ECOS Project: Using Data Analysis to Improve Marketing Response at D.C. Thomson

Tom Davenport

September 2016

Dissertation submitted in partial fulfilment for the degree of
Master of Science in Big Data

Computing Science and Mathematics
University of Stirling

Abstract

Data analysis tools and techniques provide a substantial opportunity for business. Companies have become able to store more information not only about user's personal information, but also around their interaction with the company, through their response to emails and to how they behave on their site. This has created a vast pool of data that when analysed appropriately can provide useful evidence for marketing decision making. The data however needs to be put through a series of processes to ensure that it is of a sufficient quality that analysis is grounded in fact and will be of business value.

This project was undertaken for the DC Thomson company to examine the response to emails sent during the Christmas season from October to December of 2015. It created a series of data models that would highlight what important factors were involved in making an email successful, and whether there were patterns amongst groups of users that purchased from emails. The objectives of the project were to create one model that could reflect the structural factors that made emails successful, and another model that would show what factors had an influence in leading a customer to purchase. It involved a multi-level analysis of different sources of data, creating and assessing the quality of datasets that could be used to generate data models.

The approach to modelling involved examining current methods of email data mining, and following the CRISP DM standard for approaching data mining projects. The development of the project was undertaken within an Agile environment, using daily stand up meetings and weekly Sprint meetings. Different methods for creating the analysis were considered, with the majority of data cleaning being undertaken in R, data scraping and merging using Pandas in Python. The final models were developed using R. These models generated were decision trees, a white box model that can effectively show the relationships within the data that lead to a particular outcome.

The results of the project were presented to senior management. This included suggestions of how they should test campaign emails in future, by what structural influences the model had suggested generated a good response. An audit of missing data from the user campaign was also presented, showing the importance of each variable and its proportion of missingness from the set.

The process of building these models was documented effectively which will allow them to be recreated and extended with further detail to study other campaigns. The suggestions from the models will have an impact on the organisation's decision making process in the future.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my University project except for the following:

- The removeduplicatesorderedSeq function in section 4.1.1 is adapated from [26].
- The postcode validation expression used in section 4.2.2 is from [32].

Signature **Date**

Acknowledgements

I would like to thank the Datalab for their funding and the opportunity to complete this degree. I'd like to thank my supervisors Chris Kelly and Andrea Bracialli for all their time and advice. I'd also like to thank my parents for all their support over the last two years.

Table of Contents

List of Figures

1 Introduction

This project was carried out for the DC Thomson media company who are involved in publishing, regional news, magazines, data centres, genealogy sites and also Scottish gifts and memorabilia. It aimed to improve response to marketing campaigns sent out at Christmas by the website DC Thomson Shop. Using data provided by their Single Customer View (SCV) model the project examined how customers reacted to email campaigns sent out during their Christmas season of October to December of 2015.

After gaining an understanding of the data available, two areas were considered. The first section, called the Campaign section, studied the structure of emails by analysing how users responded to different links to the website included in each email.

The second section, called the User section, took this idea further by looking at every user and their response (when they opened, whether they purchased etc.) to each marketing email sent during the Christmas period and was then merged this with everything else that the company knew about the user (for example, the total number of emails they had been sent, their geographical location, the age of their account and so on).

This brief presented several challenges, including creating datasets for analysis by scraping data from emails, cleaning data, interpreting missing values, building data models, analysing results and presenting findings effectively. The models and findings generated by the project will be used by DC Thomson to inform how future emails may be structured and the organisations data collection procedures of facts about users.

1.1 Background and Context

E-commerce, selling products or services over the internet, has become one of the main ways that retail companies interact with consumers to make a profit.[1] The internet enables companies to have an interface with customers that is open at any time, without customers needing to wait or be physically present in an establishment. It saves the organisation money in physical presence and is usually more convenient for the customer and attracts customers from all over the world. An online presence for an organisation also allows a company to engage with their customers using email marketing.

This is a powerful form of marketing as it provides a pool of customers that have physically registered that they are interested in the content and marketing material can be sent out at a very low cost to the organisation.[2] The aim for companies such as DC Thomson is to take advantage of these factors to generate more sales from their online platform.

Selling goods and service through an online platform also offers significant opportunities for data capture and analysis. Systems such as Sailthru,[3] as used by DC Thomson, allows a customer's interaction with the site and how they responded to every email they have been sent to be tracked and stored. DC Thomson approaches this capture as a Single Customer View (SCV). An SCV is where the organisation stores all the data that it ever captures about a customer from any channel in one data store, allowing the data about a customer to be queried or used by any department in the organisation.[4]

This kind of store of data is very valuable for an organisation and is suitable for data mining.[1] Data mining is the process of discovering the relationships that exist within a dataset (for example, testing whether there are patterns in the gender and location of customers of customers that buy or do not buy from an email). There is a sufficient variety of data is available in a rich format where relationships between variables can be identified. There is a sufficient volume of data, as there are many customers that interact with the site. The dataset is also constantly being updated as customers interact with the site which means that there is less opportunity for the dataset to become out of date and it will remain relevant.

This project aimed to identify the patterns in the structure of an email and whether there were patterns in general information about users that purchased or did not purchase after receiving emails. Acquiring knowledge of these customer behaviours can provide a company with a competitive advantage as they can structure their emails better, target customer groups and personalise the content in a way that can increase engagement with a campaign.

One of the main challenges to identifying these relationships is the creation of a data set from the data available that is of a sufficient quality and of enough depth that would allow a data mining model to pick them up. E-commerce data can suffer from problems such as bias and incompleteness and part of this project included finding where these problems had occurred and how solutions to these might be implemented.

R and Python provide many libraries for exploring data and creating new variables that would allow a dataset that is clean and correct to be generated. This project used R to explore and clean the user data section and used Python to scrape variables from the HTML of emails sent during the October to December Christmas period of 2015. Data sources were merged using Python's Pandas module, and the importance of variables and decision tree models were generated using R's Rattle package.

1.2 Scope and Objectives

The project was a detailed analysis of DC Thomson's E-commerce data.

The objectives of the project were to:

- Create datasets that modelled the response to emails and the characteristics of users
- Build models that identify the relationship between the structure of an email and the response by users
- Build models that identify the characteristics of users that purchase from emails
- Provide data driven suggestions for improving the structure of emails
- To create a missing data audit, identifying the importance of variables and reporting their relative absence from the data

The scope of the project limited data collection to sources that were present from the 1st of October to the 31st December 2015. The project only examined data sources that were connected to DC Thomson Shop and the users of the site. Creating a predictive model that would be implemented for general use by the company was not an aim of the project, nor was producing explicit clusters of users for marketing purposes.

1.3 Achievements

Some knowledge had to be gained to carry out the project. This included learning the R programming language, the data scraping features of Python's RE module, and the data merging features of Python's Pandas module.

The data models that were built were documented effectively. The path to creating these models was also documented in a way that others could follow the process taken to achieve them. Results of the models were also presented to senior management, who were able to take these suggestions into account and influenced their data collection and email testing strategy. Infographics of the main findings of the project were also created and presented to management at the end of the project to summarise the main findings.

The project has met all the objectives laid out above, and the accompanying documentation has shown how the models can be replicated in future. The suggestions from the project will allow the company to make data-driven decisions about their marketing strategy and will have an impact on the planning of future Christmas campaigns.

The implementation can be extended to take into account further data sources, explore other relationships or to create predictive models that can predict the success of a planned campaign. The models that are built could also be used to compare different seasonal marketing campaigns, to identify if different factors are important at different times of the year.

1.4 Overview of Dissertation

This dissertation will discuss the steps taken to achieve the objectives of this project.

Section 2 will discuss the theory in depth and will include a literature review of current work that has been done in this area to identify where the project is situated in terms of other work that has already been carried out.

Section 3 will detail how the problem was approached and justify the design considerations that were taken into account. The technologies and project methodologies that could have been used to implement the project are compared and contrasted.

Section 4 will examine the solution in depth. It will highlight problems that were encountered, how these challenges were met and where assumptions had to be made. It will go into some detail about the stages undertaken to create each dataset and build models.

Section 5 will describe what results were derived from each model, how these were prepared to be presented to senior management, and which suggestions based on the analysis were put forward.

Section 6 will provide a summary and critical examination of the project as a whole, suggestions for future work and some overall conclusions.

2 State-of-The-Art

This section will discuss E-commerce in depth and how suitable the data that it generates is for the application of data mining methods and will provide a literature review that can situate this project in the context of work that is current in this field.

This project is centred on data that has come from the E-commerce data store for DC Thomson shop. It makes use of a store of data that contains every user's long term interaction with the website or the company as a whole. This also includes the response of each customer to marketing campaigns and the marketing material itself. After some transformation and other data cleaning procedures have been carried out, these sources can be modelled to provide substantial value for business.[5]

2.1 Theory

E-commerce refers to selling goods or services over the internet. As a result of these transactions being carried out online, it is possible to track, record and store all the interactions that customers have with a company's online presence.[1] The result is a very large pool of data that is often stored in different places or formats. When the data is pulled together and analysed effectively it can be used in many applications.

One example is customer profiling, identifying groups of similar customers based on similar characteristics, where advertising or products can be targeted to groups that are more likely to be interested.[1] Another area is recommendation systems, recording which products are often bought together and presenting these as suggestions to users as they register their interest in products by selecting it or adding it to their basket.[1] Personalisation is another key application that can be derived from E-commerce data. Web pages and communication between the company and the consumer can be targeted to the individual's preferences or be modelled on their past behaviour, which allows communication to be more valuable and relevant to consumers.[6]

Another application, which this project focusses on, is being able to measure the response to marketing campaigns. Traditionally, the response to direct marketing was seen as something that is very difficult to model as the only data that used to be available was from those that purchased, which biased the results significantly.[7] It used to be much more common to mass post marketing material to a large list of customers, with the hope that some would engage with the material and make a purchase with the company. The only data that the organisation would have about the response to the campaign would be from those that had purchased, and organisations were used to a response rate of only around 1%.[7] This was wasteful for the

organisation that would have to print material and it was hard to identify which sales were as a result of the direct marketing campaign.

Email Campaigns are able to return much more data about the response to them from individuals. They represent a direct link between the customer and an organisation.[2] Users agree to receive emails so that they may be kept up to date with offers or new products and organisations are able to send out material periodically to a pool of users that will be engaged with the material. There are benefits to this approach, including being relatively inexpensive and being able to measure the success of the campaign.[2]

Customer Relationship Management (CRM) software such as Sailthru, which was used for this project, are able to measure the response to campaigns. Sailthru can show when a user opened an email and the device used to open the email, whether they went on to purchase from the email, the value of their purchase if this is true. Sailthru also records data about the users lifetime engagement with the website, for example: how many emails that they have received in total, their name and address, when they last opened an email, when they last viewed a page on the website, which newsletters and mailing lists they have signed up to and so on. In contrast to the traditional direct mail approach access to these datasets can allow emails to be targeted much more effectively to those that are likely to be more engaged with the material. [8]

Sailthru also collects the general response to an email, allowing an insight into where user's clicked on an email, how much revenue a link generated, what products were bought as a result of the email. An example is shown below in Figure 1:

Figure 1 Sailthru Email Response Example

Figure 1 shows an example of some of the different response statistics that services such as Sailthru can offer. Each of the images are of the same links and are from the same email. The link on the left shows the number of clicks in a red box that the link received. The green boxes in the middle images refer to the amount of purchases made as a result of a user clicking on

that link. The figure in the green boxes in the image on the right refers to the amount of revenue generated as a result of users navigating to the website through that link in the email. The 3 images at the top show that The Beano image received 224 clicks through to the website, this generated 6 sales that were worth £318.50.

In contrast to the traditional direct marketing approach, having the ability to see exactly where and how users engaged with the material means that the response can be measured much more effectively. The traditional approach would only be able to measure its success by the amount of purchases or total revenue of the products that may have been mentioned in the material and the data captured about the response would be from those that actually bought from the company. The email marketing approach enables a much wider set of data about the response to be gathered. For example, only 6 out of the 224 people that clicked the link may have purchased, but it also shows that there are 218 other people that were interested in the link enough to click through to the website.

The data generated from these sources is suitable for data mining. Data mining concerns the identification of useful patterns from a set of data.[5] It involves the application of machine learning algorithms to a dataset which attempt to discover between input variables and an output variable. An input variable can be thought of as a column in a table, for example a column of genders relating to a list of customers (a list of male or female) and a column of ages (a list of the age of each customer). An output variable is something that can be either predicted or classified, such as whether a customer purchased from an email (a true or false value). A simple data mining method may be to use statistical methods such as linear regression to identify whether there is a positive correlation between a person's age and gender and their likelihood to buy. Complex machine learning algorithms can take into account a much wider array of input variables, identifying the patterns between different them and calculating a prediction or classification of an output variable based on what it has learned from the data.[9] Some models such as rule based methods or decision trees are also able to reveal how the predictions or classifications were reached.[9] This allows knowledge about the relationships within the data to be revealed, not just predictions, which can be very valuable to an organisation.[9]

In order for data mining methods to work effectively, they require the data sources that will be used to have certain qualities. Data mining is about finding general patterns from data, and as such there needs to be a significant amount or a large enough volume of historical examples for a model to learn.[10] Email campaign response data can meet this condition as groups of data regarding the response to different emails can be merged together to form a much larger set. This project used emails all the emails that were sent by the organisation between 1st October and 31st December of 2015, where the response to each set was merged to create a large set of data that could be used for analysis.

In order for relationships to be identified, the degree of input variables must be large enough, or contain sufficient variety, that the data mining model will reflect the data. Data from email marketing meets this condition as since it is often highly structured, many input variables can be generated for use by a model.[10] For example, features can be extracted from emails such as the location and position of links, the time of day that emails were sent, and so on.

Another condition is that new examples of data can be captured, or there is enough velocity to the dataset.[10] Email marketing campaigns are fairly frequent, albeit much slower than most applications of data mining. New examples from campaigns as they occur however can be added to the dataset or used to test the model. Services such as Sailthru will also keep the store of data about customers updated, so the datasets used by the model will be based on accurate and up to date information.

The usefulness of patterns identified by the model however have to be taken into consideration. Rastegari [5] points out three areas where patterns identified by data mining in E-commerce situations may fail to be useful.[5]

The first area is where the relationships identified do not match the true relationships of the data.[5] This can occur often when the sample size is too small and the models created are too specific, or when the model is misled by the introduction examples that do not match the main body of examples from the dataset.

The second area is where the patterns identified describe the data rather than reality.[5] This is when the data contains structural patterns that describe the end result. For example, if a model was predicting the success of a link, which had been calculated using the amount of people sent an email divided by the amount of users that clicked a link, and both of these variables were included, the model would only identify this relationship as it would be the most direct relationship that existed in the data. This area will be covered in depth in a later section.

The third area is where the relationships that have been identified by the model are accurate and relevant, but are already known by the organisation.[5] These “truisms”[5] are not useful results to report as they do not reveal new knowledge to the organisation. This area will be covered in the results section.

By taking these situations into account, the data mining process can be targeted to identify new relationships that were not previously known by the company before, are accurate and describe reality rather than the data.

Successful identification of these hidden relationships can be of great advantage to business. For example, it can identify the important factors that led customers to buy or what structural factors in an email have a positive effect.

This can influence the strategic marketing efforts for the company.

A/B testing is a common method for testing which emails will perform well with a group of customers. This is where two sample groups of customers are sent different versions of an email.[11] The response to each is then measured, and the email that performed the best is sent to the main group. The factors identified by the model can be used to identify the changes that can be made to an email to create the different versions to test.

Some models are also able to rank the importance of variables that were used to build the model. This is used by an organisation to optimise its data capture methods by providing suggestions of which variables could be collected more effectively or where new variables that are not already collected might be useful to identify for future models.

The application of data mining methods from this project is covered in more depth in the results section.

2.2 Literature Review

Much of the current focus in the literature surrounding the use of data mining in E-commerce is centred on how it may be applied to business functions. These include customer profiling, [5] personalisation, [6] and recommendation systems. [12] Of the literature that focuses on data mining and email marketing, the main subject of consideration is centred around the content of emails, for example the words included in the email body, the subject line, rather than the structure, as is looked at in this project.[6] It is acknowledged by many authors that systems are able to capture and store details about individual user's interactions with the site,[1][5][13] but there are fewer studies that attempt to identify which of these historical factors have an influence on whether a user will purchase.

One approach that is similar to this project is presented in Mogoș and Acatrinei's recent work *Designing Email Marketing Campaigns – A Data Mining Approach Based on Consumer Preferences*.[6] Their work considers how recipients respond to emails from companies. They apply data mining methods to a data set made up of survey results from individuals that regularly receive marketing emails from companies. The survey covered three aspects: what determines an individual, what determines them to respond positively to the email and the overall influence of the email.[6] Each of these aspects had a set of questions associated with them, for example whether the email had a "catchy title" and this caused the recipient to open emails.[6] Each of the 262 recipients of the survey filled in every question and the answer to each question formed the dataset. This data set was then analysed using the WEKA data mining tool. They then use this tool to calculate the importance of each of the questions, cluster the users based on their response and used the decision rule PART algorithm to identify com-

mon rules or patterns of answering questions amongst the groups that had been identified.[6] They concluded by identifying the important factors that their model identified.

Their work raises some important issues. They identify many influential factors that cause recipients to open emails and they go into detail about why these are important. Their approach is different to the one taken by this project in that they analyse the content of emails rather than the structure. Their surveys focus on whether an email has "a funny message" "has the company name" or whether the "logo is included" would be able to identify trends amongst email campaigns in general, regardless of the content.

A major limitation of the paper however is the sample that formed the dataset. It is relatively small, with only 262 respondents, and is biased in that 91.2% of the respondents came from an urban environment and only 8.7% came from a rural background. All respondents were aged between 20 and 35 and had access to higher education.[6] Since their aim was to identify the factors that influence recipients to open emails, a dataset that covered a much wider range of groups would have been better. The researchers acknowledge this limitation, and do state that the conclusions drawn should take this into consideration.[6] Another criticism that can be made is that the authors state that during the pre-processing stage:

"the main preprocess activities done for the data mining process were: the fill out level of the data mining dataset was checked and fulfilled with null values where it was necessary"[6]

One of the main challenges of using survey data in data mining is how missing values are identified.[14] The authors do not go into detail about the quality of their dataset and the steps taken to ensure that it is suitable for data mining, which means that some of the conclusions that they make can be called into question. Missing values were a problem that was also encountered by this project. This project used sampling techniques and also imputation methods to reduce the amount of missing data from the set.

Another useful study to consider is Breur's[15] article: *How to Evaluate Campaign Response: the Relative Contribution of Data Mining Models and Marketing Execution.* In this article, Breur attempts to define how to measure the response to direct marketing campaigns that have been optimised using data mining. He explains that there are two factors involved in measuring the response: the "marketing execution" and the "data mining model".[15] The marketing execution involves the layout of the campaign and its content, and the data mining model is used to target which customers should be sent marketing material. He argues that these have different influences on the response to a campaign, and that this needs to be taken into account. Then the author goes on to describe how tests that measure these influences should be carried out, and the statistical methods that allow them to be compared. He concludes by asserting the importance of effective planning of campaigns in the design of

marketing material and the targeting of the audience. If the execution is very good, but the targeting is poor, the response will be poor. Similarly, if the targeting is good, but the execution is poor, then the response will also be poor.[15] This project aimed to provide a bridge for these ideas. By identifying patterns in the structure of marketing material, this will create a better execution. The patterns identified in groups of users will enable better targeting.

Breur's article is useful to consider because it goes into a lot of depth regarding methods to measure response. Unlike Mogos and Acatrinei, Breur places a lot of emphasis on the need for effective sampling. He also goes into detail regarding different measures of success, for example, comparing lift curves to Rosset *et al.* response to non-response ratio.[16] He does not refer to any empirical study that he has observed or carried out. This can make it difficult to follow his argument, as his suggestions and models are not then backed up by evidence.

This project fits in well with Breur's suggestions. The Campaigns section contains a focus on what makes the structure of emails successful, which is related to Breur's idea of "marketing execution". The user's section attempts to find shared characteristics amongst users to identify users that will respond positively to a campaign. This reflects Breur's focus on the need to target the audience. The project can also be seen as an extension of the work carried out by Mogos and Acatrinei. It will extend their approach by looking at the impact of email marketing on a specific case, the email of the DC Thomson shop's Christmas campaign of 2015 and the characteristics of users.

This project aims to be situated in the literature as a case study of how data mining can be applied to email marketing. The factors that need to be taken into account when conducting this kind of research are identified. This project aims to extract knowledge from data mining methods rather than present a predictive or classification model for implementation.

3 Approach

This section outlines how the project was designed and carried out. It provides a justification of the structure of the project. A critical review of the different tools and technologies that were used or could have been used to implement the project is included.

3.1 Project Methods

The project was carried out using the agile methodology.

This involved daily stand-up meetings where every day the work that would be carried out that day would be presented to the team at DC Thomson. This enabled the wider team to be kept up to date with the progress of the project and have an opportunity to share their skills to provide an input.

The project also used weekly "sprint" meetings, where the tasks for the week would be laid out. These tasks would be added to a Trello board.[19] This is a website that allows tasks to be laid out in a pin-board style design. Tasks for a "sprint" or the week would start off at one side of the board, move through "in progress" areas until they reached the "complete" side. These meetings helped to identify the direction of the project each week, where resources or expert assistance may be required and ensured that the project was on schedule. This was a useful methodology to use because it identified where problems may have arisen, ensured that the project was on track and identified other interesting ideas that were within the scope of the project that could be implemented.

An alternative method that could have been used is the waterfall approach, where each of the stages is planned out fully and the project iterates through each of these stages.[20] This approach uses different development steps. It starts at a planning stage, then an implementation stage and finally a testing stage. After a stage is fully completed, the project moves to the next one.

Although the depth of the planning stage may have enabled most of the problems associated with the project to be identified early on, the static nature of the waterfall model would not have been suitable for the project. The static nature of the development stages would not have been suitable for the time frame, would not have allowed changes to be implemented effectively and would not have allowed early results to have been analysed. The agile methods that have been outlined above enabled the project to be flexible and problems could be identified easily at sprint meetings, where the direction of the project could be changed accordingly.

The project used two iterations of the CRISP-DM approach. The first run through the data allowed an effective approach to be developed and created some early models. The second run identified problems with the initial approach, and adjusted some of the data cleaning and model construction so that the results obtained by the project were more reliable.

An alternative approach would have been only to make one pass through the data. This may have enabled more time to be spent on each section, but the project would not have benefitted as a whole from the initial results that could have been delivered from a first pass through the project.

3.2 Project Structure

The overall aim of the project was to identify what factors drove sales for the 2015 email campaign for DC Thomson Shop. After gaining an understanding of the business context and the datasets available, two main areas for consideration were identified.

The Sailthru platform is used by the organisation to store and send emails provided a dataset that held the response by users to each link to the website in an email (the number of click-throughs, purchases and revenue). This provided a core set of data that could be enriched by merging the structural aspects associated with each link – such as its position in an email, or the size of a picture that included a link. The structure of an email has an influential impact on its success,[15] and by using the response data surrounding the link to generate success metrics, models could be built that would be able to predict how well a link was likely to be received based on its structural aspects. These models would allow the relationships in the data that lead to these classifications to be revealed, and would allow the importance of each of the structural variables to be ranked. This section was called Campaigns, as it would attempt to identify the structural factors that made links in an email campaign successful.

Sailthru also keeps a record of every individual email that was sent to each recipient in an email campaign alongside how they responded, including when they opened an email, whether they clicked any links and when they made a purchase. This is stored with a user ID. Sailthru also stores a large dataset of everything else that is known by the organisation about each user, also stored with the same user ID. This includes their name, how many purchases they have made, when they signed up, the value of their last and highest purchase and some location information. By merging the list of recipients with everything else that is known about the user, in theory models can be built that can identify whether there are any common factors amongst groups of user that purchase from emails. Models can be built that are based upon predicting whether an individual will purchase from an email or not. This section was called Users be-

cause it would attempt to identify what factors in the store of data about users led them to purchase.

3.3 Process Model

The project was designed using the CRISP-DM (CRoss Industry Standard Process for Data Mining) methodology. CRISP-DM is a methodology that allows the complete path of a data mining project to be planned out into 6 phases.[17] These phases are shown in Figure 2 below:

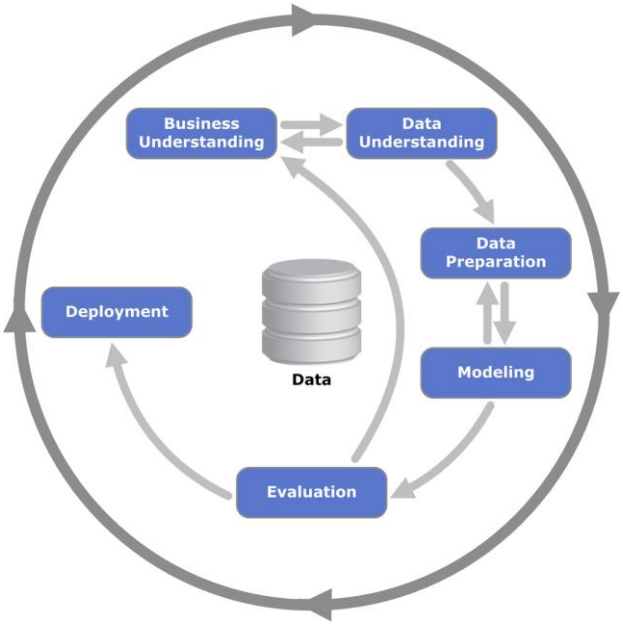


Figure 2 CRISP-DM Data Mining Project Structure.[18]

There are 6 stages to the project, and these are iterated through to reach the final stage, of deployment of a model.

The first stage was Business Understanding: business operations; the context of the problem being studied; identifying constraints and problems that arise during the project; assumptions that have to be made and other factors that have an impact or influence on the model or the project.[17]

This stage in this project highlighted several factors. The main aim was to study what factors exist in email campaigns that influence customers to buy during the Christmas period. The scope covered the Christmas period of 2015, and two levels, the reaction to campaign emails and the reaction by users to campaigns. The data provided by the Sailthru CRM platform was used as a basis, and other data sources available within the company was merged to provide extra information about the user or the campaign. This planning stage allowed the structure of

the data mining solution to be developed. The two main areas under consideration were a Campaigns section, which examined how different emails were received, and a Users section, which examined the factors that lead individuals to react in a certain way to a campaign email.

The Data Understanding stage involved identifying sources of data, exploring these sources to assess their quality and assessing ways in which they can be merged.[17]

The Campaigns section focused on the *links in emails* to the site. Sailthru stores the click through, purchases, revenue and total clicks for each link that was included in an email. This set was supplemented by looking at the email itself, to identify the position of links in the email, the size of associated pictures, whether brands were mentioned and so on. More general information about the email was gathered as well, such as the number of days between the sent date and Christmas. A snapshot of some of the variables and data from the campaign section is shown below.

id	Link.URL	position	size	distanceFr	daysToChr	sentAt	dayOfThe\	emailType	Sub.Brand	clickThrou	Sent	Click.Rate	Total.Click	Purchases	Revenue	price	Link.Click.I	rowColPos	Leading Im	Link Click F	Above Belc
5393895	https://wv	top	200	0.2	58	evening	Wednesda	reader offers		benchmarl	29154	90	103	1	57	39.99	Above Ben	Top Right	product	0.003533	Below
5393895	https://wv	top	200	0.3	58	evening	Wednesda	reader offers		benchmarl	29154	89	100	1	15	15	Above Ben	Middle Lef	product	0.00343	Below
5393895	https://wv	middle	158	0.4	58	evening	Wednesda	reader offers		benchmarl	29154	87	98	2	85.99	18.99	Above Ben	Middle Mi	product	0.003361	Below
5393895	https://wv	middle	200	0.5	58	evening	Wednesda	reader offers		benchmarl	29154	44	47	1	23.5	23.5	Below Ben	Middle Mi	product	0.001612	Below
5393895	https://wv	middle	200	0.6	58	evening	Wednesda	reader offers		benchmarl	29154	132	151	1	21	21	Above Ben	Middle Rig	product	0.005179	Above
5393895	https://wv	bottom	200	0.7	58	evening	Wednesda	reader offers		benchmarl	29154	85	96	0	0	45	Above Ben	Bottom Le	product	0.003293	Below
5393895	https://wv	bottom	200	0.8	58	evening	Wednesda	reader offers		benchmarl	29154	61	71	0	0	33.25	Below Ben	Bottom M	product	0.002435	Below
5393895	https://wv	bottom	200	0.9	58	evening	Wednesda	reader offers		benchmarl	29154	131	156	0	0	57	Above Ben	Bottom Ri	product	0.005351	Above
5393895	https://wv	bottom	180	1	58	evening	Wednesda	reader offers		benchmarl	29154	102	120	0	0	57	Above Ben	Bottom Ri	product	0.004116	Below
5393895	http://link	bottom	0	1	58	evening	Wednesda	optout		benchmarl	29154	0	0	0	0	0	Below Ben	Top Left	product	0	Below

Table 1 Snapshot of campaign dataset

Each row in table 1 refers to a link in an email. The data set was created by extracting all the information about each link in an email, then pulling all of these links all these links into one dataset.

The User section was made up of the individual emails that were sent to users. Sailthru maintains a list of every user that is sent an email for each campaign, and how they respond (if/when they opened, whether they made a purchase). Sailthru also maintains a list of aggregated information about each customer, for example the amount of emails they have been sent, their favourite device. Merging these sets together enables a picture of how users respond to emails. This enables the patterns that lead users to purchase to be identified.

This set was enriched by adding data from other sources that is stored about customers. The Oracle BI set for example contains information about their demographic region, and whether they purchased from a brochure or magazine during December. A snapshot of a sample of the User data is given in table 2:

device	timeFromSendT	timeFrom	sendPerio	openPeric	purchaseF	openedDa	purchased	purc	Click	Pagev	Lifeti	Purch	Purcha	Largest.	country	source	gender	On newsle	More_tha	timeFrom	timeFrom	ageOfAcc	openedEn	signupTim
iPad	245.6255556	1415.457	afternoon	evening	afternoon	Wednesda	Saturday	1	3	6	7	1+	4000	4000	United Kin	DCT Subsc	female	TRUE	TRUE	106	106	285	afternoon	noon
Gmail	1333.158333	1333.401	afternoon	night	night	Sunday	Sunday	1	3	5	22	1+	4200	4200	Overseas	DCTShop	female	TRUE	TRUE	179	179	343	night	noon
Gmail	1061.869722	1062.469	evening	night	night	Friday	Monday	1	3	3	54	1+	2500	2500	Overseas	DCT Produ	male	TRUE	TRUE	108	204	285	night	noon
iPad	37.97583333	810.9564	evening	morning	afternoon	Wednesda	Tuesday	1	19	32	33	1+	5700	5700	United Kin	DCTShop	female	TRUE	TRUE	132	276	343	noon	morning
Gmail	13.36861111	797.0456	afternoon	night	evening	Wednesda	Monday	1	4	3	47	1+	6800	6800	United Kin	DCT Subsc	male	TRUE	TRUE	80	254	285	morning	noon
Chrome	39.66694444	768.7367	afternoon	morning	afternoon	Wednesda	Sunday	1	1	4	48	1+	4200	4200	United Kin	DCT Subsc	male	TRUE	TRUE	79	82	285	evening	noon
Gmail	3.457222222	700.5489	noon	afternoon	afternoon	Monday	Tuesday	1	16	63	33	1+	8495	1200	United Kin	DCTShop	female	TRUE	TRUE	114	320	343	evening	noon
Internet E	665.1827778	665.4086	afternoon	morning	morning	Wednesda	Wednesda	1	12	15	49	1+	3099	3099	United Kin	DCT Subsc	female	TRUE	TRUE	75	142	285	afternoon	noon
iPad	35.65972222	661.4956	noon	evening	night	Monday	Monday	1	12	24	45	1+	8100	5700	United Kin	DCTShop	female	TRUE	TRUE	111	283	343	noon	noon
Outlook	3.465277778	658.9581	evening	evening	night	Tuesday	Tuesday	1	9	13	31	1+	3699	3000	Overseas	DCTShop	male	TRUE	TRUE	122	321	343	night	noon
iPad	196.2941667	628.2383	afternoon	evening	evening	Wednesda	Monday	1	1	1	49	1+	4000	4000	United Kin	DCT Subsc	male	TRUE	TRUE	73	263	285	morning	noon
Firefox	3.066111111	608.0703	evening	evening	night	Friday	Wednesda	1	7	14	23	1+	2098	1099	Overseas	DCTShop	male	TRUE	TRUE	147	321	343	night	morning

Table 2 Snapshot of sample user dataset

Each row refers to an individual email that was sent to a user.

The Business Understanding and Data Understanding stages are linked, because some things can arise out of looking at the data that cause the assumptions made in the Business Understanding stage to be re-evaluated.

The Data Preparation stage involved ensuring data is of appropriate quality for modelling. Data mining involves identifying general trends from data. As such, data is required to have several features that enable the process to be carried out. It must have a good distribution, where there are a clear range of variables. It must be mostly free from missing data, so that relationships can be identified. Outliers (values that are unique or far removed from other examples of the variable) have to be removed or recoded.[17]

The Campaign section required some variables to be generated, as they were not available through Sailthru (such as the distance from top, days to Christmas values). Some outliers were removed, such as one email link that received 1500 clicks where most received under 100. This will be covered in more detail in the solution section.

The User section required much more preparation, as the quantity of missing data was considerable.

Sailthru works by capturing interactions by the users that it stores in its database. For example, when a user opens an email that has been sent to them from Sailthru, this response is sent back to Sailthru, containing such details as the time it was opened, the device it was opened on, and so on. When there was no response from users, such as users that did not open the email, this resulted in a lot of missing data.

This needed some automatic imputation to be carried out (using values suggested by relationships in the data). Other factors, such as balance had to be addressed. There is a large gap between users that purchase from emails and users that do not. This means that some sampling has to be carried out to make a set that will have much more balance. Another suggestion may

be to generate new artificial purchasers based on purchaser examples from the dataset. This would enable much more of the dataset to be used. This stage also required some columns to be created, based on the values of other columns (for example the gender and region information).

The Modelling stage involves creating the data models for the project. It is closely linked with the preparation stage, as it may reveal data quality issues that need to be addressed. The right model needs to be chosen that will best suit the aim of the project.[17] This project used decision tree models. The aim of the project were to identify and return the relationships that existed within the data, and decision tree models are able to show how they reached their classification easily.[9] This is covered in more depth in the solution section.

The Evaluation stage involves looking at the current models and identifying whether they meet the requirements set out in the Business Understanding stage. If the model does not reach these requirements, the process starts again at the Business Understanding stage, revaluating the sources of data used or the methods undertaken for cleaning them.[17] Once the models are in a final stage, the project moves into the deployment stage.

The deployment stage represents the final stage of the project where the models may be put in to practise. For this project, this stage involved looking at the results from the model, and transforming these into a set of suggestions that would be useful for the company in future.

3.4 Underlying Technologies

The technologies used to implement the project were a combination of R and Python.

R is a statistical based programming language and was used to explore datasets and create models.

Python is a programming language that has extensive libraries that effectively deal with data manipulation. This project used the Pandas library for merging datasets and the RE (regular expressions) library for extracting data from emails.

The project could have been alternatively carried out using purely either language. R contains libraries and methods for regular expressions and merging, and python has packages such as matplotlib and sklearn that can be used to explore data and perform model creation respectively.

R was chosen to explore datasets and create models because the language is much more intuitive for data exploration and creating graphical plots is easy. The program Rstudio[21] enables a workspace to be stored that includes a plot window that is updated when a plot function is called. R's sophisticated indexing techniques allow subsets of the data to be extracted

naturally and was a good fit for the data. The R library Rattle[22] contains a useful GUI for data analysis, which allowed graphs concerning the distribution of variables and correlation between variables to be generated easily. The GUI also enables models to be constructed, visualised and evaluated.

Python was chosen for the data merging mainly because of its efficiency. The response data from Sailthru contained lots of small files, and Python is much more efficient at opening files and looping structures than R.[23] The merging of the different datasets was of particular interest to the organisation, so it was considered that the readability of Python code would be something that would be useful to take into account.

Although some time was required to learn how to use R, the combination of R and Python ultimately proved to be the best approach for the project.

The project could also have been implemented using the WEKA platform.[24] The WEKA GUI contains methods for creating, cleaning and building models and is built in Java.

It may have been useful to use WEKA for the project as it contains a very diverse range of models with many different options for manipulating them. The GUI is easy to use, there is extensive documentation and support for the program and there is a wide range of methods for evaluating the success of a model.

It was decided not to use WEKA for the project for several reasons. WEKA does not always cope well with large datasets, [25] and the data preparation tools are not intuitive to use. R provides a much wider range of methods for exploring and modelling the data, and much of the functionality provided by WEKA is present in Rattle, which was used for the project.

4 Solution

This section will describe the implementation of this project in detail. It is split into two main sections, Campaigns and Users, and will explain how the dataset for each was created, cleaned and modelled.

4.1 Campaigns

The campaigns section involved several stages. First, the structural information about links had to be created by scraping the HTML of emails, and then merged with the response data provided by Sailthru. This data set was then explored and modelled using R.

4.1.1 Data Scraping

The first stage was to download the HTML of each email, putting it into a file that that could be read as a JSON object. A snapshot of this file is shown in Figure 3:

Figure 3 Email Data Store

Storing the emails in a JSON style file means that they can be accessed and iterated through easily. The example above shows the structure of the JSON document. JSON works on key value pairs. The keys in the email file are the Sailthru ID for each campaign. The value is the HTML of the email. This makes it easy to access, and merge with other datasets from Sailthru that contain the ID.

The first script used for email scraping was “emailClickLocator.py”. This program was used to go through the email, extract specific pieces of data using regular expressions with Python. The benefit of using the email file in the JSON format is that it keeps all the emails in a way

that can be problematically accessed, and the emails are stored as strings which means they can be processed efficiently. Regular expressions can be used to parse every email, since each use the same format. Regular expressions are able to find patterns in a string. For example, the regular expression: “www[.][a-zA-Z]*?[.]co[.]uk” would find all the occurrences of any website that is included in the email with a .co.uk domain. All the emails under consideration follow a basic template which means that a regular expression used to capture something in one email will work on another.

The aim of this script is to turn the HTML of an email into a flat series of rows corresponding to links in the email, with columns that describe various things about the link- its position in the email, the size of pictures and so on. The script creates the following columns for each link:

- ID
- Link URL
- Type
- Position
- Size
- distancefromTop

The “ID” is captured using the keys of the JSON email store file.

The “Link URL” is captured using the following regular expression:

```
links = re.findall('title="(.+?)"', email[id]["clicks"])
```

In HTML, links to other pages on the internet usually always use a title="" element in the tag. This regular expression captures all the links in the email. However, these links may be repeated within the email, but the Sailthru statistics for a link are aggregated. For example, in figure 4, the three links referred to by the boxes 11, 3 and 1 direct the user to the same product page on the website The response statistics that are accessible from Sailthru are not stored in this way, as it stores the sum total number of clicks for each version of the link. Since the repeated links appear in roughly the same location of the email, the repeated links can be deleted, as long as the overall list of links is kept in the order that they have been seen. This is accomplished using a function adapted from[26]:

```
def removeDuplicatesOrdered(seq):
    seen = set()
    seen_add = seen.add
    return [x for x in seq if not (x in seen or seen_add(x))]

clicks = removeDuplicatesOrdered(clicks)
```
[26]

The “type” variable is an attempt to identify where the link might lead. This column is improved later, but is a basic way to check to see whether it is a product, subscription, optout or homepage is to check whether these terms are included in the link. This is originally done using:

```
def guess_content(link):
    if link == 'http://www.dcthomsonshop.co.uk':
        return "HOMEPAGE"
    if link == r"http://link.dcthomsonshop.co.uk/manage/50g/dcts-
optout":
        return "OPTOUT"
    if "subscription" in link:
        return "SUBSCRIPTION"
    else:
        return "PRODUCT"
```

This however is a fairly inaccurate way of identifying the type of the link. The else clause will cause all the links that do not mention “subscription” “homepage” or “optout” to be classified as a product where this might not necessarily be true. It was improved in the “cleanCampaignsEmails.py” script by looking at other common factors included in the link.

“Position” is calculated by splitting the list of links into three groups. For example a list of links such as:

[link1, link2, link3, link4,link5,link6]

gets split into:

[[link1,link2],[link3,link4],[link5,link6]]

or:

[[top],[middle],[bottom]]

This labels the links as either being at the “top” “middle” or “bottom”. This was later improved using a method that identifies the position of links more accurately with the variable “rowColPosition”.

“Size” is also improved in a further script, but refers to the picture size of an email associated with a link. An example image from an email is shown in Figure 4.

Figure 4 Example Image from Email

The picture of the sheep is associated with a link to the website. The following regular expression captures where these links occur:

```
pictures = re.findall('<img.+?height="([0-9]+)" width="([0-
9]+)"|<img.+? width="([0-9]+)"', email[id]["clicks"] )
```

This captures this part of the email shown in Figure 5:

```
tm_term=DC%20Shop%20-%20all%20users%20with%20permission%20true"><span style="position:relative"><span title="htt
p://www.dcthomsonshop.co.uk" style="cursor:hand;height:12px;color:red;font-weight:bold;font-size:12px;font-family:sa
ns-serif;border:2px solid red;padding:1px 4px 1px 4px; line-height:12px;position:absolute; background-color: white;o
pacity:0.8;text-decoration:none !important">21</span></span>                                    <img styl
e="margin-top: 15px;padding-bottom:15px; display: block;" src="https://my.sailthru.com/ssl?url=http%3A%2F%2Fmedia.sa
ilthru.com%2F50g%2F1jz%2Fa%2F8%2F56167f796d388.png" alt="DC Thomson Shop Logo" border="0" width="300">
                                </a>                                                    </td>
```

Figure 5 Image Capture

This returns links that have a width and a height attribute or only a width in an "<img>" tag. The homepage and optout links are the only links in an email that do not have an associated picture. This means that the size values can be assigned to the links in a list in order after the homepage link and optout link have been ignored and still refer to the correct link. The height times the width of links would give an approximate size of the picture in pixels.

This however proved not to be a good way of measuring the size of pictures. Some pictures had only a width attribute and were commonly larger than ones with both width and height, and this made this variable false. This variable was improved in a later script.

"Distance from top" is calculated by how far through the list of links the link appears (as a percentage).

For example, in a list of 5:

[link1,link2,link3,link4,link5]

becomes:

[0.2,0.4,0.6,0.8,1]

This is calculated by taking the position of the item, for example, link 1 would be 1, and dividing it by the number of items in the list. This would give a basic understanding of where links in the list lie relative to the top of the page, as they have been extracted in order.

The next step that was taken was to merge the links with the information about the response to these links that Sailthru provides. The script "cleanCampaignsEmails.py" is used for this and also creates several more variables:

- daysToChristmas
- sentAt
- dayOfTheWeek
- Clicks
- Revenue
- Purchases
- emailType
- subBrands

"daysToChristmas" takes the time that the email was sent, from the campaign statistics, and takes this away from the date of Christmas, to get the number of days between when the email was sent and Christmas day. This is shown in the following code:

```
dateSent = dateSent["send_time"][0]    #eg. "2015/11/25 12:00"
sendTime = dateSent[11:]               #"12:00"
dateSent = dateSent[0:10]              #"2015/11/25"
dateSent = datetime.date(year=int(dateSent[0:4]),
day=int(dateSent[8:10]), month=int(dateSent[5:7]))
daysToChristmas = (datetime.date(2015, 12,25 ) - dateSent).days
linksAndLocat["daysToChristmas"] = [str(daysToChristmas) for x in
range(len(linksAndLocat))]
```

"sentAt" is a measure of what time of day that the email was sent. It takes the time, as captured above, and turns it into a categorical variable with "morning", "night", "afternoon" and evening" as factors:

```
timesOfDay = ['night', 'morning', 'afternoon', 'evening']

if 0 <= sendTime.hour < 6:
    timeOfDay = timesOfDay[0]#night
elif 6 <= sendTime.hour < 13:
    timeOfDay = timesOfDay[1]#morning
elif 13 <= sendTime.hour < 19:
```

```
    timeOfDay = timesOfDay[2]
elif 19 <= sendTime.hour < 24:
    timeOfDay = timesOfDay[3]

linksAndLocat["sentAt"] = [str(timeOfDay) for x in
range(len(linksAndLocat))]
```

"dayOfTheWeek" uses an inbuilt python calendar for dates which day of the week a date fell on:

```
weekDay = calendar.day_name[dateSent.weekday()]

linksAndLocat["dayOfTheWeek"] = [str(weekDay) for x in
range(len(linksAndLocat))]
```

"Clicks", "Revenue" and "Purchases" are all merged from the Sailthru response dataset that is provided for each email. This dataset contains a column that contains the links of an email, and other columns that include the total number of users that clicked on a link, the total amount of purchases made as a result of clicking on the link, and the total revenue resulting from it.

```
for id in campaignList:

    clicksAndPurchases =
pd.read_csv(r"C:\Users\tdavenport\Desktop\data\Campaigns\EmailLocati
onAndClickData\campaignStats\export-stats-zip-%s\links-%s.csv" %(id,
id))

    merged = pd.merge(linksAndLocat, clicksAndPurchases,
on="Link URL")
```

This will merge these two files using the URL of a link, and provide the clicks, revenue and purchase variables. These variables represent the response that the link received from users, and can be used as a predictor in training a model. They can also be used to generate measures of success of a link, which will be explained in the next subsection.

"emailType" is included here to identify a more specific category for a link. It takes advantage of the fact that the path of a link from Sailthru often contains further information about a link, for example, using this link:

https://www.dcthomsonshop.co.uk/sailthru/grouped-categories/row-1/dennis-the-menace-wallet.html

the following regular expression:

```
emailType = re.findall("http.*?sailthru/(.*?)/", i)
```

will capture:

```
https://www.dcthomsonshop.co.uk/sailthru/ (** grouped-categories **)
/row-1/dennis-the-menace-wallet.html
```

“emailType” refers to the portion of the link after the Sailthru section, which refers to a general category for a link. The original “type” variable is merged with the blank values that are within “emailType”. Typically, the homepage and opt-out links do not contain this general category from Sailthru. Therefore, using the values from “type” to supplement “emailType” is acceptable and provides a much more complete and accurate column of data that describes the content of a link.

Similarly, “subBrands” is captured from the portion at the end of the URL using:

```
linkTitle = re.findall("//.*?/.*?/.*?/?.*?/(?:row-[0-
9]/)?(.*?).html", link)
#https://www.dcthomsonshop.co.uk/sailthru/grouped-categories/ (** Not
row-1 **) / (** dennis-the-menace-wallet **) .html - but skips row-1
```

This captures "dennis the menace wallet". The not clause is used to ignore a section if it contains a “row” number, so that it will capture all the links presented in this way. Not all emails use the “row” syntax, so it was necessary to ensure that this would not be captured. The next step compares this string to a list of brands, generated from another script, “brandsSearch.py”.

```
brands = ['Parragon', 'Puzzler Media Ltd', 'Findmypast', 'MyFami-
lyClub', 'Brightsolid', 'Aberdeen Citizen', 'The Press and Journal',
'Sunday Post', 'Scot-Ads', 'Evening Express', 'The Weekly News',
'Evening Telegraph', 'The Courier', 'This England', 'Shout', 'Twirly-
woos', 'LIVING', 'No.1 Magazine', 'WWE Kids Magazine', 'The Broons',
'Scottish Wedding Directory', '110% gaming', 'Commando', 'My Weekly',
'Evergreen', "The People s Friend", 'D&G EPIC Magazine', 'Oor Wul-
lie', 'Animals and You', 'The Scots Magazine', 'The Official JW Mag',
'jacqueline wilson', 'Thunderbirds Magazine', 'The Beano', "dennis
the menace", "gnasher", "broons"]
```

If a “brand” from the link is found in the list, then the “subBrand” value will be the brand from the list. If a brand isn't found, it is left blank. In the example above, “dennis the menace” will be used as the “subBrand”

The variables that have been compiled are then merged together and stored in another file:

```
merged = pd.merge(linksAndLocat, clicksAndPurchases, on="Link URL")
```

“LinksAndLocat” refers to the dataset generated by “emailClickLocator.py” and “cleanCampaignsEmails.py”. “clicksAndPurchases” is the response data for each link provided by Sailthru. Each of these datasets share a common column, “Link URL” which means that the data can be merged. These are all then saved in a new CSV file:

```
merged.to_csv(r"C:\Users\tdavenport\Desktop\data\Campaigns\EmailLocat
ionAndClickData\inprogress\merged%s.csv" % (id))
```

The next stage is to merge all these together:

```
path =
r'C:\Users\tdavenport\Desktop\data\Campaigns\EmailLocationAndClickDat
a\inprogress'
all_files = glob.glob(os.path.join(path, "*.csv"))
#list all csv files of emails
allEmails = pd.concat([pd.read_csv(f) for f in all_files], ig-
nore_index=True)
#Create a concatenated file of all links in emails
allE-
mails.to_csv(r"C:\Users\tdavenport\Desktop\data\Campaigns\EmailLocati
onAndClickData\mergedEmails\mergedCSV.csv", sep=",")
```

The "glob" method takes all the merged files and concatenates them. The result is shown in figure 6:

id	Link URL	type	position	size	distanceFi	daysToCh	sentAt	dayOfThe	emailType	Sub Brand	clickThrou	Sent	Click Rate	Total Click	Purchases	Revenue	price	Link Click Rate Benchmark
5393895	http://ww	HOMEPAG	top	300	0.1	58	evening	Wednesday			benchmar	29154	95	107	13	346.3	26.63846	Above Benchmark
5393895	https://w	PRODUCT	top	200	0.2	58	evening	Wednesd	reader offers		benchmar	29154	90	103	1	57	39.99	Above Benchmark
5393895	https://w	PRODUCT	top	200	0.3	58	evening	Wednesd	reader offers		benchmar	29154	89	100	1	15	15	Above Benchmark
5393895	https://w	PRODUCT	middle	158	0.4	58	evening	Wednesd	reader offers		benchmar	29154	87	98	2	85.99	18.99	Above Benchmark
5393895	https://w	PRODUCT	middle	200	0.5	58	evening	Wednesd	reader offers		benchmar	29154	44	47	1	23.5	23.5	Below Benchmark
5393895	https://w	PRODUCT	middle	200	0.6	58	evening	Wednesd	reader offers		benchmar	29154	132	151	1	21	21	Above Benchmark
5393895	https://w	PRODUCT	bottom	200	0.7	58	evening	Wednesd	reader offers		benchmar	29154	85	96	0	0	45	Above Benchmark
5393895	https://w	PRODUCT	bottom	200	0.8	58	evening	Wednesd	reader offers		benchmar	29154	61	71	0	0	33.25	Below Benchmark
5393895	https://w	PRODUCT	bottom	200	0.9	58	evening	Wednesd	reader offers		benchmar	29154	131	156	0	0	57	Above Benchmark
5393895	https://w	PRODUCT	bottom	180	1	58	evening	Wednesd	reader offers		benchmar	29154	102	120	0	0	57	Above Benchmark
5393895	http://lin	OPTOUT	bottom	0	1	58	evening	Wednesday			benchmar	29154	0	0	0	0	0	Below Benchmark
5394039	http://ww	HOMEPAG	top	300	0.071429	60	evening	Monday			above ber	51328	154	198	7	208.42	29.77429	Above Benchmark

Figure 6 snapshot of campaign data

"extraVarsCampaigns.py" attempts to improve some of the variables that have been introduced by the other scripts:

- Price
- Size

"Price" is improved by using a regular expression that captures every link and a figure in a price format that follows. The regular expression:

```
priceList = re.findall('href="(.*?.html).*?([0-9][0-9]?[0-9][.][0-
9][0-9])', email[str(id)]["clicks"])
```

captures all the links in an email, and a price figure that follows. This price figure is always in the same format, a link, followed by up to 3 numbers followed by a decimal point and two more numbers. By looping through the matches of the regular expression each of these results, if there is a link in the list that the regular expression, a link that contains a price can be updated to include this:

```
for link,price in priceList:
    currentEmail.ix[currentEmail["Link URL"] == link, "price"] =
price
```

"Size" is improved by looking at the width of images rather than the width * height. Wider images are always (in the set) bigger than smaller images, so it makes sense to use only the width attribute. The following regular expression captures the link to the website from each picture and its accompanying width attribute:

```
links = re.findall('href="([a-z/:.A-Z-]*?)[?]utm.*?width="([0-
9]+)">', pictureSizes)
```

This set is then used to change the size that had been recorded into the size of the width.

```
for link, width in links:
  currentEmail.ix[currentEmail["Link URL"] == link, "size"] = width
```

This captures the rest of the sizes that were missing, and removes the sizes that had been entered previously. This makes the "size" variable much more accurate.

The script, "extraCampaignVars2.py", provides the final variables for the models. It creates:

- rowColPosition
- linkClickRate

The "rowColPosition" refers to where a link appears in an email, in one of 9 places: top left, top middle, top right, middle left, middle middle, middle right, bottom left, bottom middle or bottom right.

This can be calculated by taking advantage of the fact that the email layout uses an HTML Table structure. HTML tables are laid out as follows:

<table>

<tr> <td> ... </td>… <td>… </td> </tr>

<tr> <td> ... </td>… <td>… </td> </tr>

</table>

A "<tr>" element refers to a table row. A "<td>" element refers to a table data, or a cell in the row. Most of the email is structured in this way. By splitting up the table elements in the email, the location of links can be inferred. Splitting the rows into three will give the "top", "middle", and "bottom" sections:

```
for id in campaignList:
    currentEmail = email[str(id)]["clicks"]       #for each email
    currentEmail = currentEmail.replace("\n", "") #delete new lines
    rows = re.findall('<tr>(.*?)</tr>', currentEmail)
#find all separate table rows
  rows = do_split(rows, splits)
#split into three blocks
```

```
    index = 0 #which set of rows are we looking at
 for row in rows:
     totalPosition = "" #initialise the output
     row = "".join(row)
```

This gives the top, middle or bottom position for each "<td>" element in each block. The idea is that the first third of elements will appear earlier in this list and will therefore be closer to the opening "<tr>" tag and will be on the left. The next set will be in the middle, the last on the right. If one of the links in the email is found in this position in the email, then that links "rowColPosition" will be the position at that point:

```
position = ["Top ", "Middle ", "Bottom "]
cols = re.findall('<td>(.*?)</td>', row)
cols = do_split(cols, splits)           #first third of td elements
are left, second third of elements middle, third third right.

innerIndex = 0
for col in cols:
#for each <td> element in each row
   leftMiddleRight = ["Left", "Middle", "Right"]
   totalPosition = position[index] + leftMiddleRight[innerIndex]

   for tableElement in col:  #for each <td> element </td>
       for link in cleanedMerged.ix[cleanedMerged["id"] == id,
"Link.URL"]:
       if link in tableElement:
           cleanedMerged.ix[(cleanedMerged["id"] == id) & (cleaned-
Merged["Link.URL"] == link), "rowColPosition"] = totalPosition
           innerIndex += 1

       index += 1
```

This provides a provisional attempt for finding the location of a link in an email. A better method may have been to split the rows in each top, middle and left block again by "<tr>" elements, and then split their "<td>" elements again into three for a clearer left, middle, right location. This could be implemented in future work.

The "Link Click Rate" is calculated by dividing the number of clicks a link received by the number of users that the email that the link is contained in was sent to. This shows the percentage of users that clicked the link. It is used to generate the "above below link average" benchmark described in section 4.1.2.1. This is generated using the following code:

```
for id in campaignList:
    for link in cleanedMerged.ix[cleanedMerged["id"] == id,
"Link.URL"]:
      cleanedMerged.ix[(cleanedMerged["id"] == id) & (cleaned-
Merged["Link.URL"] == link), "Link Click Rate"] =
float(cleanedMerged.ix[(cleanedMerged["id"] == id) & (cleaned-
Merged["Link.URL"] == link), "Total.Clicks"]) /
float(cleanedMerged.ix[(cleanedMerged["id"] == id) & (cleaned-
Merged["Link.URL"] == link), "Sent"])
```

This code loops through each link and divides its “total clicks” value by its “sent” value. This reveals the percentage of users that clicked on the link.

These steps ensured that the rows in the campaign dataset would represent each link to the website from every email. Each of these rows would represent the structural aspects of these links, and contain the response to them as reported by Sailthru.

4.1.2 Data Exploration

The next stage was to create some success metrics, identify any outliers in the data that may cause issues in the modelling process and explore correlations between variables.

4.1.2.1 Creating Success Metrics

Data mining methods, such as decision trees which were used for this project, often require a target variable which a model will attempt to predict or classify. Part of the project involved creating categorical target variables that would classify a link as being good or poor. The model can then be trained to learn what features make a particular link good.

It was necessary to generate a new measure to find the success of links. The usual way of measuring the success of an email in a marketing campaign is to compare its click through rate to the industry benchmark.[27] The click through refers to the amount of users that clicked on a link in an email that led to the website. It is calculated by dividing the total click throughs an email received by the amount of customers that the email was sent to. The industry benchmark level of a good retail campaign is 2.65%.[27]

The target variable “ClickThroughBenchmark” was the first attempt at providing a target variable. It is used to describe whether the email as a whole was above or below this benchmark level. Then, every link in an email will have a value of "above" "below" or "benchmark" depending on the links benchmark level.

For each email:

```
totalOpenRatesSignups =
pd.read_csv(r"C:\Users\tdavenport\Desktop\data\Campaigns\EmailLocatio
nAndClickData\campaignStats\export-stats-zip-%s\signups-%s.csv" %
(id, id))

totalSent = sum(totalOpenRatesSignups["Sent"])
totalOpen = sum(totalOpenRatesSignups["Est. Opens"])
totalClicks = sum(totalOpenRatesSignups["Clicks"])

#click rate is the percentage of total recipients that clicked any
tracked link on the campaign
clickThrough = float(totalClicks) / float(totalSent) * 100

clickThroughBenchmarks = [2.65]
```

```
#to measure how well an email has done, we'll use three terms - below
benchmark, benchmark, above benchmark
benchmarks = ["below benchmark", "benchmark", "above benchmark"]
benchmarkLevel = ""

if clickThrough < 2.15:
    benchmarkLevel = benchmarks[0]
elif clickThrough >= 2.15 and clickThrough <= 3.15:
    benchmarkLevel = benchmarks[1]
elif clickThrough > 3.15:
    benchmarkLevel = benchmarks[2]

linksAndLocat["clickThroughRateBenchmark"] = benchmarkLevel
```

This is not an entirely accurate benchmark however. Since the granularity of the dataset is based around individual links rather than entire emails, there is a possibility that a link in an email will perform poorly, but the email as a whole will perform well. The problem with using this variable as a target is that a model will pick up what makes an email as a whole good, rather than what makes an individual link good.

The "Link Click Rate Benchmark" was the next attempt at providing a target variable. It is a linear representation of whether a link reached the benchmark level. This is calculated by first dividing the total amount of emails by the number of links. This measure assumes that if every link received this number of clicks, the email would receive a perfect amount of clicks.

However, this is not expected, as the benchmark advises that only 2.65% of this amount should be clicked. First a link click rate is generated (by dividing the click rate of the link by the number of users sent the email) for each link.

Next, the target amount for that link is found by finding 2.65% of the equal amount of clicks, with a lower and upper threshold. Links with a link click rate within this threshold can then be classified as a benchmark, and ones that are above or below will be classified as such:

```
numberOfLinks = len(currentEmail)
sentAmount = currentEmail["Sent"][0]

#dividing this figure represents an equal amount of users per link

equalEmailAmount = sentAmount/numberOfLinks

#but we aim for it to be clicked 2.65% of the time
target = equalEmailAmount * 0.0265
lowerThreshold = target - target * 0.0265
higherThreshold = target + target * 0.0265
```

Then, each link in an email is checked to see whether it is above or below or benchmark level for that email.

```
#now we can measure this as above or below benchmark

#below will be below benchmark, above will be above benchmark.

currentEmail.ix[currentEmail["Click Rate"] < target, "Link Click Rate
Benchmark"] = "Below Benchmark"
currentEmail.ix[currentEmail["Click Rate"] > target, "Link Click Rate
Benchmark"] = "Above Benchmark"
currentEmail.ix[currentEmail["Click Rate"] == target, "Link Click
Rate Benchmark"] = "Benchmark"
currentEmail.ix[(currentEmail["Click Rate"] > (lowerThreshold)) &
(currentEmail["Click Rate"] < target) , "Link Click Rate Benchmark"]
= "Benchmark"
currentEmail.ix[(currentEmail["Click Rate"] < (higherThreshold)) &
(currentEmail["Click Rate"] > target), "Link Click Rate Benchmark"] =
"Benchmark"
```

There is a slight problem with this benchmark - it assigns the same level to each link, regardless of the content (it may be an opt-out link). Also, the data suggests that links in different positions have different expected levels of clicks it cannot be assumes that every link will be clicked the same amount of times. For emails with lots of links, this sets the benchmark fairly low, so most come across as above benchmark.

The final benchmark level, called "above below Link Average", that was used in the final models was to create benchmark based on the average click rate that links received in each position, as calculated by "rowColPosition". For example, the benchmark level of links classified as being in the "top left" would be based on whether the link had a click rate that was above or below the average of all links in the "top left". This idea is based on the amount of users that click on each link in the Google search results list of links.[28] It found that most users click the first link in the list of results, and each proceeding link received fewer clicks.[28] Since the dataset contained the click rate for every link, and its position could be inferred, this allowed the average click rate for each position in a link to be calculated, and compared.

The mean level is then stored by calculating the mean for each position:

```
positions = ["Top Left", "Top Middle", "Top Right", "Middle Left",
"Middle Middle", "Middle Right", "Bottom Left", "Bottom Middle",
"Bottom Right"]
meanForEachPosition = []

for p in positions:
    meanForEachPosi-
tion.append(np.mean(cleanedMerged.ix[cleanedMerged["rowColPosition"]
== p, "Link Click Rate"]))                                    #get the
average for each position - may be sensitive to outliers ( so after
the cleaning process)
```

The percentage of each link is then compared to the mean for the position that the link is in, and whether it is above or below is saved into a new column:

```
for id in campaignList:
    for link in cleanedMerged.ix[cleanedMerged["id"] == id,
"Link.URL"]:
        pos = cleanedMerged.ix[(cleanedMerged["id"] == id) &
(cleanedMerged["Link.URL"] == link), "rowColPosition"]
        pos = pos.tolist()[0]
#because Pandas returns a series, which is more difficult to check in
a boolean index check

  clickRate = cleanedMerged.ix[(cleanedMerged["id"] == id) &
(cleanedMerged["Link.URL"] == link), "Total.Clicks"]

        linkPositionPercentage = float(clickRate) / cleaned-
Merged.ix[(cleanedMerged["id"] == id) & (cleanedMerged["Link.URL"] ==
link), "Sent"]
        linkPositionPercentage =
float(linkPositionPercentage.tolist()[0])


        if linkPositionPercentage > meanForEachPosi-
tion[positions.index(pos)]:
            cleanedMerged.ix[(cleanedMerged["id"] == id) & (cleaned-
Merged["Link.URL"] == link), "Above Below Link Average"] = "Above"
#above or below the mean level - makes it sensitive to outliers
  else:
            cleanedMerged.ix[(cleanedMerged["id"] == id) & (cleaned-
Merged["Link.URL"] == link), "Above Below Link Average"] = "Below"
```

Although this method provides a more accurate way of measuring the success of links, as it is based on a mean it is very sensitive to outliers. Links that receive an abnormally large amount of clicks can bias the average click rate of a position that it is located in. Therefore prior to creating this variable, the dataset has to have had its outliers removed.

4.1.2.2 Campaign Outliers

An outlier value of a variable is a value that is far removed from the main distribution of values.[9] It was necessary to remove outliers from the Link rate variable as it was used to calculate the "Above Below Link Average" benchmark target variable. This target, described in section 4.1.2.1, is based around the average "link click rate" for each position. Links that have a "link click rate" that is far from the main distribution of values will bias this average. They can be identified using Python's Matplotlib module's boxplot function. The outliers are shown in Figure 7:

```
cleanedMerged['Link Click Rate'].plot(kind= "box")

plt.show();
```

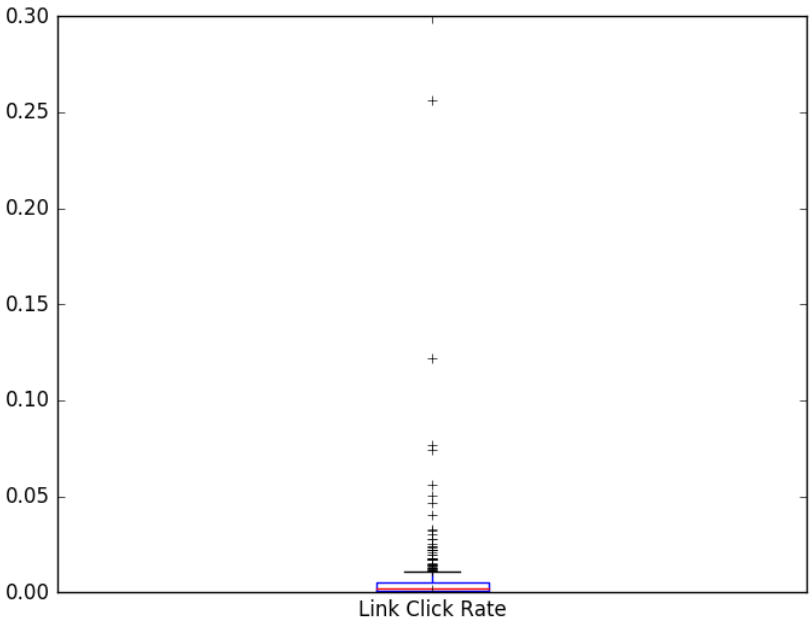


Figure 7 Distribution of Link Click Rate

Figure 7 reveals two outliers. One received around 0.255% of the total clicks of an email, and the other received around 0.13%. The main body of values appears to finish at around 0.09%. These outliers can be removed using:

```
cleanedMerged = cleanedMerged.loc[cleanedMerged['Link Click
Rate']!=cleanedMerged['Link Click Rate'].max()] # Every row that is
not the max of link click rate.
cleanedMerged = cleanedMerged.loc[cleanedMerged['Link Click
Rate']!=cleanedMerged['Link Click Rate'].max()] # do it twice remove
the next outlier as well.
```

It can be checked that the outliers are removed by plotting again. This is shown in Figure 8:

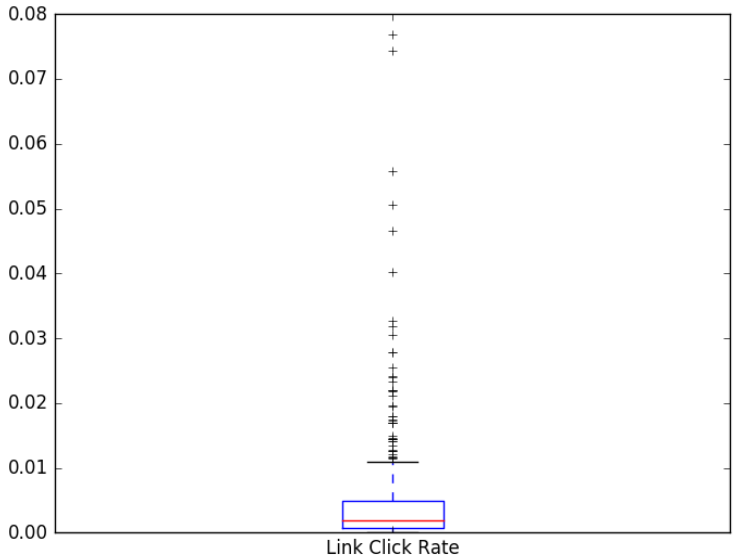


Figure 8 Distribution of Link Click Rate

These steps remove most of the outliers, and this can be seen by the scale on the left dropping from 0.3% to 0.08%, which is much more in line with the main body of values.

Each of the variables distributions were tested using similar methods. Once the outliers had been removed, statistical methods could be applied that measured the correlation between variables and models could be built.

4.1.2.3 Correlations

The corrplot package in R can be used to create a correlation plot of the numerical values in the data. Identifying whether there are correlations prior to the modelling process allows some insight into how the data is structured and some understanding of the relationship between variables in the dataset. The Pearson correlation of numeric variables in the campaign data shown in Figure 9:

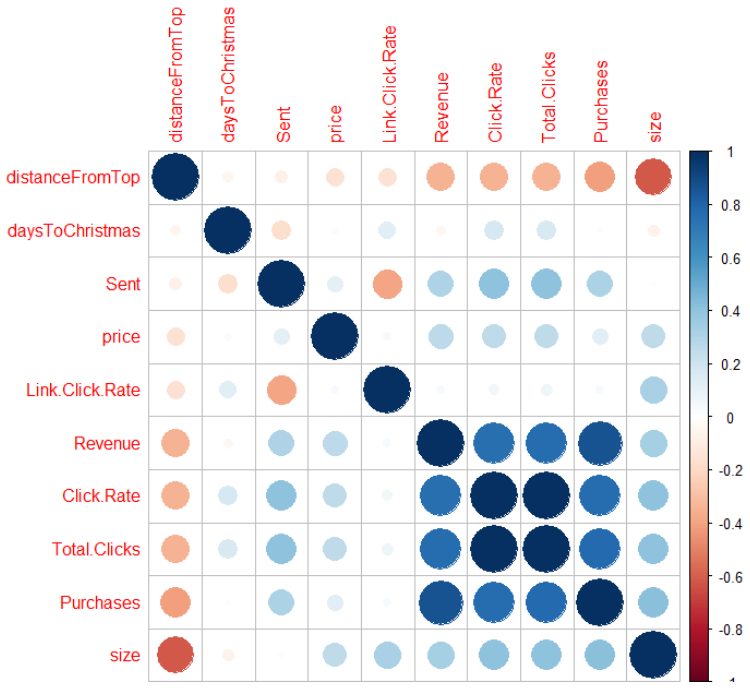


Figure 9 Pearson Correlation of Campaign data

This diagram shows the degree of linear correlation between variables in the dataset, using the Pearson correlation formula.

The colour shows whether the correlation is positive or negative. Positive correlations are shown as blue, negative correlations as red. The size of the circle and the intensity of the colour represent the strength of the correlation.

Positive correlations suggest that as the value of one variable rises, the value of the other rises. A negative correlation is where the value of one variable rises, the other falls. For

example, Purchases is shown to be negatively correlated with distance from top. This suggests that more purchases are made from links near the top of the email than those at the bottom. This diagram also suggests which variables might be linearly dependent- for example the revenue is shown to be very highly correlated with purchases, which might be expected as the number of purchases is an important part of the function of calculating the revenue.

Another set of correlations that is interesting to examine is the Spearman's correlation, shown in Figure 10:

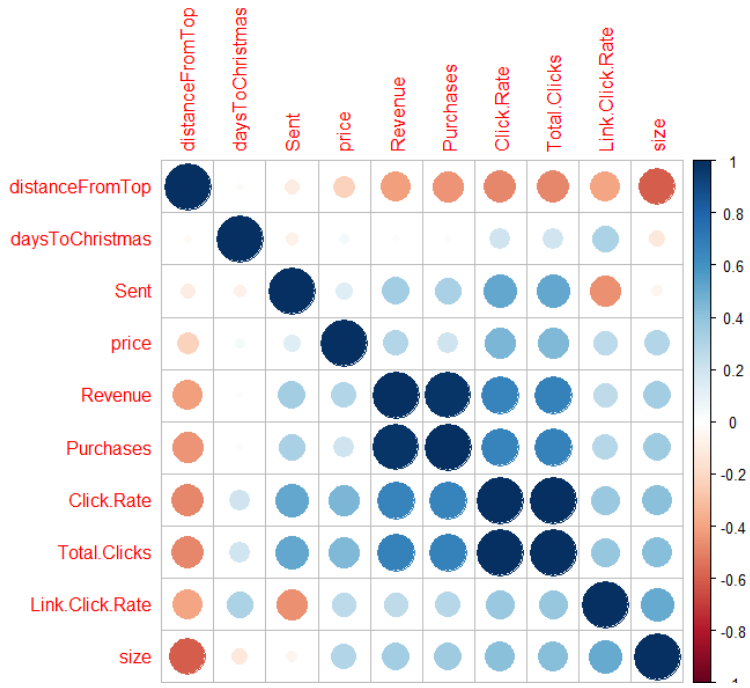


Figure 10 Spearman's Correlation

The Spearman's correlation shows which variables are monotonically correlated. It measures the strength of association between two variables. A monotonic correlation is when the value of one variable rises, the other rises or when the value of one variable falls, the other falls. Figure 11 shows the distribution plots of monotonic and non-monotonic variables:

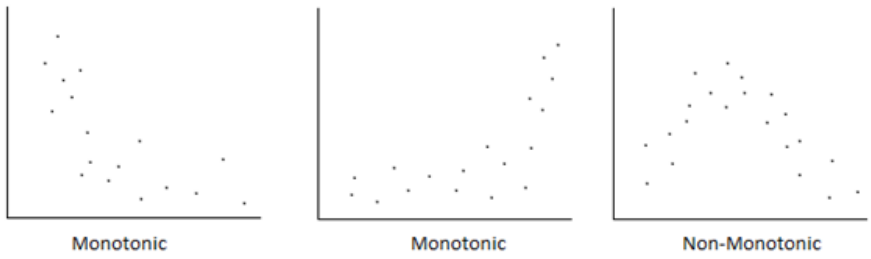


Figure 11 Examples of Correlation[29]

It is useful to also examine this method in conjunction with Pearson because it shows correlations which may exist that are not linear. This reveals that clicks and click rate are correlated

to revenue which might be expected, but also picks up on some others that Pearson seemed to miss such as between “days to Christmas” and “click rate”. Correlations that have a greater Spearman’s figure than Pearson’s will be a monotonic but not linear correlation. [29]

These correlations are useful when choosing which variables to include in the modelling process. Very highly correlated variables can have an adverse impact, as models that are built may focus on this relationship. For larger datasets, it can also be computationally inefficient, as it only takes one of the variables to describe this relationship from the data. [47]

4.1.3 Data Modelling

The project used the decision trees for modelling the data for both the campaign and user section. They were created using the Rpart package in R. This package generates decision trees based on the Gini coefficient of variables relationship to the target class.[44]

A decision tree model aims to classify a data point according to the values that it has. A data point is could be a row in a table, or in the Campaign set, it refers to a link in an email and all the variables that have been generated corresponding to it.

In order to classify data, the decision tree model requires a set of training data of historical examples that have a target class. For example, in the campaign dataset, this target could be whether a link has been classed as “above”, “below” or “benchmark”. The decision tree model attempts to partition the data in a way that each of the values for each data point in each target class have around the same values.[9]

This is achieved by passing each data point through a series of splits. The decision tree model is made up of nodes that each represent a variable from the dataset. These nodes are a split of the dataset, for example in figure 12 the first node is the “Sent” variable and the split is whether data points had a sent value of above 4186 or below. The first node is called the “root node” and splits the data the most into the correct target class which means that it reveals the most information about which target class a data point should be classified as.[9]

After each split the amount of data points in each target is calculated, and further splits, (or branches), may be required. Eventually, all the data points will be in the same target class, and the model will have reached a “leaf node” which will be used to classify new unseen data points. To test the model, new data points are queried by each branch, and their value determines which leaf node the data point will finish at.[45] The amount of times the test data point was classified correctly can be used as a measure of success of the model.

The rpart package create the splits and orders the branches by the "purity" of the data points that are present at each node. In figure 12, the colour of a node represents the majority class of data points at that point in the tree. The first line describes the majority target class at that point, for example, at the root node this is "below". The next line describes the percentage split of the target examples at that point. In the root node of figure 12, 80% are "below" examples, and 20% are above examples. The final line shows the percentage of the dataset at that point, with the root node having 100% of the dataset. The split is based on the probability that the data points resulting from a split will be in the same target class.[46] The Gini index for each split at each point is calculated, and the one with the higher score is used as the split at each point. For the example in figure 12, this is calculated by comparing the probability that a data point will be "above" and "below" for the split of each variable. The one with the highest probability will be used as the split.

Decision trees were used because of their accessibility. The aim of the project was to identify knowledge from the data and the readability of decision tree models aids this. An alternative may have to been to build "black box" models such as neural networks or random forests, which may have benefited from greater accuracy, but the reasoning behind the models would have been lost.

The first final model used all the variables except target variables and "clicks", "purchases", "revenue", "ID" and "Link URL". A full list of all the variables available for the campaign section is available in the appendix. This model used "Above Below Link Average" as a predictor, and created the following model shown in Figure 12:

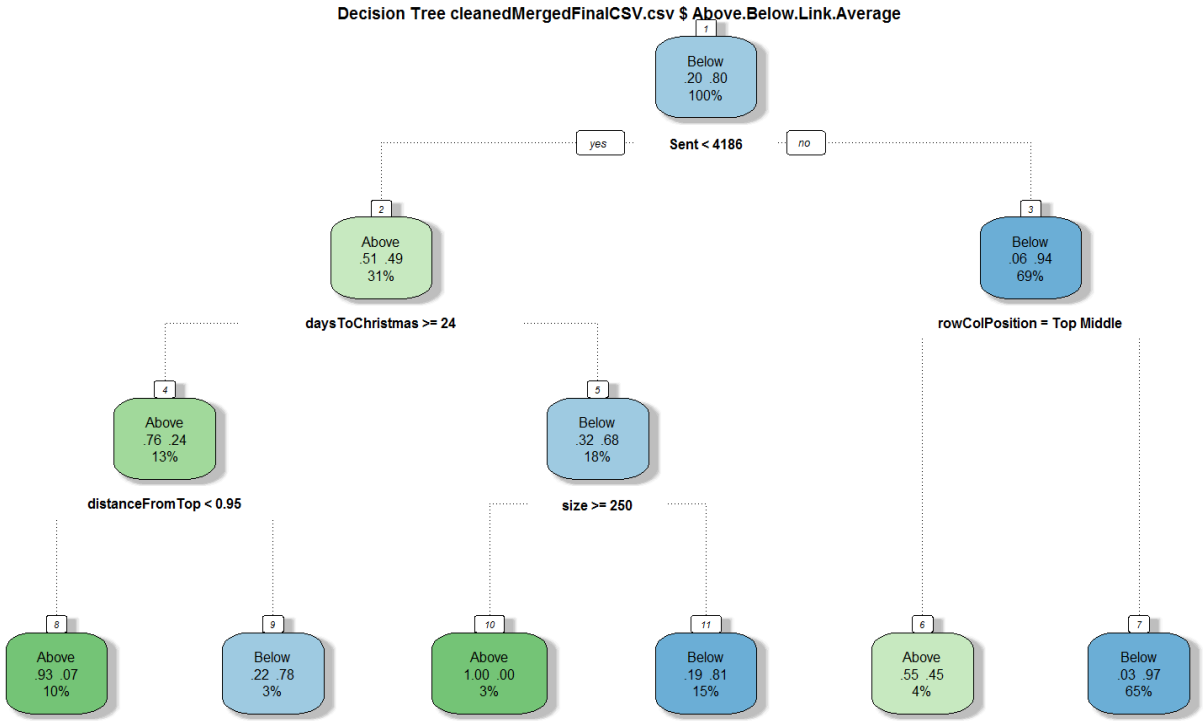


Figure 12 Final model, Above Below, Link Average as predictor

This model identifies that the targeted emails which are sent at least 24 days before Christmas perform better. Of those that were after the 24 days before deadline, smaller picture sizes had a below average amount of clicks. Those sent to more than 4186 that weren't in the top middle tended to also perform poorly. It may also be noticed that some variables are missing. This is because the model is built by choosing splits that lead a predicted example to a target class. Often, splitting on variables that will not reach the target variable more effectively will only mislead the model and will not contribute to an effective classification and will be ignored.[9] The following confusion matrix in Figure 13 describes the distribution of the correct classifications and errors that were made:

	Predicted	
Actual	Above	Below
Above	9	6
Below	0	45

Figure 13 Confusion Matrix for First Final Model

A confusion matrix shows how many times the model was correct when it was tested on a set of data. The numbers in the leading diagonal represent correct classifications and the numbers not on this diagonal represent incorrect classifications. This confusion matrix is based on the models performance on a validation set of unseen data that had not been used to test or train the model previously. This confusion matrix shows that the model classified 9 examples from

the validation set as above, and 45 as below. It also classified a further 6 examples as being below that were actually above. This means that the model itself is fairly accurate, but perhaps that the examples in the validation set is either biased towards "below" or that the target variable is biased as a whole towards "below".

Although this model had a good classification ability, it was not particularly informative. It was already known that emails sent before the cut-off date for Christmas deliveries and targeted emails perform better. Removing these variables would identify further interesting patterns in the data. The following model shown in Figure 14 uses the same variables as previously, except the "sent" and "days to Christmas" variables:

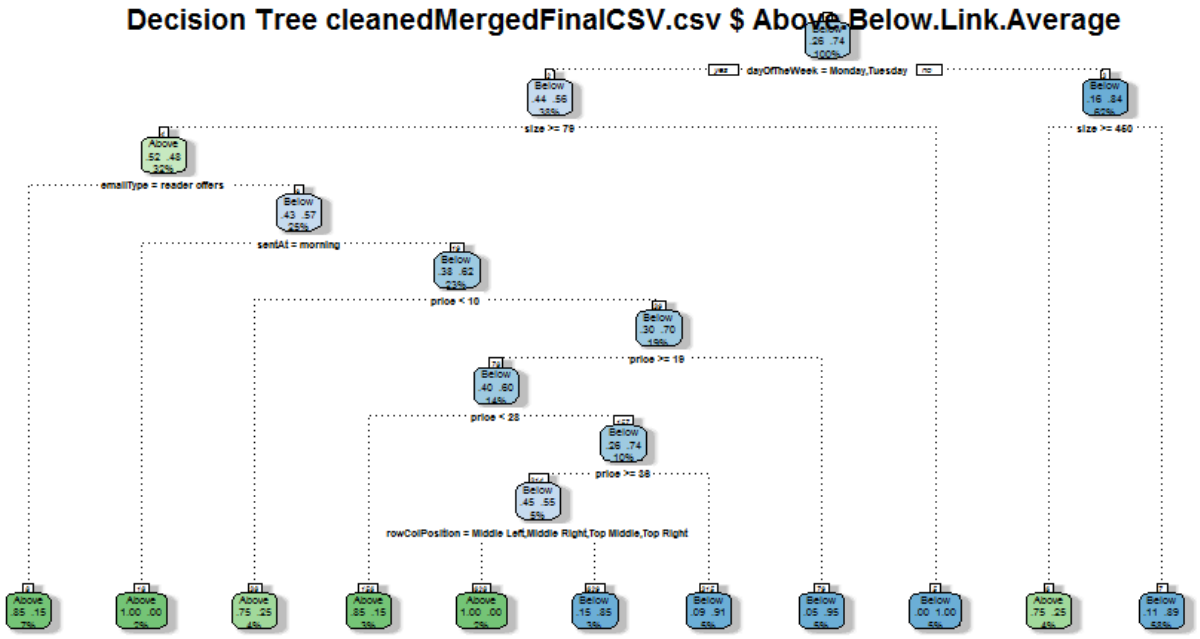


Figure 14 Second Final model. Larger version in appendix.

This model shows that subscriptions that are not near the top of the page are more often below the average link click rate for their position. It also identifies Wednesday and Friday emails as performing less well. The middle right, top middle and top right appears to be prime space for clicks (perhaps this is the right handed hand position for an iPad, the most common device for opening emails). Bigger pictures, emails that were sent in the evening or morning that had a higher price also seemed to offer a better chance of being a more successful link position. The main thing to take away is that links further down the email and emails not sent on a Monday or Tuesday performed worse. The confusion matrix for this model is shown in Figure 15:

	Predicted	
Actual	Above	Below
Above	7	8
Below	4	41

Figure 15 Second Model Confusion Matrix

This model has slightly more error, and has miss-classifying 4 "below" examples as "above", and 8 "above" examples as "below". This is still a fairly good model, but the imbalance is still present, although this may reflect the data.

It is also interesting to see what makes the email as a whole effective, rather than just individual links, and this can be measured using the "ClickThroughRateBenchmark" variable. This variable provided a classification for each link of "above" "below" or "benchmark" depending on if the email the link was included in met the industry benchmark as a whole. This tree is shown in Figure 16:

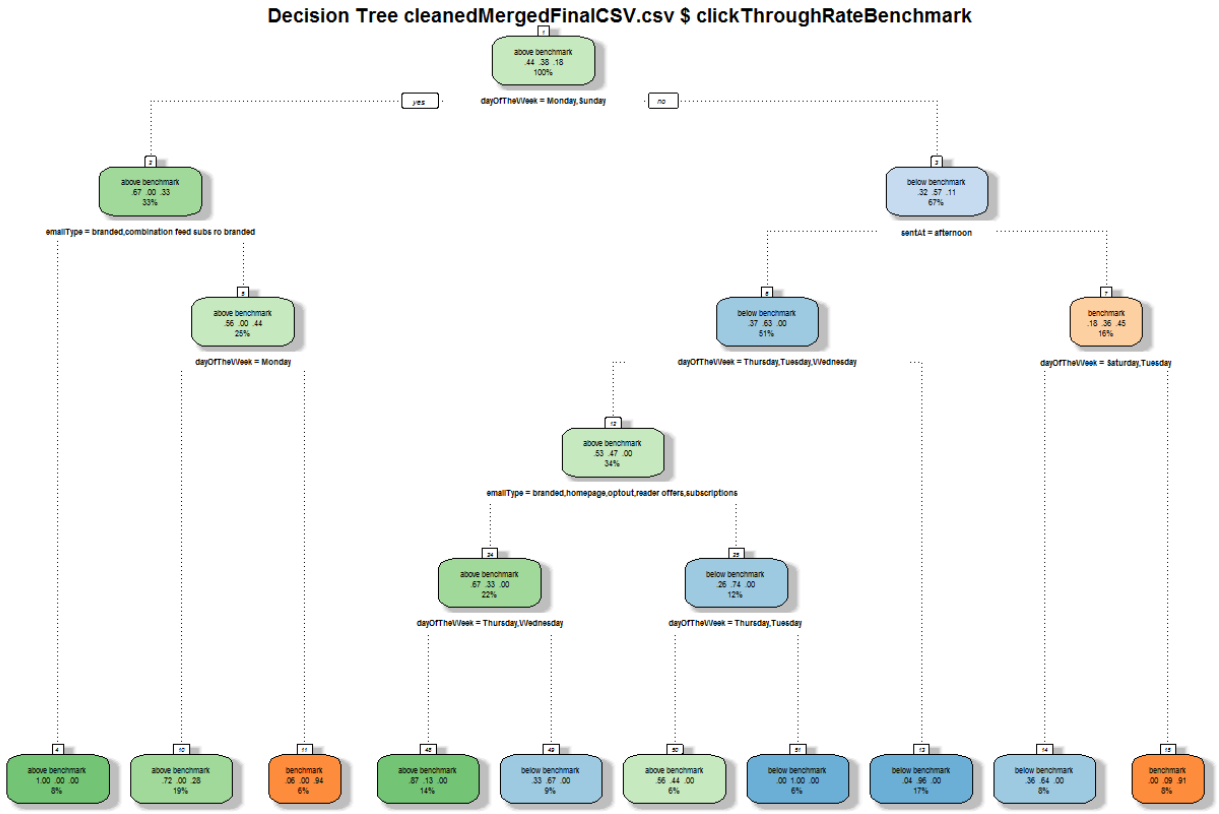


Figure 16 Click Through Rate Benchmark Tree, larger version in appendix.

On the level of a whole email (this may be biased towards emails sent to a large number with lots of links) it appears that the targeted emails are more likely to have an above benchmark level. Wednesday and Sunday emails were also less likely to perform well. Afternoon emails also performed less well. The confusion matrix for this tree is shown in Figure 17:

	Predicted		
Actual	above benchmark	below benchmark	benchmark
above benchmark	25	5	1
below benchmark	5	13	0
benchmark	2	0	9

Figure 17 Confusion Matrix for Click Through Rate Benchmark Tree

The model has a good overall classification ability, but tends to bias its predictions towards the "above" and "below" levels. This reflects the distribution of examples in the data however, as the data contains fewer "benchmark" examples.

These models allowed some conclusions to be drawn from their structure, and were presented as an infographic. The importance of variables was also calculated using some variable importance measurements based on the output of creating decision trees which is presented in the results section.

4.2 Users

The Users section aimed to identify shared characteristics between groups of users that purchased from emails. It involved several stages: exploring the dataset, preparing the dataset, creating samples from these and creating models.

4.2.1 Data Exploration

The User dataset is made up of three main datasets.

The first set was the store of data that Sailthru keeps on all users. This dataset is updated every time that the system is able to record an interaction between the user and the website. It stores the date they signed up, any mailing list that they are on, the price of the most expensive item they have purchased, the price of the last item that they purchased and more – (there is a full list of variables generated for the model in the appendix). If they have bought an item through the website, this will also capture their location from their billing address.

The second dataset extended what was known about the user. The first dataset only looked at their online behaviour, and it was also considered interesting to see whether their offline behaviour had an impact on whether they purchased through emails. Specifically, the organisation's Oracle BI database was queried to see whether customers' first names, surnames and postcodes were recorded as purchasing from the DC Thomson Shop using the telephone or in writing using a brochure. If there were customers in the first set that existed in the second set, a value for a variable called "purchased by phone" or "purchased via brochure" was recorded as "True". This dataset would allow an insight into whether these offline methods had an impact on online purchasing.

The third dataset came from the list of users that were sent each campaign email. This dataset includes their email address, when they opened the email, when they clicked through to the website, and whether they purchased from the email. As this list contains the user's email address which is present in the set of lifetime data from Sailthru, it means that these two datasets can be merged. The resultant dataset shows every individual email sent out over the Christmas period, every response by a user and all the information that is known about the user.

The combination of these datasets should be able to give some insight into what drives users to purchase from emails. One of the major challenges was missing data from the datasets. The Sailthru platform can only collect data about users when they interact with the site which meant that for unengaged users that may have only signed up to receive emails, not purchased from the site, or rarely opened emails, little information was captured about them.

This missing data caused the data to become unbalanced, as the main factor that initial models would identify would be the patterns of missing data. The models would identify users in the dataset with patterns of missing values as "non purchasers" by definition, rather than by design. This means that models trained on this unbalanced dataset would identify the patterns of missing data rather than the relationships that cause a user to "purchase" or not.

This challenge was met by generating new variables such as gender or location based on factors that were commonly present in the dataset that were probably required to be entered when the user signed up. To address the imbalance between "responders" and "non-reponders" samples of the dataset were taken that could be used to build models that could identify relationships in the data that were not only based around missing values.

4.2.2 Data Preparation

The script "userCleaning.R" is designed to explore different variables and also identify potential outliers from the Sailthru information about users that are on the DC Thomson shop mailing list. The script explores each of the variables in turn to examine their distribution of values, to identify whether they would be of use to the data mining process or if they can be used to generate other variables based on their value.

The first variable "domain" has the following distribution shown in Figure 18:

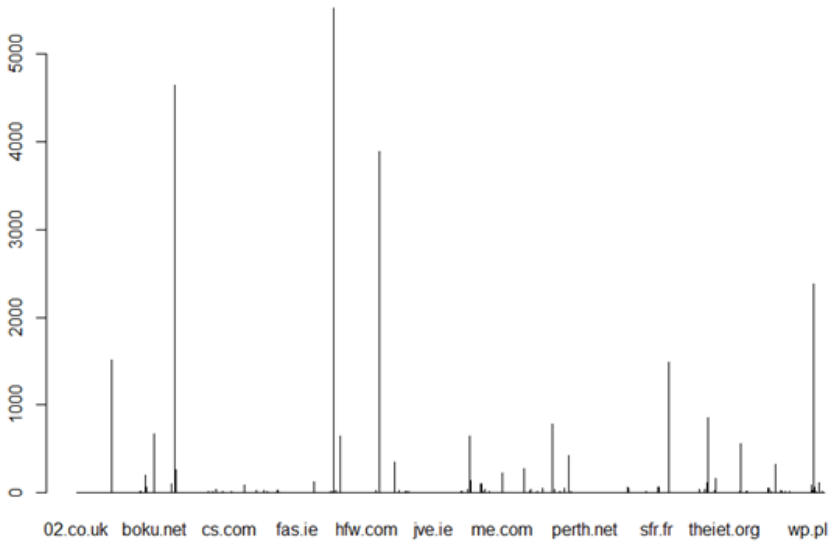


Figure 18 Distribution of Domain variable

The rows of this variable include the email domain (Hotmail, Gmail, Outlook etc.) for each user's email address. There are about 50 main ones used, as can be seen by the high peaks on the graph. The length of the x axis the histogram above shows that there is a high proportion of domains that are only have one example. By plotting the frequency of the domain names in the dataset, this reveals that there are some main domain, including "gmail.com" and "btinternet.com" but there are also 4070 examples of domains that have less than 7 occurrences. Data mining is about finding specific classifications from general data, and data that is too specific for example, an email domain that has only one example, such as "noproblemcomputing.com", cannot really be used in the data modelling process because it is not possible to generalise from one example.

Domain may be an important value for prediction as research suggests that similar demographic groups use the same email provider. [30] It was therefore considered better to try and adapt this variable so that it could be used in the data mining process. The variables with fewer examples should not just be deleted, as the fact that they have few examples may reflect the kind of domain that they have chosen to use as they include personal domains, business addresses or domains from other countries.

Examining the dataset, it was found that many of the single domains were in fact falsely entered examples of the most common domains, such as "o2.co.uk" when it should have been "02.co.uk". This meant that it was possible to correct these entries to what could be the near-

est approximate correct domain. The following code shows the process that was undertaken to correct these false entries:

```
domainCount = count(sailthruUsers, "Domain")
#Order domain frequency count in reverse - from most used domains to
least.
domainCount <- arrange(domainCount, -freq)
#from the chart and the frequency table, we can suppose that the pop-
ular domains (gmail) etc. make up the majority.

#What about poorly formatted domains? Let's take the top fifty fre-
quent (they're the majority domains) as examples of the correct
format.
head(domainCount, n=100)
correctFormats <- domainCount$Domain[1:50]
correctFormats <- droplevels(correctFormats)
correctFormats
#for the examples of one frequency, these may have been input incor-
rectly. Match domains that are similar to the top list, and replace
them with the most likely candidate.
oneExample <- subset(domainCount, freq == 1, select = Domain)
oneExample <- droplevels(oneExample)
# see if the incorrectly entered examples can be approximated
outerIndex <- 1
while (outerIndex <= nrow(oneExample)){
innerIndex <- 1 #for looping through the correct Formats list
matchList <- vector() #intialise empty vectors to hold match list
while(innerIndex <= length(correctFormats)){ #loop through each cor-
rect format - needs an outer loop to loop through main list
 count <-
adist(correctFormats[innerIndex],oneExample$Domain[outerIndex],
counts=FALSE) #Calculate the approximate string distance (see le
#venshtein distance)
 matchList <- c(matchList, count) # this will be a vector of
(1,4,5,6,7 etc) with the distance between the example each popular
domain
 innerIndex <- innerIndex + 1
 }
smallestMatchIndex <- which(matchList == min(matchList)) #find the
#index of the most likely

 if (length(smallestMatchIndex) == 1){ #if the min() function only
#finds one lowest example
 if(matchList[smallestMatchIndex] < 3){ #as long as there is only one
#example standing out as the lowest, it offers a degree of certainty
#that its similar to another example
 mapvalues(sailthruUsers$Domain, from=
sailthruUsers$Domain[oneExample$Domain[outerIndex]], to = correctFor-
mats[smallestMatchIndex])
 }
 }
 outerIndex <- outerIndex + 1
}
```

The top 50 most occurring domains were stored, and then each example of single domains from the dataset was iterated through. During each iteration, the Levenshtein distance between each example of a single domain from the dataset and each of the top 50 domains was

calculated.[31] This is a measure of how many inserted characters, deleted characters or substituted characters exist between one word being another. For example, the Levenshtein distance between "o2.co.uk" and "02.co.uk" is 1, or represented as "SMMMMMM" (where S is Substitute and M is Match) as there is only one substitution required to make "o2.co.uk" into "02.co.uk". Finding examples where the distance was only one or two meant that there is a degree of certainty that the single example is really an example of one of the common types. This method could be included in the registration process as a quick way to verify email domains before they are entered into the system. This allowed a portion of the data errors from the set to be corrected.

There were however still a significant amount of domains with low frequency in the dataset, that were not encoded incorrectly and were unique. Since the scarcity of these domains is something that defines them, it was acceptable to split those with low frequency into three main classes of: "single example", "very infrequent" and "infrequent". Figure 19 shows the "domain" variable to have a much better distribution:

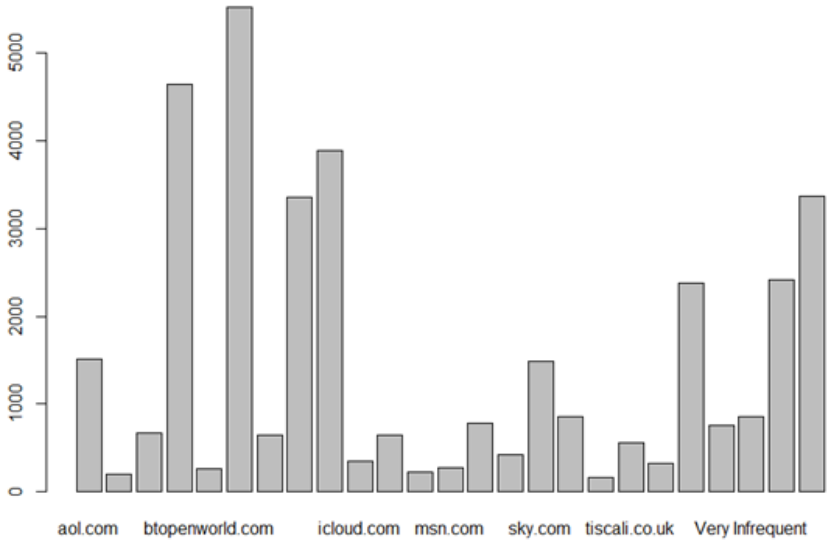


Figure 19 Distribution of Domain Variable

There are still some smaller minority classes which suggests that the levels that designate the frequency may need to be altered, but it makes the variable as a whole much more acceptable for the modelling process. It reduces the number of different domains in the dataset from around 5700 to 53.

Some variables contained outliers, and a way to visualise these is to check is to plot the variable to see the distribution as shown in Figure 20:

```
plot(sailthruUsers$Purchase.Price)
```

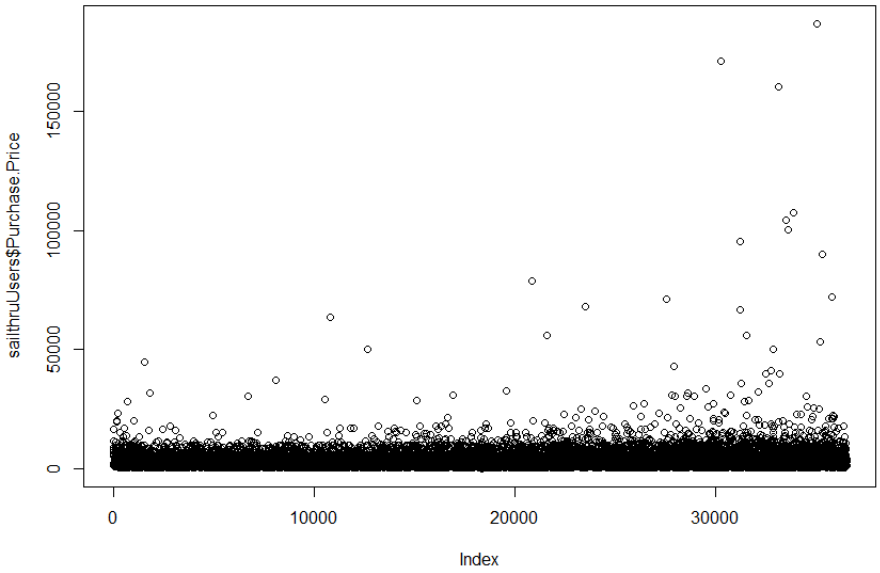


Figure 20 Distribution of Purchase Price

The graph above shows three examples which are far from the spread of the rest of the data. They can be easily removed using:

```
sailthruUsers <- sailthruUsers[-
which.max(sailthruUsers$Purchase.Price),]
sailthruUsers <- sailthruUsers[-
which.max(sailthruUsers$Purchase.Price),]
sailthruUsers <- sailthruUsers[-
which.max(sailthruUsers$Purchase.Price),]
```

This tells R to keep all the rows except the one which contains the maximum purchase price, and when repeated three times will remove the top three maximum prices.

The dataset contains several important dates pertaining to the user. Individual dates tend to be too specific to be useful in the data mining process, so two different kinds of variables were calculated from them. One variable looked at the time attached to the date, to get an understanding of the time of day that the time shows. For example, the “profile created date” variable can be changed into a "signUpTimeOfDay" variable, using:

```
datetime <-sailthruUsers$Profile.Created.Date
hour <- as.integer(substr(datetime, 12, 13))
conversion <- data.frame(datetime=datetime, hour=hour,
 period=cut(hour, c(-Inf, 6, 10, 12, 17, Inf),
 labels=c("night", "morning", "noon", "afternoon", "evening")))
#add new column
plot(conversion$period)

sailthruUsers$signupTimeOfDay <- conversion$period
```

The other calculation with dates was used to find the number of days between certain dates. For example, an "ageOfAccount" variable can be created by taking the signup date from today's date:

```
sailthruUsers$ageOfAccount <- as.Date(as.character(Sys.Date(), for-
mat="%Y/%m/%d"))-
 as.Date(as.character(sailthruUsers$Profile.Created.Date), format
="%d/%m/%Y")
ageOfAccountPlot<- data.frame(table(sailthruUsers$ageOfAccount))
```

The User data set contains the first name of every user. First names are not really useful in data mining, as on their own they tend not to provide any relevant or useful relationships, or are too uniquely distributed to suggest patterns in a model. It was however useful in that it was used to generate a new gender column. There are datasets that exist that hold the most common gender for a first name. The "gender" package that can be installed for R contains the Kantrowitz dataset which is a historical set of first names and associated genders. The following code segment recodes the gender variable as "NA" (which was originally poorly formatted and mostly missing) and then queries each first name against the Kantrowitz dataset to determine the likely gender:

```
sailthruUsers$gender <- NA

index <- 1
while(index <= nrow(sailthruUsers)){
 currentGender <- gen-
der::gender(as.character(sailthruUsers$first_name[index]), method=
"kantrowitz")
 sailthruUsers$gender[index] <- currentGender$gender
 index <- index + 1
 }
sailthruUsers$gender <- unlist(sailthruUsers$gender)
sailthruUsers$gender <- as.factor(sailthruUsers$gender)
```

The Kantrowitz package is useful, but for names where the gender could be either (such as Ashley) are labelled as "either". These examples can be sent to another external set from genderize.io which provides a RESTful API. This dataset is based on current social media profiles, and so can give a much more modern idea of the gender of names than the Kantrowitz method. It can be accessed using:

```
splitUnknown <- split(unknownGender, ceil-
ing(seq_along(unknownGender)/10))
genderizeGender <- vector()
genderizeName <- vector()

outerIndex<- 1
#loop through each set of 10
while(outerIndex <= length(splitUnknown)){
 innerIndex <- 0 #genderize likes indexes to start at 0
 eitherNames<- splitUnknown[outerIndex]
```

```
eitherNames <- unlist(eitherNames, use.names = FALSE)
outUrl <- "https://api.genderize.io/?"
outUrlEnd <- "" #we're aiming for the format
#name[0]=peter&name[1]=lois&name[2]=stevie
while(innerIndex <= length(eitherNames)-1){
 item <- paste('name[', innerIndex,']',
'=',eitherNames[innerIndex+1], sep = "") #glue these together
 outUrlEnd <- paste(outUrlEnd, item, sep = "&")
 innerIndex <- innerIndex + 1
 }
outUrlEnd <- substring(outUrlEnd, 2) #remove the first & symbol
outUrl<- paste(outUrl, outUrlEnd, sep = "")
conn <- curl_with_proxy(outUrl)
 lines <- readLines(conn)
close(conn)
 rm(conn)
 lines <- jsonlite::fromJSON(lines)

 genderizeName <- c(genderizeName, lines$name)
 genderizeGender <- c(genderizeGender, lines$gender)
randInterval <- runif(1,0.5,1.2) # a random time interval between
half a second and 1.2 seconds
Sys.sleep(randInterval) #To avoid overwhelming their server
outerIndex <- outerIndex + 1

 }
```

The API only allows bulk requests of up to ten, so the set of "either"s have to be split up. They are then formatted into a URL that is sent to the API. The response is then parsed into JSON, and the name and gender extracted into different vectors. The loop then waits for a random time interval, to avoid overwhelming the servers at genderize.io. The results are then inserted into the gender column using:

```
index <- 1
while(index <= length(genderizeName)){
 nameCheck <- genderizeName[index]
 changeIndicies <- which(sailthruUsers$first_name == nameCheck,
arr.ind = TRUE)
 sailthruUsers$gender[changeIndicies] <- genderizeGender[index]
 index <- index + 1

}
```

Postcodes on their own, for a small dataset anyway, tend not to be very useful variables to model as similar to first name, they tend to be rather unique. They can however be used to generate wider geographical variables, which can then be used to merge demographic information about those regions.

The Oracle BI system for DC Thomson has access to the Experian demographic dataset that also has a region name for a postcode (North England, Wales, Scotland etc.). The Experian demographic dataset provides a description of the group of people that live in a postcode area.[48] It is based around the idea that people who live in the same area tend to have the same socio-economic circumstances, and provides classifications for an area such as “cafés and

catchments".[48] Each of these demographic classifications are applied to many postcodes. This means that it can be used to create identify general relationships surrounding demographic classification from a model.

There were enough postcodes in the set that the BI system could be queried in batches using generated SQL statements. First the set of correct postcodes (determined by a regular expression from [32], which checks for valid formatted postcodes) were sent to a CSV file:

```
#first look at those postcodes that pass the validation check
postcodesOut <- postcodes[unlist(lapply(postcodes, function(x)
grepl("^(([gG][iI][rR] {0,}0[aA]{2})|((([a-pr-uwyzA-PR-UWYZ][a-hk-yA-
HK-Y]?[0-9][0-9]?)|(([a-pr-uwyzA-PR-UWYZ][0-9][a-hjkstuwA-
HJKSTUW])|([a-pr-uwyzA-PR-UWYZ][a-hk-yA-HK-Y][0-9][abehmnprv-
yABEHMNPRV-Y]))) {0,}[0-9][abd-hjlnp-uw-zABD-HJLNP-UW-Z]{2}))$",
x)))]
write.csv(postcodesOut,
"C://Users//tdavenport//Desktop//data//postcodeBatches.csv")[32]
```

The Experian dataset provides demographic data on three levels, regional, Scottish and UK. The regional data is used to describe what region the postcode is in such as "North West" "London" "Wales" and so on. The Scottish and UK dataset is split into two further levels, demographic group and demographic type. The demographic group is a wider demographic classification of 15 groups that describes the general socioeconomic situation of those living there, such as "Prestige Positions" or "Transient Renters". The demographic type is a further sub classification of 63 sub-groups, which aims to describe their position in further detail, such as "Central Pulse" or "Dependent Greys". The following code queries the Experian set to identify the region, whether it has a Scottish demographic classification and inputs the UK classification. If a postcode is not found, the demographic column and region column are left blank.

```
index <- 1
#Loop through each row of Users set
while(index <= nrow(sailthruUsers)){
 #extract each postcode
 postcode <- sailthruUsers$zipcode[index]
 #set as character rather than factor
 postcode <- as.character(postcode)
 #get the row info for that postcode - if it's there
 postcodeInfo <- postCodeLookUp[which(postCodeLookUp$UK.Postcode ==
postcode, arr.ind = TRUE),]
#If it's not missing from the Sailthru set
 if (is.na(postcode) != TRUE|postcode != "" ){
 #append the region info to the row for the postcode
 govRegion <- as.character(postcodeInfo$GovernmentRegion[1])
 sDemGroup <- as.character(postcodeInfo$ScotlandDemographicGroup6[1])
 sDemType <- as.character(postcodeInfo$ScotlandDemographicType6[1])
 ukDemGroup <- as.character(postcodeInfo$UkDemographicGroup6[1])
 ukDemType <- as.character(postcodeInfo$UkDemographicType6[1])
```

```
 sailthruUsers$region[index] <- govRegion
 #if it's a Scottish postcode
 if (is.na(sDemGroup) != TRUE){

 if (sDemGroup != "Not in Mosaic Scotland Area"| sDemGroup != "Un-
classfied" | sDemGroup != "Unknown"){
 sailthruUsers$scotland_demographic_group[index] <- sDemGroup
 sailthruUsers$scotland_demographic_type[index] <- sDemType
 }
 #append the uk demographic group
 sailthruUsers$ukdemographic_type[index]<- ukDemType
 sailthruUsers$ukdemographic_group[index] <- ukDemGroup

 }

 }


 index <- index + 1
}
```

Next, some other "offline" data from BI was included. "Offline" data refers to interactions with the customer that did not occur online. First, the methods of purchase for everything from the shop which included a postcode and a surname for items bought between the 1st October and the 31st December were downloaded from the BI system. This dataset shows the different channels for purchase. For example, "Individual req via Phone" refers to a telephone order, and "Individual in writing" refers to a brochure order in writing. If a postcode and surname appears together in this CSV, then they can be joined together in the following code:

```
#get the names from BI to match the column names in the sailthru set
colnames(BIBrochuresAndTelephone)[which(names(BIBrochuresAndTelephone
) == "Postal.Code")] <- "zipcode"
colnames(BIBrochuresAndTelephone)[which(names(BIBrochuresAndTelephone
) == "Surname")] <- "last_name"
#join them together
total <- inner_join(sailthruUsers, BIBrochuresAndTelephone,
type="inner")


sailthruUsers$boughtViaPhone <- NA
sailthruUsers$boughtViaBrochure <- NA

index <- 1
matches <- 0
#loop through every row
while(index <= nrow(total)){

 postcode <- as.character(total$zipcode[index])
 surname <- as.character(total$last_name[index])

 postcodeInfo <- total[which(total$zipcode == postcode & to-
tal$last_name == surname, arr.ind = TRUE),]

 brochure<- "FALSE"
 phone<- "TRUE"
```

```
 if("Individual req via Phone" %in% postcodeInfo$ChannelName){
 phone <- "TRUE"
 }
 else{
 phone<- "FALSE"
 }

 if("Individual in writing" %in% postcodeInfo$ChannelName){
 brochure <- "TRUE"
 }
 else{
 brochure <- "FALSE"
 }
 #search BI set for entries that contain a postcode and surname
 #that match the postcode and surname of the row under consideration

brochureOrPhoneIndex <- which(sailthruUsers$zipcode == postcode &
sailthruUsers$last_name == surname, arr.ind = TRUE)

 sailthruUsers$boughtViaPhone[brochureOrPhoneIndex] <- phone
 sailthruUsers$boughtViaBrochure[brochureOrPhoneIndex] <- brochure
index <- index + 1
}
```

This code loads the CSV downloaded from BI, loops through each row of the User dataset, and appends the region and demographic info that match the surname and postcode into the main set.

The unused variables, including those that are either empty, skewed or too poorly formatted are then dropped from the dataset using:

```
drops <- c("username", "address_line_one", "address_line_two", "ad-
dress_line_three", "phone_mobile", "birth_date",
 "company_name", "industry_sector", "lev-
el_of_responsibility","marketing_campaign", "business_category",
 "is_post", "is_phone", "is_sms", "is_third_party", "by_air",
"by_coach", "by_ocean_cruise", "by_river_cruise",
 "by_own_way", "submission_date", "is_prod", "subscriber_id", "sta-
tus", "website", "store", "store_code",
 "store_id", "suffix", "created_at", "recurring_payment", "sig-
nup_date", "sub_brand", "is_sub", "date_subscribed",
 "email", "site", "UID", "wedding_role", "swd_partner_1",
"swd_partner_2", "wedding_date", "campaign",
 "last_opened_date", "surname", "nickname", "wedding_area",
"phone_number", "previous_email", "list_name",
 "address_line_one_line_one", "retype_email", "brand",
"last_opned_date", "date_of_birth", "address_line_one_3",
 "address_line_one_2","address_line_one_1", "last_name",
"is_competition", "is_comp", "subscription_duration",
 "payment_method", "title", "birth_year", "birth_month", "birth_day",
"prev_sub", "by_rail", "lastname",
 "is_brochure", "salutation")

sailthruUsers = sailthruUsers[,!(names(sailthruUsers) %in% drops)]
```

4.2.3 Sampling

It is necessary to sample the dataset for use in the modelling process because of there is an uneven split between those that purchased and those that did not purchase which is used as the target variable. This skew would cause models that are built from this set to either have very poor predictive ability or will only predict any example as a non-purchaser, as that is all the model will be trained upon. Sampling the dataset means that the model that is built will be based on relationships within the data rather than on the pattern of missing data. Two different kinds of sampling techniques were used to generate datasets for the modelling process, stratified sampling and a dataset generated using SMOTE.

A stratified sample is a sample that takes a sample from a larger dataset based on a condition.[33] In this case, the condition is whether the user that received the email purchased or not. The script uses Pandas' "sample" feature to take different samples from the dataset. There are 944 examples of purchasers in the dataset. To make an equal split, 944 examples of non-purchasers can also be taken from the dataset.

```
import pandas as pd

fullDataset =
pd.read_csv(r"C:/Users/tdavenport/Desktop/data/users/mergedUserCampai
gn/idLessmergedConcattedCampaignReceps.csv")

purchasedSample = fullDataset.query("purchased == True")

noPurchaseSample = fullDataset[fullDataset.purchased !=
True].sample(944)

#100 random samples of non purchasers:
for i in range(100):
    noPurchaseSample = fullDataset[fullDataset.purchased !=
True].sample(944)
    total = pd.concat([purchasedSample,noPurchaseSample])
    to-
tal.to_csv(r"C:\Users\tdavenport\Desktop\data\users\mergedUserCampaig
n\randSamps\samp%s.csv"%(i), index= False)

total = pd.concat([purchasedSample,noPurchaseSample])
```

Here, the "purchasedSample" is the subset of the data where the "purchased" value is true. The "no purchased" sample is where this is not true. The loop runs 100 times, creating 100 different non-purchaser examples and using the same purchaser set, to create 100 stratified samples. Models can then be built and compared on each sample, which allows the average importance of a variable to be calculated. This allows the variance of the dataset to be taken into account.

SMOTE (Synthetic Minority Oversampling TEchnique) is a method for oversampling datasets.[34] It is used to generate a sample of the dataset that emphasises the role of a minority value.[34] In this case, the number of users that are "purchasers" is the minority. SMOTE takes a sample of rows from the majority class and a sample from the minority class. It then generates new artificial examples of the minority class, which are based on the values of the rows that exist already in the majority class. This causes the minority class to become "over sampled".[34] This allows a data model to pick up more of the relationships that govern the target class than would be possible if this had not been carried out. This can be implemented as follows:

```
users <- read.csv(file.choose(), na.strings = c("", NA, "NA"),
stringsAsFactors = T)

smotedSet <- SMOTE(purchased ~ ., data = users, perc.over = 200,
perc.under = 100, k = 10)
 # purchased depends on everything else.  percent over = 200, percent
under= 100, imputed values are based on the 10 (k) nearest neighbours
```

This creates a synthetic dataset that will have more examples of "purchase" emails than the stratified sets. It can be useful in that it can emphasise the relationships that exist within groups of the minority of "purchase" e-mails. The risk is that it no longer reflects the "real life" version of the data, and may cause models that are created to be "over generalised".[35] This means that the models that are created will classify items too broadly, which would reduce the accuracy of a model on unseen data. It depends on the dataset however, and these results can be tested by validation procedures.

4.2.4 Data Modelling

Some attempts were made to create models from the User set of data. A full list and explanation of variables is available in the appendix.

A major problem that was encountered in the User modelling process was missing data. The models attempted to predict whether an email sent to a user would lead the user to purchase from the email, based on their characteristics. However, as has been noted previously, a feature of the dataset is that users that frequently purchase have a lot of information stored about them, whilst those that do not have much interaction with the site there is very little known about. This meant that the models that were created would risk identifying only the features of these missing variables. For example, the location information in the main set would only be present if the user had bought something from the site. A model would identify this relationship, and use its presence to predict whether the email would be a "purchase" or not.

There are three different kinds of missing data: Missing at Random (MAR), Missing Not At Random (MNAR) and Missing Completely At Random (MCAR).[37] MCAR missing sets represent data where there is no pattern to the missing values.[37] MAR sets include data which is missing as a result of a value of another variable.[38] MNAR occur when there is a pattern to missing data, but it is not possible to infer this pattern from other variables in the dataset.[39] The missing data in the User set is an example of MAR, as the presence of some of the variables was dependent on whether the user had ever purchased from the site.

Some of the variables had to be chosen carefully for use in the modelling process, as some variables would directly infer whether a user had ever bought from the site, including from any email in the User data set. For example, if a user signed up for the mailing list from another DC Thomson site, they may have received many emails but never visited the site, meaning that a "Pageviews" count of 0 would mean that they guaranteed to be a non-purchaser in the User data set. Similarly, if the "largest purchase price" variable had a missing value, this also meant that they would not be a "purchaser" as it meant that they had not ever bought anything from the shop. These variables needed to be excluded from the model as they would obscure useful relationships and only identify the pattern of missing data from the set.

The first model used the stratified sample set to identify which variables may have this relationship. It used all the variables available and "purchased" as a target. This produced the following model shown in Figure 21:

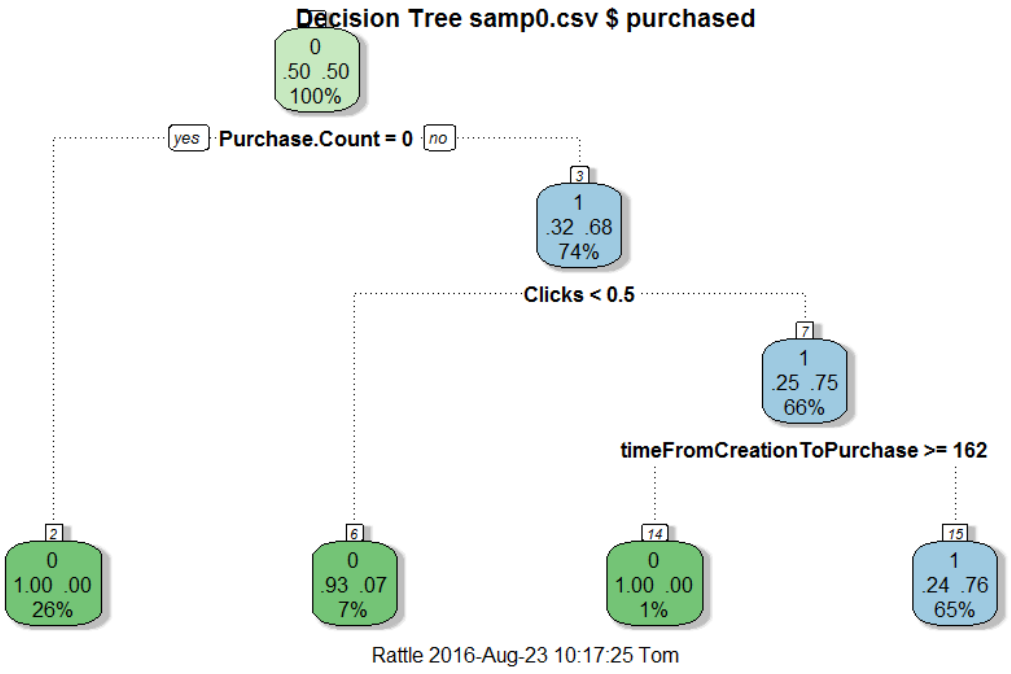


Figure 21 First User model, Stratified Sampling

In this figure, a “0” represents a “non-purchase” and a “1” represents a “purchase”. This first model reveals the variables that should not be used in the modelling process. These variables describe what makes an email “purchased” from, rather than explain why a user might have purchased. After repeating this process the following variables were ignored from future models: “Purchase Count”, “Clicks”, “timeFromCreationToPurchase”, “purchasePeriod”, “timeFromSendToPurchase”, “purchasedDayOfWeek”, “pageviews”, “purchaseCount”, “purchasePrice”, “largestPurchaseItemPrice”, “timeFromCreationToPurchase”. Using the rest of the variables available, the following model in Figure 22 was produced (a larger version is available in the appendix):

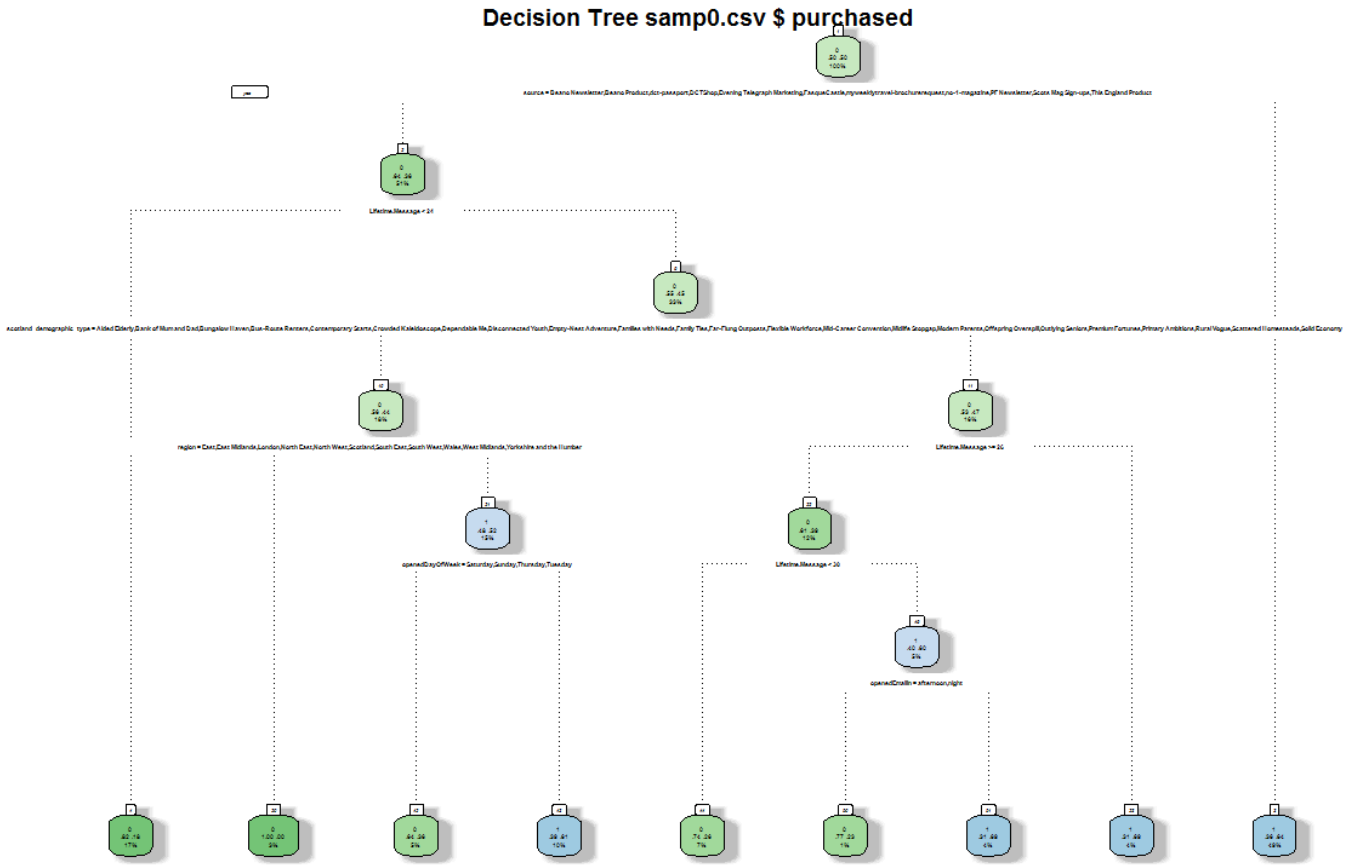


Figure 22 Second User Model, Stratified Sample Set

This model still identified some of the variables that contained missing data as can be seen in the first, third and fourth branches of the tree, with the long conditions for the split at these branches. The condition for the split at the tree at these points are actually referring to whether the variable is present in the dataset. For example, the first split is on “source” and the options are whether the data point is contained in the list of values at that split. The list of values however contains all the possible values for “source”. A data point that does not have a “source” value means that it will not have been collected, meaning that it is an example of a

"non-purchase" by default. It does however reveal some interesting relationships- emails that were not opened on a Saturday, Sunday, Thursday or Tuesday tended to be "purchaser" examples.

The variables used to build this model were then used in the SMOTE sampled set. The extra values generated by the SMOTE set can help to make the model more general and identify more relationships from the data. These variables created the following model in Figure 23:

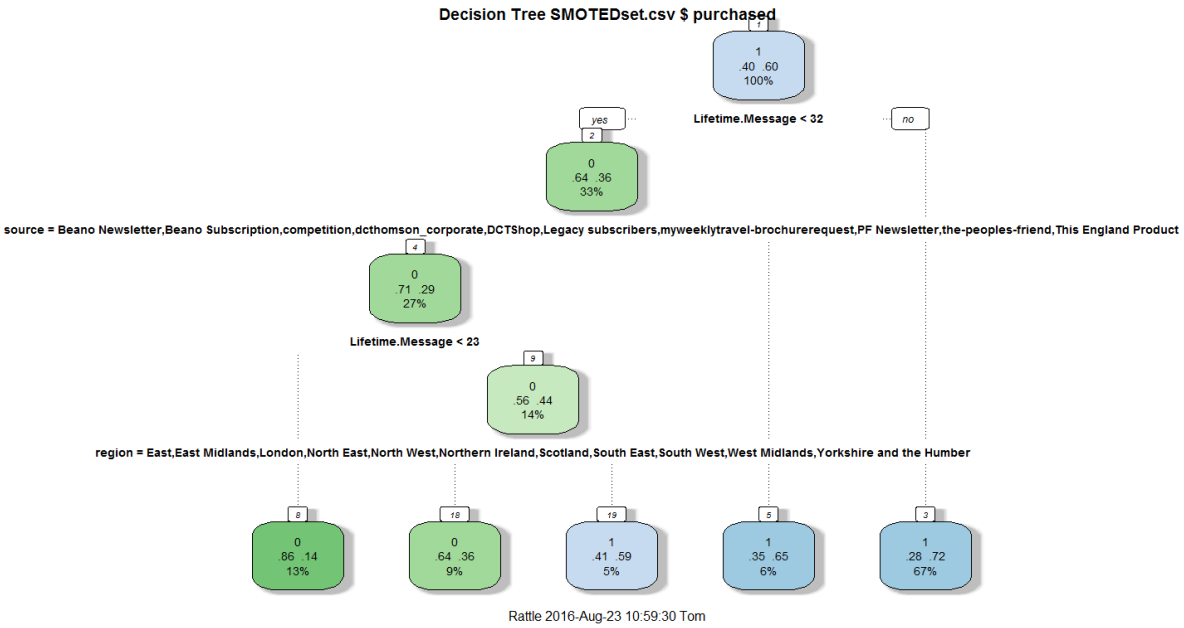


Figure 23 First Model, SMOTE Set

Unfortunately this model also seems to suffer from identifying the patterns of missing data. It does identify that there is a relationship between the number of emails a user has been sent ("lifetime message") and whether they purchased but overall the model is much too general for any other useful relationships to be extracted.

Although no firm models could be created that could identify useful patterns in the data it is possible to use decision tree models to identify which variables are important. Once an understanding of the importance of each variable has been ascertained, data collection efforts can be targeted. The method and results of these efforts are described in section 5.

5 Results

This section will describe the results that were presented to senior management at DC Thomson, along with suggestions that arose from the relationships suggested by the model.

5.1 Campaigns

The Campaign model produced some interesting insight into the dataset. Some of the interesting relationships are identified in an infographic to the company. The first Figure 24 identifies some of the important findings, Figure 25 some more interesting findings and Figure 26 is shows the amount of importance assigned to each variable in the dataset:

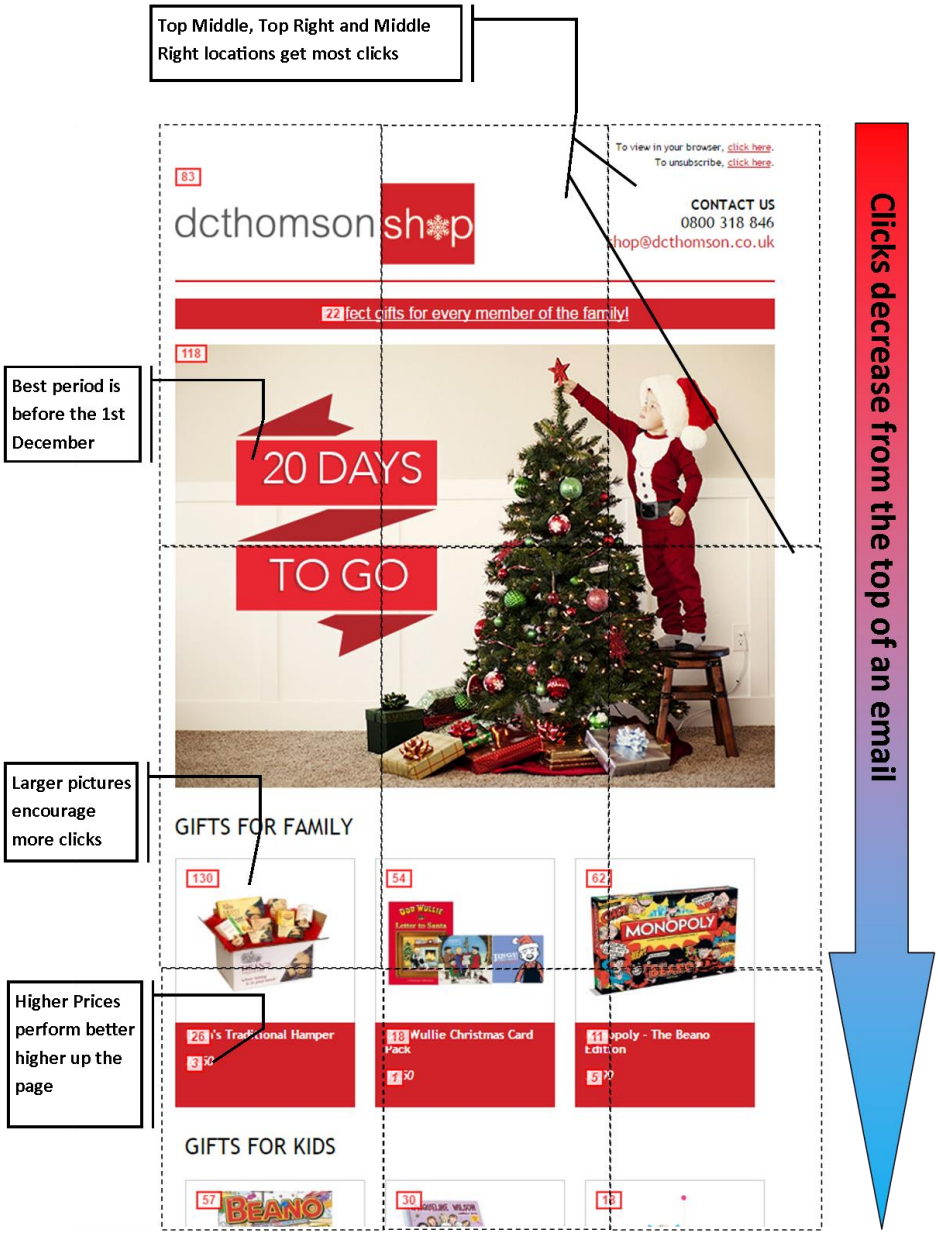


Figure 24 Important factors from Campaign models

1. The model suggests that the clickrate for a link decreases the further it is down the page, as perhaps users do not scroll all the way down the email.
2. Links in the top middle, top right and middle have a better click rate. This may be due to iPad being the most common device used to open emails, with the majority of customers being right handed, and their finger naturally being in this location.
3. The model found links that had higher prices would perform better higher up the page.
4. Larger pictures associated with links were also more likely to be successful.

Figure 25 Important factors from Campaign Models

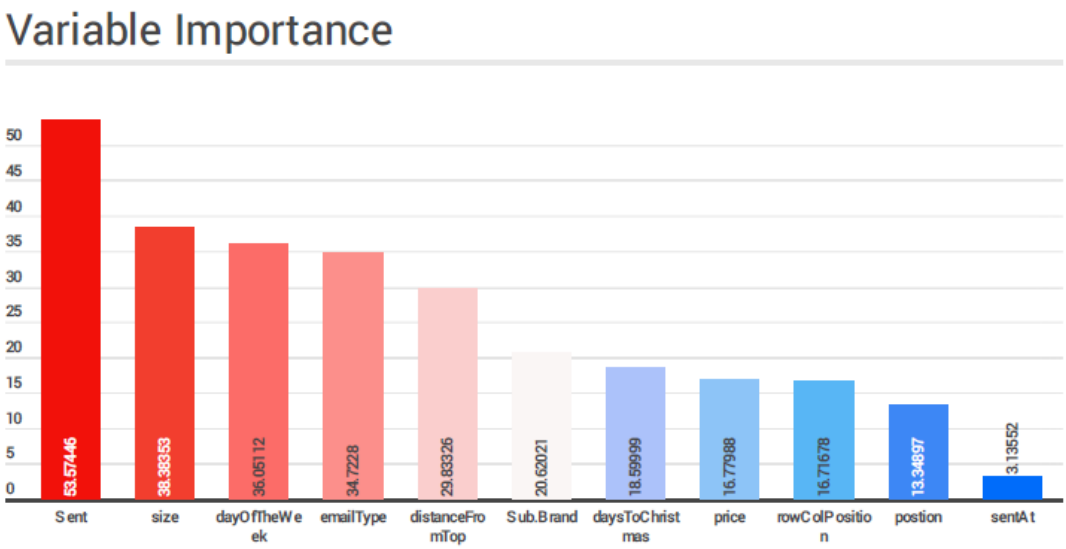


Figure 26 Variable Importance from Campaign Models

The information in these figures comes from common factors identified from the building of models. The models identified several factors that are useful to consider for future models. It highlighted that the top middle, top right and middle right were areas where more clicks than average occurred. Links at the top of the email performed better than areas lower down. Larger pictures had an impact on determining which links had better performance. If a link had a higher price associated with it, it tended to perform better if it was higher up the email than lower down. This structural information can be useful to follow when designing new emails and identifying what factors may be useful to test.

These suggestions were identified by examining the structure of the decision tree. The "root" node at the top of a decision tree shows the branch that reveals the most information.[9] The result of this split will have a large general impact on the classification of a data point. The following branches are more specific and are related to the portion of the dataset that is present at each point. The order of splits and the variables involved is very useful to consider. Since variables higher up the tree reveal more information about a variable, they can have a greater influence on the target class.[40]

The caret package in R has a method "varImp" that was used to generate the variable importance graph shown in the third figure.[40] This function calculates the importance of each

variable in a decision tree.[41] The "varImp" method examines how the decision tree model split the data. It aggregates the Gini importance (as described in section 4.13) that the decision tree model assigned to each variable.[41] This measure shows the importance of each variable from the tree, and can be used to plot the bar chart shown in figure 26.

This identified that the number of users sent the email was the dominant factor in a link's success. The size of a picture associated with an email, the day of the week an email was sent, what category of product or subscription a link referred to and how far from the top of the email a link was positioned also had an important factor. Whether a sub brand was mentioned, the price associated with an email and the position of a link in an email had less influence. The time of day an email was sent had the least impact.

This information is very useful for planning A/B testing. This form of testing involves sending two versions of a piece of marketing material to two groups, an A group and a B group. The version which has most of the group respond is the one that is adopted by the company.[42] The better version can be identified by carrying out a student's T test or a Z test.[43] The factors identified by the variable importance plot can be used to drive what changes are made to create the A/B test.

The graph identifies that the number of people sent the email had the most important impact. This suggests that first thing that should be tested is the difference in response between an email that has been sent to the whole group of users and one that is sent to a targeted set of users. The next stage in testing would be to examine the best size of pictures in emails, comparing smaller ones with larger ones. The testing cycle could iterate through each of these variables in a data driven way that would improve response.

After changes have been made, new models can be built and the importance of variables can be measured again. This allows the testing cycle to be targeted, and the success to be measured. If new structural factors are found that may have an influence on the success of the link, their importance can be identified using the importance plot.

5.2 Users

The missing data from the user models meant that the results that they produced were inconclusive. It was decided that it would be useful to create an audit of the missing data from the set, in order to target future data collection methods.

The VIM package in R can be used to generate a visualisation of the patterns of missing data. This can be seen in Figure 27:

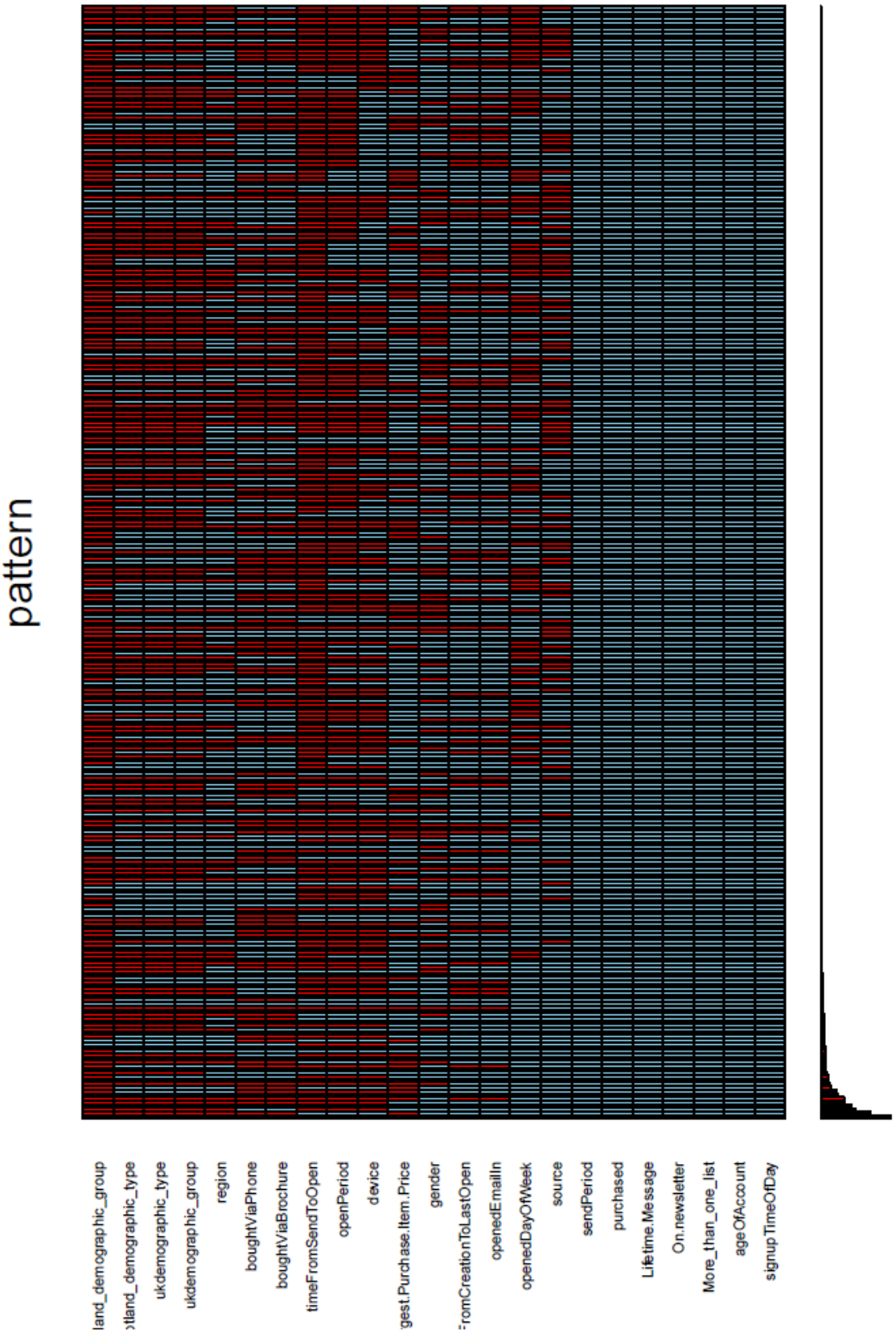


Figure 27 Pattern of Missing Variables

Figure 27 shows the distribution of missing variables in the dataset. Each row refers to a pattern of missing data. Each red square shows where a value in that pattern is missing, and each

blue square identifies where it is present. The bar chart on the right shows the proportion of the dataset that this pattern represents. For example, the most common distribution of missing values has only the demographic groups, region information, the bought via phone and bought by brochure, and largest purchase item price values missing. This distribution of missing data suggests that these are examples of users that opened the email, but have never made a purchase as this has never been captured by Sailthru. It is also interesting to note the columns where every value is blue, which suggests that they are complete with no missing data. This variables, such as "source" and "ageOfAccount" are captured when a user signs up to receive emails. This suggests that the signup process is one of the key methods that the company has to capture and store information about users.

The importance of variables was also measured in a similar way to the campaign set. However, to take the variance that is present in the much larger User set into account, 100 samples of the data were used to generate the plot. Each sample built a decision tree that predicted whether an email would purchase or not, and stored the importance of each variable. Then using the "varImp" method from the caret package, the average importance for each variable across each of the 100 samples was calculated, and used to create the graph shown in Figure 28:

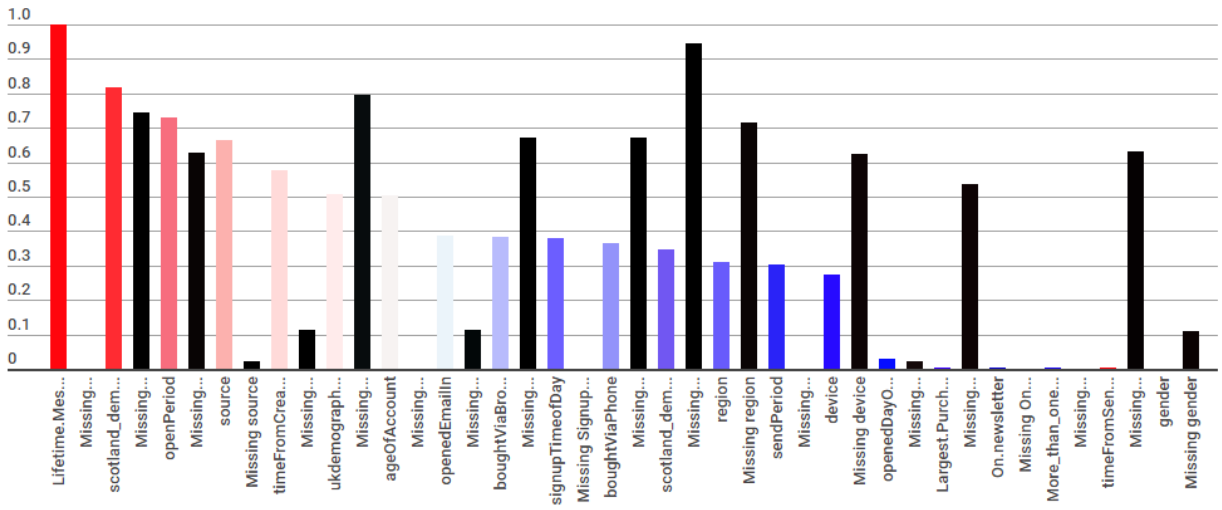


Figure 28 Importance of User variables

Fig 28 shows the (normalised) measure of importance for each variable. The importance of a variable is shown by a coloured bar, then the proportion of the variable that is missing from the data is shown in the following black bar.

This is a useful plot because it shows which variables have the most impact across the set of samples, but it also shows where data collection methods can be targeted. The amount of messages that a user has been sent ("lifetime message") has an important impact, but is not missing at all from the set. "Open period" has an important impact, but also has a high proportion of missing data, but may just suggest that it was important that people opened emails,

but it highlights the necessity to ensure that users remain engaged. The "Scotland demographic group" variable is the second most important variable, but has a very high proportion of missing data. This data was generated using the postcode of the user where it was present and if this was missing it was left blank. Other location based variables such as "region" and the other demographic variables have a similar distribution. This suggests that location has an important influence on purchasing behaviour, but if their postcode is missing, usually as a result of having not purchased from the website and thus not having access to their billing address, then less analysis of the impact of location based variables is available.

Figure 28 should aid DC Thomson in its data collection efforts. It identifies which variables are important from the dataset in producing a "purchase" outcome, and also shows the proportion that they are missing from the dataset. This means that they can plan data collection efforts, such as by organising competitions or surveys that will result in the capture of this information. This work also highlights the importance of ensuring that users are engaged, as this allows more data to be gathered and analysis to be made.

6 Summary and Conclusion

This section will provide a short summary of the project, assess the strengths and weaknesses of the approach and provide some conclusions that have been drawn.

6.1 Summary

The aim of the project was to identify the factors that caused a good response from users included in email campaigns using the data that DC Thomson has about their online site, DC Thomson shop. Splitting the dataset into two sections, as described in chapter 3, makes the best use of the data available to create models that can provide some insight into marketing response.

The Campaign section explored what structural factors of emails have an impact on response. The structural data was scraped from emails in a way that created a wide set of data around how links to the website are presented in emails. This set, when merged with the response data for each link provided by Sailthru, was clean, diverse and large enough to be used in the data mining process. Some different methods for creating a target variable were considered and each can be used to discover different relationships with in the data. These datasets were then modelled using decision trees to predict each of these target variables. The relationships identified by the model were then presented to senior management, which can be incorporated into their plans for the next Christmas season's marketing activities.

The Users section explored whether there were common relationships that can be used to describe whether a user is likely to buy from a campaign email. This dataset was made up of every email that had been sent during the Christmas season of 1st October to 31st December 2015. This set included each user's email address, which meant that it could be enriched by merging everything else that was known about the user by the company. Whether the user had bought from an email or not was used as a target variable for data mining models. This dataset underwent some data cleaning and exploration but when it was subjected to the modelling process, the amount of missing data proved a barrier to achieving any significant conclusions from the User models. The senior management were instead presented an audit of the missing data. This used a range of samples of the dataset to generate variable importance measurements that identified how important variables might be to a model, alongside how much they were missing from the dataset. This audit would allow the organisation to plan their data collection strategy so that models created in the future would allow deeper levels of analysis.

6.2 Critical Assessment of Project

One of the main strengths that emerged from the project was that it created some useful for datasets for the company. For the Campaign section, the data scraping methods can be applied to other emails as they are sent in a similar format. This means that different campaigns can be compared, for example, to see if the important structural relationships are different between the Christmas and Easter campaigns. For the Users section, it enables a framework for further user profiling to be carried out. As the User dataset becomes more enriched, more and more analysis can be derived from it.

The benchmarks created for the Campaign section are also useful. They allow new metrics for measuring the success of links in an email in a way that was not available previously. The models created for the campaign section could be adapted to predict how well a current email may be received, which would influence the way products are laid out on future emails.

The User section provides a good groundwork for further models to be included. As more data is collected its value will grow and enable clustering efforts and general profiles to be generated to describe users. This will benefit the organisation as it will have a deeper understanding of what users are likely to be interested in, and will be able to inform the advertising that they see on the website and the products that they receive in an email.

One of the main weaknesses of the project is that there is not a general interface for users to access the models through. This was not one of the main aims of the project, but it may have been useful for those that may come to improve on the work carried out by the project to have a quick front end to work with. It may therefore limit the access to the models to those with some background knowledge of machine learning and with skills in R and Python.

A weakness of the Campaign section may be that the dataset potentially operates on two levels of granularity. It may have been better to use two datasets, one with variables that describe emails themselves and another that only addresses the links to the website from every email. For example, “days to Christmas”, “email subject”, “total emails sent” could be included in an “email” dataset and “type of link”, “price”, “size” could be included in a “links” dataset. Splitting the dataset up in this way may avoid some of the granularity concerns that may arise using a dataset that includes these together. For example, there may be links in an email that performed poorly because of some structural factor, such as its location but was part of an email that was sent to a lot of people on a day where emails are commonly sent out, which mostly contained examples of links that performed well. If these wider factors had a strong impact on the model, it may cause the model to be commonly mislead.

The approach taken with the User section to predict users as a “purchaser” or “non-purchaser” may have been the wrong approach to take. This approach suggests that there is an equal

chance of an email being a “purchase” or a “non purchase”, and this may not have been the case. The missing data patterns in the dataset in general describe whether an email was a “purchase” or not, and this may have led to few useful relationships being identified by the model. An alternative would have been to predict a level of confidence instead of a binary choice. With “0” being a “non purchase” and “1” representing a “purchase”, the User models could have instead predicted the likelihood of a user purchasing from an email as a number between 0 and 1. For example, 0.6 would represent a user who was slightly likely to purchase.

It may also have been useful to take more of the contextual factors surrounding an email for the Users section. The User section currently looks at when the email was opened, but does not go into detail about what the email included. It would be beneficial for future work to incorporate more of what the email was about in the User data set, as it would identify patterns of what the email was about and whether there were groups of users that would respond well to certain emails, rather than emails in general.

6.3 Recommendations for Future Work

The campaign section could be extended to look at all emails that have been sent by the shop. This would extend the dataset significantly allowing much deeper levels of analysis to be reached and the model would be more robust.

The Campaign section could also be extended to look at different seasonal emails. For example, datasets could be created that modelled “sales” type emails or emails to do with yearly events such as Easter or summer.

The User section will benefit from extra data collection efforts. An increased dataset would allow much more analysis to be drawn as models would be able to learn more effectively. This section could also be extended to automatically generate clusters of users based on their characteristics, which could model their level of engagement with the site or their enthusiasm for clusters of products.

The project overall would benefit from a front end for general users to access and visualise the model. An R Shiny web app for example could enable users to input an email to test to identify how well it was likely to perform and allow them to make changes based upon this.

6.4 Conclusion

This project met the aims well. It provides useful datasets that can be applied to data models that can identify relationships that influence response to e-mail marketing campaigns. These datasets were created, explored and modelled effectively and suggestions from them were given to the organisation.

The Campaign section identifies important structural relationships within emails that cause a good response, and these have been presented effectively.

The User section provides a good groundwork for further models. It will benefit from further data collection efforts, which will be supported by the missing data audit described in section 5.2.

The project shows the value that E-commerce data holds for organisations and that it can be modelled to provide an insight into the success of marketing campaigns.

References

[1] Srinvasa, Raghavan, N.R. *Data Mining in E-commerce: A survey*, in Sadhana: Academy Procedings in Engineering Sciences, (2005: vol.30: 2), pp. 275-289.

[2] Pantea, C. Pop Nicolae, Al. *Email Marketing Campaigns: The easiest path from organisations to consumers – An exploratory assessment*, in The Journal of the Faculty of Economics, (2010: vol.1:1), pp. 737-742.

[3] Sailthru, *Experience Center* available: http://www.sailthru.com/product/#experience-center, accessed: 15/8/16.

[4] Econsultancy.com, Ratcliff, C. *What Is The Single Customer View and Why Do You Need it?* (9/9/14). Available: https://econsultancy.com/blog/65425-what-is-the-single-customer-view-and-why-do-you-need-it/, accessed: 15/8/16.

[5] Rastegari, H. Noor Md. Sap, M. *Data Mining and E-commerce: methods, applications and challenges,* in Jurnal Teknologi Maklumat, (2008:2), pp.116-128.

[6] Mogoşm R,I. Acatrinei, C. *Designing Email Marketing Campaigns: A data mining approach based on consumer preferences*, in Annales Univesitatis Apulensis: Series Oeconomica, (2015:vol.17:1), pp. 15-30.

[7] Ling, C. Li, C. *Data Mining for Direct Marketing: Problems and solutions*, in *International Conference on Knowledge Discovery and Data Mining*, 73-79, 1998. ISSN: 1577350707.

[8] McGarry, K. Martin, A. Addison, D. MacIntyre, J. (2002). *Data Mining and User Profiling for and E-commerce System*, in Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery.

[9] Witten, I. Frank, E. Data Mining: Practical Machine Learning Tools and Techniques (San Francisco: Elsevier, 2005).

[10] Signupto.com, Kent, T. *Email Data: It might not be big but it is clever*, available: https://www.signupto.com/news/geeky-stuff/email-data-it-might-not-be-big-but-it-is-clever/ accessed: 17/8/16.

[11] Optimizely.com, *What is A/B Testing?* Available: https://www.optimizely.com/ab-testing/, accessed: 17/8/16.

[12] Zhang, X. (2007), *Building Personalized Rcommedation System in E-commerce Using Association Rule-based Mining and Classification* in Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, ICMLC 2007, pp. 4113-4118.

[13] Suhail, A. Ron, K. Liew, M. Zijian, Z. (2001), *Integrating E-commerce and Data Mining: Architecture and Challenges* in Proceedings- 2001 IEEE International Conference on Data Mining, ICDM'01, pp. 27-34.

[14] Yu, c. Jannasch-Pennel, A. DiGangi, S. Kim, C. Andrews, S. *A Data Visualization and Data Mining Approach to Response and Non-Response Analysis in Survey Research*, in Practical Assesment, Research and Evaluation (2007: vol.12:19), pp. 1-12.

[15] Breur, T. *How to Evaluate Campaign Response - The Relative Contribution of Data Mining Models and Marketing Execution*, in Journal of Targeting, Measurement and Analysis for Marketing (2007: vol.15) pp. 103-112.

[16] Rosset, S. Neuman, E. Eick, U. Vatnick, N. Idan, I. "Evaluation of Prediction for Marketing Campaigns", in *Procedings of KDD-01,* (AAAI Press: Menlo Park, 2001), pp. 456-461.

[17] Shearer, C. *The CRISP-DM Model: The New Blueprint for Data Mining*, in Journal of Data Warehousing, (2005: vol.5:4), pp. 13-22.

[18] Jensen, K. *Process Diagram showing the relationship between the different phases of CRISP-DM* available: https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining , accessed: 22/08/16.

[19] Trello.com, *Homepage*, available: https://trello.com/, accessed: 22/8/16.

[20] Benington, H. (1987) *Production of Large Computer Programs* in ICSE '87 Proceedings of the 9th international conference on Software Engineering pp. 299-310.

[21] Rstudio.com, *Homepage*, available: https://www.rstudio.com/, accessed: 23/8/16.

[22] Togaware, *Homepage* available: http://rattle.togaware.com/ accessed: 23/8/16.

[23] DataCamp.com, *R vs Python*, available: http://blog.datacamp.com/wp-content/uploads/2015/05/R-vs-Python-216-2.png, accessed: 24/8/16.

[24] Waikato.ac.nz, *Weka*, available: http://www.cs.waikato.ac.nz/ml/weka/, accessed: 24/8/16.

[25] Wiki.pentaho.com. *Handling Large Data Sets with Weka*, available: http://wiki.pentaho.com/display/DATAMINING/Handling+Large+Data+Sets+with+Weka , accessed: 24/8/16.

[26] Bengtson, P. *Fastest Way to Uniqify a List in Python*, available: https://www.peterbe.com/plog/uniqifiers-benchmark, accessed: 5/7/16.

[27] Smartinsights.com. Chaffey, D. *Email Marketing Statistics 2016*, available: http://www.smartinsights.com/email-marketing/email-communications-strategy/statistics-sources-for-email-marketing/, accessed: 15/7/16.

[28] Marketingland.com. Schwartz, B. *A New Click Through Rate Study For Google Organic Results* (1/10/12) available: http://marketingland.com/new-click-rate-study-google-organic-results-102149, accessed: 18/7/16.

[29] Statistics.Laerd.com, *Spearman's Rank-Order Correlation* available: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php, accessed: 18/7/16.

[30] Gizmodo.com. Green-Hunch, A. *What your Email Domain Says About You* (3/9/11) available: http://gizmodo.com/5780416/what-your-email-domain-says-about-you, accessed: 30/6/16.

[31] Cran-rproject.org. Van der Loo, M. *Stringdist: Approximate String Matching and String Distance Functions*, available: https://cran.r-project.org/web/packages/stringdist/index.html accessed: 1/7/16.

[32] RegExLib.com. *Regular Expression to Match UK postcodes*, available: http://regexlib.com/REDetails.aspx?regexp_id=260&AspxAutoDetectCookieSupport=1, accessed: 2/7/16.

[33] Kim. Y, Oh, Y. Park, S. Park, H. *Stratified Sampling Design Based on Data Mining* in Healthcare Informatics Research, (2013: vol.19:3) p. 186-195.

[34] Cs.cmu.edu. Chawla, N. Bowyer, K. Hall, L. Kegelmeyer, W. P. (2002) *SMOTE: Synthetic Minority Over-sampling Technique* "SMOTE" available: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/node6.html#SECTION00042000000000000000, accessed: 30/8/16.

[35] Garcia, V. Sanchez, S. Martin-Felez, R. Mollineda, R.A. *Surrounding neighbourhood-based SMOTE for learning from imbalanced data sets* in Progress in Artificial Intelligence, (2012: vol.1:4), pp. 347-362.

[36] Missingdata.lshtm.ac.uk. available: http://missingdata.lshtm.ac.uk/index.php?option=com_content&view=article&id=78:conclusions&catid=40:missingness-mechanisms&Itemid=96 accessed: 24/8/16.

[37] Missingdata.lshtm.ac.uk. *Missing Completely At Random(MCAR),* available: http://missingdata.lshtm.ac.uk/index.php?option=com_content&view=article&id=75:missing-completely-at-random-mcar&catid=40:missingness-mechanisms&Itemid=96 , accessed: 24/8/16.

[38] Theanalysisfactor.com. Grace-Martin, K. *What is the difference between MAR and MCAR missing data?* Available: http://www.theanalysisfactor.com/mar-and-mcar-missing-data/, accessed: 24/8/16.

[39] Missingdata.lshtm.ac.uk. *Missing Not at Random (MNAR)* available: http://missingdata.lshtm.ac.uk/index.php?option=com_content&view=article&id=75:missing-completely-at-random-mcar&catid=40:missingness-mechanisms&Itemid=96, accessed: 24/8/16.

[40] Topepo.github.io. *Variable Importance*, available: http://topepo.github.io/caret/varimp.html accessed: 30/7/16.

[41] Rbloggers.com. Charpentier, A. (17/6/15), *'Variable Importance Plot' and Variable Selection,* available: https://www.r-bloggers.com/variable-importance-plot-and-variable-selection/, accessed: 30/7/16.

[42] VWO.com *The Complete Guide to A/B Testing* available: https://vwo.com/ab-testing/, accessed: 25/8/16.

[43] Gembaacademy.com. Pereira, R. (20/6/07), *How Beer Influenced Statistics* available: http://blog.gembaacademy.com/2007/06/20/how-beer-influenced-statistics/ accessed: 25/8/16.

[44] Thernau, T. Atkinson, E. *An Introduction to Recursive Partitioning Using the RPART Routines* (29/6/15) available: https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf accessed: 5/9/16.

[45] Teknomo, K. *How To Measure Impurity* available: http://people.revoledu.com/kardi/tutorial/DecisionTree/how-to-measure-impurity.htm accessed: 5/9/16.

[46] Saraswat, K. *A Complete Tutorial on Tree Based Modelling from Scratch (in R & Python)* (12/4/16) available: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/ accessed: 5/9/16.

[47] Abbot, D. *Find Correlated Variables Prior to Modelling* (7/12/04) available: http://abbottanalytics.blogspot.co.uk/2004/12/find-correlated-variables-prior-to.html, accessed: 6/9/16.

[48] Experian, *Mosaic in Detail*, available: http://www.experian.co.uk/marketing-services/products/mosaic/mosaic-in-detail.html, accessed: 5/9/16.

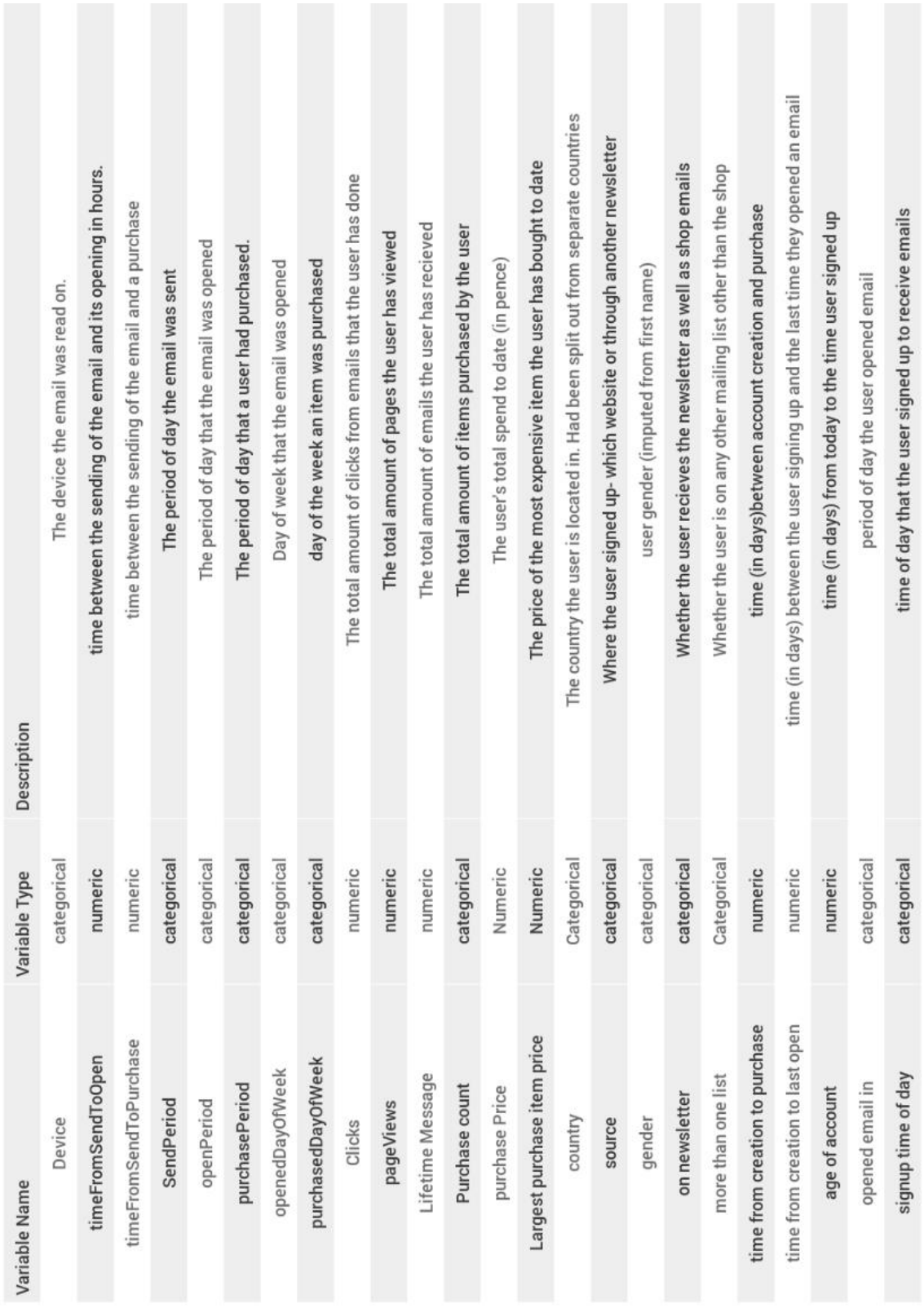
Appendix 1 – Campaign Variables

Variable Name	Variable Type	Description
ID	Categorical	A unique identifier thatSailthru uses to identify email campaigns.
Link URL	Categorical	Each row represents a link in an email
Type	Categorical	What category the email falls into based on the contents of the link.
Position	Categorical	Describes the position of the link in the email. Read from left to right the list of links is split into three and then placed in the appropriate category
Size	Numeric	Describes the width of a picture associated with a link in pixels. Not all pictures have a height element but wider pictures are bigger than thinner pictures so it works.
Distance From Top	Numeric	Describes the percentage of how far from the top the link is situated. All links in an email are kept in a list and this value is based on how far through this list the link is.
Days To Christmas	Numeric	The number of days between Christmas and the email being sent
Sent At	Categorical	What period of day the email was sent at
Day of Week	Categorical	What day of the week the email was sent
Email Type	Categorical	A clearer example of the email type taken from the link context.
Sub brand	Categorical	Potentially may perhaps be better to include a "sub brand included" of yes or no column instead as this may have a better distribution
Sent	Numeric	The number of email addresses the email was sent to
Price	Numeric	The price that is associated with a link.
Total Clicks	Numeric	The total clicks a link received (including multiple clicks from the same user)
Purchases	Numeric	The amount of purchases resulting from clicking on a link
Revenue	Numeric	The total revenue associated with users clicking on a link
Click Rate	Numeric	The number of unique clicks of a link (excluding clicks from the same user)
Link.Click.Rate	Numeric	The click rate divided by the number of emails sent
Click Through Rate Benchmark	Categorical	The overall click rate performance; compared to an industry benchmark As a predictor. Should really be superceded by Link Click Through Rate Benchmark
rowColPosition	Categorical	The location of an email that a link is found in. Calculated by splitting the email by <td> .* </td> elements; and splitting this list into thirds (top middle; bottom). Then; each of these is split by the <td> .*</td> elements within - (left ;middle; right.)
Above.Below.Link.Average	Categorical	uses the row col position and click rate values. for each link; the position is identified (eg Top Middle). Then; the mean of the click through rate divided by the sent amount for each link in the same position is taken. Each link's click rate is then compared to the average click rate for a link in this position;and measured as above or below.

Figure 29 Campaign Variable Glossary

Appendix 2 – User Variables

All of the variables created for the project. Not all used in the models.



Variable Name	Variable Type	Description
Device	categorical	The device the email was read on.
timeFromSendToOpen	numeric	time between the sending of the email and its opening in hours.
timeFromSendToPurchase	numeric	time between the sending of the email and a purchase
SendPeriod	categorical	The period of day the email was sent
openPeriod	categorical	The period of day that the email was opened
purchasePeriod	categorical	The period of day that a user had purchased.
openedDayOfWeek	categorical	Day of week that the email was opened
purchasedDayOfWeek	categorical	day of the week an item was purchased
Clicks	numeric	The total amount of clicks from emails that the user has done
pageViews	numeric	The total amount of pages the user has viewed
Lifetime Message	numeric	The total amount of emails the user has recieved
Purchase count	categorical	The total amount of items purchased by the user
purchase Price	Numeric	The user's total spend to date (in pence)
Largest purchase item price	Numeric	The price of the most expensive item the user has bought to date
country	Categorical	The country the user is located in. Had been split out from separate countries
source	categorical	Where the user signed up- which website or through another newsletter
gender	categorical	user gender (imputed from first name)
on newsletter	categorical	Whether the user recieves the newsletter as well as shop emails
more than one list	Categorical	Whether the user is on any other mailing list other than the shop
time from creation to purchase	numeric	time (in days)between account creation and purchase
time from creation to last open	numeric	time (in days) between the user signing up and the last time they opened an email
age of account	numeric	time (in days) from today to the time user signed up
opened email in	categorical	period of day the user opened email
signup time of day	categorical	time of day that the user signed up to receive emails
region	categorical	region of the UK (or overseas label) of where the user is located

Figure 30 User variable glossary

scotland demographic group	categorical	Experian scottish demographic group
scotland demographic type	categorical	Experian scottish demographic type
uk demographic type	categorical	Experian UK demographic type
uk demographic group	categorical	Experian uk demographic group
bought via phone	categorical	Whether the user bought via phone during the christmas period of 2015
bought via brochure	categorical	Whether the user bought via brochure during the Christmas period of 2015
purchased	categorical	Whether a customer purchased as a result of receiving the email or not

Figure 31 User Variable Glossary Continued

Appendix 3 – Larger Versions of Diagrams

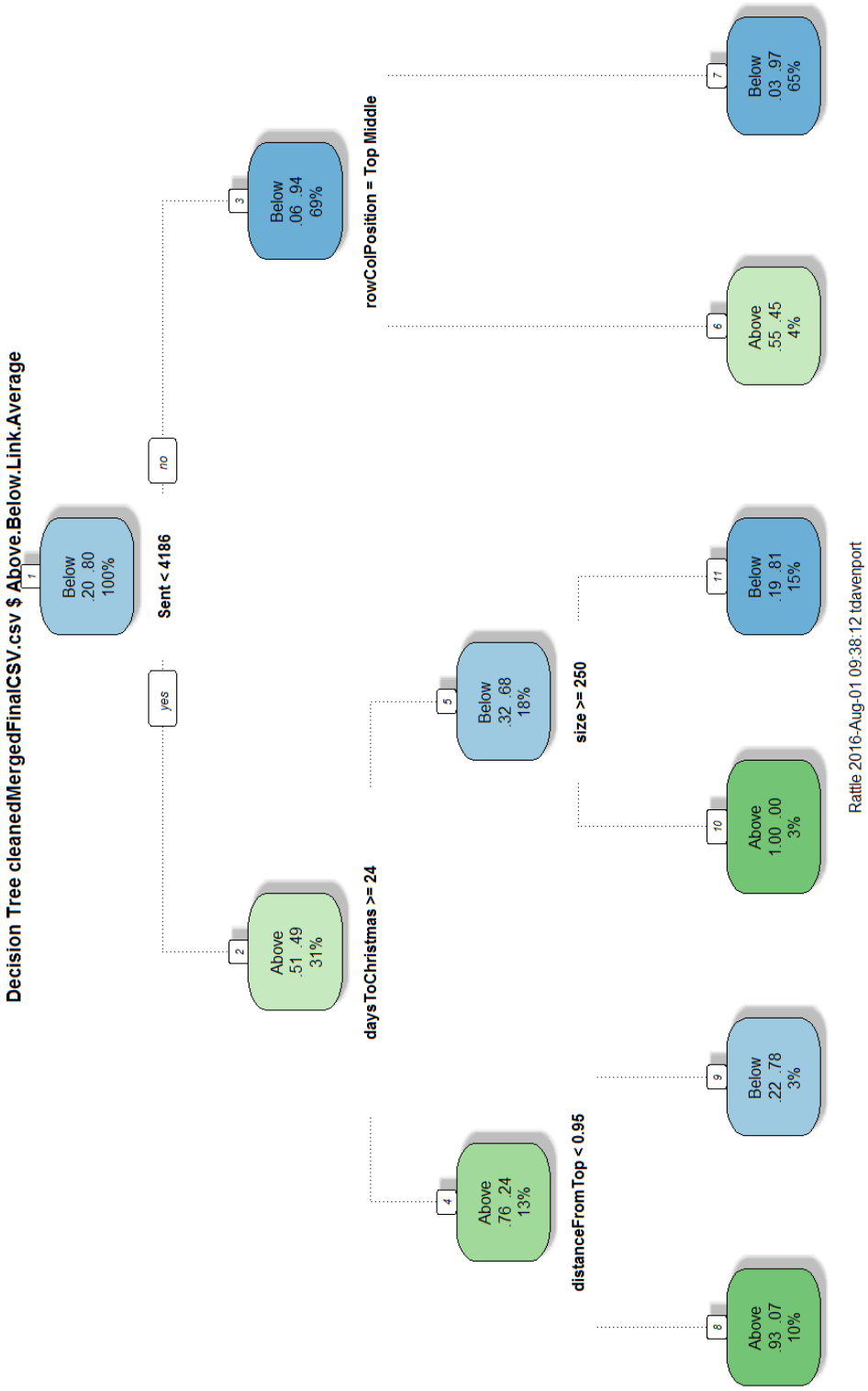


Figure 32 Larger Version figure 13

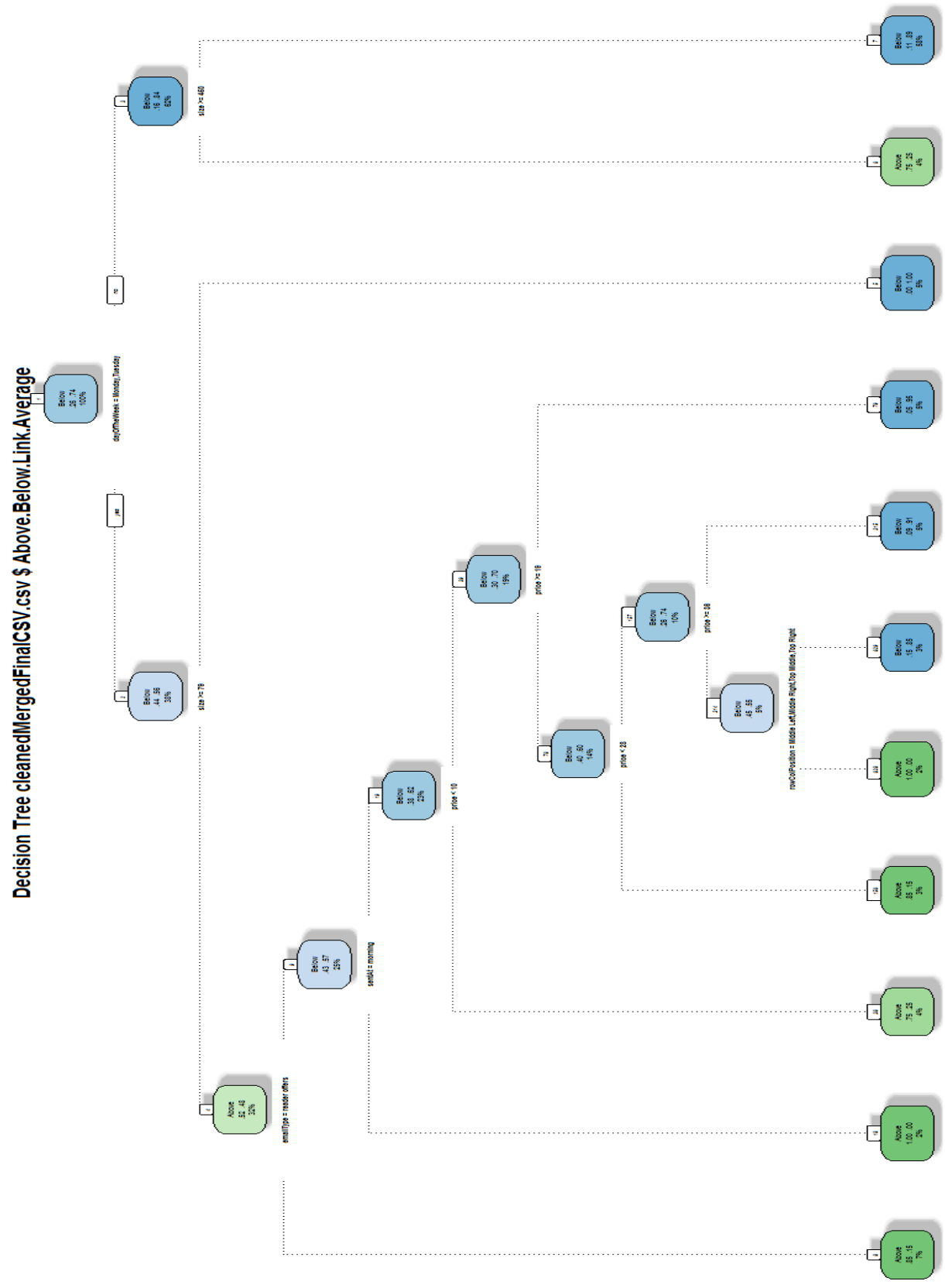


Figure 33 Larger Version Figure 14

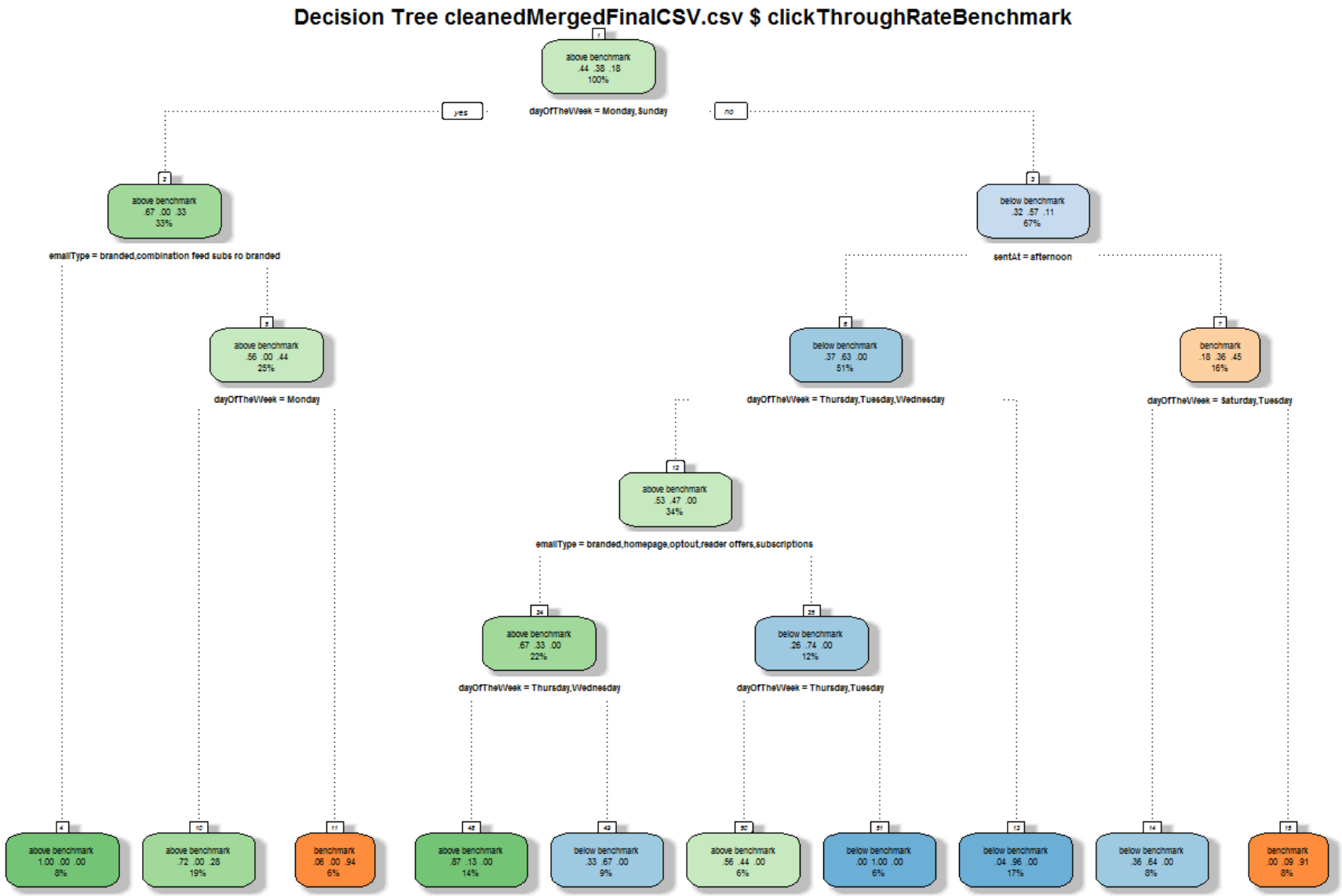


Figure 34 Larger Version Figure 16

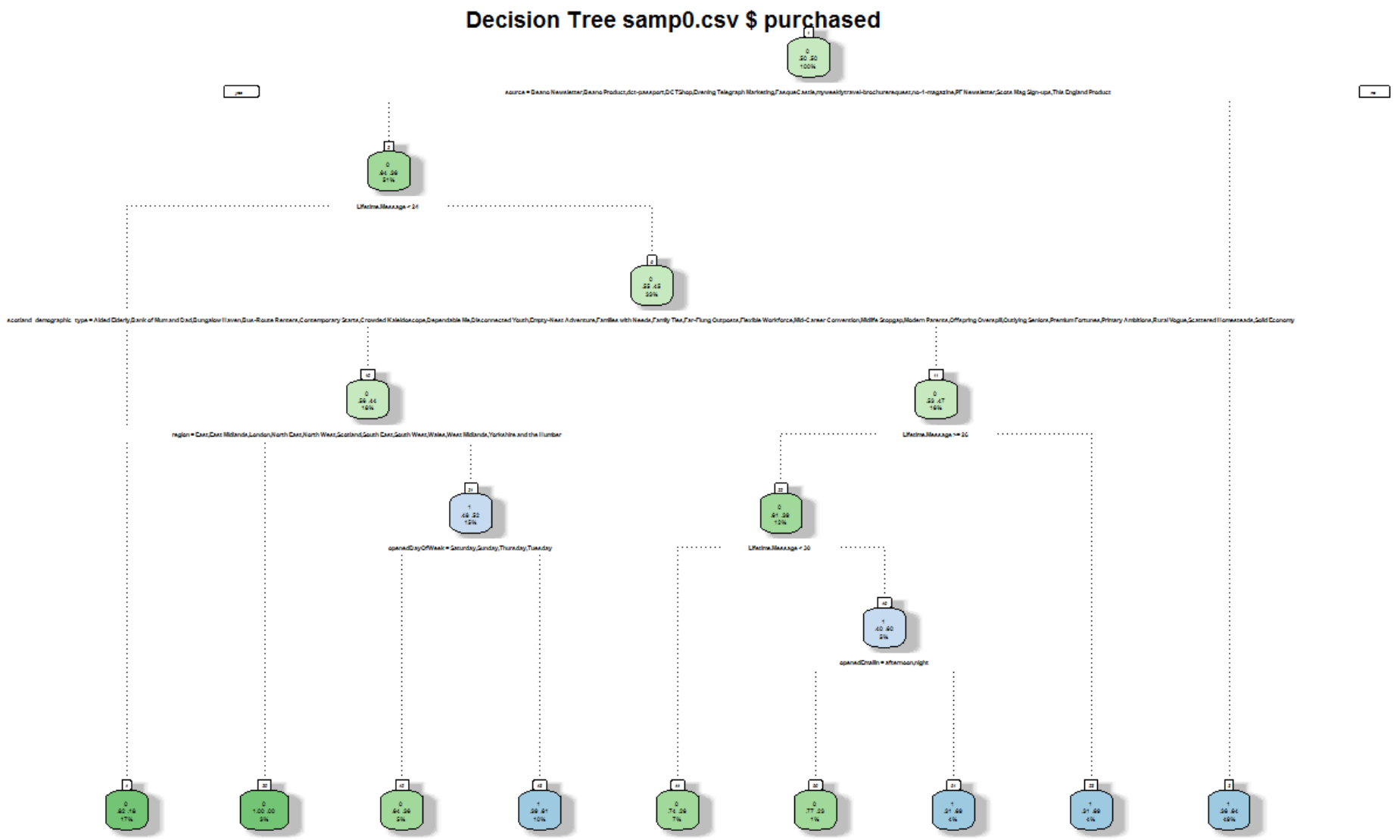


Figure 35 Larger Version of Figure 22