Abstract

Daft.ie markets house listings in Ireland. This paper outlines the use of Web Scraping & Natural Language Processing (NLP) alongside corollary analyses to deliver insights into data collected from Daft.ie as well as the composition and use of words in each advertisement/house listing.

Introduction

> *"Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications."* (Liddy, 2001)

The purpose of Natural Language Processing (NLP) is to allow the computer to read and make sense of human language for the purpose of analysis. We wished to use NLP to see if the language used on Daft.ie house listings descriptions could be cross-referenced with higher prices. To do this we used sentiment analysis, the process of which will be described in the report below. The data we sourced was web scraped from Daft.ie. *"Web scraping, also known as web extraction or harvesting, is a technique to extract data from the World Wide Web (WWW) and save it to a filesystem or database for later retrieval or analysis"* (Zhao, 2017).

We will describe the process involved to retrieve this data in detail, as well as a brief description of the dataset generated by this web scraping. We will then perform several analyses on the dataset to get a better understanding of the data, which will be described below.

Web Scraping

For web scraping, we chose daft.ie – a housing website for the rental, business, and personal home buyer. On this website, we chose to scrape 20+ pages to gather the information that we needed to carry out our analysis. To effectively scrape Daft.ie for this information – we looked through Daft.ie to find the required information such as:

- House size
- Bedroom / bathrooms
- Energy rating
- Listing description
- Who is the seller?
- Views count since being listed.

Daft.ie is structured such that property advertisement on the main page holds some of the information that we need however, if a user clicks on a property there is vastly more information given to the user. Instead of altering R code to scrape listings 1 by 1, the script was built with a number of functions. What these functions do is; Retrieves the property URL, goes inside that chosen URL, finds the chosen elements on that page, concatenates it and finally

converts it to text to allow writing to a data frame. For example, these functions are built as below:

House Size element:

```
house_size <- function(ex){ # house size sqM
  ex <- read_html(ex)
  size <- ex %>% html_nodes('.Zimjn~ .Zimjn+ .Zimjn') %>% html_text() %>% paste(collapse = '')
  return(size)
}
```

Bedrooms and bathrooms:

```
bedrooms <- function(b){ #Bedrooms number
  b <- read_html(b)
  bed <- b %>% html_nodes('.Zimjn:nth-child(1)') %>% html_text() %>% paste(collapse = '')
  return(bed)
}


bath <- function(ba){ # Bathrooms number
  ba <- read_html(ba)
  baths <- ba %>% html_nodes('.Zimjn:nth-child(2)') %>% html_text() %>% paste(collapse = '')
  return(baths)
}
```

There are 7 functions in total similar to the above, retrieving elements from each individual listing. The next step was to simply create an empty data frame. The purpose of this is so that when the web scraping script is running, the data will be constantly written to the data frame. If the data frame was placed inside of the web scraping script, each time it is written to – the data would be constantly overwritten. Therefore, the data frame will remain outside of the web scraping loop as illustrated below:

```
# Empty dataframe to write too
emp_data <- data.frame()
```

Since multiple pages needed to be scraped to get the required data, a loop was required to re-run the script.
To achieve this, we revisited daft.ie and scrolled through each page – the URL of course changed for each page however, each page is in increments of 20, meaning page 0 or page 1 is noted as 00, page 2 as 20 and so on. So as to achieve scraping over these pages Rstudios **seq()** (sequence) function was utilised.

To first start the script, we filtered daft.ie to find all types of houses up to the amount of 300k that are located in Dublin, Cork, Meath, Wicklow, and Wexford with one attribute in common; that they are all seaside counties. Once filtered, the URL was then ready to be used inside the script. To begin, the loop was first created in a sequence to count from 0 to 460 in increments of 20. After this was created, anything inside this loop was read over 24 to 25 pages. This is illustrated as overleaf:

```r
# Placing the web scraper into a loop which will scrape multiple pages and place the scrapped data into
# the above mentioned empty dataframe
for(i in seq(0,460,20)) {
web <- paste0('https://www.daft.ie/property-for-sale/ireland?salePrice_to=300000&location=dublin-city&location=meath&location=wicklow&location=
html <- read_html(web)

list_title <- html %>% html_nodes('.knPImU') %>% html_text() # Titles
list_title

price <- html %>% html_nodes('.pJtsY .gDBFnc') %>% html_text() # Listing price
price


house_type <- html %>% html_nodes('.bcaKbv') %>% html_text()
house_type

listing_url <- html %>% html_nodes('.itNYNv a') %>% html_attr('href') %>% paste('http://www.daft.ie', ., sep='') # Fidning the URL of each list
listing_size <- sapply(listing_url, house_size) # using the listing URL and the function house_size to scrape the size info

beds <- sapply(listing_url, bedrooms) # using the listing URL and the function bedrooms to scrape the size info
baths = sapply(listing_url, bath)
sellers = sapply(listing_url, seller)
views = sapply(listing_url, view)
views

energy_rating <- sapply(listing_url, energy) # using the listing URL and the function energy to scrape the size info

description <- sapply(listing_url, list_desc)# using the listing URL and the function date to scrape the size info
```

Within the script we can see what is being scraped as required in order to carry out our analysis, each vector or variable named to enhance readability. Other code was added to the loop to concurrently clean the data as not all of the data was read cleanly, for example, the description of a property wasbeing scraped along with HTML code such as /r and /n. The code was easily cleaned by simply calling the gsub() function inside of Rstudio as below:

```r
## Cleaning the data
# removing unwanted characters
description = gsub('[\r\n]', '', description)
energy_rating = gsub('[kWh/m2/yr]', '', energy_rating)
beds = gsub('[Bed]', '', beds)
baths = gsub('[Bath]', '', baths)
price = gsub('[AMV € : ,]', '', price)
listing_size = gsub('[m²]', '', listing_size)
```

At the end of the loop, each of the vectors or variables previously declared were written directly to the aforementioned data frame as below:

```r
#  writing of all the above vectors / veriables to a the empty dataframe
emp_data <- rbind(emp_data, data.frame(sellers[1:18], views[1:18], list_title[1:18], price[1:18],
                                       baths[1:18], beds[1:18],
                                       house_type[1:18], energy_rating[1:18], listing_size[1:18],
                                       description[1:18]))
```

To effectively know that each page was being scraped an indicator was required which would print each page that the script was on and scraping – this was easily solved by using Rstudio's print() function to print the web vector / variable at the end of the code. This showed the URL and the page to which the script is on, an example of which can be seen below:

```
[1] "https://www.daft.ie/property-for-sale/ireland?salePrice_to=300000&location=dublin-city&location=meath&location
=wicklow&location=wexford&location=cork&location=galway&pageSize=20&from=0"
[1] "https://www.daft.ie/property-for-sale/ireland?salePrice_to=300000&location=dublin-city&location=meath&location
=wicklow&location=wexford&location=cork&location=galway&pageSize=20&from=20"
[1] "https://www.daft.ie/property-for-sale/ireland?salePrice_to=300000&location=dublin-city&location=meath&location
=wicklow&location=wexford&location=cork&location=galway&pageSize=20&from=40"


> |
```

Notice at the end of each URL, the page number is changing by increments of 20.

Lastly, an issue arose with daft.ie in which the server / website would kick us off for loading too many URLs to quickly and reading / scraping that data. This is understandable given the script is a type of bot that crawls each web page and retrieves information from it. To remove this issue – the simple function inside of Rstudio named: Sys.sleep() effectively pauses our loop and waits an allotted amount of time to start running again. In our case 0.9 of pause time removed the error of being kicked from the daft.ie server. The code for the pausing of the script and the printing of the URL can be seen below:

```
# Printing the web page so we know what page we are scrapping
print(web)
# Putting the script to sleep so the website wont kick us off
Sys.sleep(0.9)

}
```

Decision Tree

A decision tree is a machine learning method that applies a strategy of dividing data into smaller and smaller portions to identify patterns which enable decision support. It uses a tree-like model that can be easily understood without statistical knowledge due to its visualisation and explores the possible consequences of each decision based on statistical analysis.

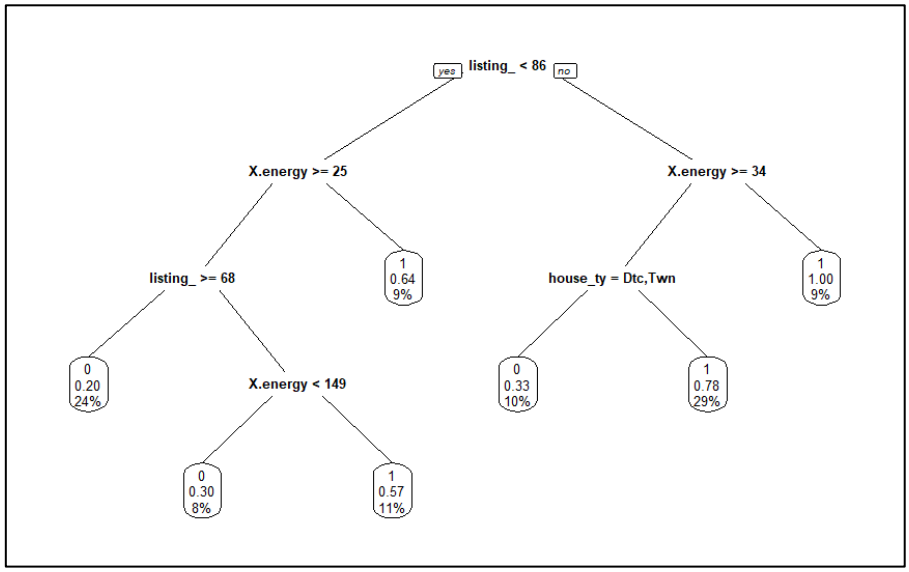
"The decision tree makes explicit all possible alternatives and traces each alternative to its conclusion in a single view, to make easy comparison among the various alternatives" (Patel & Prajapati, 2018).

The purpose of the decision tree below is to explore whether houses are above or below the median house price of the dataset. The dataset created by the above web-scraping contained eight character variables: seller, which describes the listing agency, property.views, which is the number of views on the daft website received by the property, listing_title, which is the name of the property, bathrooms, bedrooms, house_type, which is the type of property e.g. apartment, semi-detached, listing_size.m2., which is the size of the listing in metres squared, and description, which is a description of the property. There is also two numerical variables: price, which is the asking price of the listing, and X.energy_rating.Kwh, which is the energy rating in kilowatts per hour.

Due to bedroom, bathroom and listing size being character variables, despite actually being numeric, we coerced these into numeric variables. We then performed a Shapiro-Wilk Test to determine if the data was normal using a confidence interval of $\alpha = 0.05$. Returning a p-value = 1.643e-13, we cannot reject the null hypothesis, and thus can determine that the data is not normal.

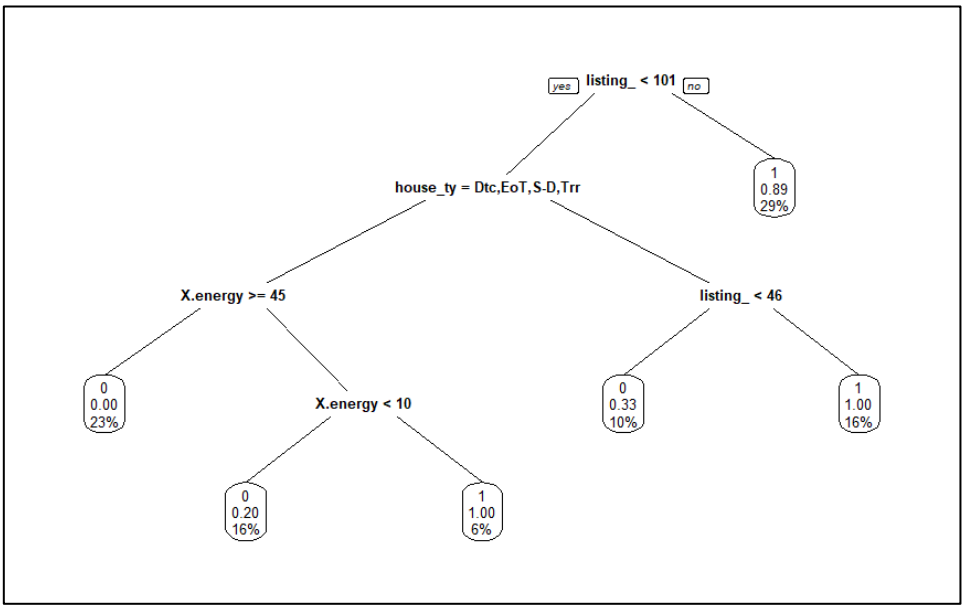
Due to the data being non-normal, we have chosen to use the median rather than the mean of the sample. To do so, we first replaced each house with a "1" indicating it was at or above the median or a "0" indicating it was below the median, which was coerced as a factor. We also coerced the property type (house_type) variable as a factor also, so that the decision tree could take this into account. The decision tree does not make use of character variables, so we removed the descriptions and addresses, as well as the seller.

We also believed that the house price is more likely to have an effect on the number of views (property.views) variable rather than vice versa, so this was removed also leaving us with energy rating, listing size, house type, number of bedrooms, number of bathrooms and the target variable, which was the price.



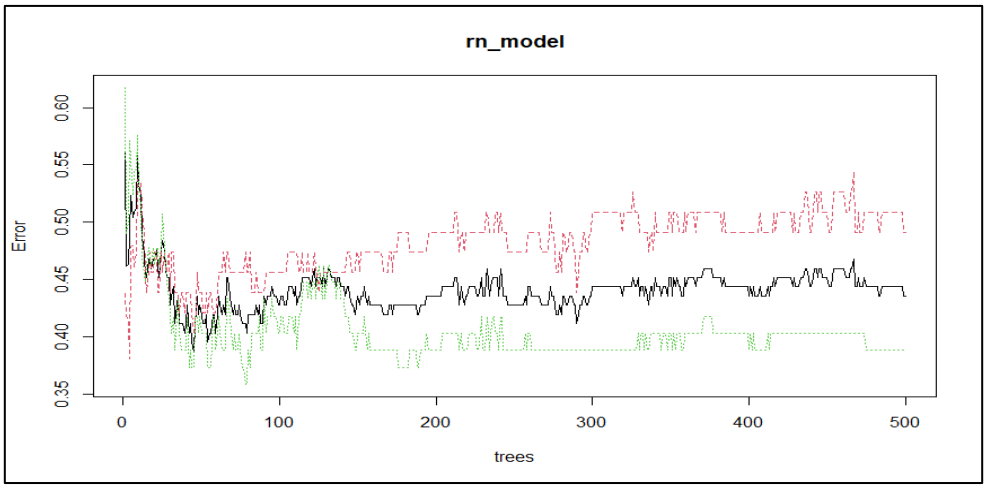
We split the dataset into a train and test set at an 80:20 split. The above decision tree is as a result of the train dataset. As we can see, the biggest influences on the pricing is the listing size and energy rating, with housing type playing a smaller role. The confusion matrix shows an accuracy of 75%. The Mcnemar's Test has a null hypothesis that there is a difference between false positives and false negatives of a confusion matrix.

The Mcnemar's Test P-Value returned p-value = 0.4725, which is not statistically significant using a 95% confidence interval, and thus we cannot reject the null hypothesis. The P-Value for testing the null hypothesis that the accuracy of the model is greater than the no information rate ([Acc > NIR]) is statistically insignificant also, with a p-value = 1.199e-06, meaning that there is likely a large difference in the classes, leading to a lack of information.



As we can see in the test set, the important variables are largely the same, with property type now playing a bigger role, while listing size and energy rating continue their important. This time the confusion matrix showed 90.32% accuracy, with the McNemar test returning 1, which is statistically significant, meaning that there is no difference between the rate of true positives vs true negatives. However, the P-Value for [Acc > NIR] still showed to be statistically insignificant, with p-value = 2.328e-05 due to a 50% no information rate.

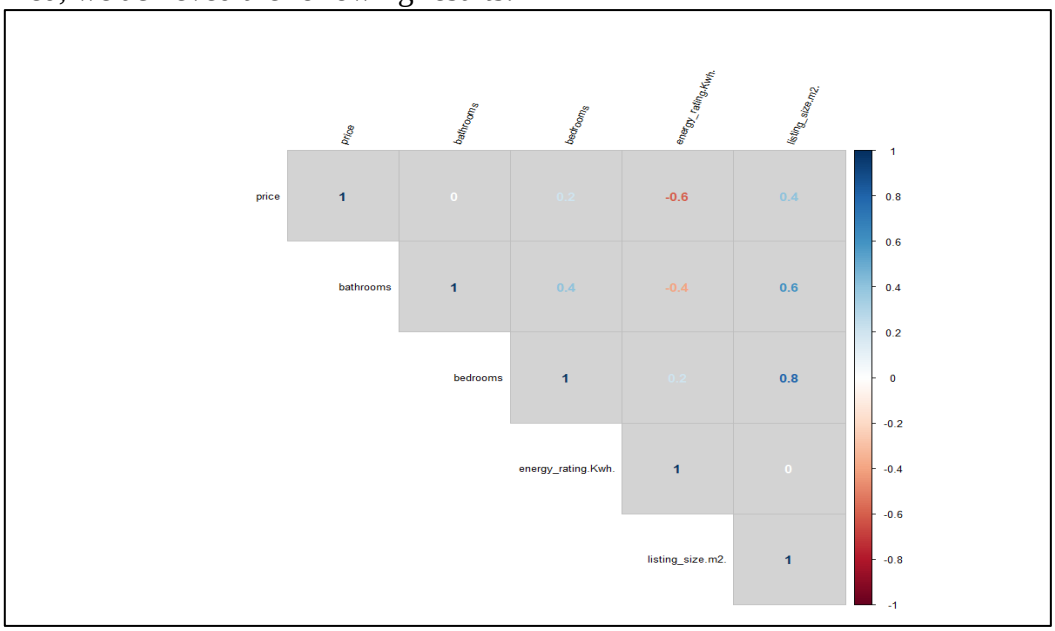
Due to this, we decided to use a random forest to test if this would improve the model. A random forest model takes the decisions of multiple decision tree, and the decision tree with the most "votes" from the forest becomes the prediction model. The random forest model relies on the wisdom of a committee of decision trees. By doing so, this weeds out the number of individual errors made by the trees. The random forest showed a decreased accuracy to 64.52%, but the P-Value for [Acc > NIR] had increased dramatically to p-value = 0.1839. However, this appears to be due to a decrease in accuracy rather than due to a decrease in the No Information Rate.



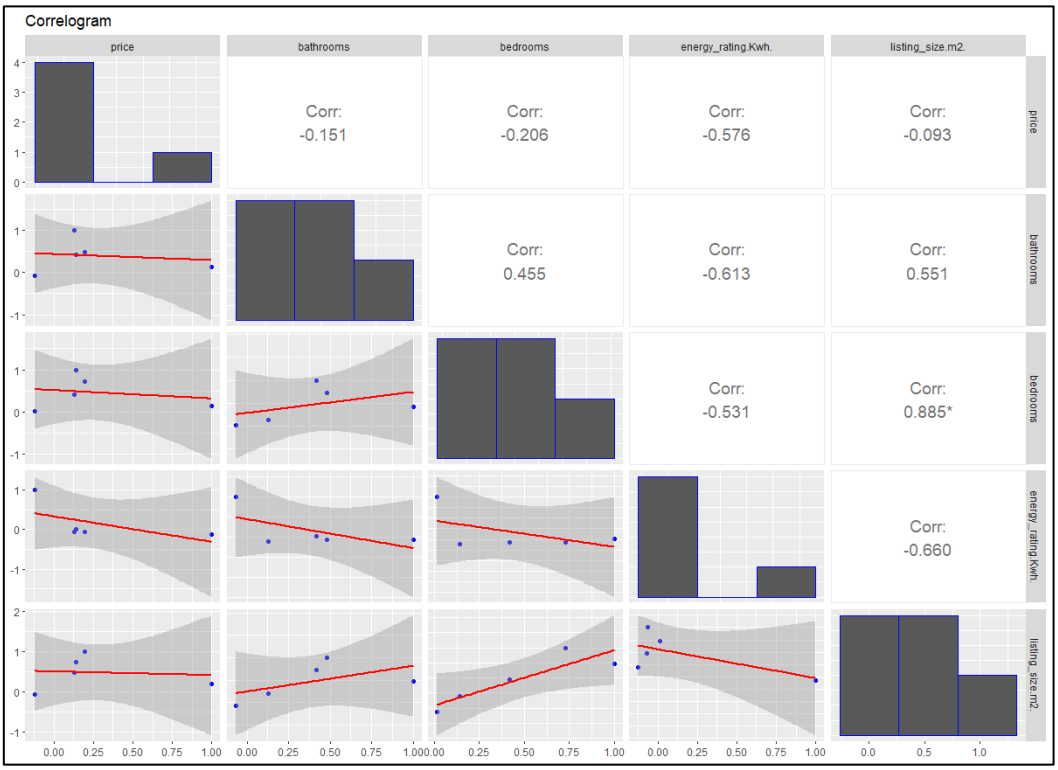
The No Information Rate (NIR) is our classifier to determine whether the decision tree is statistically significant, and in this case, it appears that it is not. This is likely due to a lack of information available between the classes. As we can see from the above, when the decision tree model is applied to the test data, this was still accurate, but more data for a variety of different classes would be required in order to ensure that it is trustworthy. Some of the house types appear a lot more frequently than others, and thus may not be split equally across the test and train sets which could lead to the high NIR.

Correlation

Different numerical variables were measured against each other so as to determine the correlation, that is, the strength of the relationship between them. The Kendall method of determining the correlation co-efficient was used as the data was non normal. When this was performed, we achieved the following results:



As can be seen there is a strong relationship between price and energy rating and between bedrooms and bathrooms. Developing on this, and producing a line of best fit to map these variables, we can see this illustrated as below:



NLP

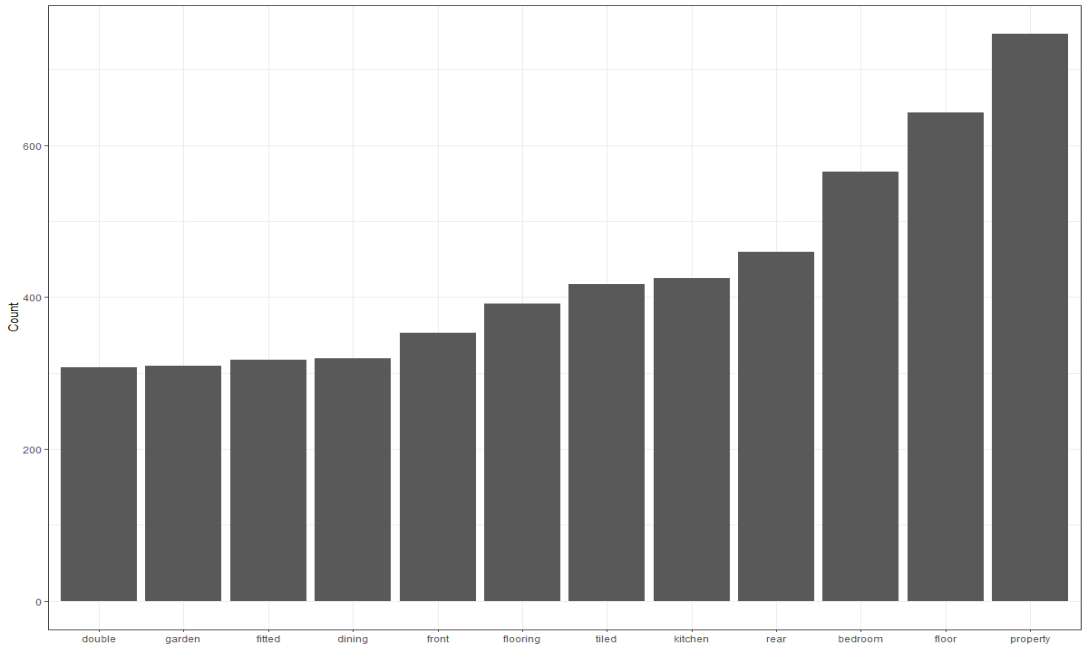
"Natural Language Processing is the analysis of linguistic data, most commonly in the form of textual data such as documents or publications, using computational methods"(Verspoor et al., 2013)

Given there were a low number of houses in the Duplex, Townhouse and Bungalow House Type Categories, these were removed from our NLP analysis. The reason for this was due to the fact that a disproportionate number of these house types could be in our Training dataset compared to our Test dataset. As such, the accuracy of our Naive Bayes model (which we developed as below) would be diminished if such house types were not removed from analysis.

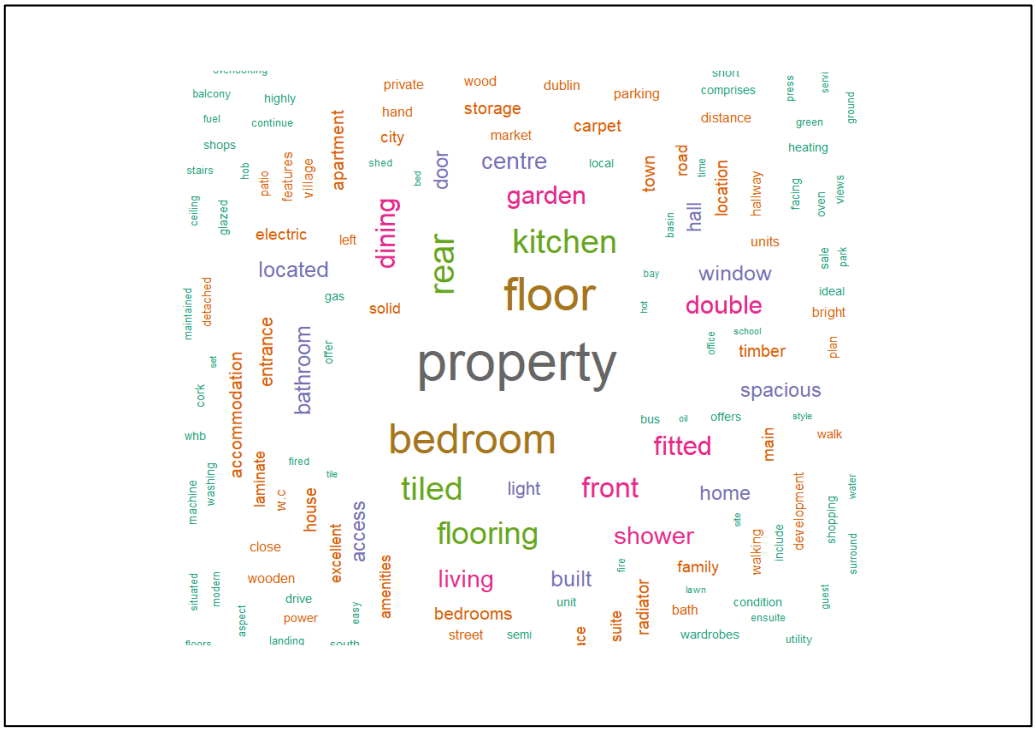
Our Naïve Bayes Model is based on the Bayes' theorem, which underpins this machine learning algorithm. Random Sampling was used so as to change around the order of our data. This was imperative so as to ensure the training and test datasets were as homogenous as possible. In creating a corpus, stop words, numbers, punctuation, and whitespaces were removed. The corpus was then changed to a document term matrix in order to develop a Naive Bayes model. When our model was called, the following results were achieved:

	y				
x	Apartment	Detached	End of Terrace	Semi-D	Terrace
Apartment	9	3	0	11	4
Detached	6	0	2	1	5
End of Terrace	0	1	0	0	0
Semi-D	4	1	1	5	7
Terrace	2	1	1	3	1

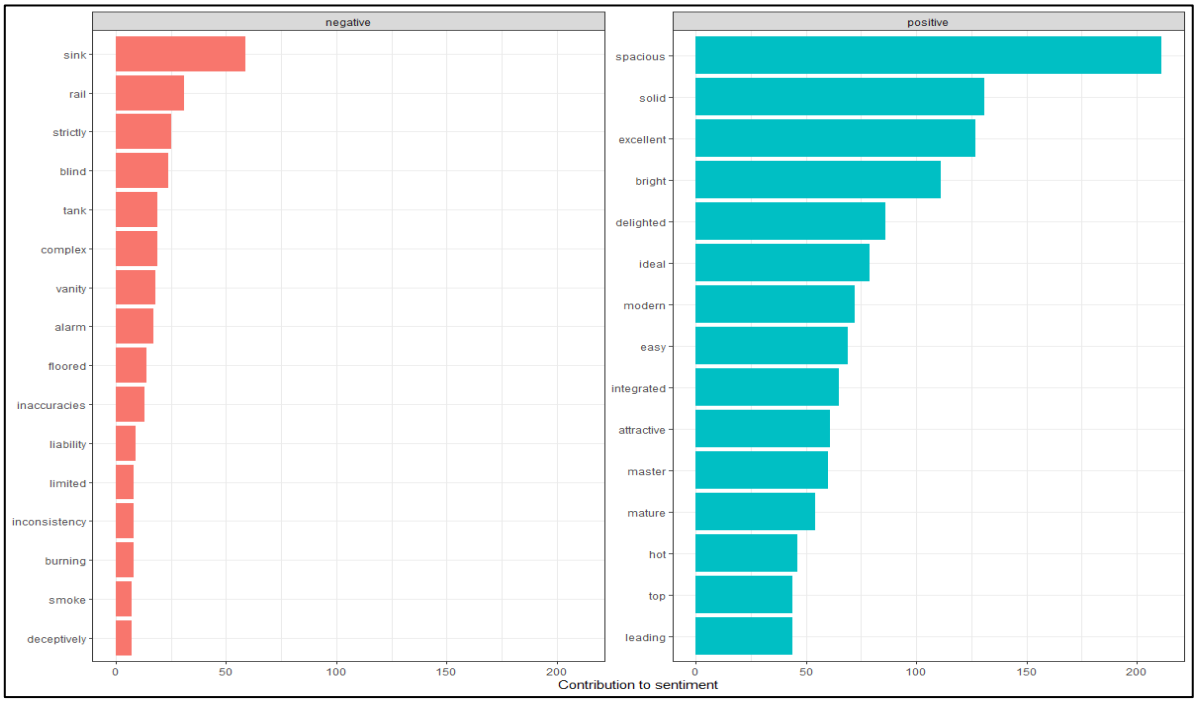
As can be seen, the model was very inaccurate with just 15 observations of 68 being accurate, reflecting an accuracy rate of 22%. In essence, only 22% of the words were correctly allocated to their respective housing type. When we analyse the words of each text, we find the following 12 words are most common:



A word cloud was developed to highlight a greater number of words according to their prevalence, as illustrated below:

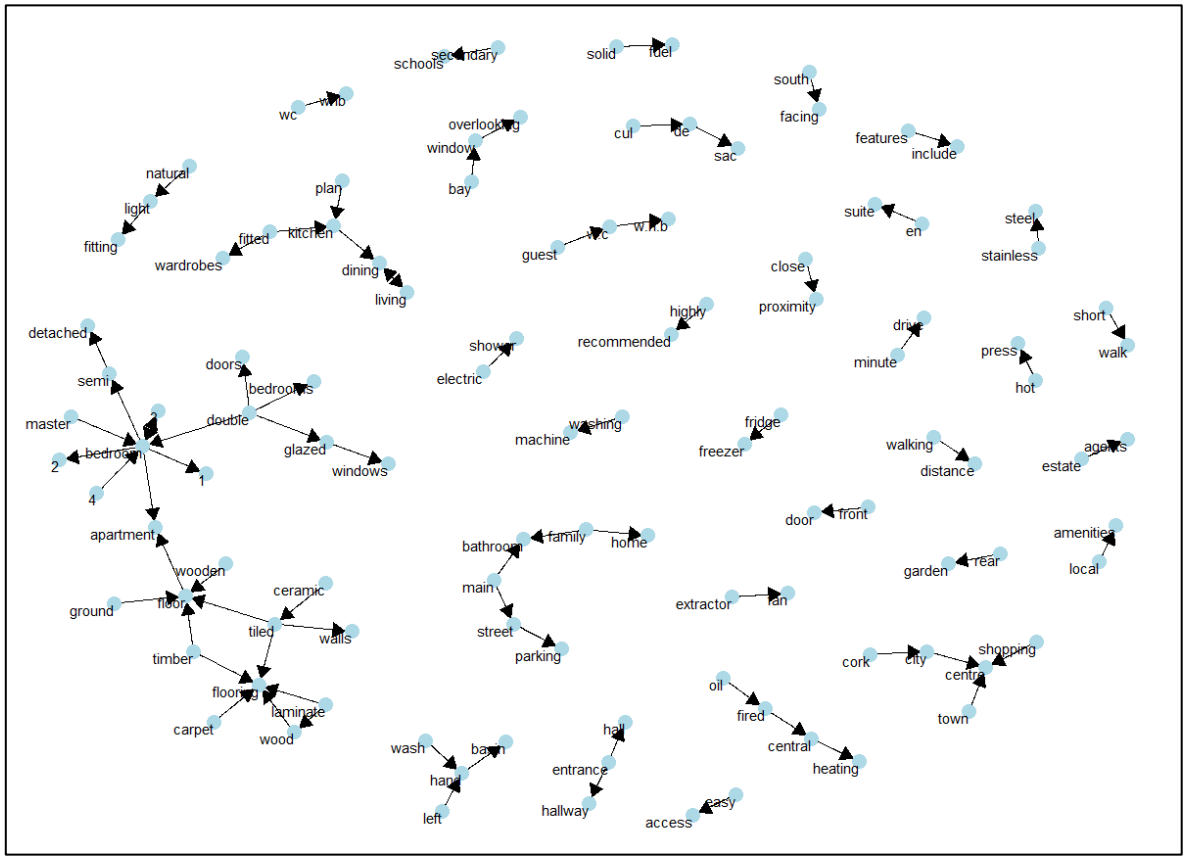


Building on this, Sentiment Analysis was undertaken so as to determine how each house listing was being described. This technique *"aims to determine the attitude of a writer with respect to some topic or the overall contextual polarity of a document*" (Lou et al, 2013). In this instance that refers to words with positive or negative associations with the top 15 most prevalent words by sentiment type illustrated below:



As can be seen, there were a far greater number of positive words than negative words. This is to be expected considering the purpose of each house listing is to try and effectively market each house to try and maximise views and potential buyers' interest, while being accurate in its description of the property in question.

Bigrams were investigated so as to investigate word pairings, that is – how each word interacts with the word immediately following or preceding it. Selecting only word pairings with above 25 instances, we can see how they relate to each other as below:

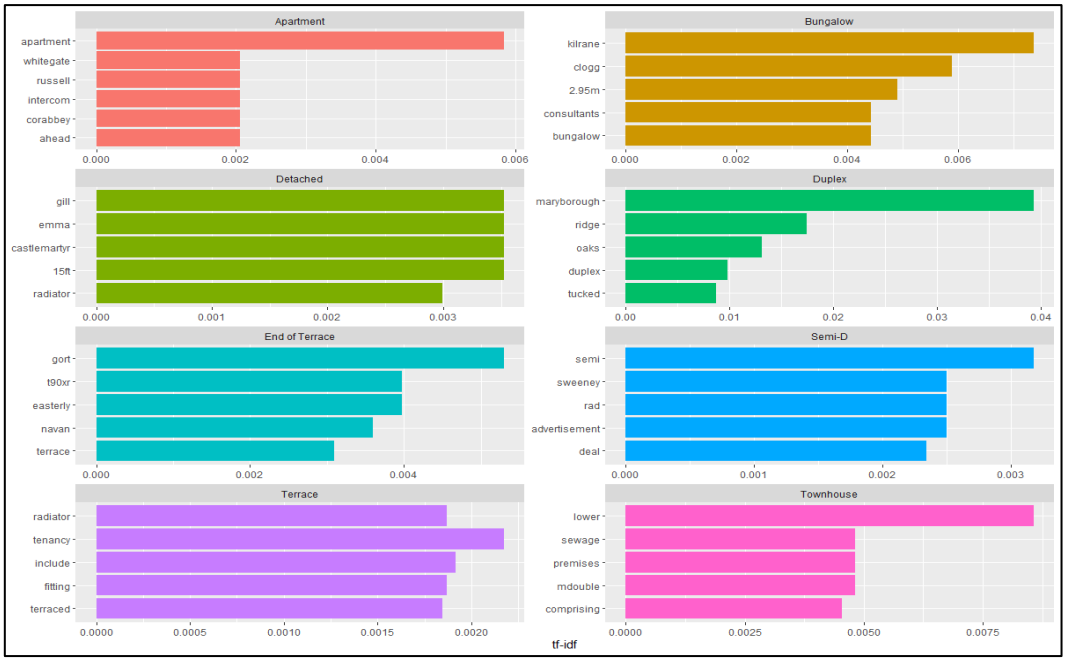


It is interested to note there are two large clusters of bigrams, namely kitchen and its associated words, and both 'bedroom' and 'flooring' both of which connected to each other via 'floor'.

Building on our analysis of House Type via use of our Naive Bayes Model, the following words were found to be most prevalent throughout each house type:

You will note “Property” does not appear in these graphs. That is because we removed it as we found it to be a very general word that does not do much to help us understand how word data is distributed by each house listing. Interestingly, in all but the Apartment category, the word to describe the actual type of house is not prevalent.

In order to investigate these observations further, a tf-idf analysis was undertaken. By tf-idf, we refer to term frequency–inverse document frequency. This highlights how important a word is to a document in a collection or corpus. In this case, how important a word is to each house listing. Without any pre-processing (minus removing stop words), we achieved the following results:



As can be seen above, the description of each house type is very important as well as the name of the property being listed. Descriptive words such as ‘include’ and ‘comprising’ were important to the Terrace and Townhouse categories, while measurements were important to the Bungalow and Detached categories.

Conclusion

Our analysis shows that there is a correlation between the wording used in the description and the house price. It also shows that the name of the property can often have a major influence on the value of the house. As shows in our decision tree, the biggest contributors are the listing size, house type and energy rating. Our models could be improved with more data, so that there is a larger number of each house type in the test set. As seen in the decision tree data, the no information rate was higher due to a difference in the number of each house type. This is why we took the decision to remove some of the house types for our Naïve Bayes analysis. Unfortunately, due to large amounts of missing data on the daft website, the dataset needed to be trimmed so as not to skew the data, leading to a smaller dataset to work with. This is one of the many problems that can arise with web scraping. Occasionally, websites may take measures to stop data scraping. Daft required data to be scraped more slowly so as not to be removed.

Bibliography

Chowdhury, G. (2003) *Natural language processing*. Annual Review of Information Science and Technology, 37. pp. 51-89. ISSN 0066-4200

D. D. Lewis, Naive (Bayes) at forty: the independence assumption in information retrieval, in: C. N´edellec, C. Rouveirol (Eds.), Machine Learning: ECML-98: 10th European Conference on Machine Learning, Chemnitz, Germany, April 21–23, Springer, Berlin/Heidelberg, 1998, pp. 4–15.

Kamiński, B. Jakubczyk, M. and Szufel P, (2017). *A framework for sensitivity analysis of decision trees*. Cent Eur J Oper Res. 2018; 26(1): 135–159. doi: 10.1007/s10100-017-0479-6

Liddy, E.D. 2001. *Natural Language Processing*. In Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc

Luo, Tiejian & Chen, Su & Xu, Guandong & Zhou, Jia. (2013). Sentiment Analysis. 10.1007/978-1-4614-7202-5_4.

Patel, Harsh & Prajapati, Purvi. (2018). Study and Analysis of Decision Tree Based Classification Algorithms. International Journal of Computer Sciences and Engineering. 6. 74-78. 10.26438/ijcse/v6i10.7478.

Rish, Irina. (2001). *An Empirical Study of the Naïve Bayes Classifier*. IJCAI 2001 Work Empir Methods Artif Intell. 3.

Verspoor, Karin & Cohen, Kevin. (2013). Natural Language Processing. 10.1007/978-1-4419-9863-7_158.

Zhao, Bo. (2017). *Web Scraping*. 10.1007/978-3-319-32001-4_483-1.