for always being willing to assist us. Finally, we would also like to thank all the statistics students for making this year, although it was challenging, such an enjoyable year.

Abstract

The research assignment is the analysis of three classification methods in determining whether an entrant in the Two Oceans Marathon would start the race and whether the person who started the race would go on to finish it. The assignment introduces the three methods: Logistic Regression, Linear Discriminant Analysis and Support Vector Machines as well as giving a brief explanation on the race itself. The aim was the project was to analyze the different results that these three different classification methods could produce. Although there are many other statistical methods that could have been used and modelled the data better, these three methods were the most interesting and caught the eye of the authors. Especially comparing the difference between the black box and transparent methods. The literature review describes the history and gives an understanding of how the methods can be used in statistical learning. In the research and methodology an in-depth explanation of each method is given, viewing how they are proven and can be used in classification problems. A comparison between the methods is given through viewing the advantages and disadvantages. In the data analysis the actual dataset is first summarized by looking at the number of observations as well as the variables that are significant and selected. The SVM is first analyzed by determining the best kernel to use in running the model. The kernel decision was between the linear, radial basis function, polynomial, and sigmoid kernel. After the kernel is selected the three different methods are programmed to determine which would result in the best method to determine whether the entrant would start or finish. The ideal method is determined by the one that results in the best performance metrics. The metrics that are used are the Area Under Curve, Receiver Operating Characteristics, Accuracy, Test Error, Sensitivity and Specificity. The classes of the classification variables are highly imbalanced so there is an analysis on the imbalanced data as well as the data is balanced to determine if there is a difference in the results. The assignment concludes by portraying the best method that would be able determine the starters and finishers of the Two Oceans Marathon.

Table of Contents

List of tables

List of figures

List of appendices

Lists of abbreviations and acronyms

MRC	Medical Research Council
LDA	Linear Discriminant Analysis
SVM	Support Vector Machine
LM	Learning Machine
RBF	Radial Basis Function
CV	Cross-Validation
TP	True Positive
TN	True Negative
FN	False Negative
FP	False Positive
DNS	Did Not Start
DNF	Did Not Finish
TPR	True Positive Rate
FPR	False Positive Rate
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
AUROC	Area Under the Receiver Operating Characteristics
SMOTE	Synthetic Minority Oversampling Technique

List of notations

Notation	Description
n	Number of observations in dataset (total sample size)
n_1	Number of observations in DNS data
n_2	Number of observations in DNF data
p	Number of original variables in the dataset
p'	Number of selected variables for model (predictor variables)
ω	Sample classes
ω_1	Sample class 1
ω_2	Sample class 2
y_1	Response variable 1 (DNS)
y_2	Response variable 2 (DNF)
e	Base of natural logarithm
X	Input variable for logistic regression
β_0	Parameter of the logistic model when $X = 0$
β_i	Parameters of the logistic model
R^2	Coefficient of determination
$\bar{x}_1$	Sample mean from class 1
$\bar{x}_2$	Sample mean from class 2
S_{B_i}	Between-class variance for the ith class
S_{W_i}	Within-class variance for the ith class
S_b	Between-class covariance matrix
S_w	Within-class covariance matrix
μ	Total mean
μ_i	The ith class mean
c	Population class
m	$(c-1)$ class
w	Projection vector
$\hat{\lambda}$	Eigenvalues of the transformation of the projection vector
$\hat{e}$	Eigenvectors of the transformation of the projection vector
X_i	Input data matrix
Y_i	Binary response variable
S	Feature space set
W	Weight vector in binary case ($W \epsilon R^2$)

w	Weight vector in n-dimensional space ($w \epsilon R^n$)
x_i	ith number of input variables
$g(x)$	Linear discriminant function
b	Bias
$J(w)$	Objective function
λ_i	Lagrange multipliers
ξ_i	Slack Variables
C	Regularisation parameter (Cost parameter)
ϕ	Mapping function phi
σ	Width parameter for kernels
γ	Gamma parameter for kernels
d	Degree parameter for kernels
r	Offset (coef0) parameter for kernels

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

The focus of this research project will be the comparison of three different statistical classification methods and to what level they perform when being tested on the data from the Medical Research Council (MRC), the Two Oceans Marathon data, specifically the post-period which ranges from years 2012-2015. The importance of classification methods in statistics is crucial, and especially with the increase in recent years with the sizes of data that is needed to be analyzed which has led to an emerging relevance of the development of tools capable of extracting knowledge from the data and performing prediction and classification (David & Balakrishnan, n.d.). A synopsis of classification is important before going in any further into this research task, as it is one method that classifies unstructured data into the structured class as well as groups which assists the researcher for knowledge and opportunity planning (Koturwar, et al., n.d.). The intellectual decision making that classification offers for a researcher is invaluable and another major reason why it is so popular in the present day. Two phases are used, first phase involves large training data sets where an analysis is done and patterns are noted, then the second phase consists of test of data sets and the results are analyzed and distinguished of each classification pattern. A supervised method will also be another focus throughout this task, as with a supervised method there is a particular target value where the target value in this case has two possible outcomes, a binary classification, from which predictions can be made about the data.

1.2 CLASSIFICATION METHOD

There are many different classification techniques available to use and analyze however the focus has been narrowed down to purely on the performance level of logistic regression, linear discriminant analysis (LDA) and support vector machine's (SVM) and the capability of each to handle the given data set. There are many other statistical methods that could handle the data better and produce better results, such as XGBoost which is a known performer on machine learning tasks. The three methods were selected due to the curiosity of the authors comparing two closely related transparent methods with high interpretability such as logistic regression and LDA that are commonly used, to comparing these methods to a black box method, SVM, that is known to be accurate but not interpretable. The analysis of the three methods is just to determine how different classification methods with different

trade-offs between interpretability and complexity compare against one another. The dataset in question can be considered to be a narrow, low dimensional as $n > p$ to a large extent, therefore with n being such a large value, this will stretch each classification method to its full capability to allow a thorough analysis to be done of which one performs the best.

Logistic regression falls under the supervised machine learning category with emphasis that lies on classification problems as it is a predictive analysis that is used to model the probability of a particular event, with binary classification being the main focal point (Pant, 2019). LDA is a classification technique that is said to be preferred sometimes however only when the classification problem is more than a two-class problem, LDA makes predictions by estimating the probability that a new set of inputs belong to each class (Brownlee, 2020). SVM is another well-known classifier that is also popular in modern day statistics, it is a method that analyzes the data and searches for patterns that can be applied to the classification procedure. With the data set into a training set, SVM constructs a model that places the data into one of the two classes that have been created.

1.3 TWO OCEANS MARATHON

The Two Oceans Marathon takes place in Western Cape, South Africa and is comprised of an Ultra and a Half – marathon, where the Ultra is comprised of a 56km race and while the Half is a 21km race. The marathon is known as ‘the world’s most beautiful marathon’ and is a test for any caliber of runner, especially the Ultra. Taking place on the Saturday of the Easter weekend with a backdrop of the Cape Peninsula. Thousands of runners from around the world take part each year, with the majority participating being from the Cape.

1.4 BRIEF EXPLANANTION OF QUESTIONNAIRE DESIGN

The online pre-race medical questionnaire was compulsory as part of the registration process, the choice of it being used in research was up to the entrant. The questionnaire comprised of simple questions such as age, gender and which race the entrant is running. The questionnaire included more detailed medical questions to detect if the results were due to any medical relations. With more focused questions such as does the entrant use any prescribed medication on a daily weekly or monthly basis to treat chronic (long-term) medical conditions or injuries, has the entrant consulted a medical doctor in the past twelve months to obtain a medical clearance that the entrant may safely participate in endurance running and if yes, did their medical practitioner clear them with any specific advice for participating in endurance running. There are levels to the question regarding whether the

doctor gave the entrant clearance to run such as the first one being their doctor did not give them clearance, the second one being their doctor gave them clearance but with some restrictions and guidelines on safe participation and lastly the third one being their doctor gave them clearance to run with no restrictions. There were specific blocks to be filled within the medical assessment question regarding what procedures were undertaken if the entrant did see a medical practitioner in the past twelve months.

1.5 DATA RESTRUCTURE

The data that is received from the questionnaire is largely imbalanced longitudinal data. Originally that three selected methods will model the imbalanced data from 2013 to 2015. Due to the independence assumption of logistic regression that all observations in the Two Oceans must be independent of each other, models of the data are also run from year to year to take this into account. The assumption is due to entrant that have participated in the race more than once. An average is taken off the three independent years rather than removing the dependent observations due to the SVM not performing well in structured data, so to account for the two classification methods that data is made semi-structured (Bassey, 2019).

1.6 CONCLUSION

The Two Oceans is a very popular race with a high demand of people who want to take part. Therefore, by testing which classification of these method is the best at determining whether an entrant would go on to start the race or finish would be very beneficial in the efficiency and planning for the race in the years to come. Although there probably other methods that could return better results, the project is analysing the best outcomes and interpretations of the methods due to the attention shown in these three selected methods.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

Classification techniques have grown a tremendous amount over the past few decades in terms of reliability, capacity they can manage and just the overall research behind them. The main few that are going to be looked at and studied are Logistic Regression, LDA and SVM. The review will provide a short history of each of the methods, as well as an insight into each of the statistical learning methods, such as limitations and advantages as well as to any recent discoveries if there are.

2.2 LOGISTIC REGRESSION

The logistic function was developed as a model of and named "logistic" by Pierre-François Verhulst in the 1830s and 1840s, under the guidance of Alphonse Quetelet who was a Belgian astronomer that turned statistician. Quetelet was aware that the indiscriminate extrapolation of exponential growth must lead to impossible values. Pierre-François Verhulst was a pupil of Quetelet (Cramer, 2002). For Verhulst first paper that he published, he did not specify how he fit the curves to the data. However, for his more detailed paper that was later released in 1845, Verhulst determined the three parameters of the model by making the curve pass through three observed points, which in the end, yielded poor predictions. The logistic function was independently developed in chemistry as a model of autocatalysis (Wilhelm Oswald, 1883).

Raymond Pearl and Lowell Reed rediscovered the logistic function as a model of population growth in 1920 published as Pearl & Reed (1920), which led to its use in modern statistics. At first, Pearl and Reed were unaware of Verhulst's work and assuming they learned about it from L. Gustave du Pasquier, however he was given nearly zero credit and his terminology was not adopted. Verhulst was acknowledged to a greater extent and his term "logistic" revived by Udny Yule in 1925 and has been the trend since. The model was first applied to the population of the United States by Pearl and Reed and following the trend of fitting the model by making it pass through three curves, following Verhulst, poor results were obtained.

The probit model was developed during the 1930s by Chester Ittner Bliss, who coined the term "probit" in Bliss (1934) and by John Gaddum in Gaddum (1933) and the model fit

by maximum likelihood estimation by Ronald A. Fisher in Fisher (1935). The development of the logistic model as an alternative to the probit model, was highly due to the reasoning of work done by Joseph Berkson.

One major advantage with logistic regression when it is compared to other similar methods and a major reason why it is so popular among medical researchers, especially with data that is similar to the one in this research task is that the exponentiated logistic regression slope coefficient (e^b) may be simply interpreted as an odds ratio (Schober & Thomas, 2021). There are many other advantages such as it is a bit more involved than linear regression, which is one of the simplest algorithms out there. It is also transparent; therefore, the process can be seen clearly and can understand what is going on at each step. Logistic regression is able to provide a feature importance assessment as well as the direction each feature affects the response value, if that is positively or negatively. When looking at coefficient weights, the absolute value shows the magnitude of the influence while the sign shows the direction (Phung, 2019). As well as this method is very quick at classifying unknown records. Logistic regression also makes zero assumptions about distributions of classes.

However, there are limitations for logistic regression such as the focus of it is intended for binary classification problems. It can be extended for multi-class classification but is mainly used for binary classification. Logistic regression can become unreliable when the classes are separated well and when there are few samples from which to estimate the parameters (Brownlee, 2016).

2.3 LINEAR DISCRIMINANT ANALYSIS

Ronald A. Fisher, in 1936, formulated linear discriminant and showed some practical uses as a classifier, it was described for a 2-class problem, and later generalized as 'Multi-class Linear Discriminant Analysis' which was a generalization of the standard two-class LDA that can handle illogical number of classes and it is based on the analysis of two scatter matrices, the *within-class scatter matrix* and *between-class scatter matrix.*

There was also 'Multiple Discriminant Analysis' put forward by C.R. Rao in 1948, which was a multivariate dimensionality reduction technique however it is not directly used to perform classification but merely used as support by yielding a compressed signal open to classification therefore this classification was useful because of most classifiers being affected by the curse of dimensionality, which is various phenomena that arise when

analysing and organizing data in high-dimensional spaces, in other words the performance of the classifier is catastrophically impaired by the overfitting problem. This problem can be reduced squeezing the signal to a lower-dimensional space which is what Multiple Discriminant Analysis does. Regarding supervised learning, LDA is the most commonly used dimensionality reduction technique. It is a pre-processing step for pattern classification and machine learning applications. (Tyagi, 2019).

The advantages of LDA is that the data points can be separated linearly, also multi-featured data can be classified. It also allows for discrimination between multiple features of a dataset. LDA can be used in facial recognition, medical treatment analysis, tools for customer identification and marketing as well as it is a very simple and applicable classification tool that can be used in many real-world examples and problems (Acharjee, 2021).

LDA consists of limitations such as the method produces C – 1 feature projections. If more features are needed after the classification error estimates establish that is what is needed, another method must be used to provide the additional features needed as well as LDA is a parametric method therefore it assumes unimodal Gaussian Likelihoods, unimodality is defined as having a unique mode, or a single highest value. If the distributions are significantly non-Gaussian, the projections from LDA may not keep the complex structure in the data that is needed during classification. (Gutierrez-Osuna, n.d.). Another issue would be that LDA could fail if discriminatory information is not found in the mean but in the variance of the data.

2.4 SUPPORT VECTOR MACHINES

2.4.1 History

The SVM methodology was derived from the Generalised Portrait Algorithm, a statistical learning theory created by the Russian scientist Vladimir Naumovich Vapnik back in 1962 (Thomé, 2012). This machine learning algorithm was then introduced by Boser et al. (1992). This approach was developed in the computer science community and has been receiving increased attention around the world as years have gone past. The statistical method is based on the Structural Risk Minimisation (SRM) principle (Kanevski & Pozdnukhov, 2001). The principle made by Vapnik posed questions that had to be addressed to design learning machines (LMs). The four questions are:

i. What are the necessary and sufficient conditions for consistency of learning process?
ii. How fast was the rate of convergence to the solution?
iii. How could we control the generalization ability of the LM?
iv. How could we construct an algorithm that implement these prerequisites?

With these it was aimed to minimise both the empirical risk and complexity of the model that would result in higher generalisation abilities (Kanevski & Pozdnukhov, 2001). Empirical risk is essentially the performance measured on a known set of training data with the complexity the sum of the following: differentiation of functions between project participants, dependencies between systems and subsystems, and the consequential impact of a decision field. The term basically measures the error on the dataset sample that is being tested (Pogančić, 2019). Model complexity is essentially the flexibility of a model, where minimizing it would simply be reducing the chances of overfitting by keeping a low variance and a higher bias. Although by minimising it too much could result in underfitting (Sejdinovic, n.d.). The basic idea of the SVM:

i. Map the data into a high dimensional feature space (margin) via a non-linear mapping.
ii. Construction of an optimal hyperplane for separating features.

SVM's have been seen to be versatile being able to perform in a variety of settings and considered as one of the best "out of the box" classifiers as an add on from the simple and intuitive classifier called the maximal margin classifier (James, et al., 2013).

2.4.2 Maximal Margin Classifier

The maximal margin classifier is the first step of the complex SVM and is a hypothetical classifier that simply explains how the methodology works in practice. Essentially being able to plug input variables in your data from a sample size into a line equation and able to calculate whether the next point would be above or below the hyperplane (Brownlee, April 20, 2016). The hyperplane is the line that separates the variable space into the two classes. The margin is thus the distance between the line and the nearest data points and therefore the maximal margin hyperplane being the optimal line that separates the classes farthest from their training points. It is assumed in this instance the training points are linearly separable and the classes can be split perfectly. The classifier can thus determine from this scenario which sides of the margin the test data would lie (Gao & Zhang, 2009).

2.4.3 Support Vector Classifier

In reality the problem arises where there are large amounts of scattered data and splitting the data into two separate classes is not that easy. The soft margin classifier, also known as the support vector classifier, allows for some points in the training to violate the hyperplane (Brownlee, April 20, 2016). Rather than seeking the maximal margin classifier so that every observation is on the correct side of the margin, the soft margin allows some observations to be on the incorrect side of the hyperplane. Slack variables are the additional set of coefficients that are established to allow extra margin space for observations. This increases the complexity of the model due to more parameters to fit the data (James, et al., 2013). The support vectors are initially all the training observations that lie within the margin and have an effect on the placement of the hyperplane. When there is a large amount of support vectors the algorithm is more flexible to the training data therefore having a high variance and low bias. Contrasting few support vectors would result in a less flexible algorithm with a low variance and higher bias. This trade-off of support vectors mirrors the well-known bias-variance trade-off which is a model's ability in minimizing bias and variance which results in an accurate model that neither underfits nor overfits (Singh, 2018).

2.4.4 Support Vector Machine

The approach naturally has been intended for binary classification, which is the classification approach in two-class setting. In most practical situations the classification problems are non-linear. The SVM can provide non-linear classification using kernel functions this being where the maximal separating margins are constructed. The kernel allows for the algorithm to be implemented in practice. The SVM is an extension form the classifier the results in the feature space enlarged using such kernels (Kanevski & Pozdnukhov, 2001). The use of different kernels obtains learning machines analogues to well-known architectures. Kernel modification can improve the overall performance in a data-dependent way, with data-dependency being the situation where a program statement refers to the data of a proceeding statement. This would allow for very flexible models that could solve a wide variety of classification tasks to be built. Enlarging such feature space would allow for the accommodation of a non-linear boundary between the classes (Ingo Steinwart, 2008). There are number different types of kernels such as: the linear, the polynomial and the radial. The linear kernel is the least complex with the distance being a

linear combination of outputs. The polynomial then becomes more complex allowing for curved lines in the feature space with the radial being even more complex and allows for a polygon shaped input space that could separate the two-classes (Brownlee, 2020).

2.5 CONCLUSION

The SVM is a much newer method compared logistic regression and LDA although it has become a more popular method on classification in recent years. LDA is one of the most popular methods especially in binary cases. Yet for beginners the use logistic regression is a great selection due to its transparency along the whole model and allowing for a clear interpretation. All three methods do have advantages compared to the others in different ways and through the assignment it will be seen which method outweighs the others in the analysis of the large Two Oceans Marathon dataset.

CHAPTER 3: RESEARCH AND METHODOLOGY

3.1 INTRODUCTION

Logistic regression, LDA and SVM are three important classification methods used worldwide by millions of statisticians in different ways with different intentions. The assumptions and reasoning behind the equations is discussed in depth throughout this section as well as an insight to comparisons made between them. The situations to which one method is preferred over the other will be discussed in this research project as well as an understanding will be gained throughout the following section.

3.2 LOGISTIC REGRESSION

The usage of logistic regression becomes appropriate when the dependent variable is binary, which in the case of our study, it is. Logistic regression is used to describe data and to give an explanation of the relationships between one dependent binary variable and one or more interval, ordinal, nominal or ratio-level independent variables which is a type of variable measurement scale that is quantitative in nature and allows any researcher to compare the differences. Logistic regression is also considered a predictive analysis with the advantage of multiple explanatory variables being evaluated by the extension of basic principles; the general equation is:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_n X_n)}} = \frac{1}{1 + e^{-(\beta_0 + \sum \beta_i X_i)}} \tag{3.1}$$

With e being the base of the natural logarithm, β_0 being a parameter of the model when $X = 0$ and β_i also being a parameter of the model and specifically adjusts how the probability fluctuates with changing X by one unit. β_0 is complicated for a researcher to interpret as the relationship between X and P is nonlinear, β_0 is usually not as complex to interpret when in ordinary linear regression.

An important part of logistic regression is the odds ratio, which represents the constant effect of predictor X, with the likelihood of the outcome occurring. However logistic regression does not analyze the odds, analysis is done on a natural logarithmic transformation of the odds, the log odds. Calculations can be very complicated when there

are multiple independent variables, computer programs play a major role in helping perform these procedures.

An important assumption is that the model is correctly specified, two parts make up correct specification, the functional form of the model is correct is the first part while the second part consists of the model including all relevant independent variables as well as zero irrelevant independent variables. The issue with the inclusion of one or more irrelevant variables is that the standard error of the parameter estimates will be increased, therefore with the reduction of the of the efficiency of the estimates, without biasing the coefficients (Menard, 2002). The only exception to the standard errors not becoming inflated would be that the irrelevant included variable is entirely uncorrelated with the other variables, however this would be exceptionally unlikely in practice. The omission of a relevant variable from the equation in logistic regression leads to biased coefficients for specifically the independent variables, to the extent that the variable that was omitted is correlated with the independent variables in the logistic regression equation. As in linear regression (Berry & Feldman, 1985) the direction of bias depends on the parameter of the excluded variable, the direction of the effect of the excluded variable on the dependent variable, as well as the direction of the relationship between the excluded and included variables. Magnitude of the bias depends on the strength of the relationship between the variables that have been included as well as excluded. Similar to what was stated before, however with the excluded variable being entirely uncorrelated with the included variables, the coefficients again, may be unbiased. Bias is largely referred to as a bigger issue than inefficiency, however mass inefficiency would be more worried about than a small amount of bias.

The available theories that have failed to identify the all the causes of a dependent variable, relevant predictors may be a reason that omitted variable bias occurs as well as variables that are theoretically relevant have just been omitted. The functional form of the model being mis specified may also lead to the pattern characteristic of omitted variable bias.

The general equation was stated earlier in Equation 3.1, we use the logistic function in logistic regression:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{3.2}$$

After manipulation of the above:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X} \tag{3.3}$$

By taking the logarithm of both sides:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \tag{3.4}$$

The left-hand side of Equation 3.4 is known as the $logit$. Nonlinearity in the logit, with the change in $logit(Y)$ is constant for a one-unit change in X and it does not depend on the value given to X, therefore the logistic regression model has a relationship that is linear in the $logit$. Therefore, the change in $logit(Y)$ for a one-unit change in X is equal to the logistic regression coefficient. If a non-linear relationship is present, the change in $logit(Y)$ for a one-unit change in X is therefore not constant and depends of the value given to X. Numerous techniques are available for detecting nonlinearity in the relationship between the dependent variable, $logit(Y)$, and each of the independent (Hosmer and Lemeshow, 1989, pp. 88-91). One of the methods is to treat each of the independent variables as if they were categorical variables and use an orthogonal polynomial contrast, which is the comparison mean values for two or more time points of an independent variable to test for linear, quadratic, cubic, as well as higher order effects in multiple logistic regression or in a bivariate logistic regression model. Another method for detecting nonlinearity is one called the Box-Tidwell transformation, which contains adding a term of X multiplied by $\ln(X)$to the equation (Hosmer, et al., 2013). Evidence is present of nonlinearity in the relationship of $logit(Y)$ and X if the coefficient for this specific variable is significant. If the result is the relationship being nonlinear, there must be further analysis to determine the pattern of the nonlinearity present.

The next assumption, non-additivity, transpires when the change in the dependent variable associated with a one-unit change in the independent variable depends on the value of another one of the other independent variables. Nonlinearity is slightly simpler to pick up in either linear or logistic regression, non-additivity is more challenging for a researcher to

detect. Theory may provide some guidance in this sense but if not, it is common to be left with either assuming an additive model, testing for interaction effects that seem to be the most plausible or even all possible ones.

Independent variables being correlated with each other develops an issue called collinearity. An independent variable being a perfect linear combination of the other independent variables is perfect collinearity. A way to obtain perfect collinearity, would be to for each independent variable to be treated as the dependent variable in a model and all other independent variables as predictors, the outcome is an $R^2 = 1$ for each individual independent variable. Perfect collinearity results in an endless amount of possible combinations of logistic regression coefficients that would all work as well as each other, obtaining one unique estimate of the regression coefficients would be impossible. Less than perfect collinearity is the usual. The increase of collinearity amongst the independent variables, leads to the logistic regression coefficients being unbiased and efficient to the largest possible extent, taking into account the relationships between the independent variables.

Comparing logistic regression to SVM, logistic regression can be considered faster than a technique such as kernel SVM however when considering that accuracy of the method is slightly less than that of a kernel SVM, the justification of the methods is similar. The next analysis that will be discussed is LDA, a method regarded as very similar to logistic regression and questions are raised such as to why one would use LDA when logistic regression is readily available and one of the reasons to this is the parameter estimates for the logistic model can be relatively unstable if the classes are well-separated, this is an issue that does not occur for LDA. Another reason is if n is small and the distribution of X is approximately normal in terms of each of the classes, this leads to a more stable linear discriminant model when compared to a logistic regression model (James, et al., 2013). LDA is also a very established method when the response classes are larger than two. Similarities between the two methods are evident as they both generate linear decision boundaries, the difference within this procedure would be that the parameters of logistic regression are estimated using maximum likelihood and on the other hand, the parameters of LDA are generated using the estimated mean and the variance from a normal distribution. The same relationship is present between logistic regression and LDA for multidimensional data with $p > 1$ (James, et al., 2013). Therefore, since the only difference between LDA and logistic regression is present during the fitting procedures of both, similar results and output is expected from the two methods however that is not the case on some instances. LDA assumes that the covariance matrices are identical in each class and observations are

drawn from a Gaussian distribution and therefore with the assumption holding, LDA is considered to have improvements over logistic regression. On the other hand, with the assumption not holding, logistic regression can outperform LDA.

3.3 LINEAR DISCRIMINANT ANALYSIS

A major part of LDA, is to know which population the data observations in the initial sample, known as the training sample, belong to as this aids in making future predictions of the classification of data observations which population they belong to is unknown. LDA is most often used to classify patterns between classes, as well as classifying multiple patterns, either binary or the latter which is a multi-class classification setup. Focus will be made on the binary side of it as this it was the study entails. An assumption of LDA is that all classes are linearly separable and according to the multiple linear discrimination function in which several hyperplanes are created in the feature space which allow for the classes to be distinguished (Vaibhaw & Pattnaik, 2020). For binary classification, then one hyperplane is constructed by LDA, and the data is projected such that the space between the two classes is maximized. This hyperplane is constructed according to two conditions taken into account simultaneously:

i. The minimization of variance between each class.
ii. The maximization of distance between the means of each class.

Three steps are needed to achieve the goal of LDA, which is to predict the original data matrix onto a lower dimensional space. The first step that is taken into account is to calculate the distance between the means of the two classes, the separability between the two classes, known as the between-class variance. The second step that is taken into account is to calculate distance between the sample and the mean of each class, known as the within-class variance. Third step is to create a lower dimensional space that maximizes the between-class variance and minimizes the within-class variance (Tharwat, et al., 2017), therefore a more in-depth explanation to the bullet points above.

Fisher put forward LDA in 1935, when focusing on differentiating between the two populations. Normality is not assumed within Fisher's method however the same variance-covariance matrix is assumed for both classes, namely class 1, ω_1, class 2, ω_2. The training data that is available will be from n_1 observations from class 1 and n_2 observations from class 2. Observation means that are sampled from class 1 is denoted by $\bar{x}_1$ with the mean

of observations sampled from class 2 denoted by $\bar{x}_2$. Sample size of all observations is denoted by $n = n_1 + n_2$.

Calculating the between-class variance, which for the ith class is known as S_{B_i} which shows the distance between the ith class mean, μ_i, and the total mean, μ. The main focus is maximizing the distance between the two classes.

Therefore, S_b is as follows:

$$S_b = \frac{1}{n}\sum_{i=1}^{c} n_i(\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T \tag{3.5}$$

In the Equation 3.5, $\bar{x}$ represents the overall mean and $\bar{x}_i$ represents the mean for class i, where $i = 1, .., c$.

In terms of S_{W_i}, LDA investigates for a lower dimensional space, which focuses on minimizing the within-class variance. Thus, S_w is as follows:

$$S_w = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{ni}\left(x_{ij} - \bar{x}_i\right)\left(x_{ij} - \bar{x}_i\right)^T \tag{3.6}$$

Where x_{ij} represents the ith sample in the jth class. In this study, c = 2, because of the two populations that are being focused on. Therefore, this indicates that the S_w and S_b both have dimensions consisting of p x p, and due to there being many variables this indicates that the dimensions are very large. As mentioned before, the goal of LDA is project the data into a lower dimensional space which is exactly the goal in this case which is to transform the p x p dimensions into a lower dimension. If this study consisted of multi-class LDA then then the matrix used would have been a projection matrix however with binary LDA, a projection vector w is found by solving a maximisation problem:

$$max\left\{\frac{w^T S_b w}{w^T S_w w}\right\} \tag{3.7}$$

This formula in Equation 3.7 can be reformulated as below:

$$S_w w = \hat{\lambda} S_b w \tag{3.8}$$

Where $\hat{\lambda}$ is representing the eigenvalues of the transformation of the projection vector (w). The solution to this problem can be found through the calculation of the eigenvalues which are nonzero and denoted by $\hat{\lambda}_1, \hat{\lambda}_2, \dots, \hat{\lambda}_m$ as well as the eigenvectors denoted by $\hat{e}_1, \hat{e}_2, \dots, \hat{e}_m$ of $w = S_w^{-1} S_b$, if S_w is non-singular. In the case of binary LDA, $m = c - 1 = 1$. Therefore, this provides information about the LDA space as the eigenvalues are scalar values and the eigenvectors are nonzero vectors (Tharwat, et al., 2017). The new spaces direction is shown by the eigenvectors and the eigenvalues that correspond show the magnitude of the eigenvectors as well as the scaling factor. Consequently, in terms of the LDA space, one axis is represented by one eigenvector with the associated eigenvalue indicating the robustness of this specific eigenvector. This aspect is important as the robustness of the eigenvectors demonstrates its capability to differentiate between different classes, for example the increasing of the between-class variance and the decreasing of the within-class variance of each specific class therefore relating to what was earlier states with the two conditions as to why a hyperplane is constructed. Through Fisher's method, the data is projected from p dimensions to $c - 1$ dimensions.

3.4 SUPPORT VECTOR MACHINE

SVM's are positive in the way that they are very versatile as different kernels can be used for different models, they are also very effective in high dimensional spaces where the number of dimensions is greater than the sample size it could produce accurate results. The dataset in this research project is a binary classification problem yet SVM's can also be used in regression problems unlike LDA and Logistic Regression, as well as being less vulnerable to overfitting then the latter (Srivastava, 2020). Although there are such disadvantages such as the statistical approach is low interpretability as it does not directly provide the probability estimates, as the classifier just works by putting observations above and below a hyperplane (Dhiraj, 2019). When the data points are not separated nicely it can cause the classes to overlap which does cause the approach to perform poorly.

3.4.1 Margin Maximization

Suppose we have a $i \times n$ data matrix X_i that consists of i training observations in an n-dimensional space (training set):

$$x_1 = \begin{pmatrix} x_{11} \\ . \\ . \\ . \\ x_{1n} \end{pmatrix}, \dots, x_i = \begin{pmatrix} x_{i1} \\ . \\ . \\ . \\ x_{in} \end{pmatrix} \tag{3.9}$$

Here let $x_i \in R^n$,with $i = 1, .., n$,be the i^{th} point of a set S with each x_i being an n-dimensional vector. Each x_i belongs to one of two classes that is labelled by $Y_i \in [-1, +1]$, with -1 representing the one class ω_1 and $+1$ representing the other class ω_2 (James, et al., 2013).

According to Mikhail Kanevski, 2001, in a linearly separable case set S of points x_i is given in R^2, as it is working in a two-dimensional case with: $x_i = [x_1, x_2]$. A plane is defined in geometry as to have one less dimension than a space. A hyperplane is therefore a subspace of one dimension less than its surrounding space. The objective is to establish an equation of a hyperplane that divides S leaving all points of the same class on the same side. S is linearly separable if there exists weight vector W where $W \in R^2$and bias b where $b \in R$, such that the linearly separable problem given training sample $[X_i, Y_i]$ is:

$$Y_i(W^T X_i + b) \geq +1, i = 1, \dots, n \tag{3.10}$$

The pair (W, b) can define a hyperplane with the equation:

$$(W^T X + b) = 0 \tag{3.11}$$

From Equation 3.11 of the linear discriminant function it helps define the SVM methodology of a two-class classifier which is similarly shown:

$$g(x) = w^T X + b \tag{3.12}$$

This function represents a surface that separates the feature space so that the two classes (ω_1, ω_2) lie on either side (Amarappa & Sathyanarayana, 2014). The SVM that determines the optimal linear discriminant function with a hyperplane where $w \in R^n$, $x_i \in R^n$ and $b \in R$ is written as:

$$w^T x_i + b = 0, \text{ where } Y_i = 0 \text{ for } i = 1, \dots, n \tag{3.13}$$

The points that equate to Equation 3.13 are known as the support vectors. However, in most cases the hyperplane cannot separate the feature space in two perfect classes and does create uncertainty where x_i is assigned to one of the classes arbitrarily. The distance in this instance for any x_i from the hyperplane is equal to $\frac{g(x)}{||w||}$, such that $g(x)$ result in -1 and $+1$ for the points closest to class ω_1 and ω_2 (Awad & Khanna, 2015). The equation would thus result in:

$$\frac{1}{||w||} + \frac{1}{||w||} = \frac{2}{||w||} \tag{3.14}$$

From Equation 3.14 the maximum margin between two classes during the training phase can be found by determining two support planes, a positive and negative class, which is seen as:

Positive class: $w^T x_i + b = +1,$ where $Y_i = +1$, and
Negative class: $w^T x_i + b = -1$, where $Y_i = -1$.

Stated by Quitadamo, 2017, to maximize the margin the objective function, $J(w)$, is used in the minimization of the norm for: $J(w) = \frac{1}{2}||w||^2$ such that for:

$w^T x_i + b \geq +1 \; for \; Y_i = +1$ (if x_i belongs to class ω_1) and,
$w^T x_i + b \leq -1 \; for \; Y_i = -1$ (if x_i belongs to class ω_2).

The above optimization problem can be solved by making use of Lagrange multipliers. The Lagrangian function that is used to optimize the objective function is:

$$L(w, b, \lambda) = \frac{1}{2}||w||^2 \sum_{i=1}^{n} \lambda_i (Y_i(w^T x_i + b) - 1) \tag{3.15}$$

Where λ_i's are the Lagrange multipliers and $w\ and\ b$ are considered the prime variables. Solving this optimization problem and determining $w, b\ and\ \lambda$ would result in a distinctive maximal margin (Awad & Khanna, 2015).

In the training phase the x_i's belongs to either of the two classes but in the final instance, the test phase the x_i's are assigned to one of the two classes according to which side of the hyperplane the observation lies. One of the beneficial properties of the SVM is the weight vector w only depends on the training pattern the lie on the margin, the support vectors: $w^T x_i + b = \pm 1$. Aswell as it is very straight forward to determine the bias b when given the weight vector w (Quitadamo, 2017).

3.4.2 Soft Margin: Non-Separable Case

Margin maximization is the perfect way to perform classification when the separating hyperplane exists. The problem arises when the data set is non-linearly separable, and no separating hyperplane can divide the feature space into two classes without allowing a classification error. It is best to find a hyperplane with the lowest of two errors: misclassifications and within-the-margin anomalies. The misclassification errors occur when observations (x_i's) are on the wrong side of the hyperplane. The within-the-margin anomalies occur when the observations (x_i's) are within the maximum margin even though they on the correct side of the hyperplane (Quitadamo, 2017). The generalization to the non-separable case is known support vector classifier or soft margin classifier where the SVM can classify most of the x_i's correctly but allowing the misclassification of a few in the space. Slack variables (ξ_i's) are created to allow individual training x_i's to be on the wrong side of the hyperplane or margin. These variables soften the maximum margin hence the model is referred as the soft margin SVM also known as the C-SVM. Where will $\xi_i = 0$ if x_i is correctly classified on the right side of the separating hyperplane but if x_i is misclassified then ξ_i is equal to the distance of x_i from the hyperplane. It is seen that if $\xi_i > 0$ then a within-the-margin anomaly has occurred but if $\xi_i > 1$ then a misclassification error has occurred (James, et al., 2013). The problem now is the minimization of the objective function $J(w)$ which is now:

$$J(w,\xi) = \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{n}\xi_i$$

(3.16)

With C as the box constraint which is a regularisation parameter that's size depends on the optimization goal. The sum of the slack variables ($\sum_{i=1}^{n}\xi_i$) is characterized as the overall error. C is bound to the sum of the slack variables and therefore determines the number of errors that will be tolerated known as the penalty term (Quitadamo, 2017). Yet the constraints for the optimization problem have altered to become:

$$Y_i\,[w^T x_i + b] \geq 1 - \xi_i\,, for\ i = 1, \dots, n \text{ and}$$
$$\xi_i \geq 0\,, for\ i = 1, \dots, n$$

(3.17)

From the two constraints in Equation 3.17, it can be familiarised to the equations for margin maximization such that:

$$w^T x_i + b \geq 1 - \xi_i\ for\ Y_i = +1 \text{ and,}$$
$$w^T x_i + b \leq \xi_i - 1\ for\ Y_i = -1$$

(3.18)

According to Mikhail Kanevski, 2001, the parameter C controls the trade-off between the complexity of the model and total of non-separable points. C is determined by the experimenter such that if a large value is selected it will result in a tighter margin that will cause the hyperplane to commit less errors on the training dataset, and vice versa if C is given a small value then the result would be a larger margin that would result in more violations being allowed on the training set (Awad & Khanna, 2015). A tight margin will result in a low bias with a large variance due to the classifier being highly fit to the data and the wider margin will have a higher bias and lower variance as the fit to the data is less firm. The selection of C is very important as it can lead to overfitting due to C values that are too high, and underfitting due to C values that are too low, with both the two scenarios resulting in poor performance on future data (James, et al., 2013).

Quitadamo, 2017, published that the problem with the C-SVM is there is no direct interpretation to the parameter C other then the trade-off that is explained above, therefore resulting in the relationship to be quite difficult to visualize and calculate. Another soft

margin case that can be computed is the v-SVM, which provides more interpretable parameters v and p. With the parameters $v \in [0,1]$ and $p \geq 0$ the penalty term in $J(w, \xi)$ becomes:

$$-vp + \sum_{i=1}^{n} \xi_i \tag{3.19}$$

The advantage of the v parameter is that it provides an implicit meaning and indication of the fraction of errors that can be accepted. The choice of v, for any $p \geq 0$ as p is linearly related to the margin size, acts as both an upper bound on the fraction of errors in the margin (the two misclassification errors); $[1 - \alpha < v]$ and lower bound on the fraction of errors of support vectors lying on or within the margin; $[\%SV's > v]$ (Norton, et al., 2017). The optimization problem with the chosen trade-off parameters then results as:

$$Y_i \, [w^T x_i + b] \geq p - \xi_i \, , for \; i = 1, \dots, n \text{ and,}$$
$$\xi_i \geq 0 \tag{3.20}$$

In comparison of the soft margins, the v-SVM is beneficial in the fact that the interpretation is clearer and more direct although it needs the optimization for two parameters, v and p, where C-SVM only needs one, C.

3.4.3 Kernel Trick: Non-linearly Separable Case

The soft margin classifier is a good approach in binary classification if the boundary between the two classes ($\omega_1 \; and \; \omega_2$) are linear. However, in most practical situations the problems are not linearly separable, and the soft margin is basically useless as it cannot separate the hyperplane to minimize the two classification errors and determine the two classes. In such instance kernels can be used to transform the data so that it is linearly separable by enlarging the feature space to a higher-dimensional space known as the kernel space (Awad & Khanna, 2015). In determining the function of a kernel, the symbol $\mathbb{K}$ is first explained which is used to treat the real and complex cases simultaneously. A $\mathbb{K}$-Hilbert space H is considered a real space when $\mathbb{K} = \mathbb{R}$ and is considered a complex space when $\mathbb{K} = \mathbb{C}$ (Ingo Steinwart, 2008) . Given R^n and a $\mathbb{K}$-Hilbert space exists, let $\phi: R^n \rightarrow H$ which denotes as the mapping function then function $k: R^n \times R^n \rightarrow \mathbb{K}$ is called a kernel on

R^n. To train the SVM model each input x_i is mapped to $\phi(x_i)$. As explained by (James, et al., 2013), suppose the solution to the optimization problem involves only the inner products of the x_i's and not the x_i's themselves such that the inner product of $x_i \; and \; x_i'$ is:

$$(x_i, x_i') = \sum_{j=1}^{n} x_{ij}, x_{ij}' \tag{3.21}$$

In terms of the kernel trick there must exist a function k (as explained above) so that:

$$k(x_i, x_i') = \big(\phi(x_i), \phi(x_i')\big) \text{ for all } x_i, x_i' \in R^n \tag{3.22}$$

This is beneficial part is as given the kernel k the computation of mapping function $[\phi(x_i)]$ and the feature space $[H]$ is avoided. The kernel function should be a positive semi-definite matrix and has to satisfy Mercer's theorem, which is defined in determining the kernel that all pairs of x_i's are positive and semidefinite. The function needs to reflect the relationship of the x_i's, where $k(x_i, x_i')$ will be large if the inputs are similar and small if they are different (Quitadamo, 2017). There are many different types of kernels, and the selection is very dependent on the characteristics of the dataset that is being modelled. The linear kernel is the simplest of all the types and not very useful as it basically returns the support vector classifier and is modelled as above:

$$k(x_i, x_i') = x_i . x_i' \tag{3.23}$$

If the classification problem is not linearly separable there are three more popular kernel types that are often used in SVM's; the polynomial function, the Gaussian radial basis function (RBF) and the Hyperbolic tangent (sigmoid) (Kanevski & Pozdnukhov, 2001). Let the degree $d \geq 0$ which is an integer and the offset $r \geq 0$ which is a real number (Karatzoglou, et al., 2006). The polynomial kernel is equated as:

$$k(x_i, x_i') = (\gamma(x_i . x_i') + r)^d \tag{3.24}$$

The degree (d) of the kernel determines the flexibility of the model where a high degree would allow a more flexible case. If the kernel has the lowest degree, it will essentially return a linear kernel which would not be preferred if there was a non-linear relationship between the features (Savas & Dovis, 2019). This kernel function is often used in image processing for classification problems. The Gaussian RBF is the next type that has a width σ and is defined such that:

$$k_\sigma(x_i, x_i') = \exp\{-\frac{\left||x_i - x_i'|\right|^2}{2\sigma^2}\}, \sigma \neq 0$$
$$= \exp\left(-\gamma\left||x_i - x_i'|\right|^2\right),$$

(3.25)

This function in Equation 3.25 is commonly used in practical applications of kernel methods. The gamma (γ) is inversely proportional to the width and is equal to: $\gamma\ \alpha\frac{1}{\sigma}$ (Sreenivasa, 2020).If the width is close to zero, the SVM model is at risk of overfitting meaning it would use all the training points as support vectors. Where assigning a large width value may result in underfitting and classifying all the points into one class (Savas & Dovis, 2019). Lastly the Hyperbolic tangent also known as the Sigmoid kernel has the function where most of the time the gamma, $\gamma < 0$ and the offset, $r > 0$ of:

$$k(x_i, x_i') = \tanh(\gamma(x_i.x_i') + r)$$

(3.26)

The last two kernels mentioned, the RBF and Sigmoid, both are very beneficial when there is an absence of prior knowledge in the model (Awad & Khanna, 2015). The two kernels have a similarity in kernel elements, but it is important to know that it does not directly mean they will have similar generalization performance. The RBF has been preferred due to it being harder to select suitable parameters for the sigmoid kernel (Lin & Lin, 2003). When deciding the type of kernel function the parameters that are used should be set by the experimenter, for example in the polynomial kernel the size of degree and offset must be chosen. The decision of the size of the parameters as well as the type of kernel are very important in influencing the performance of the classification problem. Grid-search is a popular technique that is often used to determine the most suitable values that would have the best influence (Quitadamo, 2017).

SVM's in the case of this research project could be very beneficial as it is a two-class classifier. With it having a very simple structure being able to classify both numerical and non-numerical data accurately. (Amarappa & Sathyanarayana, 2014).

3.5 CONCLUSION

The three methods functions are compiled differently with different assumptions. With logistic regression taking the logarithms and odds to make sure that the model is correctly specified, LDA uses a hyperplane to attempt to both minimize the variance between each class as well as to maximize the distance of the means of the two classes. Lastly the SVM is the most complex model with simply using a hyperplane and trying to maximize the margin, but in large non-linear cases such as the Two Oceans dataset must make use of soft-margin classifiers and kernel tricks to help return the two separate classes. All three methods functions will be programmed in R and analysed to see which method handles the data the best.

CHAPTER 4: DATA ANALYSIS

4.1 INTRODUCTION

That data set that is examined in the research project is based on the people who have entered Two Oceans Marathon. The Old Mutual Two Oceans is a world-renowned race that takes place in Cape Town, South Africa every year since 1970 (Africa, 2002). The data contains the information on both the people who ran the Ultra Marathon (56 km) and the Half Marathon (21.1 km) from the years 2012 to 2015. The dataset contains all the information on the people who signed up for the Two Oceans between these years, but the project will only research from 2013 onwards due to the 2012 data having a substantial amount of missing data in the variables that have been selected.

4.2 SUMMARY STATISTICS

The number of observations exceed the number of variables $(n > p)$ and therefore the dataset is determined as narrow. The project focuses on two questions: What are the differences in the starters versus the non-starters (n_1) and the differences in finishers versus the non-finishers (n_2) in the Two Oceans? In the three years from 2013 to 2015 and after removing any missing data there were 60 425 entrants $(n_1 = 60425)$ with 49230 of the entrants' actual starters and 11 195 of them did not start the race.

Table 4.1: Summary of the Two Oceans Marathon DNS Data:

Starters	Non-Starters	Total Entrants
49230	11195	60425

From the dataset our response (dependent) variable, y_1, represents the starters and non-starters as these results are the focus of the research and is the outcome determined by the various explanatory (independent) variables. The responses are made up of 0's and 1's with the result of 0 if the entrant started the race and 1 if they did not. The Two Oceans data has then two population classes $(\omega = 2)$; 49230 observations in class 1 (ω_1) which are represented by 0's in response variable y and 11195 observations in class 2 (ω_2) which are represented by 1's in response variable y. The next part of the research project will look at those who started race and whether they finished or not. From the starters data it is known

a total of 49230 people start running the race ($n_2 = 49230$) with 48386 of the runners' finishing the race and only 844 not finishing the race.

Table 4.2: Summary of the Two Oceans Marathon DNF Data:

Finisher	Non-Finisher	Total Runners
48386	844	49230

The dependent variable, y_2, is denoted by the finishers and non-finishers and once again the responses are made of 0's and 1's, which indicate the two sample classes ($\omega = 2$). The runners that finished take up a substantial proportion of the data with 48386 out of the total 49230 are represented by 0's in class 1 (ω_1). The 844 participants that did not finish are represented by 1's in class 2 (ω_2). The whole data set includes 1144 predictor variables ($p = 1144$) but many of them were non-significant to the analysis especially for the difference between starters and non-starters. The predictor variables are cut down to substantially to 9 significant variables ($p' = 9$) that influence both response variables ($y_1$and y_2). The predictors are:

Table 4.3: Summary of the Predictor Variables:

Variable	Label	Class	Levels	Range
The age of the entrant:	(age)	Numeric		[16 – 86]
The age category the entrant falls in:	(agecat)	Factor	1: less than 30 years, 2: 31 to 40 years, 3: 41 to 50 years, 4: greater than 51 years.	
The average weekly distance the entrant ran in the last 12 months (kilometres per week):	(avdist)	Numeric		[4 – 160]
The gender of the entrant:	(gender)	Factor	1: Male 2: Female	

The type of race the entrant signed up for:	(racetype)	Factor	1: Ultra Marathon 2: Half Marathon	
The race year the entrant signed up for:	(year)	Factor	1: 2013 2: 2014 3: 2015	
Had the entrant been prescribed medication to treat chronic medical conditions/injuries.	(presmed)	Factor	0: No 1: Yes	
Had the entrant consulted with a medical doctor in the last 12 months to obtain medical clearance for endurance running.	(consdoc)	Factor	0: No 1: Yes	
If yes to (consdoc) above, did the medical practitioner clear the entrant with any specific advice for participating in endurance running?	(cleardoc)	Factor	0: Did not consult a doctor 1: The doctor did not give clearance to run 2: The doctor gave clearance to run but with some restrictions 3: The doctor gave clearance to run with no restrictions	

The variables were run with the data in the models to test if there was any significant difference in the medical questions having any influence on whether the entrant would start the race or finish. The results that were returned showed that the three medical questions did not have much effect on the outcome, such that for the “cleardoc” variable just over 50%

of the people that got cleared but with some restrictions went on to start the race, but almost the same result occurred those that got clearance with no restrictions and those that had not been cleared.

4.2.1 Training and Test data

Splitting data into a training and test set is an important part of machine learning, especially in terms of classification procedures. Known as the train-test split procedure which is used to evaluate the performance of the machine learning algorithms such as the classification procedures at hand, as they are used for predictions on the data that was not used to train the model, thus the test set. The procedure comprises of the dataset being divided into two subsets with the first subset being the training set and used to fit the model with the second subset being referred to as the test set which as stated earlier, is not used to train the model but rather the input element from the dataset is provided to the model being tested, therefore predictions are made and compared to the expected values that were generated. An important condition for the dataset is that it is large enough that when it is split up in the ratio decided, as in the case of this research task the data is split up 80/20 with the training set being 80% and the test set being 20%. The importance is in regard to the train and test sets being large enough to provide an appropriate representation of the data as a whole, especially with the test set large enough to return statistical meaningful results.

The problem with a dataset that is not large enough entails that during the training procedure there will not be sufficient data for the model to learn an efficient mapping of inputs to outputs (Brownlee, 2020). As well as not sufficient amount of data in the test set efficiently assess the model's performance. The projected performance could swing either way in terms of being overfit or underfit. An alternative method that operates to the same efficiency is the k-fold cross-validation set approach. The most commonly used split percentages involve the 80/20, 70/30, 67/33 and 50/50.

4.2.2 k-Fold Cross Validation

Cross-validation is a resampling procedure used on a data set to evaluate a machine learning model. The procedure involves a parameter k, which signifies the number of the groups that the data set is divided into. This is the reason for the name of k-fold cross-validation (CV) as well as the number of groups selected can be used in place of the name, such as for the research task at hand, k = 5 was chosen, therefore a 5-fold CV is being

used. The reasoning for a value of 5 being chosen as the value for k is because values such as 5 or 10 are very popular as these values have been shown to produce test error rate estimates that do not usually suffer from a data set that consists of high bias or high variance. It is also assumed that selecting a lower k value can lead to results being more biased and therefore less suitable with a larger value of k being less biased but can suffer from high variance. The aim behind the k-fold CV is to estimate the capability of a machine learning model, such as the classification models that have been put forward in this research task, on unseen data. K-Fold CV is useful when a data set is typically large as it will have high bias and can be seen as quite simple for a researcher to understand as well as results usually less biased estimation of the model at the hand whereas a normal straight train/test split does not always. The procedure roughly involves five steps, with the first step consisting of the data set being randomly divided into k-subsets and in the case of this task, 5 subsets. With the second step consisting of the one subset being kept while the other subsets are used for training of the model. The third step involves testing the model on the subset that was kept aside during the first step and taking note of the prediction error. The fourth step is repeated until each of the k subsets has served its purpose and the prediction error is recorded from each. The fifth and final step involves the computation of all the k errors that were recorded, and the average is calculated therefore the k-fold CV is a robust procedure for estimating the accuracy of a chosen model.

4.2.3 Accuracy, Sensitivity and Specificity

There are several terms that are commonly used along with the description of sensitivity, specificity, and accuracy. They are true positive (TP), true negative (TN), false negative (FN), and false positive (FP).

In reference to the dataset whether a person did not start (DNS) or did not finish (DNF) the race, if an entrant is proven to not start or to not finish the race and the predicted test also indicates the entrant did not start or did not finish then the result of the predicted test is considered true positive (TP). Similarly, if an entrant is proven to start or to finish the race and the predicted test suggests the entrant did start or did finish as well, then the test will result in true negative (TN). Both true positive and true negative suggest a consistent result between the predicted test and the observed condition (also called standard of truth).
Although the tests are not always perfect, if the predicted test indicates the person did not start or did not finish the race but they in fact did, the test result is false positive (FP). Likewise, if the test suggests the entrant did start or did finish when they in fact did not then

the predicted test would result in a false negative (FN). Both false positive and false negative indicate that the test results are opposite to the actual condition (Zhu, et al., 2010). Sensitivity represents the proportion of the true positives correctly identified by the predicted test, which basically portrays how good the test is at detecting an entrant did not start or did not finish the race given the person did not start or did not finish the race. Sensitivity is also referred to as true positive rate (TPR) and is equated by the following formula:

$$\text{Sensitivity} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{4.1}$$

The numerical value that results from Equation 4.1 is the probability the predicted test identifies the entrants who do in fact not start or not finish the race. The higher the value the better, as it results in a better chance the test results in a true positive. Specificity thus represents the proportion of the true negatives identified correctly by the predicted test, which implies how good the test is at identifying that an entrant did start or did finish the race given the runner did start or finish, the normal (negative) condition (Shabani, et al., 2018). The formula for specificity has direct relation to the formula of the false positive rate (FPR), which are given by:

$$\text{Specifictiy} = \frac{\text{TN}}{(\text{TN} + \text{FP})} \tag{4.2}$$

$$\text{where FPR} = 1 - \text{Specificity} = \frac{\text{FP}}{(\text{FP} + \text{TN})} \tag{4.3}$$

The value that results from the formula in Equation 4.2, is the probability the test determines a result without giving false positive results. With $(TP + FN)$ the sum of all positive conditions, and $(TN + FP)$ the sum of all the negative conditions. It is significant to understand that sensitivity and specificity are independent of one another, thus each having no effect on the outcome of the other. Both factors are indeed important, and a great test would be one that results in a high sensitivity and specificity, thus being able to predict both true values (positive and negative) well. The proportion of predicting the true values in the population is known as the accuracy. It essentially measures the degree of true positives and negatives of a predicted test on a condition (Zhu, et al., 2010). The formula is given by adding the sensitivity and specificity:

$$\text{Accuracy} = \frac{(\text{TN} + \text{TP})}{(\text{TN} + \text{TP} + \text{FN} + \text{FP})} \tag{4.4}$$

Table 4.4: Confusion Matrix for DNS

	Actual	
Predicted	Did Not Start (Train Positive)	Did Start (Train Negative)
Did Not Start (Test Positive)	True Positive (TP)	False Positive (FP)
Did Start (Test Negative)	False Negative (FP)	True Negative (TN)
	Sensitivity: $= \frac{\text{True Positive}}{\text{Train Positive}}$	**Specificity:** $= \frac{\text{True Negative}}{\text{Train Negative}}$

4.2.4 The ROC and AUC

Regarding machine learning and focusing on classification technique such as the three that consist of Logistic Regression, LDA and SVM that are specific to this study, the measurement of the performance of these three different methods is of high importance. Analyzing, interpreting, and visualizing the performance of the classification problem is done with the Receiver Operating Characteristics (ROC) as well Area Under Curve (AUC) which is also known as Area Under the Receiver Operating Characteristics (AUROC). AUC is one of the most commonly used techniques as well as a reliable evaluation metric for analyzing a classification model's performance which is what the main focus of this research task compels.

The AUC – ROC curve helps the researcher studying understand how well the classification technique implemented is performing, therefore it is a performance measurement for the classification techniques at different threshold settings. The probability curve, another name for the ROC and level of separability is represented by the AUC, therefore this entails that

the AUC shows how well the model being used is able to distinguish between the two classes which are in our data, if the entrant started the race versus did not start as well as if the entrant finished the race versus if the entrant did not finish. In other words, the higher the AUC, the better the model is at predicting the correct classes such as predicting a 0 class as 0 and a 1 class as 1. If the AUC of a tested model is 1 then the model can perfectly predict the outcome and the level of separability is excellent, on the other hand if a model has an AUC of 0 then the model is predicting 0's as 1's and 1's as 0's. A model with an AUC of 0.5 is considered to have zero class separation and cannot distinguish between 0's and 1's, the reason for the referral to 0's and 1's is due to the fact of the research task focusing on binary classification.

To add to this, when referring specifically to the plot of the ROC curve, the y-axis that consists of the sensitivity (TPR) versus the x-axis that is 1 - specificity (FPR) across different cut-offs, generates the ROC curve. With an ROC curve that is generally located more to the left-hand corner near the upper side generally corresponds progressively greater discriminant capacity of diagnostic tests (Hajian-Tilaki, 2013). In regards to what was stated earlier on with a model with an AUC of 0.5, this is the same when examining and ROC curve that lies on the diagonal which shows the performance of a classification model that is no better than a diagnostic test taking a random guess to which class the observation belongs to, therefore is unrelated to the true status of whether the entrant started the marathon or not, as well as if the entrant finished the marathon or not. The AUC therefore summarizes the whole area underneath the plotted ROC curve and is an effective and combined measure of sensitivity and specificity that describes the legitimacy of the classification models that were tested.

In terms of comparing AUC to accuracy, AUC is preferred over accuracy when the classification is binary as one of the goals of AUC is to deal with scenarios where the sample that is being used has a very skewed distribution and there is a danger of overfitting to one of the classes. As well as with AUC, it is a single metric facilitating comparison between tests without the necessity of manipulating sensitivity and specificity. AUC is very popular when the circumstances are leaning towards a skewed distribution as the class sizes are balanced before taken into account on the graph, as for accuracy, if 95% of the objects are in the same class this will lead to a high accuracy of around 95%. This can be very misleading for a researcher when analyzing the given output. AUC also is known to measure overall diagnostic performance as AUC is averaged across all diagnostic thresholds (Halligan, et al., 2015). AUC is also known for taking into account different thresholds therefore accounting for different readers as well as the curves are simple to

understand and compare from a visual perspective. Issues of misclassification which can be considered a problem for AUC is argued to only be taken into account once the intrinsic performance of the classification method is known.

4.3 KERNEL DECISION

To determine the best SVM model a type of kernel needs to be selected to transform the data appropriately. The four types of kernels that will be tested for the model are the linear function, the radial basis function (RBF), the polynomial function and the sigmoid function. Although there are many types of different kernels these four have been selected due to the history of them being the better ones to use for this type of classification dataset with a large number of observations. A tuning framework is used on the different kernels to establish the best fixed hyper-parameters to use for either one. There are several hyper-parameters the kernels use, such as the cost of the constraints, violation C which is known as the regularization parameter and is needed for all four kernels, the gamma parameter is used for all except the linear kernel, the default gamma value is $\frac{1}{data\ dimension}$, therefore the hypothesis for the best gamma value is 0.1 due to the 10 independent variables use for both datasets. The offset (coef0) parameter is needed for both the polynomial and sigmoid function and lastly the polynomial function also includes the degree parameter. The tune framework basically runs a combination of different hyper-parameters to determine the best set which would result in the lowest error rate.

Table 4.5: Tuning Framework for the Different Hyper-Parameters

	Tuning Values:	Linear	Radial	Polynomial	Sigmoid
Cost (C)	$(0.5, 1,2, 3)$	✓	✓	✓	✓
Gamma (γ)	$(0.001, 0.01, 0.1, 1)$		✓	✓	✓
Offset (coef0)	$(0, 1, 2)$			✓	✓
Degree	$(2, 3, 4)$			✓	

Due to the dataset having so many observations a few of the models were unable to run the function and therefore a random sample of 5000 rows was used in the tuning framework. The results obtained were very similar among the different kernels and different parameter combinations.

Table 4.6: Results of Best Hyper-Parameters

	Classification Error (DNS):	Classification Error (DNF):	Optimal Parameters:
Linear	0.198	0.016	$C = 0.5$
Radial	0.198	0.016	$C = 0.5, \gamma = 0.1$
Polynomial	0.198	0.016	$C = 0.5, \quad \gamma = 0.1, coef0 = 0, degree = 2$
Sigmoid	0.198	0.016	$C = 0.5, \quad \gamma = 0.1, coef0 = 0$

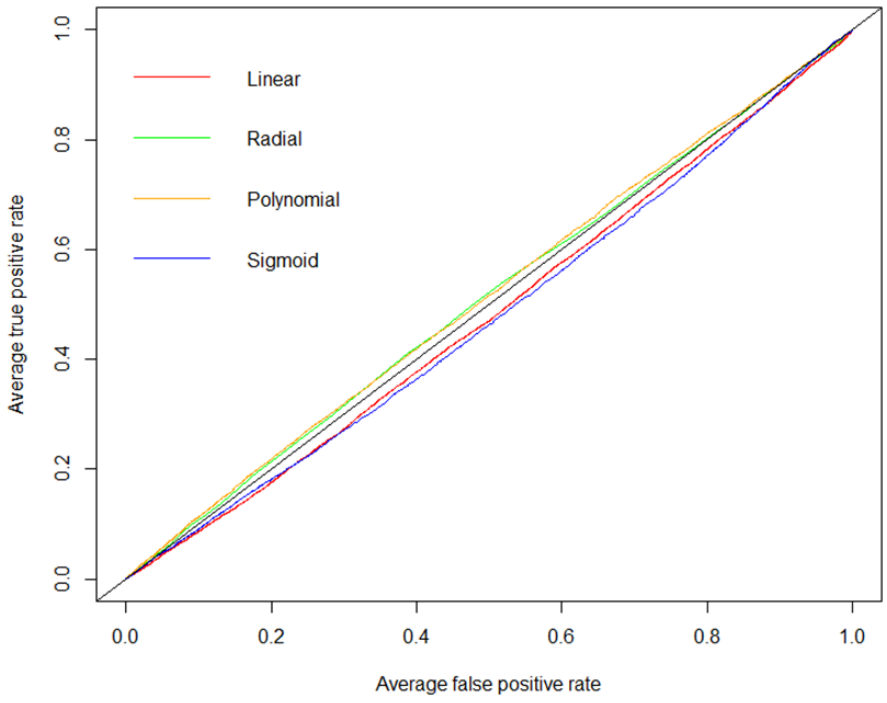
The identical classification errors portray that the different types of kernels and hyper-parameters seem to not really have any significant effect on the dataset in terms of the other types. The main reason may be due to the large number of observations and the highly imbalanced classes, the did not start (dns) and did not finish (dnf) dependent variables. Due to the tune framework being unsuccessful in the selection of the kernel to determine the best hyperplane model, each kernel would be run in the SVM model with their ideal fixed parameter combinations and the result with the best ROC curve analysis and accuracy would be selected. The degree result for the polynomial kernel of 2 is higher enough to influence the non-linearly separable case. The gamma result of 0.1 as said earlier corresponds to the width inversely and would result in a width size of 10. The width value is quite large and could risk assigning the points into one class, this is expected due to the classes being imbalanced. The only benefit does come for the sigmoid kernel as suitable parameters were able to be selected for it and the kernel may be used in the SVM model. The reason why these analyses may work better is that the classification error is based on one particular cut-off, where the ROC curve attempts all the cut-offs, plotting both the sensitivity and the specificity (Nanda, et al., 2018).

In the selection of the best kernel the k-fold cross-validation (CV) method is used. Although there are a few more cross-validation approaches, the k-fold CV approach will be used throughout the classification analysis with k=5, which partitions the data into 5 equally sized segments commonly known as 'folds'. The performance of each kernel in predicting the model will be decided using the mean performance metrics, area under the curve (AUC), accuracy (atmathew, 2015). Both the best AUC and accuracy are defined by the largest resulting value.

Table 4.7: Performance of Kernels for the DNS Variable

Kernel:	AUC:	Accuracy:	Sensitivity:	Specificity:
Linear	0.4776912	0.8147290	0.0000000	1.0000000
Radial	0.5099363	0.8147290	0.0000000	1.0000000
Polynomial	0.5143934	0.8147290	0.0000000	1.0000000
Sigmoid	0.4755014	0.7105337	0.1987536	0.8269162

Figure 4.1: ROC curve for the four different kernels for the variable DNS



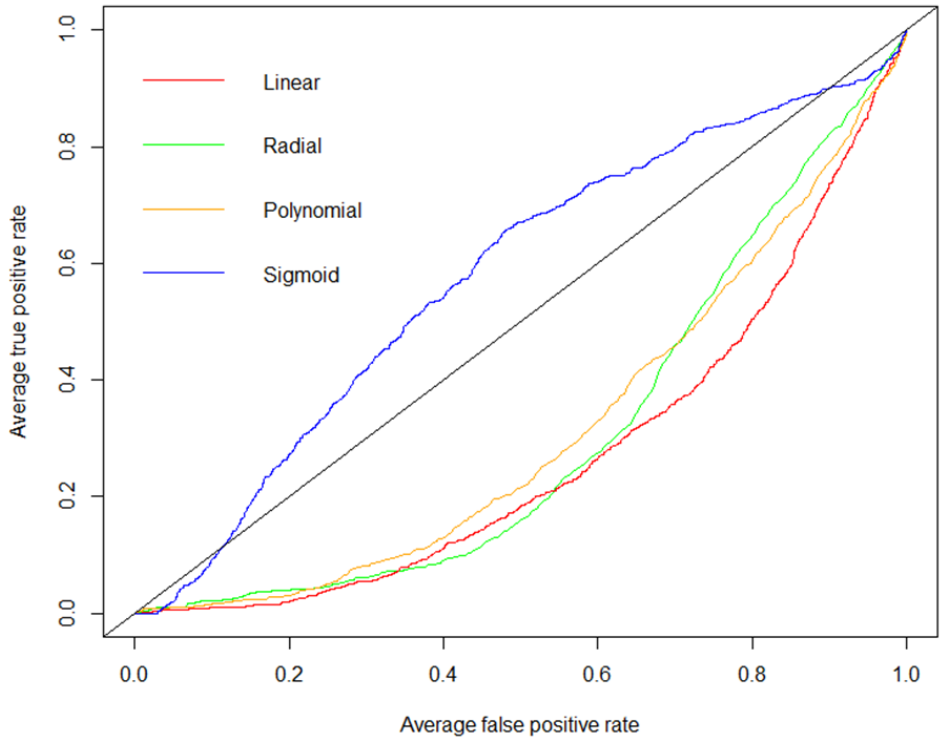
As seen from the Table 4.7, the SVM with the best accuracy rate is 0.8147, that entails that 81.47% of the time the test result is accurate, which is obtained using the linear, radial, and polynomial kernels. The reason the three kernels have the same accuracy is due to the sensitivity and specificity being identical between them. The sensitivity values of 0 fundamentally means that the predicted test identifies a probability of 0% of the entrants

that did not start the race given the entrant did not start the race, where the specificity value of 1 essentially means the predicted test identifies 100% of the entrants that started the race given the entrant started the race. The sigmoid kernel does result in a better sensitivity with a probability of 19.9% of the test correctly predicting the entrant did not start the marathon given the entrant did not start, yet the specificity and more importantly the accuracy rate is lower than the other three kernels. The best AUC is therefore the performance metric used to determine the best kernel function out of the three. The polynomial kernel has the highest AUC value of 0.514 and is therefore the kernel function used for the SVM model in the analysis of the dependent variable (y_1).

Table 4.8: Performance of Kernels for the DNF Variable

Kernel:	AUC:	Accuracy:	Sensitivity:	Specificity:
Linear	0.2597994	0.9828560	0.0000000	1.0000000
Radial	0.3043581	0.9828560	0.0000000	1.0000000
Polynomial	0.3083284	0.9828560	0.0000000	1.0000000
Sigmoid	0.57626398	0.97158237	0.03920032	0.98786895

Figure 4.2: ROC curve for the four different kernels for the variable DNF



The SVM with the best accuracy from the table is once again from using the linear, the radial, and the polynomial kernel with rate of 0.9829, which is excellent as it means that 98.29% of the time the predicted test result is accurate. The sigmoid kernel also results in a very high and impressive accuracy of 97.16% as well as returning an AUC value of 0.5763. This is the reason the sigmoid kernel is used for the SVM model for the DNF dependent variable, as its AUC value is almost double the values of the other three kernels, with the polynomial kernel having the second largest value of only 0.3083.

In comparison between the two dependent variables, $y_1$and y_2, it is seen that the kernels have much better accuracies for y_2 (DNF) this is because the classes are way more imbalanced than the classes for y_1(DNS). As seen in Table 2, 48386 entrants that started the race in fact finished where only 844 of the entrants did not. The reason for the high accuracies is the kernels were able to predict the true values of entrants finishing the race because such a large proportion of 98.29% them did. This proportion is identical the accuracy rate of the linear, radial, and polynomial kernels due to the sensitivity being equal to 0 and the specificity being equal to 1 for the kernels. This basically means that the kernels were able to predict the 48386 finishers 100% of the time but predicted the 844 runners that did not finish 0% of the time. Similarly, from Table 1 it is seen that 49230 entrants started from a total of 60425 resulting in a proportion of 81.47% which is the same as the accuracy rate for the linear, radial, and polynomial kernels for the dependent variable y_1. The reason is equivalent to the DNF variable where the sensitivity is 0 and the sensitivity is 1. It can be seen why the sigmoid kernel for both dependent variables has the only different accuracy rates, due it being able to predict some percentage of the entrants that did not start or did not finish, the sensitivity.

4.4 ANALYSIS OF IMBALANCED MODELS

Table 4.9: Comparison of performance on the three classification methods for DNS variable.

	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.5583007	0.1915598	0.5207700	0.5690142	0.8084402
LDA	0.5580245	0.1927182	0.5241794	0.5663646	0.8072818

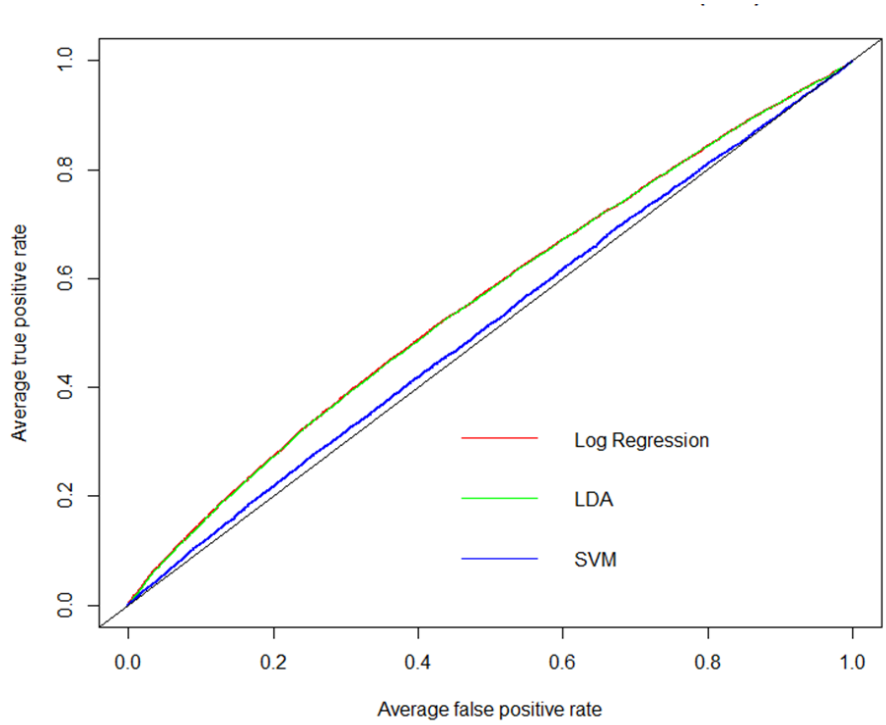
SVM using Polynomial kernel	0.5143934	0.1852710	0.0000000	1.0000000	0.8147290

When analyzing the output from the comparisons of the three classification methods, firstly focusing on sensitivity and specificity. From the table, it can be seen that the logistic regression model has a sensitivity of 0.521 and a specificity of 0.569 which entails that the logistic model has 52.1% chance of correctly predicting that an entrant does not start, given in fact that the specific entrant did not start the race. As well as specificity, the logistic model has a 56.9% chance of correctly predicting of correctly predicting that an entrant does in fact start the race, given they started the race. Moving on to the next classification model, with sensitivity, it can be seen that LDA has a 52.4% chance of correctly predicting that an entrant does not start, given that the entrant does not start. The specificity of LDA, shows that the model has a 56.6% chance of correctly predicting that an entrant does start, given that they do start. The SVM using a polynomial kernel with the fixed hyper-parameters, is seen to have a sensitivity of 0.000 therefore shows that this model has 0% chance of correctly predicting that an entrant does not start, given that they do not start. As well with a specificity of 1.000, shows that this model has a 100% chance of predicting that an entrant does start, when in fact they do start the race. The reason for these drastic contrasts in sensitivity and specificity for the SVM model is due to the issue of SVM not being able to deal with highly imbalanced data, even though it is effective for balanced classification. Reasoning behind the issue is the algorithm behind the SVM finds a hyperplane decision boundary that best splits the two classes; however the split is weakened by the fact that the margin allows for points to be misclassified (Brownlee, 2020). By default, the margin supports the majority class on imbalanced data set such as the one in this specific research task.

To add to this, when observing the accuracy of the three models it can be seen that with the logistic model with an accuracy of 0.808 entails that the model is accurate 80.8% of the time in terms of the test result. This can be seen to be slightly better than the accuracy of LDA which is 0.807 which shows that LDA is accurate about the test result 80.7% of the time, although very similar to the logistic model. The SVM model has the highest accuracy with a value of 0.814 therefore correct about the test result 81.4% of the time, although a slightly misleading value as the data is highly imbalanced therefore this must be taken into account as it will skew the accuracy to a favorable value when this should not always be

the case. This directly effects the test error as for logistic, LDA and SVM the test error rates are 19.2%, 19.3% and 18.5% respectively. Therefore, this shows SVM with the lowest test error rate however the misleading accuracy and the effect it has on the test error rate must be kept in mind.

Figure 4.3: ROC Curve for the three different classification methods for the imbalanced variable DNS.



Analyzing the above ROC of the three different classification methods where the main focus is on the AUC of each classification method where visually it can be seen that SVM is performing the worse out the three methods as it can be seen as the lowest curve out of the three. Referring back to the table, the AUC is known to be 0.514 for SVM. The main goal of this analysis is to find the classification method with the highest AUC. Examining the curves for logistic regression and LDA can be seen to be quite difficult to distinguish between the two for which one has a higher AUC however referring back to the table it is known that the AUC for logistic regression is 0.558 and the AUC for LDA is also 0.558. When extending the decimal places, the AUC for logistic regression is slightly better than that of LDA however to a minor extent.

Table 4.10: Comparison of performance on the three classification methods for the DNF variable.

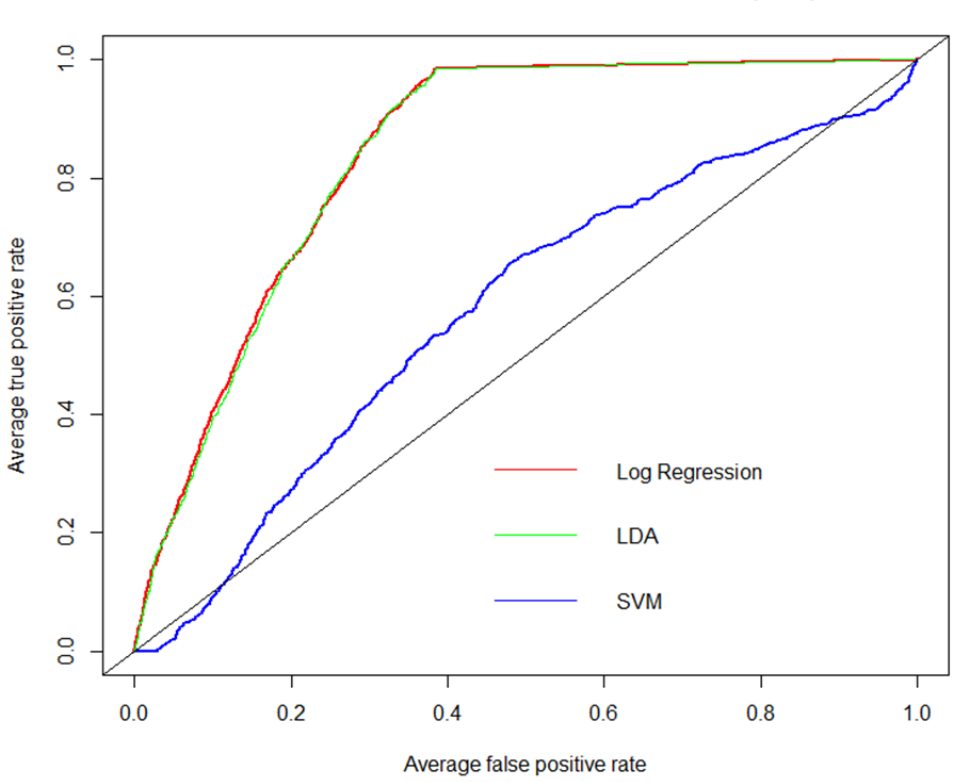
	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.84247256	0.03300833	0.85855879	0.71055120	0.96699167
LDA	0.84105307	0.03402397	0.84551533	0.71797865	0.96597603
SVM using Sigmoid kernel	0.57626398	0.02841763	0.03920032	0.98786895	0.97158237

When analyzing the output from the comparisons of the three classification methods, again, firstly focusing on sensitivity and specificity with taking the logistic model into account it can be seen that the sensitivity and specificity is 0.859 and 0.711 respectively. This entails that the logistic model has an 85.9% chance of correctly predicting that the runner does not finish the race, given they did not finish the race as well as the model has a 71.1% chance of correctly predicting the runner finished the race, given that the runner finished the race. The LDA model has similar sensitivity and specificity with values consisting of 0.846 and 0.718 respectively, therefore has an 84.6% chance of correctly predicting that a runner does not finish the race, given they did in fact not finish the race. A 71.8% chance of correctly predicting a runner does finish the race, when they did essentially finish the race. The similar results come to a halt when analysis is brought upon the SVM using a sigmoid kernel, where the sensitivity has a value of 0.039 and a specificity value of 0.988 therefore a the SVM model has a 3.92% chance of correctly predicting that a runner does not finish the race when it is given the runner did not complete the full race. On the other side of the scale, the SVM model has a 98.8% chance of correctly predicting the runner does finish the marathon given they did indeed finish the marathon. The reasoning for this is simple, the highly imbalanced data again leads to the SVM model classifying the odds of predicting the runner to finish to such a high degree because the percentage of runners that did in fact finish is so highly skewed.

Moving on to the analysis of the accuracy of each model, with the logistic model it can be seen that the value is 0.967 therefore the model is accurate 96.7% of the time in terms of the test result. The accuracy can be seen to be slightly higher than that of LDA, where the accuracy has a value for LDA has a value 0.966 therefore showing that the model is

accurate 96.6% of the time when it comes to the test result. Both of these accuracies are lower than that of the SVM where the value is of 0.972, therefore showing the SVM model is accurate about the test result 97.2% of the time. All three accuracies are very elevated with SVM being slightly higher however the inflation due to SVM not being entirely correct for handling large imbalanced data sets must be kept in mind. Again, directly relating to the test error for the three classification methods of logistic, LDA and SVM where the percentages consist of 3.3%, 3.4% and 2.8% respectively.

Figure 4.4: ROC curve for the three different classification methods for the imbalanced variable DNF



With analyzing the above ROC curve for the three different classification methods for the variable DNF, where the AUC is the main focus in this section. Visibly, it is simple to see that the worst performing classification method in terms of AUC is the SVM with the sigmoid kernel as it is the lowest curve out of the three. There is a point near the bottom left as well as the bottom right where the SVM curve is below the random diagonal line therefore meaning there is section in the SVM where there is misclassification as in a portion of 0's are being classified as 1's and a portion of 1's are being classified as 0's because of the curve going below the gradient of 0.5 in two small sections however the total AUC is 0.576. With the main goal of finding the highest AUC, it can be seen that the two competing for this is again, the logistic model and the LDA model. With the naked eye it is quite tricky to differentiate between the two but referring back to the table it cane noted that the AUC for logistic is 0.842 as well as the AUC for the LDA model being 0.841, therefore very similar however with logistic being regarded as having a slightly higher value but to a minor extent. Both these two models can be regarded having very good AUC's.

4.5 ANALYSIS OF INDEPENDENT AND BALANCED MODELS

An importance assumption of logistic regression is multicollinearity, therefore logistic regression requires the observations to be independent of each other. The observations should not come from repeated measures, also requiring there to be slight multicollinearity or none at all. To deal with this issue as the results above have had repeated measures however the belief is that the results are still valid to a large extent as it is believed that logistic regression will still be the best performer when removing multicollinearity. To prove this, each of the years 2013, 2014 and 2015 have been split and modeled individually and the average of the three years has been taken. Therefore, removing any sort of observations dependent of each other and a similar procedure will be run however on each year separately.

The data set that has been used throughout this research task, can be classified as imbalanced, especially the section consisting of the DNF variable as what has been explained before such that a large portion of entrants finish the race compared to the ones that don't. Therefore, has led to focus being applied to AUC over accuracy when comparing the three different classification methods as accuracy has been inflated due to this complication. To deal with this, one common approach for dealing with an imbalanced

dataset is a resampling technique, where it consists of oversampling and under-sampling. Over-sampling is preferred to under-sampling, as over-sampling, the minority class is repeated multiple times and with under-sampling, an item being removed may have been holding important information (Satpathy, 2020). The technique that was implemented in the research task to deal with the imbalanced data set is a program known as Synthetic Minority Oversampling Technique (SMOTE). The function deals with the issues caused by random overfitting or underfitting. It addresses imbalanced class problems by either; over-sampling the minority class, under-sampling the majority class, synthesis of new minority class instances or tweaking the cost function. In the instance of the assignment the minority class will be over-sampled by increasing the weights of the DNF and DNS variables (Data, 2021). This will allow accuracy to become an important statistic when comparing the three classification methods.

4.5.1 The Did Not Finish Variable

With analyzing the DNF variable for each year separately, therefore following the independence assumption and then taking all three years as well as using the SMOTE function in R to balance the data set. An average was calculated of each statistic for all three years and tabulated as above therefore to allow for simple interpretation.

Table 4.11: Average performance of the three classification methods for the DNF variable.

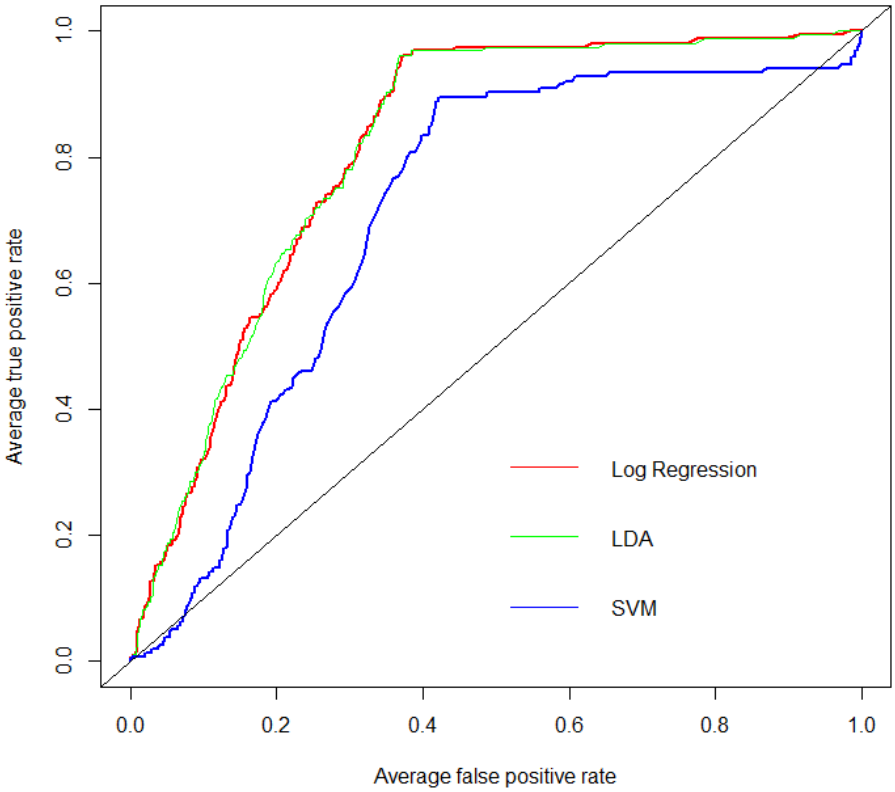
	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.83091895	0.07599082	0.8632855	0.7074967	0.92400918
LDA	0.83110712	0.08166681	0.8761844	0.699779	0.91833319
SVM (Sigmoid)	0.72583830	0.3315951	0.6962716	0.6681426	0.66840487

As can be seen from Table 4.11 compared to Table 4.10, the SVM's sensitivity and specificity has altered significantly. The large increase in sensitivity to a value of 0.6963 means that the model is now able to predict 69.6% of the time that the runner will not finish the race given that the runner did not finish. This increase compared to the 3.92% from the model on the imbalanced data is a great improvement in the model's prediction. The specificity of 66.8% was decrease from the imbalanced 98.8%. This is due to the new model

portraying much more balanced weights which does allow for a significant and fair test. The SVM is now able to classify the odds to whether a runner did finish or did not more equally. The sensitivity and specificity results for logistic regression and LDA are very similar to the imbalanced data this is due to the class sizes being balanced before they are run through the models in both cases, thus not having such considerable effect compared to the SVM model.

When comparing the accuracy of the three methods to Table 4.11, what would be expected is for accuracy to decrease as now the data is no longer imbalanced therefore the accuracy can no longer considered to be misleading. The accuracy for logistic regression and LDA both decreased by margins of around 4%, with LDA decreasing slightly more leaving logistic regression with a higher accuracy of around 92.4% compared to LDA's 91.8%. Logistic regression has a better chance of predicting the correct test result. SVM succumbed to a significant drop in accuracy and with a balanced data set consequently the misleading accuracy of around 97.2% has dropped to 66.8% and SVM's concern with dealing with large datasets becomes apparent. The AUC for both logistic and LDA decreased again, but to a much smaller margin of only around 1% with values for both being around 83.1% but when looking into more decimals, LDA is seen to be slightly higher however to a miniscule margin. AUC for the SVM model was exposed to a large increase as it went from 57.6% to 72.6%, therefore predicting 0's as 0's and 1's as 1's to a better extent than previously however still not to the effect of logistic and LDA. The test error for the logistic, LDA and SVM are 7.6%, 8.2% and 33.2% respectively thus with SVM being significantly higher and logistic with the lowest test error.

Figure 4.5: ROC curve for the three different classification methods for the balanced 2013 variable DNF.



From the analysis of Figure 4.5, the major difference noticed compared to Figure 4.4 is the much larger AUC of the SVM, although it is still the weakest model. The logistic regression and LDA line seem to follow similar paths with the LDA even having the better AUC from the analysis. Although due to the data now being balanced the accuracy rate is the performance metric that judges the best classification method which is the logistic regression.

4.5.2 The Did Not Start Variable

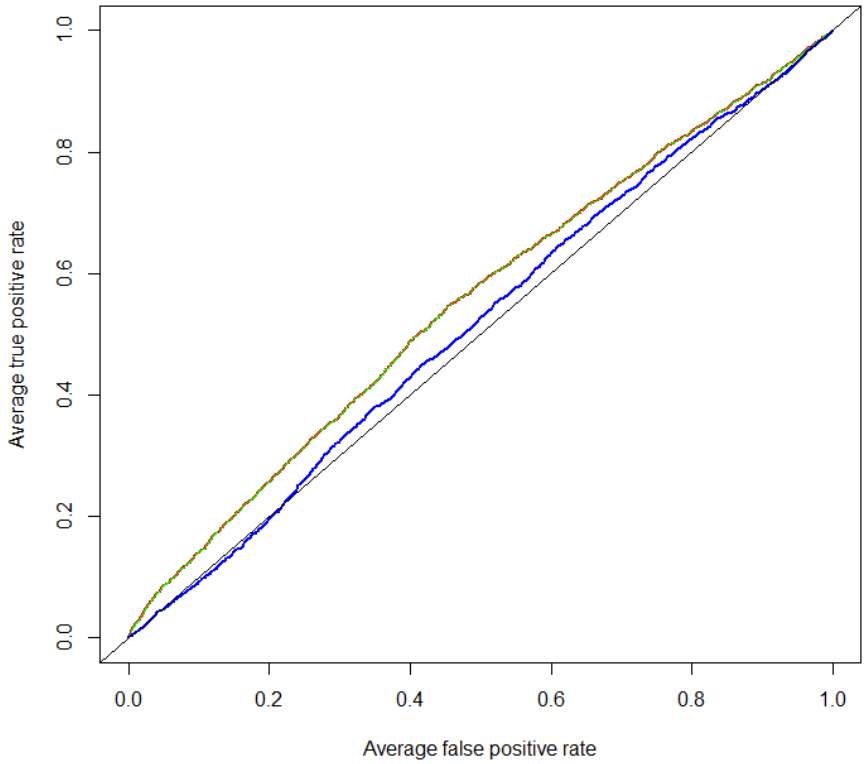
Using the Synthetic Minority Oversampling Technique by over-sampling the minority class of the DNS variable the results for the 2013 year are analyzed.

Table 4.12: The performance of the three classification methods for the 2013 DNS variable.

	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.5514010	0.2243088	0.5343813	0.5635383	0.7756912
LDA	0.5514097	0.2266562	0.5350470	0.5625644	0.7733438
SVM	0.5109437	0.4877667	0.5009967	0.5150066	0.5122333

In Table 4.12, the AUC results for all three classification methods are very similar in comparison to the results from the models that used the imbalanced data. The LDA does have a fractionally larger AUC value but as stated earlier the accuracy rate is the best performance metric which logistic regression does have, being 77.6% of the time accurate with LDA just less with 77.3%. Reversing with the accuracy rates the logistic regression model does produce the lowest test error of 22.4%. The sensitivity and specificity for the SVM model did change to both values being around 50%. This is obviously due to the minority class values receiving increased weight values, so the data is imbalanced. However due to the data for the DNS variable being balanced 80/20, with 81.5% of the people starting and 18.5% not starting, compared to the high imbalanced DNF variable having 98.3% of the people finishing and 1.7% not finishing. It was deemed unnecessary to go further with the balancing of the DNS variable data and the original data was seen to be significant for the analysis.

Figure 4.6: ROC curve for the three different classification methods for the balanced 2013 variable DNS.



4.6 CONCLUSION

From the analysis of the different models, it is determined that logistic regression is the best method out of the three. The independency assumption raised problems with the original data, although when the data was restructured the method still performed the best. The LDA results were very close, and the method still handled the data well, yet it was interesting to see the results of the SVM model and how badly it struggled to model against imbalanced data. The three different methods have their own benefits in different ways and was fascinating to see the results when compared in analysis.

CHAPTER 5: CONCLUSION

5.1 INTRODUCTION

This study has provided an overview of the use of the three statistical classification methods: Logistic Regression, LDA and SVM. The methods were discussed in detail and compared in the analysis of the Two Oceans dataset. The data contained variables about all the entrants from years 2013 to 2015. Ten significant variables were selected in the analysis of the data determining whether the person would start the race as well whether they would finish the race. The Two Oceans Marathon is a world-renowned race with many people applying every year. It is a problem when people enter and do not even start the race due to such a high demand of people who want to participate. In this assignment the best classification method will be selected in order to be able determine what people would not start the race due to certain variables as well as how the variables could have an effect on the person finishing the race.

5.2 SUMMARY OF FINDINGS

The data that was analyzed was split into two sets with two dependent variables, y_1 and y_2, to compare how accurate the three methods could be in determining whether an entrant started or did not start the race (y_1) and whether they finished or did not (y_2). Both sets of data had highly imbalanced classes with large proportions of each set favouring one class. Due to this it was hypothesized that the SVM would struggle as it does not separate well when classes are imbalanced especially in a dataset that has a large number of observations. Kernel functions had to be used for the SVM model to assist in making the data more linearly separable. The four types of kernels that were attempted were: the linear, the radial, the polynomial, and the sigmoid. To use the best kernels, the fixed hyper-parameter for each kernel must be decided by a tune framework that resulted in the best kernels and parameters with the lowest cross-validation errors. As the data was imbalanced the framework failed in determining the best kernel to use but did return the optimal hyper-parameters to use for each kernel in the SVM model. Since the classes were unjust the accuracy rates for the kernels were very high due to very dominant specificity values, resulting in the area the under the curve (AUC) values also playing a role in picking the best kernel. The polynomial kernel was narrowly the best to use for DNS data (y_1) where for the DNF data (y_2) the sigmoid kernel was the best due its' much larger AUC.

Table 5.1: Best Kernel Functions with the AUC and Accuracy

Data:	Kernel:	AUC:	Accuracy:
DNS	Polynomial	0.5143934	0.8147290
DNF	Sigmoid	0.57626398	0.97158237

After the SVM's with each kernel were decided the main analysis took place between the three classification methods. The binary logistic regression and LDA results were very similar among the performance metrics, but logistic regression seemed to be just a bit better and higher among both data analyses.

Table 5.2: Best Classification Method with the AUC and Accuracy

Data:	Method:	AUC:	Accuracy:
DNS	Logistic Regression	0.5583007	0.8084402
DNF	Logistic Regression	0.84247256	0.96699167

Even though the SVM using the selected kernels resulted in best accuracies as seen in Table 1 and Table 2, the AUC and ROC curve were more important in determining the best method. The AUC obtained for logistic regression and LDA in the DNF data for the dependent variable y_2, was the best from the whole analysis. LDA is known to outperform logistic regression when classes are well separated which is clearly not the case. Thus, logistic regression in the binomial case is concluded to be the best classification method to analyze such a large dataset with such differently proportioned classes, although LDA is not too far behind, but the SVM does indeed struggle to correctly evaluate these types of data.

Due to the logistic regression function requiring independency of the observations a further analysis had to be done by modelling the three years separately and taking the average of the three years. With the data also being imbalanced, especially for the DNF variable, the Synthetic Minority Oversampling Technique was used in the program to help balance to more equal weights. This technique was used in modelling the three different years. The hypothesis was that the logistic regression would still be the best statistical method in the determination of whether the person would start the race or finish the race. After receiving the different models in the analysis for the DNF variable the results were that the logistic regression and LDA were still better than the SVM.

Table 5.3 Best Classification Methods from the Balanced DNF Variable

Data:	Method:	AUC:	Test Error
DNF	Logistic Regression	0.83091895	0.07599082
	LDA	0.83110712	0.08166681

Although the LDA results in the best AUC value the logistic regression was still concluded as the best classification method due to method returning the lowest test error. Which for the balanced data meant that the logistic regressions model handled the data the best. Therefore, logistic regression is determined as the best classification method in the prediction of people who will start the race as well as whether they will finish the race. The method can take the variables of the entrants and determine the outcome of that entrant who would prove very beneficial for the applications in the following years. Another benefit on the logistic regression and LDA models were that they were transparent throughout the different models, and able to track the performance of the variables at the given outputs. Even though they are less complex for the Two Oceans dataset it is probably more beneficial to use classification methods that are more interpretable.

5.3 RECOMMENDATIONS FOR FURTHER RESEARCH

It is evident that the performance metrics were consistent between the sets of data, but it was also evident from the results, especially for the DNS data, that the statistical methods did not perform well at all in the AUC. In this type of binary classification with large imbalanced data there is room for further research in other methods. Even though this research project was an analysis between the three methods there may be other classification methods that could statistically examine the data better.

5.4 CONCLUSION

In conclusion of the research project, it was to study these three different classification methods as they were of interest to the authors, even though they were probably not the ideal methods to select for this analysis. All three methods returned some thought-provoking results and has given a different perspective on the use and how to use the certain methods in the foreseeable future.

REFERENCES

Acharjee, S., 2021. *Linear Discriminant Analysis: A Simple Overview In 2021.* [Online]
Available at: https://www.jigsawacademy.com/blogs/ai-ml/linear-discriminant-analysis
[Accessed 3 August 2021].

Africa, B. S., 2002. *Brand South Africa.* [Online]
Available at: https://www.brandsouthafrica.com/people-culture/sport/twooceans
[Accessed 13 10 2021].

Amarappa, S. & Sathyanarayana, S., 2014. Data Classification Using Support Vector Machine (SVM), a Simplified Approach. *International Journal of Electronics and Computer Science Engineering,* III(4), p. 11.

atmathew, 2015. *Evaluating Logistic Regression Models.* [Online]
Available at: https://www.r-bloggers.com/2015/08/evaluating-logistic-regression-models/

Awad, M. & Khanna, R., 2015. *Support Vector Machines for Classification,* Beirut: ResearchGate.

Bassey, P., 2019. Logistic Regression Vs Support Vector Machines (SVM). *Axum Labs*, 19 September, p. 1.

Berry, W. D. & Feldman, S., 1985. *Multiple Regression in Practice.* 1st Edition ed. s.l.:s.n.

Brownlee, J., 2016. Logistic Regression Tutorial for Machine Learning. *Machine Learning Algorithms.*

Brownlee, J., 2020. Cost-Sensitive SVM for Imbalanced Classification. *Imbalanced Classification,* Volume 1.

Brownlee, J., 2020. Linear Discriminant Analysis for Machine Learning. *Machine Learning Algorithms.*

Brownlee, J., 2020. Train-Test Split for Evaluating Machine Learning Algorithms. *Machine Learning Mastery.*

Brownlee, J., April 20, 2016. Support Vector Machines for Machine Learning. *Machine Learning Algorithms.*

Cramer, J., 2002. The Origins of Logistic Regression. In: Sandee, ed. *Logit Models from Economics and Other Fields.* Amsterdam: s.n., pp. 3-6.

David, J. M. & Balakrishnan, K., n.d. *Significance of Classification Techniques in Prediction of Learning Disabilities,* Cochin: s.n.

Dhiraj, K., 2019. Top 4 Advantages and Disadvantages of Support Vector Machine or SVM. *Dhiraj K*, 13 June, p. 1.

Gao, J. & Zhang, J., 2009. Sparse Kernel Learning and the Relevance Units Machine. In: *Advances in Knowledge Discovery and Data Mining.* Berlin: Springer, pp. 326-346.

Girmscheid, C. B. a. G., 2007. Complexity of Megaprojects, in: CIB World Building Congress: Construction for Developmen. *CIB.*

Gutierrez-Osuna, R., n.d. *Linear Discriminants Analysis,* Texas: CSCE 666 Pattern Analysis.

Hajian-Tilaki, K., 2013. Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian Journal of Internal Medicine,* Issue 1.

Halligan, S., Altman, D. G. & Mallett, S., 2015. Disadvantages of Using the Area Under the Receiver Operating Characteristic Curve to Assess Imaging Tests: a Discussion and Proposal for an Alternative Approach. *European Radiology,* Issue 1.

Hosmer, D., Lemeshow, S. & Sturdivant, R., 2013. *Applied Logistic Regression.* 3rd Edition ed. s.l.:Wiley.

Ingo Steinwart, A. C., 2008. *Support Vector Machines.* New York: Springer.

James, G., Witten, D. & Hastie, T., 2013. *An Introduction to Statistical Learning.* 7th ed. New York: Springer.

James, G., Witten, D., Hastie, T. & Tibshirani, R., 2013. *An Introduction to Statistical Learning with Applications in R.* 7th Edition ed. s.l.:Springer.

Kanevski, M. & Pozdnukhov, A., 2001. *Support Vector Machines for Classification and Mapping of Reservoir Data,* Valais: IDIAP.

Kanevski, M. & Pozdnukhov, A., 2001. *Support Vector Machines for Classification and Mapping of Reservoir Data,* Martigny: IDIAP.

Karatzoglou, A., Meyer, D. & Hornik, K., 2006. Support Vector Machines in R. *Journal of Statistical Software,* XV(9), pp. 1-28.

Koturwar, P., Girase, S. & Mukhopadhyay, D., n.d. *A Survey of Classification Techniques in the Area of Big Data.,* s.l.: Maharashtra Institute of Technology Pune.

Lin, H.-T. & Lin, C.-J., 2003. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods. *Neural Computation*, June, pp. 1-33.

Menard, S., 2002. *Applied Logistic Regression Analysis.* 2nd Edition ed. s.l.:s.n.

Nanda, M., Seminar, K., Nandika, D. & Maddu, A., 2018. A Comparison Study of Kernel Functions in the Support Vector Machine and Its Application for Termite Detection. *Information,* pp. 1-14.

Norton, M., Mafusalov, A. & Uryasev, S., 2017. Soft Margin Support Vector Classification as Buffered Probability Minimization. *Journal of Machine Learning Research,* I(1), p. 43.

Pant, A., 2019. *Introduction to Logistic Regression,* s.l.: Towards Data Science.

Phung, T., 2019. *Logistic Regression: Advantages and Disadvantages.* [Online] Available at: https://tungmphung.com/logistic-regression-advantages-and-disadvantages/ [Accessed 1 August 2021].

Pogančić, M. V., 2019. *Learning Theory: Empirical Risk Minimization.* [Online] Available at: https://towardsdatascience.com/learning-theory-empirical-risk-minimization-d3573f90ff77

Quitadamo, L. R., 2017. Support Vector Machines to Detect Physiological Patterns for EEG and EMG-based Human–Computer Interaction: a Review. *Journal of Neural Engineering,* I(1), p. 28.

Savas, C. & Dovis, F., 2019. The Impact of Different Kernel Functions on the Performance of Scintillation Detection Based on Support Vector Machines. *Sensors*, 21 October, pp. 1-16.

Schober, P. M. & Thomas, R. M., 2021. Logistic Regression in Medical Research. *Anesthesia & Analgesia,* 132(2), pp. 365-366.

Sejdinovic, D., n.d. *HT2015: SC4:Statistical Data Mining and Machine Learning.* [Online] Available at: http://www.stats.ox.ac.uk/~sejdinov/teaching/sdmml15/materials/HT15_lecture12-nup.pdf

Shabani, F., Kumar, L. & Ahmadi, M., 2018. Assessing Accuracy Methods of Species Distribution Models: AUC, Specificity, Sensitivity and the True Skill Statistic. *Global Journal of Human-Social Science,* pp. 6-18.

Singh, S., 2018. Understanding the Bias-Variance Tradeoff. *Towards data science,* I(1), p. 1.

Sreenivasa, S., 2020. Radial Basis Function (RBF) Kernel: The Go-To Kernel. *Towards data science*, 12 Ocotober, p. 1.

Srivastava, A., 2020. Differentiate between Support Vector Machine and Logistic Regression. *GeeksforGeeks*, 17 July, p. 1.

Tharwat, A., Gaber, T., Abdelhameed, I. & Hassanien, A. E., 2017. Linear Discriminant Analysis: a Detailed Tutorial. pp. 2-3.

Thomé, A. C. G., 2012. SVM Classifiers – Concepts and Applications to Character Recognition. *InTech,* pp. 25-50.

Vaibhaw, J. & Pattnaik, P., 2020. An Industrial IOT Approach for Pharmaceutical Industry Growth. *Brain–computer interfaces and their applications.*

Zhu, W., Zeng, N. & Wang, N., 2010. *Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS®,* Fort Washington: K&L consulting services.

APPENDIX A: R CODE

A.1 IMBALANCED DATA

```
library(tidyverse)
library(dplyr)
library(tableone)
library(glmnet)
library(tree)
library(randomForest)
library(rfUtilities)
library(ROCR)
library(MASS)
library(e1071)

######## DNS ############
library(readxl)
fulldata<-read_excel("C:/Users/Callum&Matt/Desktop/Stats
2021/Project/Two Oceans Full Clean.xlsx")
fulldata
fulldata<-as.data.frame(fulldata)
fulldata<-fulldata[c(-2)]
fulldata<-na.omit(fulldata)

#Factor variables
factors <- c("dns", "agecat", "gender","racetype",
"year","presmed","consdoc","cleardoc")
fulldata[factors] <- lapply(fulldata[factors], factor) #make
factors
str(fulldata)
variables <- c("dns","age","agecat", "avdist",
"gender","racetype", "year","presmed","consdoc","cleardoc")

table1 <- CreateTableOne(vars = variables,
                         strata = "dns",
                         data = fulldata,
                         factorVars = factors,
                         addOverall = T,
```

```
                              test = F)
table1 <- print(table1, exact = "stage", quote = FALSE, noSpaces
= TRUE, printToggle = FALSE)
```

A.2 LOGISTIC REGRESSION FOR DNS

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lmod <- c() #store auc
accur_lmod<-c()

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lmod <- glm(dns ~ age + agecat + avdist + gender +
  racetype + year + presmed + consdoc + cleardoc, train, family =
"binomial") #apply logistic reg
  #pred_stroke <- rep(0, nrow(test))
  prob_data <- predict(lmod, newdata = test, type = "response")

  #AUC value
  pred <- prediction(prob_data, test$dns)
  perf <- performance(pred, measure = "auc")
  auc_lmod[i] <- perf@y.values[[1]]

  #pred_stroke[prob_stroke > 0.5] <- 1
 #accur_lmod[i] <- mean(test$stroke == pred_stroke) #accuracy
  assign(paste("prob", i, sep = "_"), prob_data) #assign
posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
```

```
}


predictions_lmod <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_lmod <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

mean_auc_lmod <- mean(auc_lmod) #mean auc
meanprob1<-mean(prob_3)

# Make ROC curve
pred_lmod <- prediction(predictions_lmod, labels_lmod)
perf_lmod <- performance(pred_lmod, "tpr", "fpr")
plot(perf_lmod,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for Logistic Regression Model") # ROC curve
for logmod
abline(a=0, b=1) # The completely random

# Find AUC value
auc_perf_lmod <- performance(pred_lmod,measure = "auc")
mean_auc_lmod <- sum(auc_perf_lmod@y.values[[1]],
auc_perf_lmod@y.values[[2]],
                     auc_perf_lmod@y.values[[3]],
auc_perf_lmod@y.values[[4]],
                     auc_perf_lmod@y.values[[5]])/5 #gives you
the mean AUC value

# Find the optimal cutoff + sensitivity and specificity
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
```

```
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
} # define opt cut function
opt_lmod <- opt.cut(perf_lmod, pred_lmod)
opt_sens_lmod <- mean(opt_lmod[1,])
opt_spec_lmod <- mean(opt_lmod[2,])
opt_cut_lmod <- mean(opt_lmod[3,])

# Make Accuracy Plot
acc_perf_lmod = performance(pred_lmod, measure = "acc")
plot(acc_perf_lmod,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lmod <- (slot(acc_perf_lmod, "x.values")[[1]] +
                   slot(acc_perf_lmod, "x.values")[[2]] +
                   slot(acc_perf_lmod, "x.values")[[3]] +
                   slot(acc_perf_lmod, "x.values")[[4]] +
                   slot(acc_perf_lmod, "x.values")[[5]])/5

ind_cut_lmod = 176
acc_lmod = slot(acc_perf_lmod, "y.values")[[1]][ind_cut_lmod]

# Sensitivity vs. Specificity
sen_perf_lmod <- performance(pred_lmod, "sens", "spec")
plot(sen_perf_lmod,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity")

f1score<- (2*(opt_sens_lmod*(1-opt_spec_lmod))/(opt_sens_lmod+(1-
opt_spec_lmod)))

print(c(
  "AUC" = mean_auc_lmod,
  "Test Error" = (1-acc_lmod),
```

```
  "Sensitivity" = opt_sens_lmod,
  "Specificity" = opt_spec_lmod,
  "opt cut-off" = opt_cut_lmod,
  "Accuracy" = acc_lmod
))
```

A.3 LDA FOR DNS

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lda <- c() #store auc

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lda_fit <- lda(dns ~ age + agecat + avdist + gender +
                   racetype + year + presmed + consdoc +
cleardoc, data = train) #apply lda
  lda_pred=predict(lda_fit, test)

  #AUC value
  pred <- prediction(lda_pred$posterior[,2], test$dns)
  perf <- performance(pred, measure = "auc")
  auc_lda[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), lda_pred$posterior[,2])
#assign posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
}
```

```
predictions_lda <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5)
labels_lda <- data.frame(label_1, label_2, label_3, label_4,
label_5)

mean_auc_lda <- mean(auc_lda)

#ROC plot
pred_lda <- prediction(predictions_lda, labels_lda)
perf_lda <- performance(pred_lda, "tpr", "fpr")
plot(perf_lda,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for LDA Model") # ROC curve for lda
abline(a=0, b=1) # The completely random line

#AUC value
auc_perf_lda <- performance(pred_lda, measure = "auc")
mean_auc_lda <- sum(auc_perf_lda@y.values[[1]],
auc_perf_lda@y.values[[2]],
                    auc_perf_lda@y.values[[3]],
auc_perf_lda@y.values[[4]],
                    auc_perf_lda@y.values[[5]])/5

# Find the optimal cutoff + sensitivity and specificity
opt_lda <- opt.cut(perf_lda, pred_lda)
opt_sens_lda <- mean(opt_lda[1,])
opt_spec_lda <- mean(opt_lda[2,])
opt_cut_lda <- mean(opt_lda[3,])

# Make Accuracy Plot
acc_perf_lda = performance(pred_lda, measure = "acc")
plot(acc_perf_lda,
     avg= "vertical",
     main  = "Average Accuracy")
```

```
#Find accuracy for opt cutoff
avg_cut_lda <- (slot(acc_perf_lda, "x.values")[[1]] +
                  slot(acc_perf_lda, "x.values")[[2]] +
                  slot(acc_perf_lda, "x.values")[[3]] +
                  slot(acc_perf_lda, "x.values")[[4]] +
                  slot(acc_perf_lda, "x.values")[[5]])/5
ind_cut_lda = 188
acc_lda = slot(acc_perf_lda, "y.values")[[1]][ind_cut_lda]

# Sensitivity vs. Specificity
sen_perf_lda <- performance(pred_lda, "sens", "spec")
plot(sen_perf_lda,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity for LDA
Model")

f1score<- (2*(opt_sens_lda*(1-opt_spec_lda))/(opt_sens_lda+(1-
opt_spec_lda)))

print(c(
  "AUC" = mean_auc_lda,
  "Test Error" = (1-acc_lda),
  "Sensitivity" = opt_sens_lda,
  "Specificity" = opt_spec_lda,
  "opt cut-off" = opt_cut_lda,
  "Accuracy" = acc_lda
))
```

A.4 SVM FOR DNS

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
```

```
accur_svm <- c() # store accuracy
auc_svm <- c() #store auc
sens_svm <- c() #store sensitivity
spec_svm <- c() #store specificity
for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  svmfit <- svm(dns ~ age + agecat + avdist + gender +
                  racetype + year + presmed + consdoc + cleardoc,
data=train, kernel="polynomial",
cost=0.5,gamma=0.1,coef0=0,degree=2,
                scale=T, decision.values = T)
  prediction_table <- predict(svmfit, test)
  accur_svm[i] <- mean(test$dns == prediction_table) #store
accuracy

  #Sensitivity and Specificity
  conf_matrix <- table(prediction_table, test$dns)
  sens_svm[i] <- conf_matrix[2,2]/(conf_matrix[2,2] +
conf_matrix[1,2]) #sensitivity
  spec_svm[i] <- conf_matrix[1,1]/(conf_matrix[1,1] +
conf_matrix[2,1]) #specificity

  pred_auc <- attributes(predict(svmfit, test, decision.values =
T))$decision.values

  pred <- prediction(pred_auc, test$dns)
  perf <- performance(pred, measure = "auc")
  auc_svm[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), pred_auc) #assign posterior
prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
}
```

```
mean_accur_svm <- mean(accur_svm) #Mean accuracy
mean_sens_svm <- mean(sens_svm) #mean sensitivity
mean_spec_svm <- mean(spec_svm) #mean specificity

mean_auc_svm <- mean(auc_svm)

f1score<- (2*(mean_sens_svm*(1-mean_spec_svm))/(mean_sens_svm+(1-
mean_spec_svm))) #can't get it due 0 sensitivty value

print(c(
  "Mean AUC" = mean_auc_svm,
  "Test Error" = (1-mean_accur_svm),
  "Mean Accuracy" = mean_accur_svm,
  "Mean Sensitivity" = mean_sens_svm,
  "Mean Specificity" = mean_spec_svm
))

predictions_svm <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_svm <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

pred_svm <- prediction(predictions_svm, labels_svm)
perf_svm <- performance(pred_svm, "tpr", "fpr")
plot(perf_svm,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve") # ROC curve for qda
abline(a=0, b=1) # The completely random line
```

A.5 ROC PLOTS FOR DNS

```
plot(perf_lmod,
     avg= "threshold",
     lwd= 2,
```

```
     col = "red",
     main = "ROC curve of the 3 Classification Methods (DNS)")
#lmod curve
plot(perf_lda,
     avg= "threshold",
     add = T,
     lwd= 1,
     col = "green") #lda curve
plot(perf_svm,
     avg= "threshold",
     add = T,
     lwd= 2,
     col= "blue") #svm curve
abline(a=0, b=1, )
legend("bottomright", c("Log Regression", "LDA", "SVM"),
       lty = 1, col = c("red", "green", "blue"), bty = "n",
       inset = c(0, 0.015))

## FINISHED ##
library(readxl)
fulldata<-read_excel("C:/Users/Callum/Desktop/Stats
2021/Project/Two Oceans Full Clean.xlsx")
fulldata
fulldata<-as.data.frame(fulldata)
fulldata<-fulldata[c(-1)]
fulldata<-na.omit(fulldata)

#Factor variables
factors <- c("dnf", "agecat", "gender","racetype",
"year","presmed","consdoc","cleardoc")
fulldata[factors] <- lapply(fulldata[factors], factor) #make
factors

variables <- c("dnf","age","agecat", "avdist",
"gender","racetype", "year","presmed","consdoc","cleardoc")

table1 <- CreateTableOne(vars = variables,
```

```
                          strata = "dnf",
                          data = fulldata,
                          factorVars = factors,
                          addOverall = T,
                          test = F)
table1 <- print(table1, exact = "stage", quote = FALSE, noSpaces
= TRUE, printToggle = FALSE)
```

A.6 LOGISTIC REGRESSION FOR DNF

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lmod <- c() #store auc

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lmod <- glm(dnf ~ age + agecat + avdist + gender +
                racetype + year + presmed + consdoc + cleardoc,
train, family = "binomial") #apply logistic reg
  #pred_stroke <- rep(0, nrow(test))
  prob_data <- predict(lmod, newdata = test, type = "response")

  #AUC value
  pred <- prediction(prob_data, test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_lmod[i] <- perf@y.values[[1]]

  # pred_stroke[prob_stroke > 0.5] <- 1
  # accur_lmod[i] <- mean(test$stroke == pred_stroke) #accuracy
```

```
  assign(paste("prob", i, sep = "_"), prob_data) #assign
posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
}

predictions_lmod <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_lmod <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

mean_auc_lmod <- mean(auc_lmod) #mean auc

# Make ROC curve
pred_lmod <- prediction(predictions_lmod, labels_lmod)
perf_lmod <- performance(pred_lmod, "tpr", "fpr")
plot(perf_lmod,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for Logistic Regression Model") # ROC curve
for logmod
abline(a=0, b=1) # The completely random

# Find AUC value
auc_perf_lmod <- performance(pred_lmod,measure = "auc")
mean_auc_lmod <- sum(auc_perf_lmod@y.values[[1]],
auc_perf_lmod@y.values[[2]],
                     auc_perf_lmod@y.values[[3]],
auc_perf_lmod@y.values[[4]],
                     auc_perf_lmod@y.values[[5]])/5 #gives you
the mean AUC value

# Find the optimal cutoff + sensitivity and specificity
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
```

```
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
} # define opt cut function
opt_lmod <- opt.cut(perf_lmod, pred_lmod)
opt_sens_lmod <- mean(opt_lmod[1,])
opt_spec_lmod <- mean(opt_lmod[2,])
opt_cut_lmod <- mean(opt_lmod[3,])

# Make Accuracy Plot
acc_perf_lmod = performance(pred_lmod, measure = "acc")
plot(acc_perf_lmod,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lmod <- (slot(acc_perf_lmod, "x.values")[[1]] +
                   slot(acc_perf_lmod, "x.values")[[2]] +
                   slot(acc_perf_lmod, "x.values")[[3]] +
                   slot(acc_perf_lmod, "x.values")[[4]] +
                   slot(acc_perf_lmod, "x.values")[[5]])/5

ind_cut_lmod = 176
acc_lmod = slot(acc_perf_lmod, "y.values")[[1]][ind_cut_lmod]

# Sensitivity vs. Specificity
sen_perf_lmod <- performance(pred_lmod, "sens", "spec")
plot(sen_perf_lmod,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity")

f1score<- (2*(opt_sens_lmod*(1-opt_spec_lmod))/(opt_sens_lmod+(1-
opt_spec_lmod)))

print(c(
```

```
  "AUC" = mean_auc_lmod,
  "Test Error" = (1-acc_lmod),
  "Sensitivity" = opt_sens_lmod,
  "Specificity" = opt_spec_lmod,
  "opt cut-off" = opt_cut_lmod,
  "Accuracy" = acc_lmod
))
```

A.7 LDA FOR DNF

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lda <- c() #store auc

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lda_fit <- lda(dnf ~ age + agecat + avdist + gender +
                   racetype + year + presmed + consdoc +
cleardoc, data = train) #apply lda
  lda_pred=predict(lda_fit, test)

  #AUC value
  pred <- prediction(lda_pred$posterior[,2], test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_lda[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), lda_pred$posterior[,2])
#assign posterior prob to prob_i
```

```
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
}

predictions_lda <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5)
labels_lda <- data.frame(label_1, label_2, label_3, label_4,
label_5)

mean_auc_lda <- mean(auc_lda)

#ROC plot
pred_lda <- prediction(predictions_lda, labels_lda)
perf_lda <- performance(pred_lda, "tpr", "fpr")
plot(perf_lda,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for LDA Model") # ROC curve for lda
abline(a=0, b=1) # The completely random line

#AUC value
auc_perf_lda <- performance(pred_lda, measure = "auc")
mean_auc_lda <- sum(auc_perf_lda@y.values[[1]],
auc_perf_lda@y.values[[2]],
                    auc_perf_lda@y.values[[3]],
auc_perf_lda@y.values[[4]],
                    auc_perf_lda@y.values[[5]])/5

# Find the optimal cutoff + sensitivity and specificity
opt_lda <- opt.cut(perf_lda, pred_lda)
opt_sens_lda <- mean(opt_lda[1,])
opt_spec_lda <- mean(opt_lda[2,])
opt_cut_lda <- mean(opt_lda[3,])

# Make Accuracy Plot
acc_perf_lda = performance(pred_lda, measure = "acc")
```

```
plot(acc_perf_lda,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lda <- (slot(acc_perf_lda, "x.values")[[1]] +
                  slot(acc_perf_lda, "x.values")[[2]] +
                  slot(acc_perf_lda, "x.values")[[3]] +
                  slot(acc_perf_lda, "x.values")[[4]] +
                  slot(acc_perf_lda, "x.values")[[5]])/5
ind_cut_lda = 188
acc_lda = slot(acc_perf_lda, "y.values")[[1]][ind_cut_lda]

# Sensitivity vs. Specificity
sen_perf_lda <- performance(pred_lda, "sens", "spec")
plot(sen_perf_lda,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity for LDA
Model")

f1score<- (2*(opt_sens_lda*(1-opt_spec_lda))/(opt_sens_lda+(1-
opt_spec_lda)))

print(c(
  "AUC" = mean_auc_lda,
  "Test Error" = (1-acc_lda),
  "Sensitivity" = opt_sens_lda,
  "Specificity" = opt_spec_lda,
  "opt cut-off" = opt_cut_lda,
  "Accuracy" = acc_lda
))
```

A.8 SVM FOR DNF

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
```

```
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
accur_svm <- c() # store accuracy
auc_svm <- c() #store auc
sens_svm <- c() #store sensitivity
spec_svm <- c() #store specificity
for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  svmfit <- svm(dnf ~ age + agecat + avdist + gender +
                  racetype + year + presmed + consdoc + cleardoc,
data=train, kernel="sigmoid", cost=0.5,gamma=0.1,coef0=0,
                scale=T, decision.values = T)
  prediction_table <- predict(svmfit, test)
  accur_svm[i] <- mean(test$dnf == prediction_table) #store
accuracy

  #Sensitivity and Specificity
  conf_matrix <- table(prediction_table, test$dnf)
  sens_svm[i] <- conf_matrix[2,2]/(conf_matrix[2,2] +
conf_matrix[1,2]) #sensitivity
  spec_svm[i] <- conf_matrix[1,1]/(conf_matrix[1,1] +
conf_matrix[2,1]) #specificity

  pred_auc <- attributes(predict(svmfit, test, decision.values =
T))$decision.values

  pred <- prediction(pred_auc, test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_svm[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), pred_auc) #assign posterior
prob to prob_i
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
```

```
}

mean_accur_svm <- mean(accur_svm) #Mean accuracy
mean_sens_svm <- mean(sens_svm) #mean sensitivity
mean_spec_svm <- mean(spec_svm) #mean specificity

mean_auc_svm <- mean(auc_svm)

print(c(
  "Mean AUC" = mean_auc_svm,
  "Test Error" = (1-mean_accur_svm),
  "Mean Accuracy" = mean_accur_svm,
  "Mean Sensitivity" = mean_sens_svm,
  "Mean Specificity" = mean_spec_svm
))

predictions_svm <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_svm <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

pred_svm <- prediction(predictions_svm, labels_svm)
perf_svm <- performance(pred_svm, "tpr", "fpr")
plot(perf_svm,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve") # ROC curve for qda
abline(a=0, b=1) # The completely random line
```

A.9 ROC PLOTS FOR DNF

```
plot(perf_lmod,
     avg= "threshold",
     lwd= 2,
     col = "red",
     main = "ROC curve of the 3 Classification Methods (DNF)")
```

```
plot(perf_lda,
     avg= "threshold",
     add = T,
     lwd= 1,
     col = "green") #lda curve
plot(perf_svm,
     avg= "threshold",
     add = T,
     lwd= 2,
     col= "blue") #svm curve
abline(a=0, b=1, )
legend("bottomright", c("Log Regression", "LDA", "SVM"),
       lty = 1, col = c("red", "green", "blue"), bty = "n",
       inset = c(0, 0.015))
```

A.10 KERNEL PLOTS

```
for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  svmfit <- svm(dnf ~ age + agecat + avdist + gender +
                  racetype + year + presmed + consdoc + cleardoc,
data=train, kernel="radial", cost=0.001,gamma=0.5,
                scale=T, decision.values = T)
  prediction_table <- predict(svmfit, test)
  accur_svm[i] <- mean(test$dnf == prediction_table) #store
accuracy

  #Sensitivity and Specificity
  conf_matrix <- table(prediction_table, test$dnf)
  sens_svm[i] <- conf_matrix[2,2]/(conf_matrix[2,2] +
conf_matrix[1,2]) #sensitivity
  spec_svm[i] <- conf_matrix[1,1]/(conf_matrix[1,1] +
conf_matrix[2,1]) #specificity
```

```
  pred_auc <- attributes(predict(svmfit, test, decision.values =
T))$decision.values

  pred <- prediction(pred_auc, test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_svm[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), pred_auc) #assign posterior
prob to prob_i
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
}



for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  svmfit <- svm(dns ~ age + agecat + avdist + gender +
                  racetype + year + presmed + consdoc + cleardoc,
data=train, kernel="radial", cost=0.001,gamma=0.5,
                scale=T, decision.values = T)
  prediction_table <- predict(svmfit, test)
  accur_svm[i] <- mean(test$dns == prediction_table) #store
accuracy

  #Sensitivity and Specificity
  conf_matrix <- table(prediction_table, test$dns)
  sens_svm[i] <- conf_matrix[2,2]/(conf_matrix[2,2] +
conf_matrix[1,2]) #sensitivity
  spec_svm[i] <- conf_matrix[1,1]/(conf_matrix[1,1] +
conf_matrix[2,1]) #specificity

  pred_auc <- attributes(predict(svmfit, test, decision.values =
T))$decision.values
```

```
  pred <- prediction(pred_auc, test$dns)
  perf <- performance(pred, measure = "auc")
  auc_svm[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), pred_auc) #assign posterior
prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
}

############ Kernel PLot

plot(perf_svm1,
     avg= "threshold",
     add = T,
     lwd= 1,
     col= "red")
plot(perf_svm2,
     avg= "threshold",
     lwd= 1,
     col = "green",
     main="ROC Curve of the Kernels (DNS)")
plot(perf_svm3,
     avg= "threshold",
     add = T,
     lwd= 1,
     col = "orange")
plot(perf_svm4,
     avg= "threshold",
     add = T,
     lwd= 1,
     col= "blue")
abline(a=0, b=1, )
legend("topleft", c("Linear", "Radial", "Polynomial","Sigmoid"),
       lty = 1, col = c("red","green", "orange", "blue"), bty =
"n",
       inset = c(0, 0.015))
```

A.11 BALANCED DATA

A.12 SMOTE PACKAGE

```
library(smotefamily)
library(DMwR2)
library(readxl)
fulldata<-read_excel("C:/Users/Callum&Matt/Desktop/Stats
2021/Project/Two Oceans Full Clean.xlsx")
fulldata
fulldata<-as.data.frame(fulldata)
fulldata<-fulldata[c(-1)]
fulldata<-na.omit(fulldata)
fulldata<-subset(fulldata,year=="2013")
fulldata<-subset(fulldata,year=="2014")
fulldata<-subset(fulldata,year=="2015")
fulldata<-fulldata[c(-7)]
fulldata
fulldata$dnf<-factor(fulldata$dnf)

data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds

test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
test <- data_shuffle[test_indexes, ] #test set
train <- data_shuffle[-test_indexes, ] #train set
lmod <- glm(class ~ ., newtrain, family = "binomial") #apply
logistic reg

table(train$dnf)
prop.table((table(train$dnf)))
newtrain <- SMOTE(train[,-1],train[,1],K=5)
newtrain <- newtrain$data # extract only the balanced dataset
newtrain$class <- as.factor(newtrain$class)

table(newtrain$class)
prop.table((table(newtrain$class)))
```

A.13 LOGISTIC REGRESSION FOR DNF ON BALANCED DATA

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lmod <- c() #store auc

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lmod <- glm(class ~ ., newtrain, family = "binomial") #apply
logistic reg
  #pred_stroke <- rep(0, nrow(test))
  prob_data <- predict(lmod, newdata = test, type = "response")

  #AUC value
  pred <- prediction(prob_data, test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_lmod[i] <- perf@y.values[[1]]

  # pred_stroke[prob_stroke > 0.5] <- 1
  # accur_lmod[i] <- mean(test$stroke == pred_stroke) #accuracy
  assign(paste("prob", i, sep = "_"), prob_data) #assign
posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
}

predictions_lmod <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
```

```
labels_lmod <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

mean_auc_lmod <- mean(auc_lmod) #mean auc

# Make ROC curve
pred_lmod <- prediction(predictions_lmod, labels_lmod)
perf_lmod <- performance(pred_lmod, "tpr", "fpr")
plot(perf_lmod,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for Logistic Regression Model") # ROC curve
for logmod
abline(a=0, b=1) # The completely random

# Find AUC value
auc_perf_lmod <- performance(pred_lmod,measure = "auc")
mean_auc_lmod <- sum(auc_perf_lmod@y.values[[1]],
auc_perf_lmod@y.values[[2]],
                     auc_perf_lmod@y.values[[3]],
auc_perf_lmod@y.values[[4]],
                     auc_perf_lmod@y.values[[5]])/5 #gives you
the mean AUC value

# Find the optimal cutoff + sensitivity and specificity
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
} # define opt cut function
opt_lmod <- opt.cut(perf_lmod, pred_lmod)
opt_sens_lmod <- mean(opt_lmod[1,])
opt_spec_lmod <- mean(opt_lmod[2,])
```

```
opt_cut_lmod <- mean(opt_lmod[3,])

# Make Accuracy Plot
acc_perf_lmod = performance(pred_lmod, measure = "acc")
plot(acc_perf_lmod,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lmod <- (slot(acc_perf_lmod, "x.values")[[1]] +
                   slot(acc_perf_lmod, "x.values")[[2]] +
                   slot(acc_perf_lmod, "x.values")[[3]] +
                   slot(acc_perf_lmod, "x.values")[[4]] +
                   slot(acc_perf_lmod, "x.values")[[5]])/5

ind_cut_lmod = 176
acc_lmod = slot(acc_perf_lmod, "y.values")[[1]][ind_cut_lmod]

# Sensitivity vs. Specificity
sen_perf_lmod <- performance(pred_lmod, "sens", "spec")
plot(sen_perf_lmod,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity")

f1score<- (2*(opt_sens_lmod*(1-opt_spec_lmod))/(opt_sens_lmod+(1-
opt_spec_lmod)))

print(c(
  "AUC" = mean_auc_lmod,
  "Test Error" = (1-acc_lmod),
  "Sensitivity" = opt_sens_lmod,
  "Specificity" = opt_spec_lmod,
  "opt cut-off" = opt_cut_lmod,
  "Accuracy" = acc_lmod
))
```

A.14 LDA FOR DNF ON BALANCED DATA

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lda <- c() #store auc

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lda_fit <- lda(class ~ ., newtrain) #apply lda
  lda_pred=predict(lda_fit, test)

  #AUC value
  pred <- prediction(lda_pred$posterior[,2], test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_lda[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), lda_pred$posterior[,2])
#assign posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
}

predictions_lda <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5)
labels_lda <- data.frame(label_1, label_2, label_3, label_4,
label_5)

mean_auc_lda <- mean(auc_lda)
```

```
#ROC plot
pred_lda <- prediction(predictions_lda, labels_lda)
perf_lda <- performance(pred_lda, "tpr", "fpr")
plot(perf_lda,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for LDA Model") # ROC curve for lda
abline(a=0, b=1) # The completely random line

#AUC value
auc_perf_lda <- performance(pred_lda, measure = "auc")
mean_auc_lda <- sum(auc_perf_lda@y.values[[1]],
auc_perf_lda@y.values[[2]],
                    auc_perf_lda@y.values[[3]],
auc_perf_lda@y.values[[4]],
                    auc_perf_lda@y.values[[5]])/5

# Find the optimal cutoff + sensitivity and specificity
opt_lda <- opt.cut(perf_lda, pred_lda)
opt_sens_lda <- mean(opt_lda[1,])
opt_spec_lda <- mean(opt_lda[2,])
opt_cut_lda <- mean(opt_lda[3,])

# Make Accuracy Plot
acc_perf_lda = performance(pred_lda, measure = "acc")
plot(acc_perf_lda,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lda <- (slot(acc_perf_lda, "x.values")[[1]] +
                  slot(acc_perf_lda, "x.values")[[2]] +
                  slot(acc_perf_lda, "x.values")[[3]] +
                  slot(acc_perf_lda, "x.values")[[4]] +
                  slot(acc_perf_lda, "x.values")[[5]])/5
ind_cut_lda = 188
```

```
acc_lda = slot(acc_perf_lda, "y.values")[[1]][ind_cut_lda]

# Sensitivity vs. Specificity
sen_perf_lda <- performance(pred_lda, "sens", "spec")
plot(sen_perf_lda,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity for LDA
Model")

f1score<- (2*(opt_sens_lda*(1-opt_spec_lda))/(opt_sens_lda+(1-
opt_spec_lda)))

print(c(
  "AUC" = mean_auc_lda,
  "Test Error" = (1-acc_lda),
  "Sensitivity" = opt_sens_lda,
  "Specificity" = opt_spec_lda,
  "opt cut-off" = opt_cut_lda,
  "Accuracy" = acc_lda
))
```

A.15 SVM FOR DNF ON BALANCED DATA

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
accur_svm <- c() # store accuracy
auc_svm <- c() #store auc
sens_svm <- c() #store sensitivity
spec_svm <- c() #store specificity
for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
```

```
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  svmfit <- svm(class ~ ., newtrain, kernel="sigmoid",
cost=0.5,gamma=0.1,coef0=0,
                scale=T, decision.values = T)
  prediction_table <- predict(svmfit, test)
  accur_svm[i] <- mean(test$dnf == prediction_table) #store
accuracy

  #Sensitivity and Specificity
  conf_matrix <- table(prediction_table, test$dnf)
  sens_svm[i] <- conf_matrix[2,2]/(conf_matrix[2,2] +
conf_matrix[1,2]) #sensitivity
  spec_svm[i] <- conf_matrix[1,1]/(conf_matrix[1,1] +
conf_matrix[2,1]) #specificity

  pred_auc <- attributes(predict(svmfit, test, decision.values =
T))$decision.values

  pred <- prediction(pred_auc, test$dnf)
  perf <- performance(pred, measure = "auc")
  auc_svm[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), pred_auc) #assign posterior
prob to prob_i
  assign(paste("label", i, sep = "_"), test$dnf) #store true
labels to label_i
}

mean_accur_svm <- mean(accur_svm) #Mean accuracy
mean_sens_svm <- mean(sens_svm) #mean sensitivity
mean_spec_svm <- mean(spec_svm) #mean specificity

mean_auc_svm <- mean(auc_svm)

print(c(
  "Mean AUC" = mean_auc_svm,
```

```
  "Test Error" = (1-mean_accur_svm),
  "Mean Accuracy" = mean_accur_svm,
  "Mean Sensitivity" = mean_sens_svm,
  "Mean Specificity" = mean_spec_svm
))

predictions_svm <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_svm <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

pred_svm <- prediction(predictions_svm, labels_svm)
perf_svm <- performance(pred_svm, "tpr", "fpr")
plot(perf_svm,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve") # ROC curve for qda
abline(a=0, b=1) # The completely random line
```

A.16 ROC PLOTS FOR DNF ON BALANCED DATA

```
plot(perf_lmod,
     avg= "threshold",
     lwd= 2,
     col = "red",
     main = "ROC curve of the 3 Classification Methods (DNF)")
plot(perf_lda,
     avg= "threshold",
     add = T,
     lwd= 1,
     col = "green") #lda curve
plot(perf_svm,
     avg= "threshold",
     add = T,
     lwd= 2,
     col= "blue") #svm curve
```

```
abline(a=0, b=1, )
legend("bottomright", c("Log Regression", "LDA", "SVM"),
       lty = 1, col = c("red", "green", "blue"), bty = "n",
       inset = c(0, 0.015))
```

A.17 LOGISTIC REGRESSION FOR DNS ON BALANCED DATA

```
library(smotefamily)
library(DMwR2)
library(readxl)
fulldata<-read_excel("C:/Users/Callum/Desktop/Stats
2021/Project/Two Oceans Full Clean.xlsx")
fulldata
fulldata<-as.data.frame(fulldata)
fulldata<-fulldata[c(-2)]
fulldata<-na.omit(fulldata)
fulldata<-subset(fulldata,year=="2013")
fulldata<-fulldata[c(-7)]
fulldata
fulldata$dns<-factor(fulldata$dns)

data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds

test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
test <- data_shuffle[test_indexes, ] #test set
train <- data_shuffle[-test_indexes, ] #train set
lmod <- glm(class ~ ., newtrain, family = "binomial") #apply
logistic reg

table(train$dns)
prop.table((table(train$dns)))
newtrain <- SMOTE(train[,-1],train[,1],K=5)
newtrain <- newtrain$data # extract only the balanced dataset
newtrain$class <- as.factor(newtrain$class)

table(newtrain$class)
prop.table((table(newtrain$class)))
```

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lmod <- c() #store auc
accur_lmod<-c()

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lmod <- glm(class ~ ., newtrain, family = "binomial") #apply
logistic reg
  #pred_stroke <- rep(0, nrow(test))
  prob_data <- predict(lmod, newdata = test, type = "response")

  #AUC value
  pred <- prediction(prob_data, test$dns)
  perf <- performance(pred, measure = "auc")
  auc_lmod[i] <- perf@y.values[[1]]

  #pred_stroke[prob_stroke > 0.5] <- 1
  #accur_lmod[i] <- mean(test$stroke == pred_stroke) #accuracy
  assign(paste("prob", i, sep = "_"), prob_data) #assign
posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
}


predictions_lmod <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_lmod <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels
```

```
mean_auc_lmod <- mean(auc_lmod) #mean auc



# Make ROC curve
pred_lmod <- prediction(predictions_lmod, labels_lmod)
perf_lmod <- performance(pred_lmod, "tpr", "fpr")
plot(perf_lmod,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for Logistic Regression Model") # ROC curve
for logmod
abline(a=0, b=1) # The completely random

# Find AUC value
auc_perf_lmod <- performance(pred_lmod,measure = "auc")
mean_auc_lmod <- sum(auc_perf_lmod@y.values[[1]],
auc_perf_lmod@y.values[[2]],
                     auc_perf_lmod@y.values[[3]],
auc_perf_lmod@y.values[[4]], auc_perf_lmod@y.values[[5]])/5
#gives you the mean AUC value

# Find the optimal cutoff + sensitivity and specificity
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
} # define opt cut function
opt_lmod <- opt.cut(perf_lmod, pred_lmod)
opt_sens_lmod <- mean(opt_lmod[1,])
opt_spec_lmod <- mean(opt_lmod[2,])
opt_cut_lmod <- mean(opt_lmod[3,])
```

```
# Make Accuracy Plot
acc_perf_lmod = performance(pred_lmod, measure = "acc")
plot(acc_perf_lmod,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lmod <- (slot(acc_perf_lmod, "x.values")[[1]] +
                   slot(acc_perf_lmod, "x.values")[[2]] +
                   slot(acc_perf_lmod, "x.values")[[3]] +
                   slot(acc_perf_lmod, "x.values")[[4]] +
                   slot(acc_perf_lmod, "x.values")[[5]])/5

ind_cut_lmod = 176
acc_lmod = slot(acc_perf_lmod, "y.values")[[1]][ind_cut_lmod]

# Sensitivity vs. Specificity
sen_perf_lmod <- performance(pred_lmod, "sens", "spec")
plot(sen_perf_lmod,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity")

f1score<- (2*(opt_sens_lmod*(1-opt_spec_lmod))/(opt_sens_lmod+(1-
opt_spec_lmod)))

print(c(
  "AUC" = mean_auc_lmod,
  "Test Error" = (1-acc_lmod),
  "Sensitivity" = opt_sens_lmod,
  "Specificity" = opt_spec_lmod,
  "opt cut-off" = opt_cut_lmod,
  "Accuracy" = acc_lmod
))
```

A.18 LDA FOR DNS ON BALANCED DATA

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
auc_lda <- c() #store auc

for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
  lda_fit <- lda(class ~ ., newtrain) #apply lda
  lda_pred=predict(lda_fit, test)

  #AUC value
  pred <- prediction(lda_pred$posterior[,2], test$dns)
  perf <- performance(pred, measure = "auc")
  auc_lda[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), lda_pred$posterior[,2])
#assign posterior prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
}

predictions_lda <- data.frame(prob_1, prob_2,prob_3, prob_4,
prob_5)
labels_lda <- data.frame(label_1, label_2, label_3,label_4,
label_5)

mean_auc_lda <- mean(auc_lda)

#ROC plot
```

```
pred_lda <- prediction(predictions_lda, labels_lda)
perf_lda <- performance(pred_lda, "tpr", "fpr")
plot(perf_lda,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve for LDA Model") # ROC curve for lda
abline(a=0, b=1) # The completely random line

#AUC value
auc_perf_lda <- performance(pred_lda, measure = "auc")
mean_auc_lda <- sum(auc_perf_lda@y.values[[1]],
auc_perf_lda@y.values[[2]],
                    auc_perf_lda@y.values[[3]],
auc_perf_lda@y.values[[4]], auc_perf_lda@y.values[[5]])/5

# Find the optimal cutoff + sensitivity and specificity
opt_lda <- opt.cut(perf_lda, pred_lda)
opt_sens_lda <- mean(opt_lda[1,])
opt_spec_lda <- mean(opt_lda[2,])
opt_cut_lda <- mean(opt_lda[3,])

# Make Accuracy Plot
acc_perf_lda = performance(pred_lda, measure = "acc")
plot(acc_perf_lda,
     avg= "vertical",
     main  = "Average Accuracy")

#Find accuracy for opt cutoff
avg_cut_lda <- (slot(acc_perf_lda, "x.values")[[1]] +
                  slot(acc_perf_lda, "x.values")[[2]] +
                  slot(acc_perf_lda, "x.values")[[3]] +
                  slot(acc_perf_lda, "x.values")[[4]] +
                  slot(acc_perf_lda, "x.values")[[5]])/5
ind_cut_lda = 188
acc_lda = slot(acc_perf_lda, "y.values")[[1]][ind_cut_lda]
```

```
# Sensitivity vs. Specificity
sen_perf_lda <- performance(pred_lda, "sens", "spec")
plot(sen_perf_lda,
     avg = "threshold",
     colorize = T,
     main = "Average Sensitivity vs. Average Specificity for LDA
Model")

f1score<- (2*(opt_sens_lda*(1-opt_spec_lda))/(opt_sens_lda+(1-
opt_spec_lda)))

print(c(
  "AUC" = mean_auc_lda,
  "Test Error" = (1-acc_lda),
  "Sensitivity" = opt_sens_lda,
  "Specificity" = opt_spec_lda,
  "opt cut-off" = opt_cut_lda,
  "Accuracy" = acc_lda
))
```

A.19 SVM FOR DNS ON BALANCED DATA

```
set.seed(0)
#make 5-folds
data_shuffle <- fulldata[sample(nrow(fulldata)),] #randomly
shuffle data
folds <- cut(seq(1, nrow(data_shuffle)), breaks=5, labels=FALSE)
#make 10 equally sized folds
accur_svm <- c() # store accuracy
auc_svm <- c() #store auc
sens_svm <- c() #store sensitivity
spec_svm <- c() #store specificity
for(i in 1:5){
  #make testing and training data
  test_indexes <- which(folds==i,arr.ind=TRUE) #segment the data
  test <- data_shuffle[test_indexes, ] #test set
  train <- data_shuffle[-test_indexes, ] #train set
```

```
  svmfit <- svm(class ~ ., newtrain, kernel="sigmoid",
cost=0.5,gamma=0.1,coef0=0,
                scale=T, decision.values = T)
  prediction_table <- predict(svmfit, test)
  accur_svm[i] <- mean(test$dns == prediction_table) #store
accuracy

  #Sensitivity and Specificity
  conf_matrix <- table(prediction_table, test$dns)
  sens_svm[i] <- conf_matrix[2,2]/(conf_matrix[2,2] +
conf_matrix[1,2]) #sensitivity
  spec_svm[i] <- conf_matrix[1,1]/(conf_matrix[1,1] +
conf_matrix[2,1]) #specificity

  pred_auc <- attributes(predict(svmfit, test, decision.values =
T))$decision.values

  pred <- prediction(pred_auc, test$dns)
  perf <- performance(pred, measure = "auc")
  auc_svm[i] <- perf@y.values[[1]]

  assign(paste("prob", i, sep = "_"), pred_auc) #assign posterior
prob to prob_i
  assign(paste("label", i, sep = "_"), test$dns) #store true
labels to label_i
}

mean_accur_svm <- mean(accur_svm) #Mean accuracy
mean_sens_svm <- mean(sens_svm) #mean sensitivity
mean_spec_svm <- mean(spec_svm) #mean specificity

mean_auc_svm <- mean(auc_svm)

f1score<- (2*(mean_sens_svm*(1-mean_spec_svm))/(mean_sens_svm+(1-
mean_spec_svm))) #can't get it due 0 sensitivty value

print(c(
```

```
  "Mean AUC" = mean_auc_svm,
  "Test Error" = (1-mean_accur_svm),
  "Mean Accuracy" = mean_accur_svm,
  "Mean Sensitivity" = mean_sens_svm,
  "Mean Specificity" = mean_spec_svm
))

predictions_svm <- data.frame(prob_1, prob_2, prob_3, prob_4,
prob_5) #Predicted values
labels_svm <- data.frame(label_1, label_2, label_3, label_4,
label_5) #true labels

pred_svm <- prediction(predictions_svm, labels_svm)
perf_svm <- performance(pred_svm, "tpr", "fpr")
plot(perf_svm,
     avg= "threshold",
     colorize=TRUE,
     lwd= 3,
     main= "ROC Curve") # ROC curve for qda
abline(a=0, b=1) # The completely random line
```

A.20 ROC PLOTS FOR DNS ON BALANCED DATA

```
plot(perf_lmod,
     avg= "threshold",
     lwd= 2,
     col = "red",
     main = "ROC curve of the 3 Classification Methods (DNS)")
#lmod curve
plot(perf_lda,
     avg= "threshold",
     add = T,
     lwd= 1,
     col = "green") #lda curve
plot(perf_svm,
     avg= "threshold",
     add = T,
```

```
     lwd= 2,
     col= "blue") #svm curve
abline(a=0, b=1, )
legend("bottomright", c("Log Regression", "LDA", "SVM"),
       lty = 1, col = c("red", "green", "blue"), bty = "n",
       inset = c(0, 0.015))
```

APPENDIX B: PRESENTATION OF TABLES AND FIGURES

B.1 PRESENTATION OF TABLES

Table 4.1: Summary of the Two Oceans Marathon DNS Data:

Starters	Non-Starters	Total Entrants
49230	11195	60425

Table 4.2: Summary of the Two Oceans Marathon DNF Data:

Finisher	Non-Finisher	Total Runners
48386	844	49230

Table 4.3: Summary of the Predictor Variables:

Variable	Label	Class	Levels	Range
The age of the entrant:	(age)	Numeric		[16 – 86]
The age category the entrant falls in:	(agecat)	Factor	1: less than 30 years, 2: 31 to 40 years, 3: 41 to 50 years, 4: greater than 51 years.	
The average weekly distance the entrant ran in the last 12 months (kilometres per week):	(avdist)	Numeric		[4 – 160]
The gender of the entrant:	(gender)	Factor	1: Male 2: Female	
The type of race the entrant signed up for:	(racetype)	Factor	1: Ultra Marathon 2: Half Marathon	
The race year the entrant signed up for:	(year)	Factor	1: 2013 2: 2014 3: 2015	
Had the entrant been prescribed	(presmed)	Factor	0: No 1: Yes	

medication to treat chronic medical conditions/injuries.				
Had the entrant consulted with a medical doctor in the last 12 months to obtain medical clearance for endurance running.	(consdoc)	Factor	0: No 1: Yes	
If yes to (consdoc) above, did the medical practitioner clear the entrant with any specific advice for participating in endurance running?	(cleardoc)	Factor	0: Did not consult a doctor 1: The doctor did not give clearance to run 2: The doctor gave clearance to run but with some restrictions 3: The doctor gave clearance to run with no restrictions	

Table 4.4: Confusion Matrix for DNS

	Actual	
Predicted	Did Not Start (Train Positive)	Did Start (Train Negative)
Did Not Start (Test Positive)	True Positive (TP)	False Positive (FP)
Did Start (Test Negative)	False Negative (FP)	True Negative (TN)

	Sensitivity: $= \frac{\text{True Positive}}{\text{Train Positive}}$	Specificity: $= \frac{\text{True Negative}}{\text{Train Negative}}$

Table 4.5: Tuning Framework for the Different Hyper-Parameters

	Tuning Values:	Linear	Radial	Polynomial	Sigmoid
Cost (C)	$(0.5, 1,2, 3)$	✓	✓	✓	✓
Gamma (γ)	$(0.001, 0.01, 0.1, 1)$		✓	✓	✓
Offset (coef0)	$(0, 1, 2)$			✓	✓
Degree	$(2, 3, 4)$			✓	

Table 4.6: Results of Best Hyper-Parameters

	Classification Error (DNS):	Classification Error (DNF):	Optimal Parameters:
Linear	0.198	0.016	$C = 0.5$
Radial	0.198	0.016	$C = 0.5, \gamma = 0.1$
Polynomial	0.198	0.016	$C = 0.5, \quad \gamma = 0.1, coef0 = 0, degree = 2$
Sigmoid	0.198	0.016	$C = 0.5, \quad \gamma = 0.1, coef0 = 0$

Table 4.7: Performance of Kernels for the DNS Variable

Kernel:	AUC:	Accuracy:	Sensitivity:	Specificity:
Linear	0.4776912	0.8147290	0.0000000	1.0000000
Radial	0.5099363	0.8147290	0.0000000	1.0000000
Polynomial	0.5143934	0.8147290	0.0000000	1.0000000
Sigmoid	0.4755014	0.7105337	0.1987536	0.8269162

Table 4.8: Performance of Kernels for the DNF Variable

Kernel:	AUC:	Accuracy:	Sensitivity:	Specificity:
Linear	0.2597994	0.9828560	0.0000000	1.0000000
Radial	0.3043581	0.9828560	0.0000000	1.0000000
Polynomial	0.3083284	0.9828560	0.0000000	1.0000000
Sigmoid	0.57626398	0.97158237	0.03920032	0.98786895

Table 4.9: Comparison of performance on the three classification methods for DNS variable.

	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.5583007	0.1915598	0.5207700	0.5690142	0.8084402
LDA	0.5580245	0.1927182	0.5241794	0.5663646	0.8072818
SVM using Polynomial kernel	0.5143934	0.1852710	0.0000000	1.0000000	0.8147290

Table 4.10: Comparison of performance on the three classification methods for the DNF variable.

	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.84247256	0.03300833	0.85855879	0.71055120	0.96699167
LDA	0.84105307	0.03402397	0.84551533	0.71797865	0.96597603
SVM using Sigmoid kernel	0.57626398	0.02841763	0.03920032	0.98786895	0.97158237

Table 4.11: Average performance of the three classification methods for the DNF variable.

	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.83091895	0.07599082	0.8632855	0.7074967	0.92400918
LDA	0.83110712	0.08166681	0.8761844	0.699779	0.91833319
SVM (Sigmoid)	0.72583830	0.3315951	0.6962716	0.6681426	0.66840487

Table 4.12: The performance of the three classification methods for the 2013 DNS variable.

	AUC	Test error	Sensitivity	Specificity	Accuracy
Logistic	0.5514010	0.2243088	0.5343813	0.5635383	0.7756912
LDA	0.5514097	0.2266562	0.5350470	0.5625644	0.7733438
SVM	0.5109437	0.4877667	0.5009967	0.5150066	0.5122333

Table 5.1: Best Kernel Functions with the AUC and Accuracy

Data:	Kernel:	AUC:	Accuracy:
DNS	Polynomial	0.5143934	0.8147290
DNF	Sigmoid	0.57626398	0.97158237

Table 5.2: Best Classification Method with the AUC and Accuracy

Data:	Method:	AUC:	Accuracy:
DNS	Logistic Regression	0.5583007	0.8084402
DNF	Logistic Regression	0.84247256	0.96699167

Table 5.3 Best Classification Methods from the Balanced DNF Variable

Data:	Method:	AUC:	Test Error

DNF	Logistic Regression	0.83091895	0.07599082
	LDA	0.83110712	0.08166681

B.2 PRESENTATION OF FIGURES

Figure 4.1: ROC curve for the four different kernels for the variable

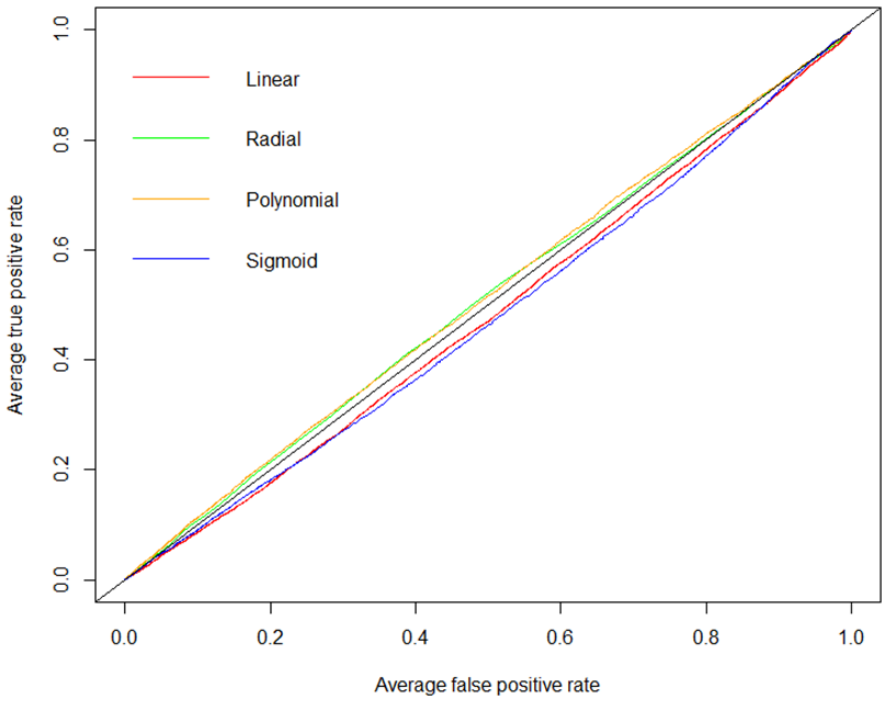


Figure 4.2: ROC curve for the four different kernels for the variable DNF

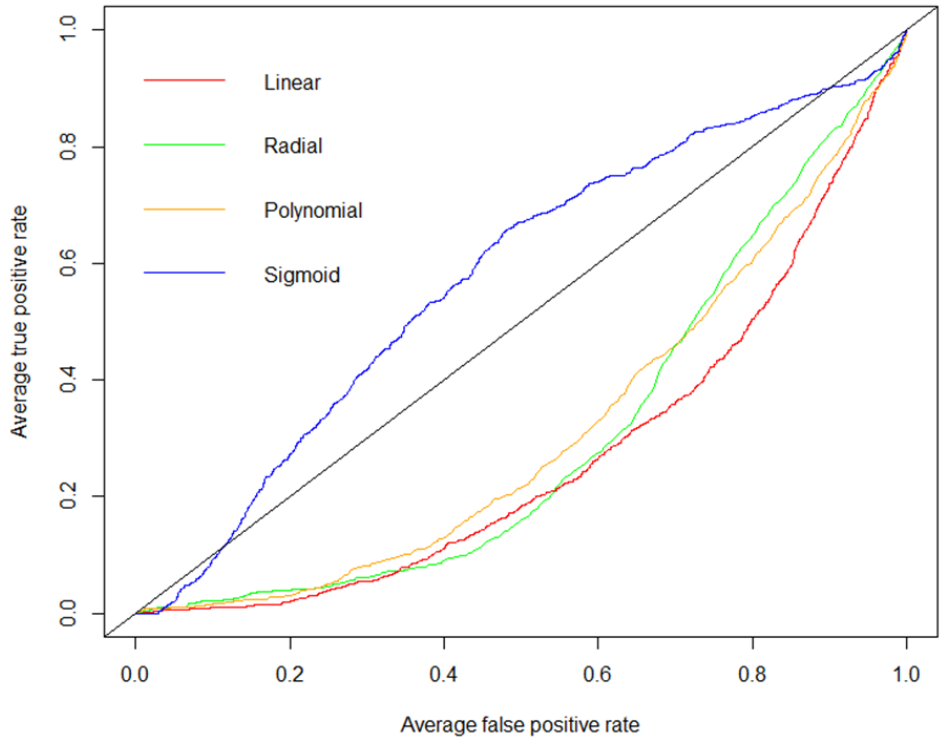


Figure 4.3: ROC Curve for the three different classification methods for the imbalanced variable DNS.

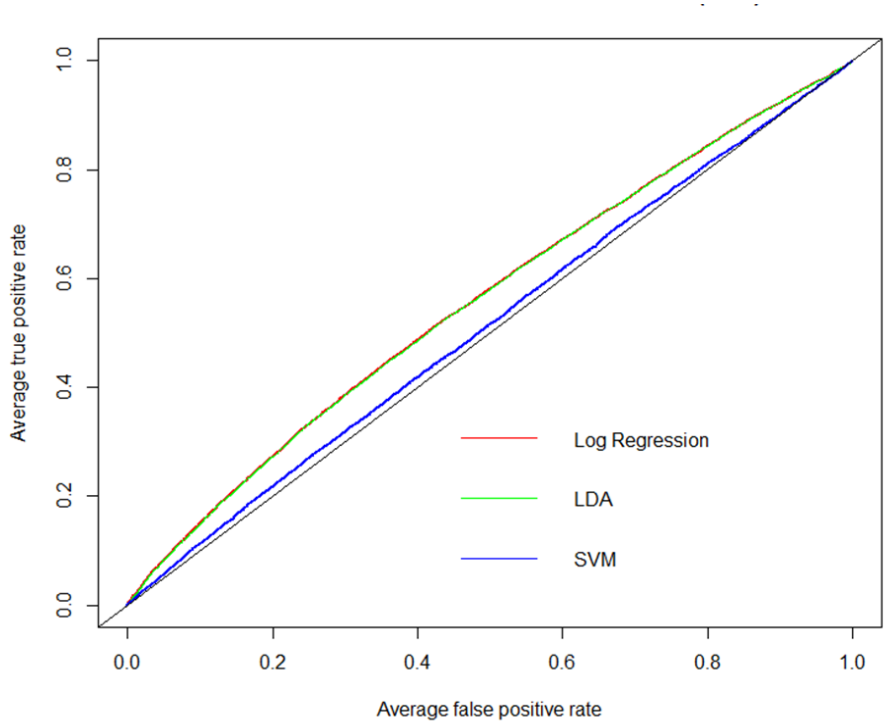


Figure 4.4: ROC curve for the three different classification methods for the imbalanced variable DNF

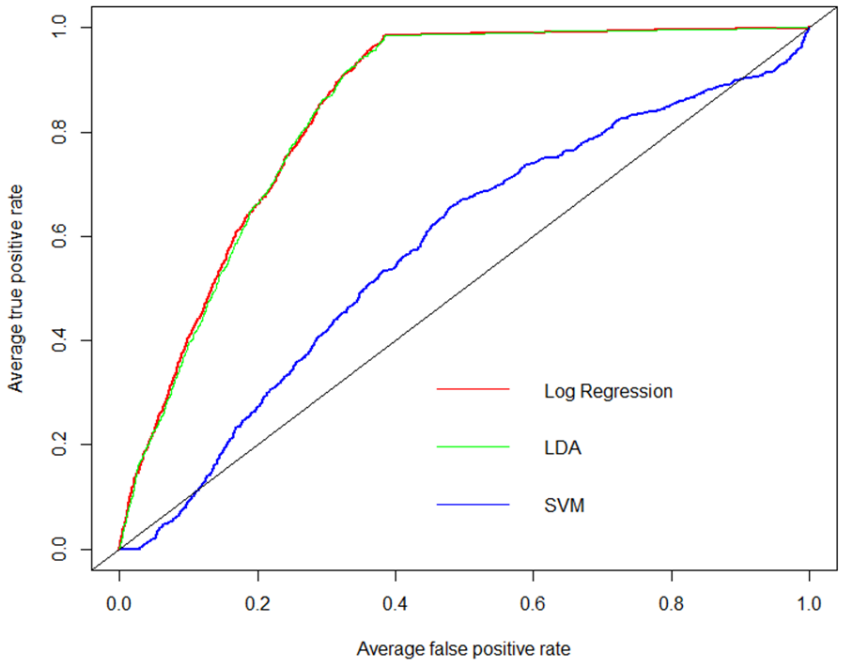


Figure 4.5: ROC curve for the three different classification methods for the balanced 2013 variable DNF.

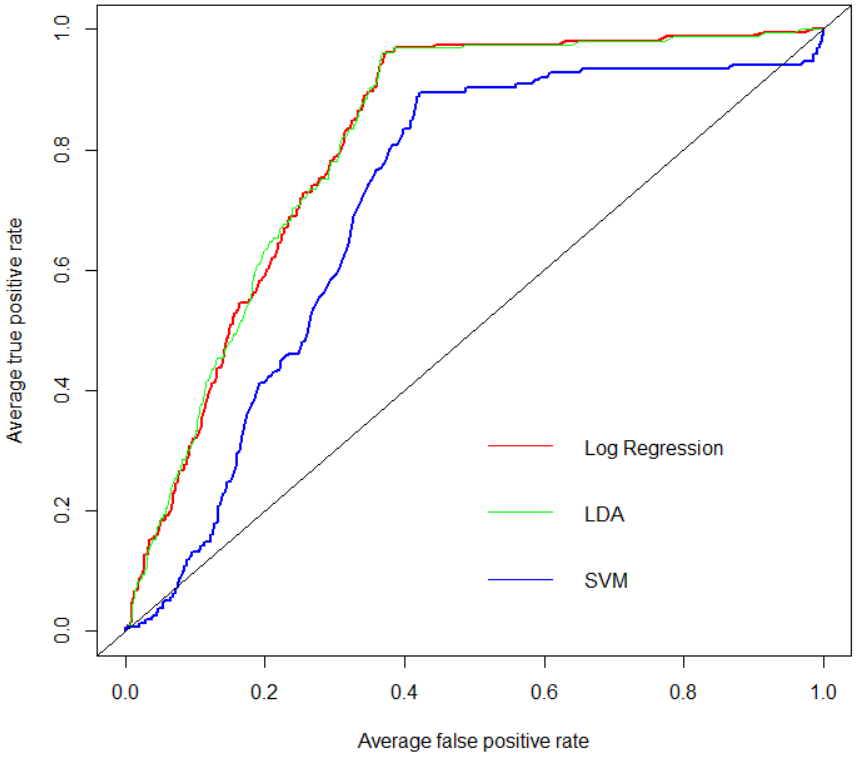


Figure 4.6: ROC curve for the three different classification methods for the balanced 2013 variable DNS.