UNIVERSITY OF
LINCOLN

Data Science

CMP3036M **Item 1**

Raymond Kirk
KIR14474219
14474219@students.lincoln.ac.uk

Contents

Data Programming Tools

R and RStudio

R is a programming language that is typically used for statistical computation and graphics. R is supported by the R foundation for statistical computing (R Core Team, 1993). The language syntax of R is similar to that of C, however semantically it shares a stronger relation to the functional paradigm that is encompassed by languages such as Haskell and Lisp (R Core Team, 2000). Within R, expressions are typed into an R command line prompt which will interpret the expressions and execute them. Usually users will type functions, a collection of many statements to be interpreted to solve problems algorithmically. Users can also write scripts to run a batch of statements and functions, stored on the file system under an extension '.r'. R supports procedural programming through the use of functions, sharing some object-orientated principles. One example of an object-orientated principle it employs is polymorphism, which in relation to R means functions can be applied to many different types of objects. An example of this is the 'plot()' function, which as shown in *Figure 1* will give different outputs depending on the data passed into the function (Maechler, 2017).

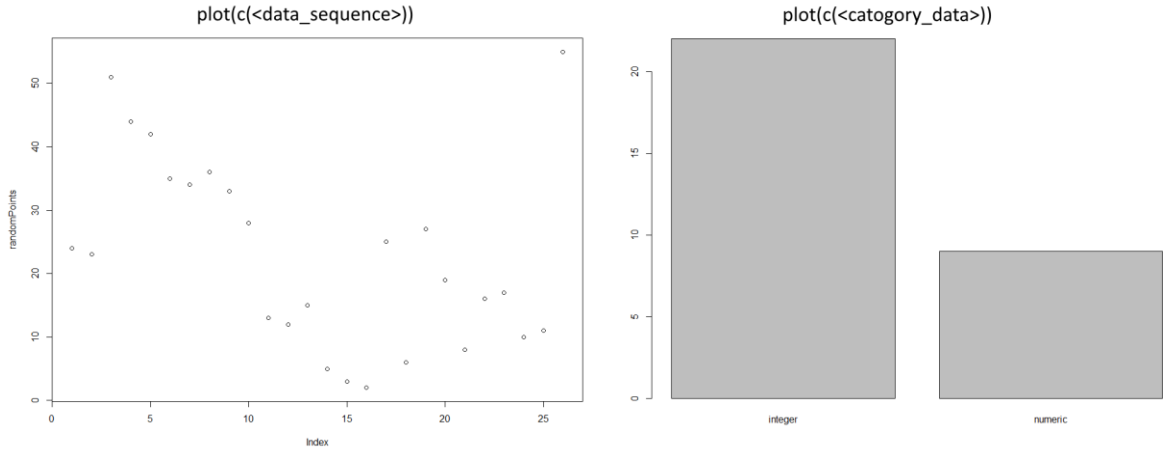

Figure 1 - Showing the different plot output based on input (Left - Numeric Data, Right - Categorical Data)

R is also the most popular language for developing statistical software, used in many different fields such as biochemistry and banking (Matloff, 2011). R has the capability to accommodate for a vast number of statistical problems. R integrates a suite of separate software facilities that enable effective data storage and handling, graphical functions for data representation, operators to perform calculations on arrays, a large collection of inbuilt tools for data analysis and user defined conditionals, loops and recursive functions (Ohri, 2014). One disadvantage to using R directly is that it doesn't have a native GUI (Graphical User Interface) which can slow down the development process and debugging processes. However if the user is not proficient with using CLIs (Command Line Interfaces) a number of GUI options are available to essentially wrap the functionality of R into a user-friendly interface (Matloff, 2008). One example of this is RStudio, an open source and enterprise ready software for R (RStudio, 2011). R and RStudio were used in the assignment for manipulation of the datasets, it provided a platform for good data representation, visualisation and modifications. Many techniques applied to the data were directly supported by R and didn't require external libraries, increasing the concentration of time on achieving better results. R was easily extended with packages distributed by CRAN, external packages can provide more functionality such as graphical or statistical techniques not packaged within R (Zhao, 2012).

Comparison of R and Other Tools

Python is another popular programming language used within Data Science, it emphasises simple and readable code. Similar to R, it has a large user base and a large community that follow it. However the community within Python is distributed throughout multiple areas since Python is generally used for general purpose tasks whereas R is almost exclusive to statistical tasks. This results in a disproportionate

amount of packages to community size relating to this assignment and data science. One trait that R and Python share is that they're both interpreted and don't have any coupled IDE. Python does not have any dedicated statistical in-built functions like that of R either, so generally is heavily dependent on other packages. However a great number of powerful packages are still available to Python and it can also interface directly with R objects and scripts. A benefit that Python has over R is that it can implement algorithms for production use within other applications such as web-apps or dedicated programs because of its flexibility to general purpose problems. This benefit usually correlates to software being developed in languages in R and then later ported to Python for production and integration with established systems (McKinney, 2012).

R Packages

caret

The 'caret' package, acronym, for Classification and Regression Training, is a set of functions that are generally used to create predictive models, many components that are included in this package are techniques that are arduous and routine processes within Data Science. This package facilitates a more streamlined approach to splitting data, selecting features, tuning models with resampling, variable importance and pre-processing. It aims to provide a more uniform framework for training models, standardising the syntax so common tasks are easier to perform (Kuhn, 2017). Three models that are later used in this report were wrapped by the 'caret' package, the three models were Logistic (Generalised linear model with binomial distribution), Naïve Bayes and Neural Nets. Their packages respectively were 'stats' (packaged with R), 'klaR' and 'nnet' (Venables and Ripley, 2002) (Chen et al., 2017). 'caret' also facilitated the assignment in finding correlating variables.

FSelector

'FSelector' (Romanski and Kotthoff, 2017) provided an array of functions in R that facilitated attribute selection in the assignment. Due to no context being present in the dataset, analysing attributes to use as predictors is more difficult; resulting in a more statistical approach to feature selection. Feature selection is a process of maximising the removal of redundant features in a dataset. Techniques such as Pearson's correlation, Spearman's correlation and Entropy were used to find combinations of important features within the dataset. 'FSelector' also provided methods in which the selected features from the aforementioned techniques could be subset into a smaller feature selection better representing the dataset. The one used in the assignment, split the features selected by the biggest difference in importance, resulting into two subsets of important and un-important features.

pROC

Comparing the performance of statistical models was a crucial step within the assignment, ascertaining the accuracy of a model estimates the future effectiveness and error rate. 'pROC' is a tool for "visualizing, smoothing and comparing receiver operating characteristic (ROC curves)" (Robin et al., 2015). It can compute partial area under the curve (AUC) which indicates the statistical significance of the ROC curve. AUC is commonly used to evaluate performance of classifiers for binary classification problems; fitting the criteria, the assignment was evaluated using this metric (Fawcett, 2004).

ggplot2

'ggplot2' was used within the assignment to facilitate clear and easy plot creation. It takes a different approach than some other graphical visualisation packages, such that of the inbuilt plot package. The approach that it takes is to declaratively define graphics based on 'The Grammar of Graphics', which allows complex plots to be constructed by specifying simple parameters (Wickham, 2009). Within the assignment it was used to plot numerous different visualisations from bar plots to corrolation plots as seen in *Figure 6.*

Training Data Pre-processing and Analysis

Summary

Two data sets were provided for the assignment, both a test set and training set. The training included 371 variables and 38010 observations, one of the variables being the response variable 'TARGET' comprised of two binary levels 0 and 1. Out of the 370 variables some were constrictive of accurate results, the following pre-processing steps aim remove these redundant variables from the training dataset. The column names of the dataset indicated no significance to data contained within them, generally being formed of similar words such as 'saldo', 'ind' and 'var' as shown in *Figure 2*.

```
[1] "ID"                    "var3"                    "var15"
[4] "imp_ent_var16_ult1"    "imp_op_var39_comer_ult1" "imp_op_var39_comer_ult3"
```

Figure 2 - Summary of nonsensical naming of data from training data

Pre-Processing & Data Analysis Findings

In order to reduce the dimensionality of the training set many methods were employed, resulting in a feature list representative of the dataset. The first pre-processing step that was undertaken was the removal of missing values, overall mean imputation (Donders et al., 2006) was implemented and applied to the dataset however no missing values were found. The data types of columns are important as the data is analysed through appropriate associative models. Within the training data only two data sets were present; numerical and integers as shown on *Figure 3*.

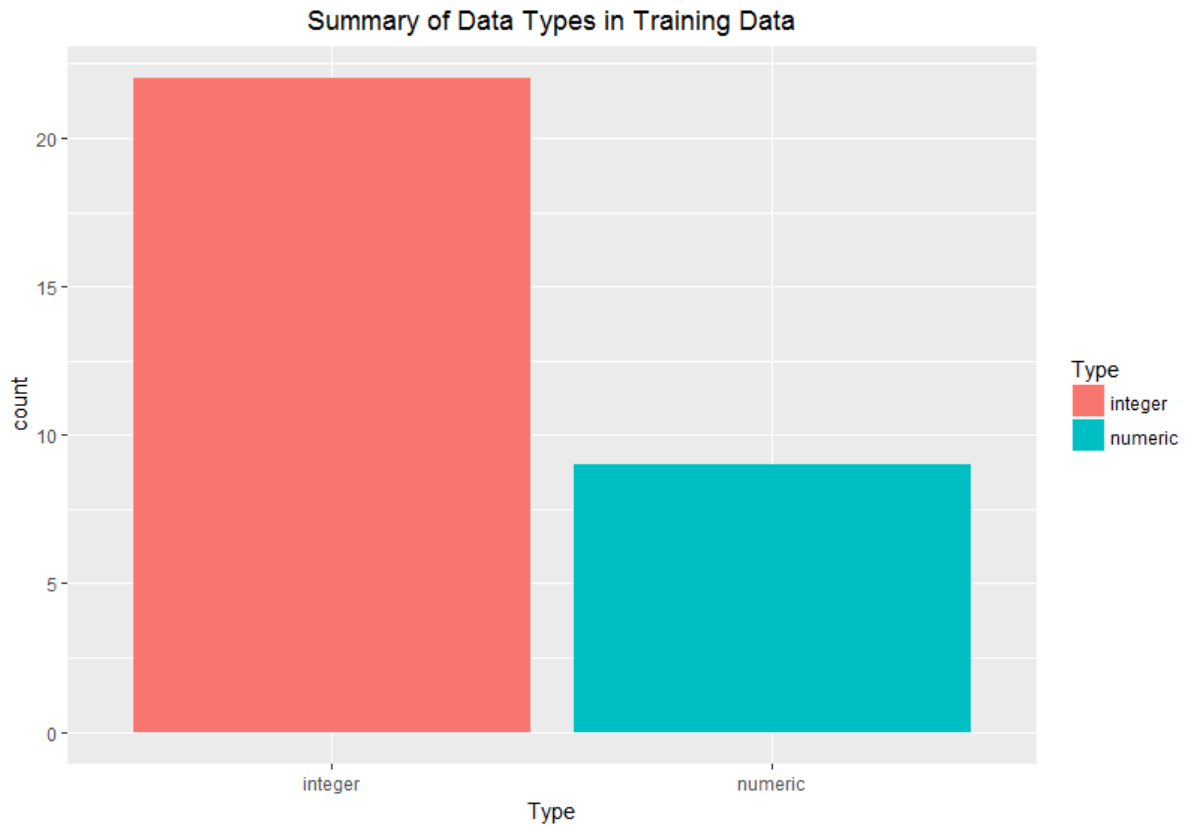

Figure 3 - Summary of Data Types in Training Data

A large population of the variables in the dataset had either zero or very low variance as shown in *Figure 4.* These variables have no significance on the 'TARGET' variable as they remain constant with only one unique value. Consideration was given as their removal could be unjustified if the data was split and the variables were poorly represented in the split data; since the training set represented the possible predictors in their entirety well they were identified and removed (Guyon and Elisseeff, 2003).

The deployment of this technique resulted in a substantial number of variables being removed. Before this stage 370 variables are present and after only 57 remain. In *Figure 4* below 262 predictors have been observed to be near-zero variance which classifies them as both having a low number of unique values compared to the number of values present and having a similar ratio of the two modal frequencies (Kuhn, 2008).

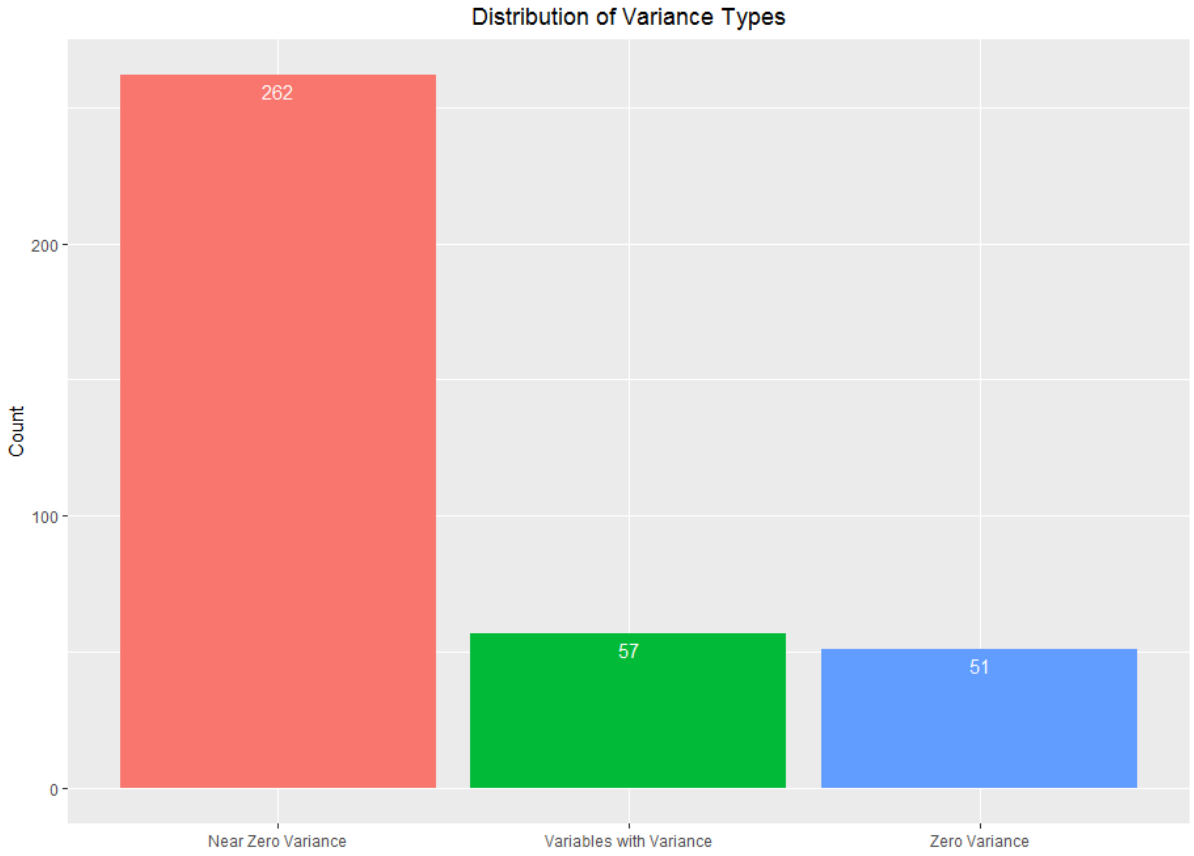

Figure 4 - Distribution of Variance Types

The 'ID' column was removed after removing zero variance variables, it was removed because it had one hundred percent uniqueness which wouldn't assist the prediction of the response variable. It was discovered that contained within the data set were some predictors that contained the same variables but had different column names. These variables were identified, validated and then removed. The two variables that were identical in the training data were 'ind_var37' and 'ind_var37_0' as shown in *Figure 5*. This reduced the total of variables to 55 not including the response variable. Duplicate information can produce misleading statistics which was the justification for their removal (Rahm and Do, 2000). This also progressed to reasoning for performing correlation analysis, shown on *Figure 6*, to provide more a more in-depth analysis of the covariance.

ind_var13_0	ind_var13	ind_var30	ind_var37_cte	ind_var37_0	ind_var37
Min. :0.00000	Min. :0.00000	Min. :0.0000	Min. :0.00000	Min. :0.00000	Min. :0.00000
1st Qu.:0.00000	1st Qu.:0.00000	1st Qu.:0.0000	1st Qu.:0.00000	1st Qu.:0.00000	1st Qu.:0.00000
Median :0.00000	Median :0.00000	Median :1.0000	Median :0.00000	Median :0.00000	Median :0.00000
Mean :0.05293	Mean :0.05167	Mean :0.7349	Mean :0.07156	Mean :0.06446	Mean :0.06446
3rd Qu.:0.00000	3rd Qu.:0.00000	3rd Qu.:1.0000	3rd Qu.:0.00000	3rd Qu.:0.00000	3rd Qu.:0.00000
Max. :1.00000	Max. :1.00000	Max. :1.0000	Max. :1.00000	Max. :1.00000	Max. :1.00000

Figure 5 - Statistical Data for Variables with Similar Names

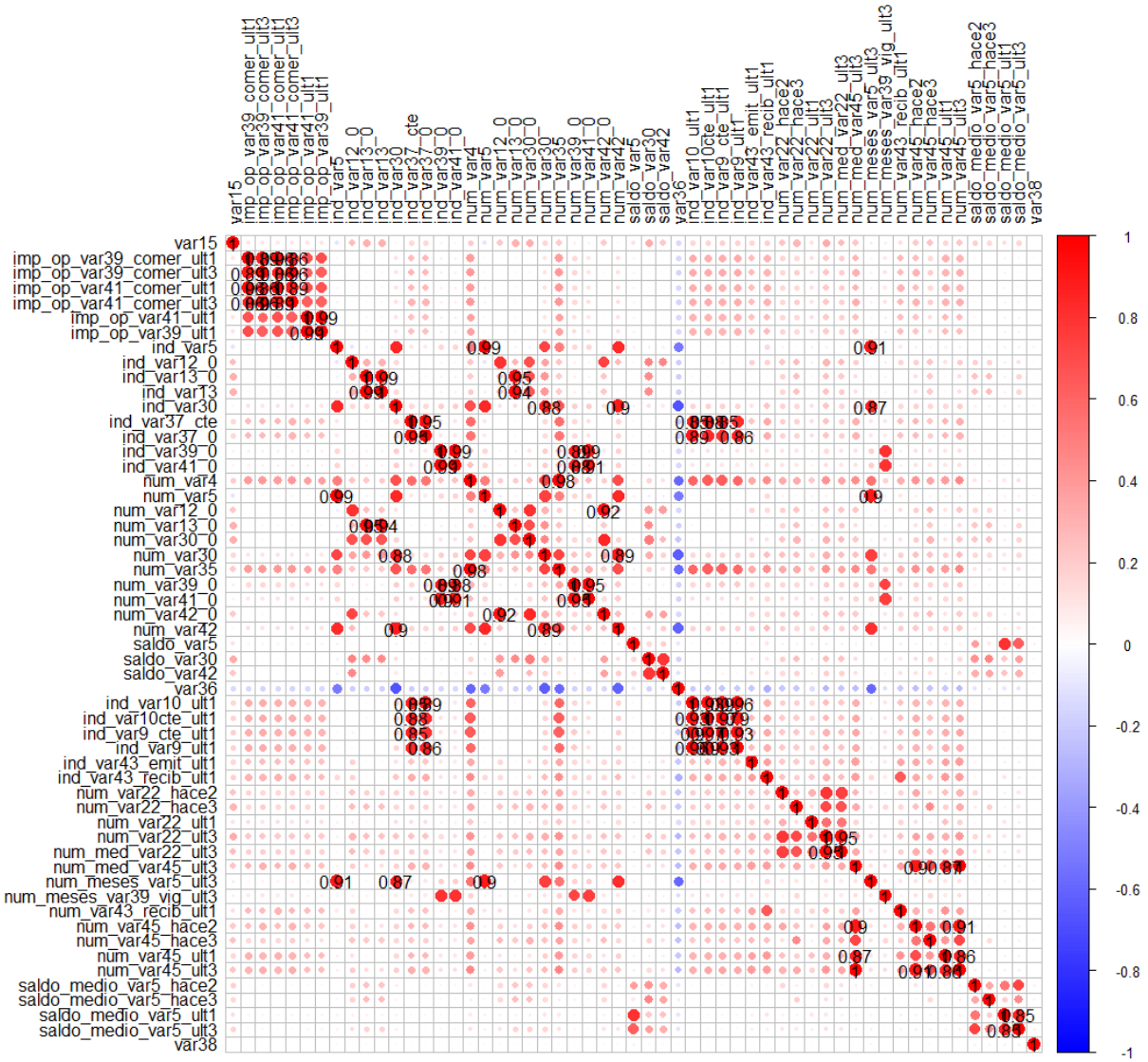

Figure 6 - Correlation between features in training data

Some models better utilise high large populations of correlating features and some benefit from fewer correlating features (Höskuldsson, 2001). Within the training data it was found that a large percentage pair-wise correlations also shared names, which is an indicator they are duplicated using unnecessary resources. When two variables were found that were highly correlating, one of them would be indexed to be later removed from the dataset. The decision on which variable to remove was made based on which one had the largest absolute mean correlation. Only variables that had a considerable correlation were removed from the data set in the assignment, this led to only one feature being removed. Only extremely high correlating features were removed because they are directly proportional to increasing accuracy and still removing some increases accuracy of column selecting for other features. *Figure 6* shows the correlating features, where red is more correlating and blue is less correlating. Covariance of the columns was calculated to determine the measure of correlation between them. Statistical correlation between two sets X and Y is given as the derived quantity where $i, j = 1, 2, \dots, n$, and $cov(X_i, X_j)$ denotes the covariance of the two variables. Elements V_{ii} are within the covariance matrix that is given by the function 'cov' in the built in stats package of R (Weisstein, 2017).

$$cor(X_i, X_j) = \frac{cov(X_i, X_j)}{\sqrt{V_{ii} V_{jj}}},$$

At this stage the data set is cleaner, with no missing data and most of the redundant information has been removed, allowing for a clearer refinement and granularity for further feature selection. The following processes aim to split the number of features into a subset to achieve the greatest accuracy on a classification model, and then split the data into two subsets for training and testing for cross-validation and model performance measurements. Recursive feature elimination (RFE) is a greedy stepwise selection approach to subset features (Derksen and Keselman, 1992). It's the process of selecting an estimator that assigns importance (weights) to attributes, then training the estimator on the initial set of features until it has assigned weights to all of the features. When trained, larger weights are kept while smaller weights are incrementally pruned until a specified number of features remain (Granitto et al., 2006). Typically termination of RFE is determent of cross validation. Random forest was used as the estimator in the implementation to find the optimal number of features, while using cross-validation to accurately measure the accuracy at each stage. *Figure 6* shows the optimal number of variables to be eight over the 55 possible predictors.

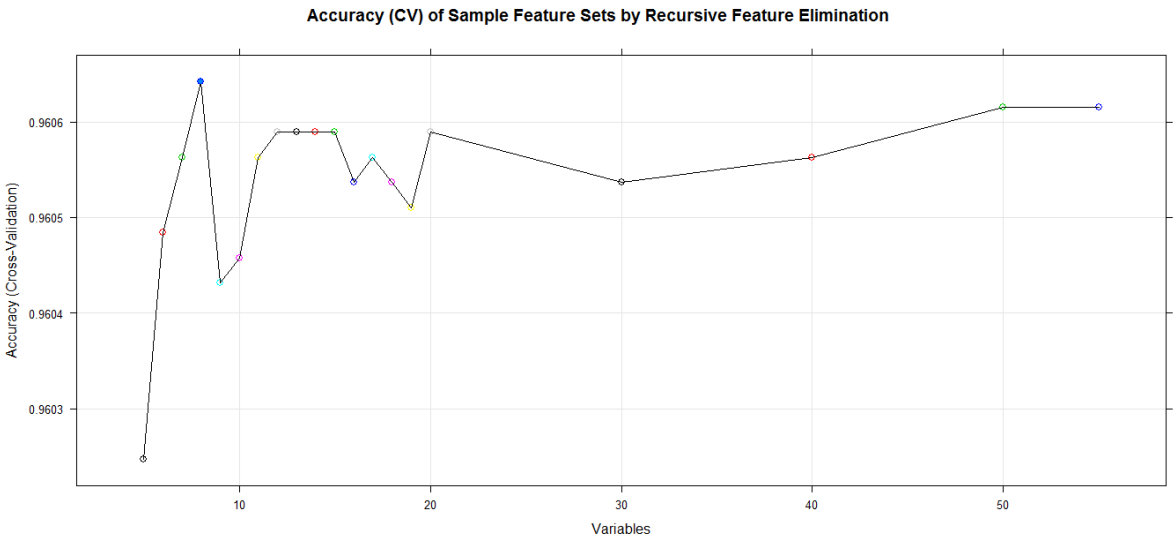

Figure 7 – Recursive Feature Elimination

The problem with RFE is that it's not generalised in respect to applicability to other models. This is why the 'information.gain' function in 'FSelector' was used to calculate features based on entropy. Subset features resultant of this approach expose the relationships between the features rather than being resultant of assumptions on a model. Information gain is the expected reduction of entropy caused when segmenting (splitting) features respective to a class variable. Initially the entropy of the class variable is calculated, then the data is split by features and entropy is calculated for each split. The information gain is then given as $Gain\ (T, X) = Entropy(T) - Entropy(T, X)$ where T is 'TARGET' and X is the feature split (Amro, 2009).

Predictive Models

After the pre-processing stages the data is now stored in one place, for cross validation later and reservation of future predictability it must be split into two-sections. A common problem with data splitting is the proportions reserved for training and testing. Too little information reserved for testing or training is detrimental to the ability to predict future observations or validate the data. Optimal splitting of the data would include consideration to the trade-off of both these points (Picard and Berk, 1990). For observations in the implementation a stratified sample of 80/20 ratio was applied respectively as suggested in *"A Critical look at procedures for validating growth and yield models"* (Huang et al., 2003), the sample created shared the same class distribution as the original dataset to preserve the model fit.

Logistic Regression (Classification)

Logistic regression is an extremely popular form of regression in statistics, originally developed by statistician David Cox in 1958 (Cox, 1958). Cox describes in his original paper that binary classification is dependent on one or more independent variables. In context to the implementation the dependant variable was 'TARGET' and independent variables were the feature selection from the processes aforementioned. The logistic model is used to estimate the probability of the dependant variable based on the feature variables (independent). The dependant variable is observed to have two levels in the training set (0 and 1) in which the outcome is described as being the result of a Bernoulli trial; a random experiment with two possible outcomes (Feller, 1968). The training set independent variables contain continuous numeric data. The output of a logistic function is dichotomous; always bounded between the values zero and one, which can be interpreted as a probability (Hosmer et al., 2013). The logistic model can handle multiple independent variables unlike the linear model. In the implementation multiple features are used to classify 'TARGET'. Mathematically multiple linear regression is stated as $y = \beta 0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p} + \varepsilon$ where y is the response is variable, x is the predictor, β is the intercept and ε is the error. The probability of the dependant variable being true $(P(x))$ with one independent variable in context to the assignment can be expressed in the following logistic function.

$$P(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Due to the classification nature of logistic models they are extremely applicable for prediction the binary dependant variable in the assignment. Predictions are given for logistic regression as a probability that the response variable is true, formatting values similar to that of 'TARGET' is simply derived as the highest probability between the two binary events; 'ONE' being $P(x)$ and 'ZERO' being $1 - P(x)$. In the implementation the model was constructed with the 'glm' function, specifying parameters as x and y, the independent and dependant variable. Cross validation was used to assess the accuracy of the models created. The 'glm' function is used for generalised linear models, for multiple logistic regression in the implementation the family was set to binomial with link logit (Statistical Data Analysis, 2017).

Naïve Bayes

Naïve Bayes classifier is a collection of probabilistic classifiers that apply Bayes theorem, which describes event occurrence probabilistically of prior knowledge that could affect the occurrence. This describes a concept called conditional probability which is simply the probability of an event occurring given another event has happened. The simple mathematical equation for Bayes theorem where A and B are events and $B \neq 0$ is $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ also stated in Bayesian terminology as $posterior = \frac{prior \times likelihood}{evidence}$ (Rosen, 2007). A classifier is a defined as a function f that will map feature vectors $x \in X$ from a feature space X to class labels $y \in \{1, \ldots, C\}$. The feature space X is usually either a real number $\mathbb{R}$ or a binary number $0|1$ (Murphy, 2006). The caveat of the naïve Bayes classifier is that it's fundamentally assuming all features are independent given the dependant class (mutually exclusive), although Rish (2001) states that it is remarkably successful in practice, evidencing medical diagnosis and text classification as particularly successful implementations. Naïve Bayes is more effective when the feature space is comprised of low correlating values, when highly correlating values are present their importance is inflated. This is further justification for the pairwise correlation feature reduction aforementioned (Shi and Manduchi, 2003).

Naïve Bayes was implemented in the assignment using the 'klaR' (Roever et al., 2014) package, it was especially advantageous due to its ability to predict response of testing data. It is also well known for good performance in multi-class predictions. Contextualised to the training data, naïve Bayes performed well, the normally degrading caveat of variable independence is well suited to the problem space. Features and their empirical correlation was impossible to ascertain from the data provided, suitably

matching with the integral working of Naïve Bayes. *Figure 8* shows the feature density of 'var36' within Naïve Bayes.

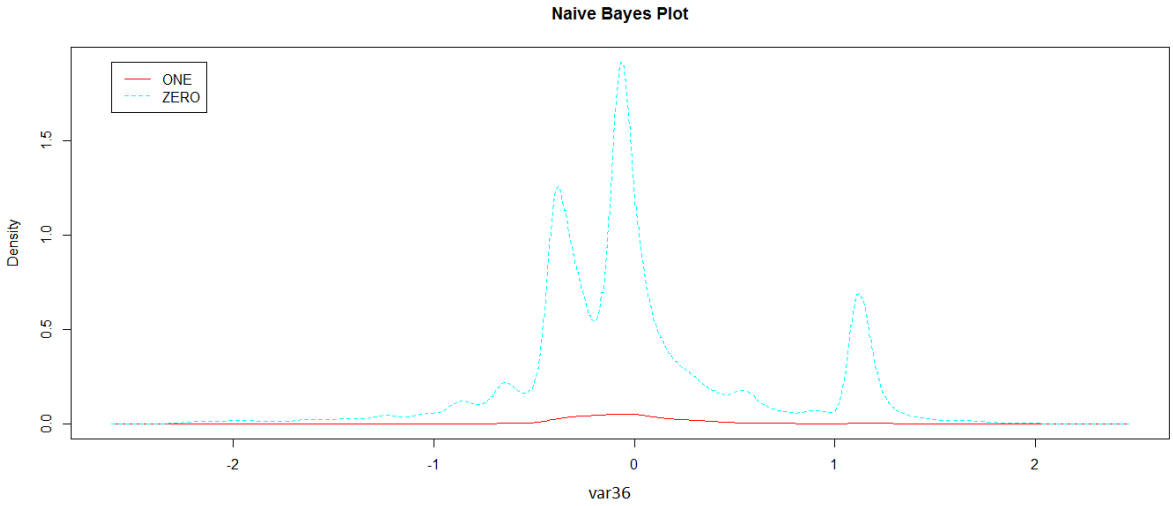

Figure 8 - Plot of Feature Density in Naive Bayes

Neural Network

Neural networks are systems that are modelled similarly to the understanding of the human brain and nervous system. Unlike many approaches in computer science and data science they are self-learning and trained. This is contrasting to the traditional hard programmed systems. Problems that are generally hard to model using explicit programs may be easily solved by neural networks (Gemal et al., 1992). In relation to data-science neural networks have been observed to be good at feature detection that are difficult to portray using available explicit interfaces (Levi and Weiss, 2004). Typically the neural network is made up of layers in either two or three dimensions of interconnected nodes. Each layer is connected to the next layer where signals will propagate in this direction as shown in *Figure 9.*

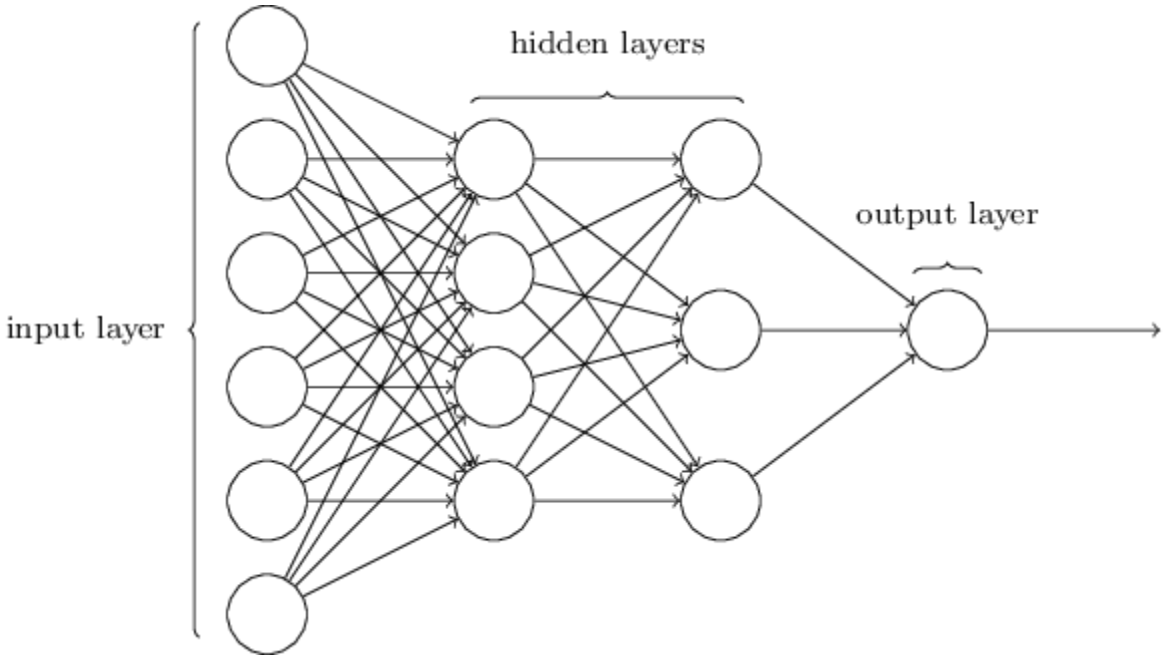

Figure 9 - Architecture of a typical neural network (Nielsen, 2017)

The goal of a neural network is to arrive at a problem solution through a process similar to how the human brain would, the learning process of a neural network typically spans a large number of cycles/iterations to reinforce task solutions. *Figure 8* clearly labels descriptors or each of the layers, the

hidden layers are essentially just nodes that do not receive input or output as that of the far right and left nodes on the figure. The input layer of a neural network is descriptive of the data used to find a solution. If the purpose of a neural network is to classify if the intensity of a grayscale 2x2 image is black or white then 4 input neurons would exist (one for each pixel) and the value contained at each neuron (typically bounded between the values 0 and 1) would represent the intensity. The output of the neural network in this example would have a value or either true or false dependant on the node value $F =< 0.5 | T =\geq 0.5$. This is a basic description of how neural networks function and arrive at solutions.

Neural networks have been observed to give similar results in classification models to that of linear discriminant analysis methods in two level classification problems, however linear analysis predictably give better results. In the implementation they were implemented using the neural network package wrapped by 'caret' (Kuhn, 2017). A contrasting point and the original justification of inclusion in the assignment was due to the fact neural networks are found to perform better when the number of features is higher and also when the classification has greater complexity and reduced feasibility modelled traditionally (Kumar,2005).

Critical Reflection

The assignment posed some challenging problems that had to be overcome to arrive at the solution presented in this report. The author of this paper had minimal experience in R which constricted the understanding of some of the functions due to the syntax being unfamiliar, this was overcome eventually but initially led to some misunderstanding of function outcomes. General understanding of the inner workings of models was gained early in the module from lectures, however there is a vast number of accepted of models in the data science community and optimal usage of each is a topic where immense research is available. This resulted in a resource expensive process of implementations of several models throughout development iterations.

Eventually the three models used comparatively in the assignment were Logistic Regression, Naïve Bayes and Neural Network. Neural Networks weren't delivered in the lectures or workshops but many principles taught throughout were shared and contributed to understanding how they function and how they could be implemented in R. An extensive number of user friendly packages are available in R, this facilitated the development of the implementation. However it was quickly observed by the author that processes undertaken by the packages performed many processes automatically, so a manual implementation helped in conceptual understanding.

Model Evaluation

The three models that were created and evaluated in the assignment were Neural Networks, Naïve Bayes Classifier and Logistic Regression. All of the algorithms discussed have their advantages and disadvantages as aforementioned. Reflection on what would be changed if time permitted to the implementation would be further refinement on feature selection and data analysis for each of the models, the features inputted to each model made the most considerable impact of model performance. At each stage in the assignment the accuracy of the models were assessed and critically compared in order to formulate a base-line for improvements/refinements.

Cross validation was a critical tool in assessing the models, it's a measure of how well the model will predict future values (Kohavi, 1995). It is a statistical tool commonly deployed when the objective is to predict values of data that is not known based on known data. In context to the assignment it provided a measure of how accurate the model would be at predicting values of the test partition of the training data set and the test data that was to be populated with predicted 'TARGET' variables. Cross validation also helped in limiting the effect of over fitting on the models. It was generally applied to all three of the models with 10 folds. A major benefit of performing cross validation is that it increases the number of observations that the models can be trained on since there isn't a need to split the data into two partitions prior for future validation as this happens during the training phase (Xu and Liang, 2001).

The type of cross validation used within the assignment was k-fold cross validation. This is a non-exhaustive example of cross validation as it does not compute all permutations of data splits. In k-fold partitioning the sample is split into k subsets and a single subset is retained so that it can be later used to validate data for the model. The results from the testing in the assignments were averaged of all k-samples (Boyd et al., 2013).

For each model predictions were made on the split training set data, the test set and the respective ROC curves were plotted. ROC stands for receiver operating characteristic curve and is a plot that gives an indication of how well binary classification performed. The ROC curve is plotted by comparing the sensitivity (true positive rate) and specificity (false positive rate). Sensitivity is the proportion of correctly identified values of the response variable, specificity is the proportion of incorrect values predicted that have been identified as incorrect (Cortes and Mohri, 2003); both are statistical measures that indicate binary classification performance. Analysing the ROC curves produced by the models in the assignments allowed optimal models to be chosen. *Figure 10* indicates how a ROC curve is calculated.

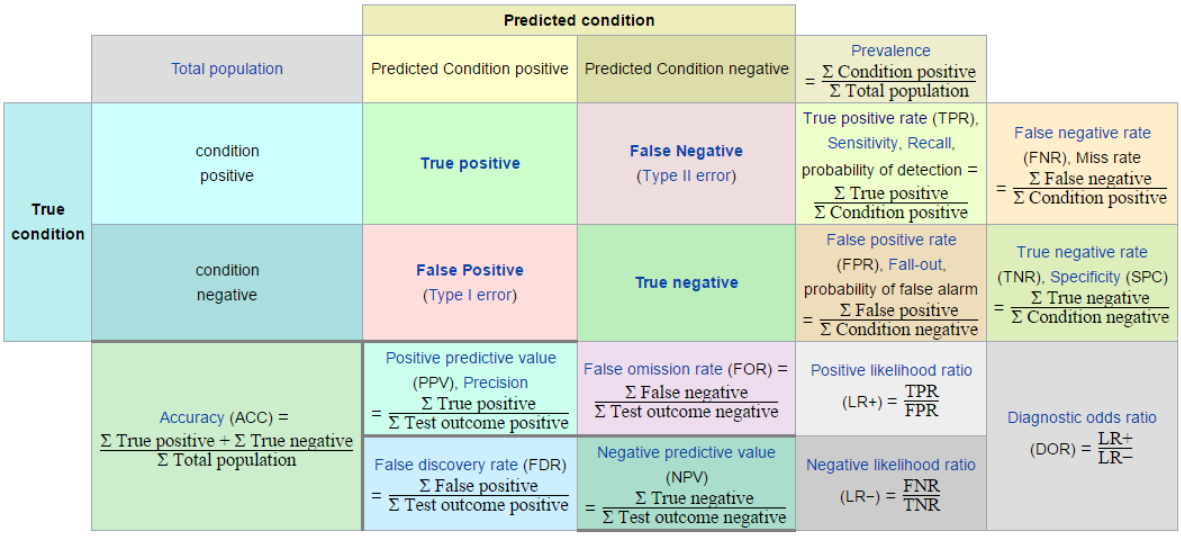

Figure 10 - ROC Curve Calculation (Wikipedia, 2017)

AUROC stands for area under the receiver operating characteristic curve, it is the expected proportion of positives ranked before a uniformly drawn random negative. Since the roc curve is bounded on its plot between 0-1 sensitivity and specificity the AUC will lay between 0 and 1. If an AUC value is high it means the proportion of positives is scored higher than all of the negatives, as the AUC approaches 0.5 it is said to be more random (Flach, 2011). As ROC is said to be a performance measure of binary classifiers, AUC is used to determine the performance of models with a single value and is comparable to other models to determine the optimal choice.

Higher AUC values a determinate of a model performance predicating more accurate classes (Pencina, 2008). *Figure 11* below shows the ROC curves of the models used in the assignments and their respective AUC values. Naïve Bayes ranked the highest and was the model used to predict the values in the submission file. The AUC for Naïve Bayes incrementally increased over the duration of the assignment, this was directly a consequence of being able to rank optimal models. A base-line for improvement was easily obtained and indicated an improved model.

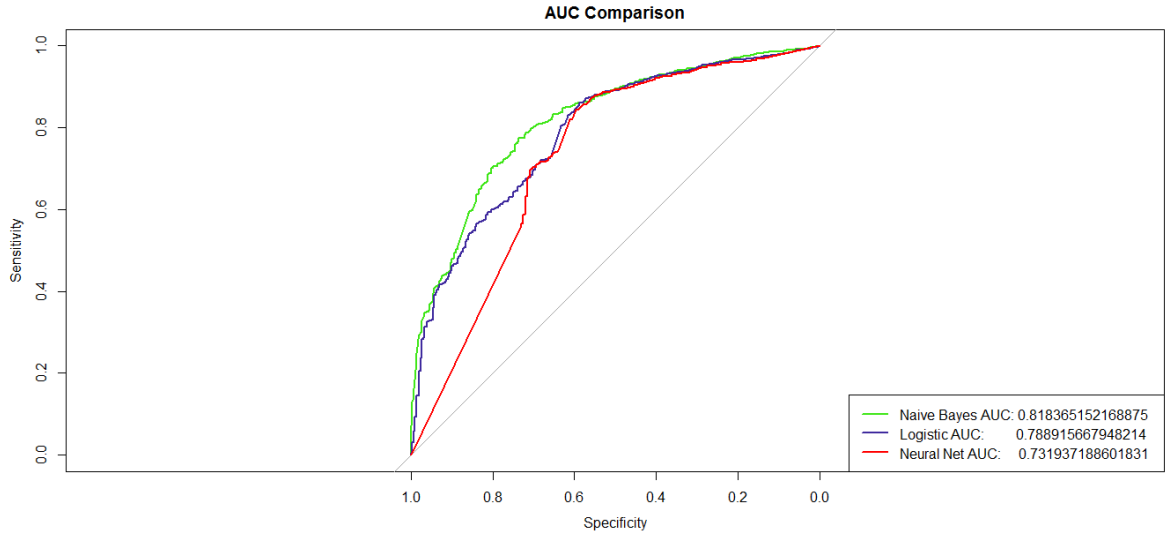

Figure 11 - ROC Curve and AUC Values for all models

References

Amro, A. (2009) *What is "entropy and information gain"?* StackOverflow. Available from http://stackoverflow.com/questions/1859554/what-is-entropy-and-information-gain [accessed 25 Jan. 2017]

Boyd, K., Eng, K.H. and Page, C.D. (2013) *Area under the precision-recall curve: Point estimates and confidence intervals.* Springer Berlin Heidelberg: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 451-466.

Cortes, C. and Mohri, M. (2003) *AUC optimization vs. error rate minimization*. NIPS 9, 10

Cox, D.R. (1958) *The regression analysis of binary sequences.* Journal of the Royal Statistical Society. Series B (Methodological), 215-242.

Derksen, S. and Keselman, H.J. (1992) *Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables*. British Journal of Mathematical and Statistical Psychology, 45(2) 265-282.

Donders, A.R.T., van der Heijden, G.J., Stijnen, T. and Moons, K.G. (2006) *Review: a gentle introduction to imputation of missing values*. Journal of clinical epidemiology, 59(10) 1087-1091.

Fawcett, T. (2004) ROC graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1) 1-38.

Feller, W. (1968) A*n introduction to probability theory and its applications*. New York: John Wiley & Sons.

Flach, P.A. (2011) *ROC analysis*. Springer US: Encyclopaedia of machine learning. 869-875

Geman, S., Bienenstock, E. and Doursat, R. (1992) *Neural networks and the bias/variance dilemma*. Neural computation, 4(1) 1-58.

Granitto, P.M., Furlanello, C., Biasioli, F. and Gasperi, F. (2006*) Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products*. Chemometrics and Intelligent Laboratory Systems, 83(2) 83-90.

Guyon, I. and Elisseeff, A. (2003). *An introduction to variable and feature selection*. Journal of machine learning research 3(1) 1157-1182.

Höskuldsson, A. (2001) *Variable and subset selection in PLS regression.* Chemometrics and intelligent laboratory systems, 55(1) 23-38.

Hosmer Jr, D.W., Lemeshow, S. and Sturdivant, R.X. (2013) *Applied logistic regression (Vol. 398).* John Wiley & Sons.

Huang, S., Yang, Y. and Wang, Y. (2003) *A critical look at procedures for validating growth and yield models*. Modelling forest systems, 271-293.

Kohavi, R. (1995) *A study of cross-validation and bootstrap for accuracy estimation and model selection.* 14(2) 1137-1145.

Kuhn, M. (2008) *Caret package*. Journal of Statistical Software, 28(5) 1-26.

Kuhn, M. (2017) *caret* [software]. GitHub: Max Kuhn. Available from https://github.com/topepo/caret [accessed 23 January 2017].

Levi, K. and Weiss, Y. (2004) *Learning object detection from a small number of examples: the importance of good features*. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 2(1) II-II

Maechler, M. (2017) *R: Generic X-Y Plotting.* Statistical Data Analysis. Available from https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/plot.html [accessed 23 Jan. 2017].

Matloff, N. (2008) *R for Programmers.* Los Angeles: University of California.

Matloff, N. (2011) *The art of R programming: A tour of statistical software design*. California: No Starch Press.

Murphy, K.P. (2006) *Naive Bayes classifiers*. University of British Columbia.

McKinney, W. (2012) *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. Sebastopol, California: O'Reilly Media.

Nielsen, M. (2017) *Neural Networks and Deep Learning*. Available from http://neuralnetworksanddeeplearning.com/chap1.html [accessed 26 Jan. 2017].

Ohri, A. (2014) *R for Cloud Computing.* New York: Springer.

Pencina, M.J., D'Agostino, R.B. and Vasan, R.S. (2008) *Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond*. Statistics in medicine, 27(2) 157-172.

Picard, R.R. and Berk, K.N. (1990) *Data splitting.* The American Statistician, 44(2) 140-147.

Rahm, E. and Do, H.H. (2000) *Data cleaning: Problems and current approaches*. IEEE Data Eng. Bull, 23(4) 3-13.

R Core Team (1993) R language [software] New Zealand: R Development Core Team. Available from https://www.r-project.org/ [accessed 23 January 2017]

R Core Team (2000) *R language definition*. Vienna, Austria: R foundation for statistical computing.

Rish, I. (2001) *An empirical study of the naive Bayes classifier*. IBM New York: IJCAI 2001 workshop on empirical methods in artificial intelligence 3(22) 41-46.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J., Müller, M. (2015) *pROC* [software] CRAN. Available from https://CRAN.R-project.org/package=pROC [accessed 24 January 2017]

Roever, C., Raabe, N., Luebke, K., Ligges, U., Szepannek, G., Zentgraf, M. (2014) *klaR* [software] CRAN. Available from https://CRAN.R-project.org/package=klaR [accessed 23 January 2017]

Romanski, P., Kotthoff, L. (2017) *FSelector* [software] CRAN. Available from https://CRAN.R-project.org/package=FSelector [accessed 24 January 2017]

RStudio, Inc (2011) *RStudio* [software] Boston, MA: RStudio, Inc. Available from https://github.com/rstudio/rstudio [accessed 23 January 2017]

Statistical Data Analysis. (2017) *Fitting Generalized Linear Models.* Available from https://stat.ethz.ch/R-manual/R-patched/library/stats/html/glm.html [accessed 26 January 2017].

Venables, W. N., Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. New York: Springer.

Weisstein, E. (2017) *Statistical Correlation.* MathWorld. Available from http://mathworld.wolfram.com/StatisticalCorrelation.html [accessed 25 January 2017)

Wickham, H. (2009) *ggplot2: Elegant Graphics for Data Analysis* [software] New York: Springer-Verlag. Available from http://ggplot2.org [accessed 24 January 2017]

Wikipedia (2017) Receiver operating characteristic. Available from http://en.wikipedia.org/wiki/Australian_patent_law [accessed 26 January 2017].

Xu, Q.S. and Liang, Y.Z. (2001) *Monte Carlo cross validation.* Chemometrics and Intelligent Laboratory Systems, 56(1) 1-11.

Zhao, Y. (2012) *R and data mining: Examples and case studies.* Cambridge: Academic Press.