SDSS Take Home Exam

Pierre Michel B. Hardy | I6117019

Part One

Question A

First review:

"I'm grading this MVD only compared to other, similar establishments, like compared to other DMV's that I've been to. As such, it gets 4 stars. If I were writing this review and comparing this place to, say, Disneyland, it would definitely get like a 1 or a 2.

Most DMV's smell like.... ugh. Death? Vomit? Vomiting corpses? B.O.? Cumin? Hatred? This one doesn't smell bad and I was here in July in Phoenix, now that's saying something.

If you go to the DMV in California, you'd better not plan on meeting anyone because you WILL need to call them and say, "Goddamnit, this is taking longer than I expected." Also, California's real dick about licenses. Arizona is apparently the most lax state in the country about giving people the right to drive. I went in, gave them my California license, they took my picture and I was in and out in about 12 minutes. And my license doesn't expire until like 2345, so by then I'll be aboard the USS Enterprise... D. The Next Generation.

It kind of scares me how easy it is to get a license given how old people love to drive through farmers markets and kill dozens of people. Oh well. I got mine!"

Directive | Commissive | Assertive

I answer this question by highlighting parts of the review that are directives in yellow, commissives in green, and assertives in blue.

Assertive statements are those that "asserts" information. In a way, they are like neutral journalists that only conveys information. The first highlight merely states the context of the review. The second highlight asserts the information that the place does not smell bad. The third highlight shows the reviewer's journey, not explicitly saying if 12 minutes is fast or slow. The fourth highlight asserts the, probably exaggerated, information of the license's expiration date, not explicitly saying if 2345 is a good year for a license to expire. The last highlight simply asserts the information that the transaction was successful.

Directive statement conveys commands that directs the reader on what to do. The only directive statement I found is the reviewer directing the reader not to plan anything else if the reader is visiting the DMV in California. Though it is not a directed to the specific DMV it is reviewing, it’s still technically a directive statement.

Commissive statements conveys commitments in a form of a vow or promise. I found no commissive statements. However, I was debating whether this part of the review, “if I were writing this review and comparing this place to, say, Disneyland, it would definitely get like a 1 or a 2,” was a commissive statement since it seems to commit to a different rating, but it’s more of a conditional statement than commissive. However, conditional is not included as a kind of illocution under the Speech Act Theory and I remain divided. In the end, I decide not to identify the sentence as a commissive statement because I am not thoroughly convinced that it is.

Question B

Original review

I will show the full review in each step of the way to better illustrate the changes that took place while preprocessing the dataset.

"I'm grading this MVD only compared to other, similar establishments, like compared to other DMV's that I've been to. As such, it gets 4 stars. If I were writing this review and comparing this place to, say, Disneyland, it would definitely get like a 1 or a 2. \n\nMost DMV's smell like.... ugh. Death? Vomit? Vomiting corpses? B.O.? Cumin? Hatred? This one doesn't smell bad and I was here in July in Phoenix, now that's saying something.\n\nIf you go to the DMV in California, you'd better not plan on meeting anyone because you WILL need to call them and say, \"Goddamnit, this is taking longer than I expected.\" Also, California's real dick about licenses. Arizona is apparently the most lax state in the country about giving people the right to drive. I went in, gave them my California license, they took my picture and I was in and out in about 12 minutes. And my license doesn't expire until like 2345, so by then I'll be aboard the USS Enterprise... D. The Next Generation. \n\nIt kind of scares me how easy it is to get a license given how old people love to drive through farmers markets and kill dozens of people. Oh well. I got mine!"

Remove punctuation

First, we remove the punctuations within the reviews. As we can see below, apostrophes, commas, periods, and exclamation points were taken away.

"Im grading this MVD only compared to other similar establishments like compared to other DMVs that Ive been to As such it gets 4 stars If I were writing this review and comparing this place to say Disneyland it would definitely get like a 1 or a 2 \n\nMost DMVs smell like ugh Death Vomit Vomiting corpses BO Cumin Hatred This one doesnt smell bad and I was here in July in Phoenix now thats saying something\n\nIf you go to the DMV in California youd better not plan on meeting anyone because you WILL need to call them and say Goddamnit this is taking longer than I expected Also Californias real dick about licenses Arizona is apparently the most lax state in the country about giving people the right to drive I went in gave them my California license they took my picture and I was in and out in about 12 minutes And my license doesnt expire until like 2345 so by then Ill be aboard the USS Enterprise D The Next Generation \n\nIt kind of scares me how easy it is to get a license given how old people love to drive through farmers markets and kill dozens of people Oh well I got mine"

Remove numbers

Next, we remove the numbers. We only want text in the end. As we can see below, the year "2345" is gone.

"Im grading this MVD only compared to other similar establishments like compared to other DMVs that Ive been to As such it gets stars If I were writing this review and comparing this place to say Disneyland it would definitely get like a or a   \n\nMost DMVs smell like ugh Death Vomit Vomiting corpses BO Cumin Hatred This one doesnt smell bad and I was here in July in Phoenix now thats saying something\n\nIf you go to the DMV in California youd better not plan on meeting anyone because you WILL need to call them and say Goddamnit this is taking longer than I expected Also Californias real dick about licenses Arizona is apparently the most lax state in the country about giving people the right to drive I went in gave them my California license they took my picture and I was in and out in about minutes And my license doesnt expire until like so by then Ill be aboard the USS Enterprise D The Next Generation \n\nIt kind of scares me how easy it is to get a license given how old people love to drive through farmers markets and kill dozens of people Oh well I got mine"

Lowercase

Next, we make everything in lowercase. As we can see below, there is no capital letter in sight.

"im grading this mvd only compared to other similar establishments like compared to other dmvs that ive been to as such it gets stars if i were writing this review and comparing this place to say disneyland it would definitely get like a or a   \n\nmost dmvs smell like ugh death vomit vomiting corpses bo cumin hatred this one doesnt smell bad and i was here in july in phoenix now thats saying something\n\nif you go to the dmv in california youd better not plan on meeting anyone because you will need to call them and say goddamnit this is taking longer than i expected also californias real dick about licenses arizona is apparently the most lax state in the country about giving

people the right to drive i went in gave them my california license they took my picture and i was in and out in about minutes and my license doesnt expire until like so by then ill be aboard the uss enterprise d the next generation \n\nit kind of scares me how easy it is to get a license given how old people love to drive through farmers markets and kill dozens of people oh well i got mine"

Take out Whitespace

Next, we take out the unnecessary whitespaces. As we can see below, the "\n" were gone. "\n" means new line, which is empty space.

"im grading this mvd only compared to other similar establishments like compared to other dmvs that ive been to as such it gets stars if i were writing this review and comparing this place to say disneyland it would definitely get like a or a most dmvs smell like ugh death vomit vomiting corpses bo cumin hatred this one doesnt smell bad and i was here in july in phoenix now thats saying something if you go to the dmv in california youd better not plan on meeting anyone because you will need to call them and say goddamnit this is taking longer than i expected also californias real dick about licenses arizona is apparently the most lax state in the country about giving people the right to drive i went in gave them my california license they took my picture and i was in and out in about minutes and my license doesnt expire until like so by then ill be aboard the uss enterprise d the next generation it kind of scares me how easy it is to get a license given how old people love to drive through farmers markets and kill dozens of people oh well i got mine"

Remove common English words

Next, we want to shorten the review and condense the review to only the most informative words. The function has a pre-defined set of English stop words that was automatically applied to the review. As we can observe below, several common words were taken away such as "if," "I," and "were," among others.

"im grading mvd compared   similar establishments like compared   dmvs ive gets stars   writing review comparing place say disneyland   definitely get like dmvs smell like ugh death vomit vomiting corpses bo cumin hatred one doesnt smell bad   july phoenix now thats saying something   go   dmv california youd better plan meeting anyone   will need   call   say goddamnit   taking longer   expected also californias real dick licenses arizona apparently   lax state   country giving people right drive went gave   california license took picture   minutes   license doesnt expire like   ill aboard uss enterprise d next generation kind scares   easy   get   license given old people love drive farmers markets kill dozens people oh well got mine"

Remove special characters

Next, since we want only text, we still need to make sure that there are no lingering special characters in the review. These special characters are: [][!#$%()*,.:;<=>@^_`|~.{}&¼½¾¥''.+-].

"im grading mvd compared similar establishments like compared dmvs ive gets stars writing review comparing place say disneyland definitely get like dmvs smell like ugh death vomit vomiting corpses bo cumin hatred one doesnt smell bad july phoenix now thats saying something go dmv california youd better plan meeting anyone will need call say goddamnit taking longer expected also californias real dick licenses arizona apparently lax state country giving people right drive went gave california license took picture minutes license doesnt expire like ill aboard uss enterprise d next generation kind scares easy get license given old people love drive farmers markets kill dozens people oh well got mine"

Stem the words to their roots

Lastly, it would be better for the analysis to condense the text further into their root. This removes the difference that tenses make on words that would otherwise mean the same thing. As we can see below, “dozens” is turned into “dozen” and “license” and “licenses” were just turned into “licens”.

"im grade mvd compar similar establish like compar dmvs ive get star write review compar place say disneyland definit get like dmvs smell like ugh death vomit vomit corps bo cumin hatr one doesnt smell bad juli phoenix now that say someth go dmv california youd better plan meet anyon will need call say goddamnit take longer expect also california real dick licens arizona appar lax state countri give peopl right drive went gave california licens took pictur minut licens doesnt expir like ill aboard uss enterpris d next generat kind scare easi get licens given old peopl love drive farmer market kill dozen peopl oh well got mine"

Question C

To extract the most important and least important terms, I took the average per column (per term) of the TF-IDF matrix. Taking the average tf-idf would show the average importance of a word in reviews. A high average means that the term is important in plenty of reviews and vice versa. Below are the top ten and bottom ten terms and their corresponding tf-idf statistic.

Top Ten Most Important Terms

1	great	0.01768110
2	food	0.01753520

3	place	0.01669992
4	good	0.01583382
5	love	0.01569821
6	servic	0.01338557
7	order	0.01268296
8	time	0.01241904
9	chicken	0.01237483
10	just	0.01221714

Bottom Ten Least Important Terms

10	albaricoqu	5.008818e-05
9	andr	5.008818e-05
8	arroz	5.008818e-05
7	banda	5.008818e-05
6	bogavant	5.008818e-05
5	brief	5.008818e-05
4	brusela	5.008818e-05
3	degere	5.008818e-05
2	dream	5.008818e-05
1	ensalada	5.008818e-05

We can observe the the bottom 10 all have similarly low it-idf score and are arranged alphabetically. This indicates that there are more that shares the lowest score, however, I will only show ten as there are in total thirty-six terms with that low score.

Question E

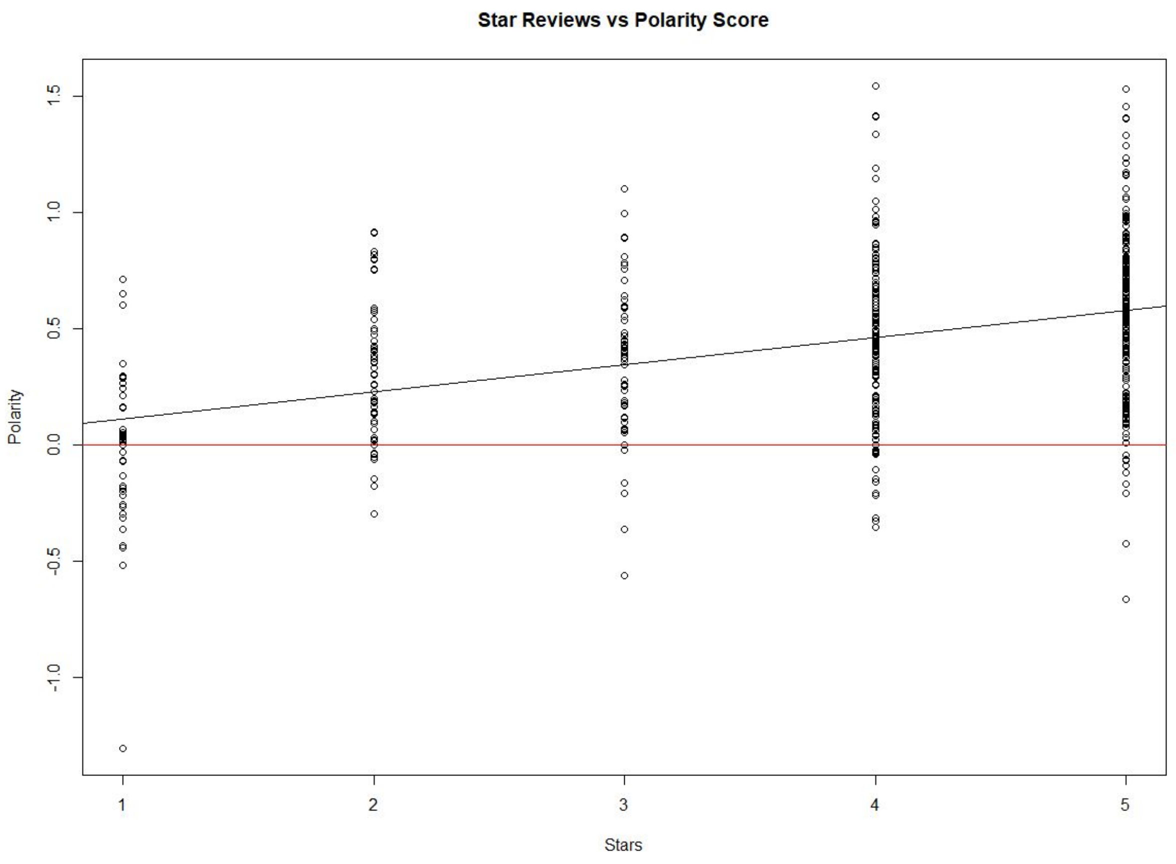


Figure 1: Star Reviews vs Polarity Score Scatterplot

I used a scatter plot to study the relationship between the star rating and sentiment analysis. In the x-axis is the star rating and in the y-axis is the polarity score. In order to see clearly if there is a relationship between the two variable, I fitted a regression line. As we can see, they are positively correlated to the tune of 0.12 with a p-value of 0.00 making it very significant. I also put a red line through a 0 polarity score for better interpretation.

Question F

Besides variables already provided in the business dataset, I decided to transform some of these. First, I transformed the times open from Monday to Sunday into how many hours they are open. These would be more understandable values than the times in the original business dataset.

Second, I decided to dummify the categories variable. Since there are multiple categories in one cell, I first separated these categories then turn each category into a column in a binary fashion. However, there were 334 unique categories. What I decided to do is to manually check all categories with above average counts within the whole dataset (there were 67 out of 334). I subjectively chose the most common and high level category in that short list of 67 categories. For example, I decided to only keep the “restaurants” category and do away with the categories indicating the cuisine. I ended up picking 20, which are:

"Restaurants"	“Beauty & Spas"	"Casinos"
"Nightlife	"Desserts"	"Automotive"
"Event Planning & Services"	"Active Life"	"Wine & Spirits"
"Arts & Entertainment"	"Bakeries"	"Hotels & Travel"
"Shopping"	"Local Services"	"Fashion"
"Coffee & Tea"	"Home Services"	"Buffets"
	"Fast Food"	"Health & Medical"

Question I

Below are two tables that represents the output of the multinomial logistic regression models. For less clutter, I have only included the coefficients that managed to get significant p-values in this section. I have included the full tables of tables 1 and 2 in Appendix 1.

The insights that have been found applies mostly to restaurant owners and how they can achieve higher star ratings and better reviews. However, the business model of Yelp relies on gathering advertising fees from business owners. Yelp can either use these insights as an enticing product to charge to business owners or as an add-on to their existing advertising bundles. Either way, satisfied business owners could lead to more business owners advertising in Yelp, which in the end helps Yelp’s bottomline. In other words, what businesses can learn and what Yelp can learn from these insights are intertwined, meaning that if the insight benefits one, it also benefits the other.

Table 1 below shows the factors that affects the star rating given to a review. The analysis showed that one of the most significant factors is the location. Some locations affect the star ratings for better or worse. For example, the city of Surprise is associated with getting a high star rating by quite a lot. By contrast, the City of Glendale is associated with lower star ratings. Additionally, the certain types of establishments also has an effect on star ratings. The analysis finds that, for example, the category “shopping” is associated with higher star ratings while the category “automotives” is associated with lower star ratings. There were also certain days where the opening hours affects the star ratings, like Thursday (negatively) and Wednesday (positively). These findings can help business owners such as which cities have strong competition (associated with high star ratings) and which days customers value having a business open more.

The analysis has also shown that better reviews with higher sentiment scores are associated with higher star ratings, which makes sense. This could be valuable for Yelp since it is a good sign that there is congruence to the reviews in their site and the corresponding star ratings. The analysis also found that for every 1% increase in the length of a review, the star ratings get lower by 0.45%. This can mean that dissatisfied reviews tend to be longer than satisfied ones.

Variable	Value	Std. Error	t value	p value	sig
cityAurora	1.026305653	1.60E-01	6.40E+00	0.00	*
cityConcord	-1.165243377	2.09E-01	-5.57E+00	0.00	*
cityGlendale	-1.812017394	2.39E-01	-7.57E+00	0.00	*
cityGoodyear	-0.612043577	1.18E-01	-5.19E+00	0.00	*
cityMississauga	-0.567791402	1.54E-01	-3.69E+00	0.00	*
cityNew Kensington	6.146889928	4.95E-04	1.24E+04	0.00	*
cityNorth Las Vegas	1.732493082	6.14E-01	2.82E+00	0.00	*
cityNorth York	-0.986809337	8.70E-02	-1.13E+01	0.00	*
cityPeoria	-2.847766631	4.92E-02	-5.79E+01	0.00	*
citySurprise	19.46626441	8.59E-09	2.26E+09	0.00	*
stars.y	1.632424293	1.87E-01	8.74E+00	0.00	*
thursday.hr.open	-0.569900455	2.00E-01	-2.85E+00	0.00	*
Shopping	1.334351157	4.45E-01	3.00E+00	0.00	*
polarity	1.371076098	5.67E-01	2.42E+00	0.02	*
word.count	-0.004498566	1.98E-03	-2.27E+00	0.02	*
cityMadison	1.511769985	6.93E-01	2.18E+00	0.03	*
wednesday.hr.open	0.581944202	2.62E-01	2.22E+00	0.03	*
cityMontrÃ©al	1.343711658	6.60E-01	2.04E+00	0.04	*
`Beauty & Spas`	1.548012015	7.45E-01	2.08E+00	0.04	*
Desserts	-1.014926191	5.03E-01	-2.02E+00	0.04	*

Automotive	-1.017814796	5.08E-01	-2.00E+00	0.05	*

Table 1 - Dependent variable: star rating of individual reviews

Table 2 shows the factors that affect the sentiment score of a review.

The star ratings are significantly corresponding to the sentiment of the reviews accordingly, save for reviews with 3-star ratings, which seems to be associated the most with positive reviews. This insight could be informative for Yelp and how reviewers are drawn to middle ratings. Somehow at odds with table 1, it seems that a 1% increase in review length lead to a 0.44% increase in review sentiment. Similar to the findings in table 1, certain categories may lead to better or worse reviews like fashion (positively) and hotels and travel (negatively). This finding is also interesting for Yelp since it could mean that for certain categories, people only review if the experience is bad more than usual. Yelp can use this insight to incentivize leaving positive reviews for categories that seem to have more bad reviews than the average.

Reviewers also seem to have more favorable reviews on certain locations. For example, the businesses from the city of Gilbert seems to generate plenty of positive reviews. Conversely, customers seem to be more prone for a bad time in North York. Business owners can also benefit from the finding that customers value a shop being open on Sunday very much in the reviews.

Variables	Value	Std.Error	t value	p value	sig
stars.x3	1.82E+00	0.429728189	4.23492567	0.00	*
stars.x4	1.39E+00	0.388369953	3.56984289	0.00	*
stars.x5	1.75E+00	0.378846358	4.62923148	0.00	*
cityAurora	1.35E+00	0.268832168	5.02167467	0.00	*
cityGilbert	2.67E+00	0.691005593	3.8610022	0.00	*
cityMarkham	1.34E+00	0.41358364	3.24830961	0.00	*
cityNew Kensington	1.36E+00	0.129186726	10.50656878	0.00	*
cityNorth Las Vegas	1.51E+00	0.322973921	4.68593376	0.00	*
cityNorth York	-2.20E+00	0.094829256	-23.1785671	0.00	*
cityPeoria	3.66E+00	0.281313098	13.00935296	0.00	*
citySurprise	-1.42E+00	0.08988269	-15.75055903	0.00	*

sunday.hr.open	1.78E-01	0.06155951	2.8899106	0.00	*
Shopping	-1.45E+00	0.459023754	-3.1505366	0.00	*
Automotive	-2.05E+00	0.305529257	-6.71369336	0.00	*
is_open	-9.67E-01	0.387645101	-2.49454876	0.01	*
cityMississauga	6.26E-01	0.27022272	2.31484721	0.02	*
word.count	4.35E-03	0.001803719	2.41149906	0.02	*
\`Hotels & Travel\`	-1.13E+00	0.470023021	-2.41324376	0.02	*
stars.x2	1.04E+00	0.471376899	2.20925377	0.03	*
Fashion	9.42E-01	0.434235964	2.17045801	0.03	*

Table 2 - Dependent variable: sentiment score of review

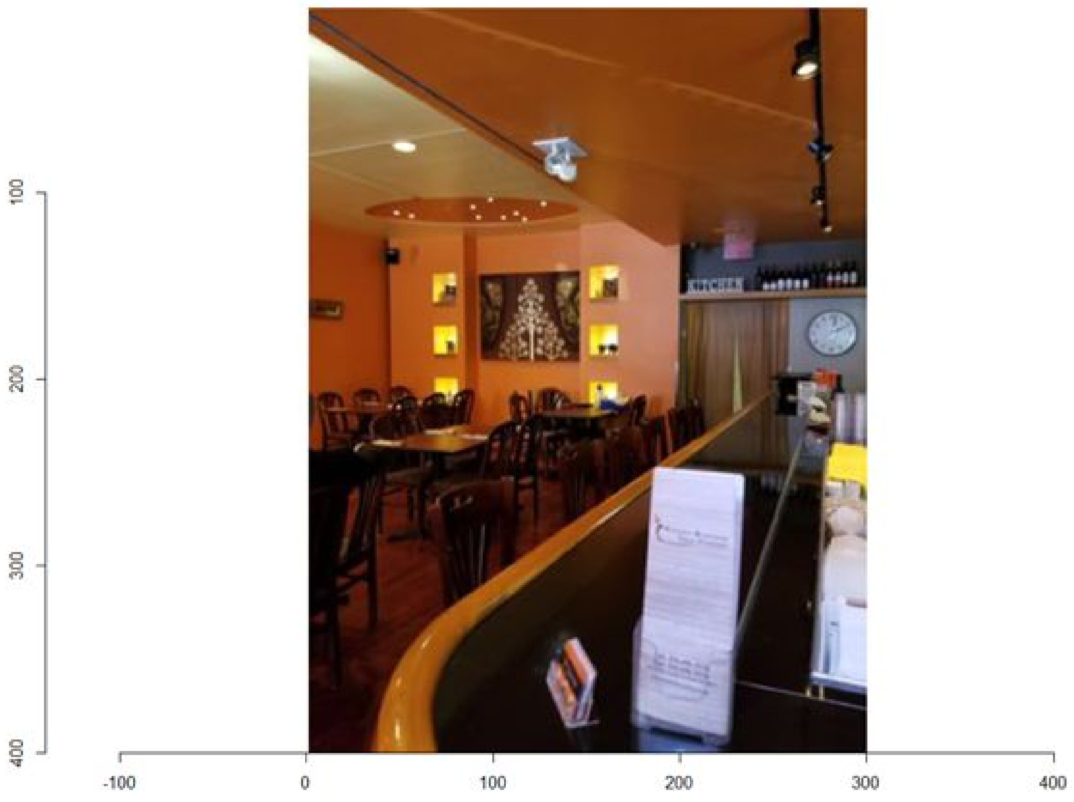
Part Two

Question C

The two features I chose captures the color models and shape of the image. To capture the color models, I employed a color histogram and get each image'S RGB and HSV information to extract the image's color composition and how "vibrant" the photo is. As for the shape, I decided to extract the amount of lines in a image. I forewent texture features since I believe that in the context of the project, color and lines are more relevant. Given more time and a better computer, however, I would have included it too.

Before showing a sample photo, here are some notes on my process. I had to remove 4 photos from the 400 since 2 causes errors in the color histograms (2 images were png's hidden as jpeg's) and 2 that causes the houghline() function to crash RStudio. I have forced my laptop to extract the lines for the remaining photos by dividing the work into four chunks and clearing some of the variables in the Global Environment to free RAM space in between those chunks. It is admittedly not the most elegant or efficient code, but I believe time is of second priority in this case and it managed to get the job done.

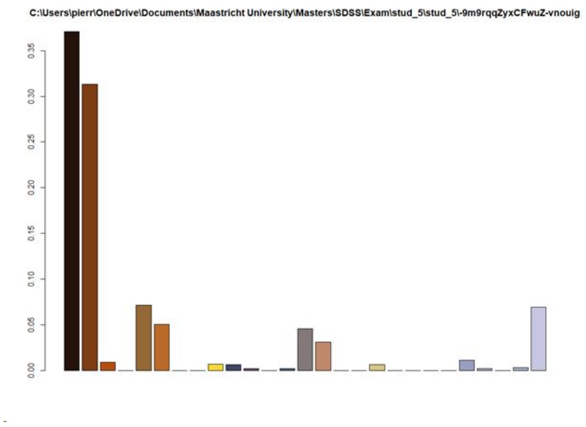
To illustrate the process of extracting the features, I will use the first image in my sample. The actual image is shown in figure 2.

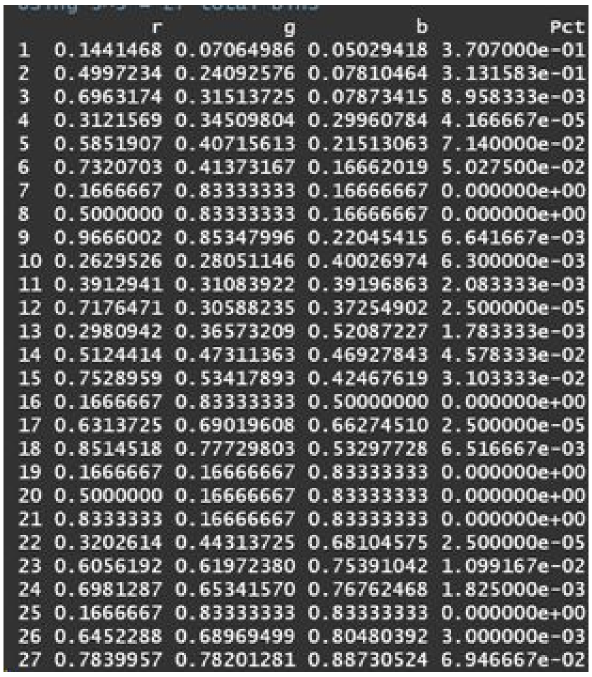


-

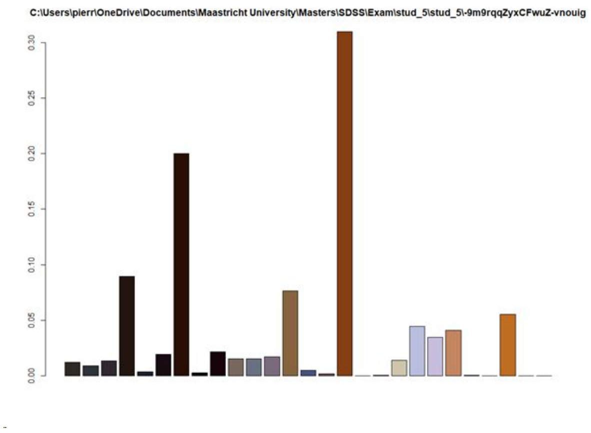
Figure 2: Actual image

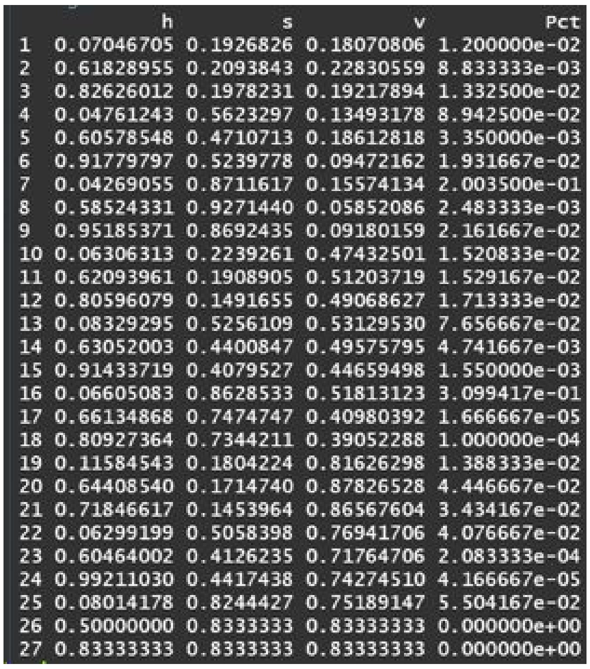
First, I generate a color histogram to capture the color and “vibrancy” elements. Figure 3 shows the rgb color histogram on the upper left and the hsv color histogram on the lower left. Their corresponding quantified representation is in the right side. Color histograms quantify the characteristics of an image by binning the amount of pixels that shows off which light and at which intensity. This can be refashioned into a vector that describes the image.





```
               r          g          b          Pct
1  0.1441468 0.07064986 0.05029418 3.707000e-01
2  0.4997234 0.24092576 0.07810464 3.131583e-01
3  0.6963174 0.31513725 0.07873415 8.958333e-03
4  0.3121569 0.34509804 0.29960784 4.166667e-05
5  0.5851907 0.40715613 0.21513063 7.140000e-02
6  0.7320703 0.41373167 0.16662019 5.027500e-02
7  0.1666667 0.83333333 0.16666667 0.000000e+00
8  0.5000000 0.83333333 0.16666667 0.000000e+00
9  0.9666002 0.85347996 0.22045415 6.641667e-03
10 0.2629526 0.28051146 0.40026974 6.300000e-03
11 0.3912941 0.31083922 0.39196863 2.083333e-03
12 0.7176471 0.30588235 0.37254902 2.500000e-05
13 0.2980942 0.36573209 0.52087227 1.783333e-03
14 0.5124414 0.47311363 0.46927843 4.578333e-02
15 0.7528959 0.53417893 0.42467619 3.103333e-02
16 0.1666667 0.83333333 0.50000000 0.000000e+00
17 0.6313725 0.69019608 0.66274510 2.500000e-05
18 0.8514518 0.77729803 0.53297728 6.516667e-03
19 0.1666667 0.16666667 0.83333333 0.000000e+00
20 0.5000000 0.16666667 0.83333333 0.000000e+00
21 0.8333333 0.16666667 0.83333333 0.000000e+00
22 0.3202614 0.44313725 0.68104575 2.500000e-05
23 0.6056192 0.61972380 0.75391042 1.099167e-02
24 0.6981287 0.65341570 0.76762468 1.825000e-03
25 0.1666667 0.83333333 0.83333333 0.000000e+00
26 0.6452288 0.68969499 0.80480392 3.000000e-03
27 0.7839957 0.78201281 0.88730524 6.946667e-02
```





```
            h         s          v          Pct
1  0.07046705 0.1926826 0.18070806 1.200000e-02
2  0.61828955 0.2093843 0.22830559 8.833333e-03
3  0.82626012 0.1978231 0.19217894 1.332500e-02
4  0.04761243 0.5623297 0.13493178 8.942500e-02
5  0.60578548 0.4710713 0.18612818 3.350000e-03
6  0.91779797 0.5239778 0.09472162 1.931667e-02
7  0.04269055 0.8711617 0.15574134 2.003500e-01
8  0.58524331 0.9271440 0.05852086 2.483333e-03
9  0.95185371 0.8692435 0.09180159 2.161667e-02
10 0.06306313 0.2239261 0.47432501 1.520833e-02
11 0.62093961 0.1908905 0.51203719 1.529167e-02
12 0.80596079 0.1491655 0.49068627 1.713333e-02
13 0.08329295 0.5256109 0.53129530 7.656667e-02
14 0.63052003 0.4400847 0.49575795 4.741667e-03
15 0.91433719 0.4079527 0.44659498 1.550000e-03
16 0.06605083 0.8628533 0.51813123 3.099417e-01
17 0.66134868 0.7474747 0.40980392 1.666667e-05
18 0.80927364 0.7344211 0.39052288 1.000000e-04
19 0.11584543 0.1804224 0.81626298 1.388333e-02
20 0.64408540 0.1714740 0.87826528 4.446667e-02
21 0.71846617 0.1453964 0.86567604 3.434167e-02
22 0.06299199 0.5058398 0.76941706 4.076667e-02
23 0.60464002 0.4126235 0.71764706 2.083333e-04
24 0.99211030 0.4417438 0.74274510 4.166667e-05
25 0.08014178 0.8244427 0.75189147 5.504167e-02
26 0.50000000 0.8333333 0.83333333 0.000000e+00
27 0.83333333 0.8333333 0.83333333 0.000000e+00
```

Figure 3: Color histograms

The next feature to extract are the lines in the image. First, the function cannyEdges is used to extract the edges in an image, as presented in figure 4. Afterwards, the function hough_lines is used to extract how many lines are present in figure 4. Scores are assigned to the lines denoting importance. We can count the top most important lines and this number of lines is another quantified feature of an image. For this example, this figure has 4000 important lines, which is shown in figure 5.

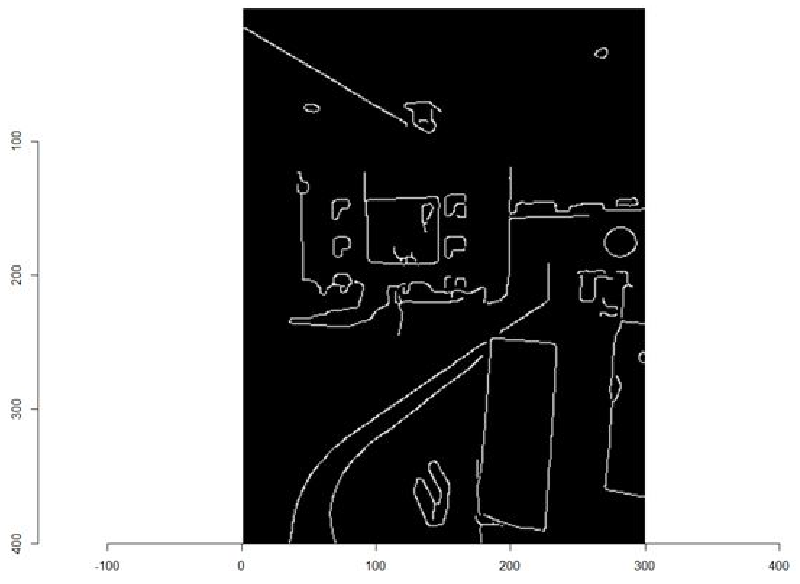


Figure 4: Edges of the image

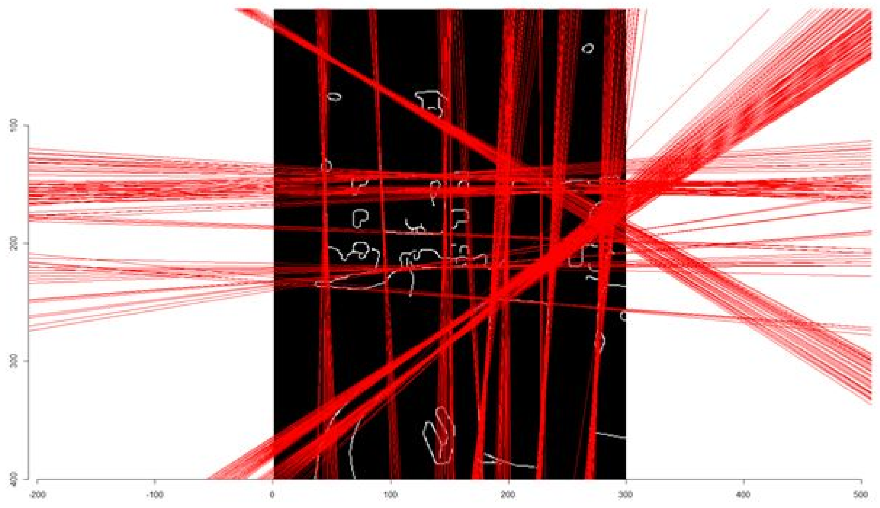


Figure 5: Important lines

And with this quantified information about an image, it is possible that there is a pattern within this sea of uninterpretable numbers that the computer can find that can be used to create a machine learning model that can identify images based on these hidden patterns.

Question F

There are plenty of classifiers to test. For this exam, I chose Support Vector Machine and Gradient Boosted Regression Trees by using the XGBoost function.

First, I chose SVM mainly due to its prevalence in image analytics and its good performance within image analytics papers.

Second, I chose GBRT since it is good at mapping somewhat abstract forms of data like images. The main weakness, however, is the lack of interpretability.

Below are the metrics I collected while trying different methods. First, I computed the baseline model, which is a model that just predicts everything as the major target attribute (in this case, it's the "food" category). The baseline model achieves an accuracy of 60%. This number is then the accuracy to beat.

Then, I divided the dataset into 80% training set and 20% test set. Table 3 shows the results. The SVM acted exactly like the baseline model. This is a useless model overall. The GBRT has managed to achieve an accuracy that is two percentage points more than the baseline model. This is not much but it is also not nothing. It is also worth noting the elusiveness of the F1-Score. This is mainly caused by my models not predicting a certain classification at all during tests. This problem persisted even after balancing the dataset. Due to this, I am forced to stick with using the accuracy measure.

Classifier	Accuracy		Recall		F1-Score	
	Train Set	Test Set	Train Set	Test Set	Train Set	Test Set
Baseline Model	59.31%	60.26%	NA	NA		
SVM (unbalanced)	59.31%	60.26%	0%	0%	NaN	NaN
XGBoost (unbalanced)	93.06%	62.82%	100%	0%	100%	NaN
SVM (balanced)	99.53%	60.23%	99.69%	0%	99.77%	NaN
XGBoost (balanced)	96.99%	60.26%	99.69%	NA	99.84%	

Table 3: Evaluation metrics of different classifiers

Noting how the SVM acted like a baseline model, I checked the balance of the test data set. Table 4 shows the proportion of the four labels. The original training set is severely unbalanced with the overwhelming majority being “foods” and images showing “outside” and “drink” are underrepresented.

To fix this, I used the SMOTE package and function to generate a more balanced dataset. Table 4 shows the new dataset.

Training Set	Drink	Food	Inside	Outside
Unbalanced	9.15%	59.31%	25.24%	6.31%
Balanced	25.38%	30.29%	20.55%	23.79%

Table 4: Balances of training datasets.

I re-run the models using the balanced dataset and quite frankly, the results have been surprising and disappointing, as can be observed in Table 3. In the end, I have decided to produce the predictions for the test set of 100 images (of which, two pictures have been removed since one is a png and the other causes RStudio to crash) using the XGBoost model trained with the unbalanced datasets.

I am not satisfied with these results but due to the lack of time, I am forced to submit this as is. However, the next steps I would’ve taken is to extract texture features and try different classifiers.

Part Three

Question A

A/B testing can be used in this situation. It’s a clear choice since the question has outlined two clear differences that they want to investigate (all positive reviews vs mixed reviews). Yelp could employ online A/B testing because of a couple of reasons. First, testing it directly to Yelp users is representative of the target population. Second, A/B online testing is cost effective.

They could randomly select a small sample of users and divide them into two. In the first group, they would show users only positive reviews. In the second group, they would show users a mixed number of positive and negative reviews. Afterwards, Yelp must take note of the star ratings left by the users in each respective groups. Then they can employ a simple ANOVA test to see if there is a significant difference between the two groups and the star ratings left by the users.

Question B

Benefits:

1. It is cost effective.
2. It is easier to capture a representative population sample.

Limitations:

1. Depending on the company, coding the changes may be time-intensive for some companies and there would be an endless variety of tweaks.
2. A/B Testing can only capture very specific effects. For example, the number of review may make a difference in star ratings but there might be another underlying factor that affects the star ratings much more.

Question C

Since Yelp is a website, the presentation of the website is very important and slight chances can have profound effects. The idea of morphing is to automatically adjust a website's appearance based on some objective. Asking which segmenting variables could be used is too broad of a question that is challenging to properly answer without more context. But if one is forced to speculate, different days of the week may have different effects to the star ratings, as well seasons. Yelp could adapt the outline of their website depending on what day it is or what month. Similarly, there may be different tastes in different locations. Perhaps, people may also have different preferences for the mixture of reviews depending on the category. For example, personally, an expensive purchase would warrant a more critical review, meaning I would prefer a mixed number of reviews. Basically, everything has the potential to be a segmentation variable. This may not be the expected answer from this question however that is what I strongly believe.

Appendix 1 - Full Tables

Variable	Value	Std. Error	t value	p value	sig
cityAurora	1.026305653	1.60E-01	6.40E+00	0.00	*
cityConcord	-1.165243377	2.09E-01	-5.57E+00	0.00	*
cityGlendale	-1.812017394	2.39E-01	-7.57E+00	0.00	*
cityGoodyear	-0.612043577	1.18E-01	-5.19E+00	0.00	*
cityMississauga	-0.567791402	1.54E-01	-3.69E+00	0.00	*
cityNew Kensington	6.146889928	4.95E-04	1.24E+04	0.00	*
cityNorth Las Vegas	1.732493082	6.14E-01	2.82E+00	0.00	*
cityNorth York	-0.986809337	8.70E-02	-1.13E+01	0.00	*
cityPeoria	-2.847766631	4.92E-02	-5.79E+01	0.00	*
citySurprise	19.46626441	8.59E-09	2.26E+09	0.00	*

stars.y	1.632424293	1.87E-01	8.74E+00	0.00	*
thursday.hr.open	-0.569900455	2.00E-01	-2.85E+00	0.00	*
Shopping	1.334351157	4.45E-01	3.00E+00	0.00	*
polarity	1.371076098	5.67E-01	2.42E+00	0.02	*
word.count	-0.004498566	1.98E-03	-2.27E+00	0.02	*
cityMadison	1.511769985	6.93E-01	2.18E+00	0.03	*
wednesday.hr.open	0.581944202	2.62E-01	2.22E+00	0.03	*
cityMontrÃ©al	1.343711658	6.60E-01	2.04E+00	0.04	*
\`Beauty & Spas\`	1.548012015	7.45E-01	2.08E+00	0.04	*
Desserts	-1.014926191	5.03E-01	-2.02E+00	0.04	*
Automotive	-1.017814796	5.08E-01	-2.00E+00	0.05	*
sunday.hr.open	0.11510379	6.52E-02	1.77E+00	0.08	
cityPittsburgh	0.970137977	5.99E-01	1.62E+00	0.11	
Casinos	0.833297731	5.34E-01	1.56E+00	0.12	
saturday.hr.open	-0.112037239	7.58E-02	-1.48E+00	0.14	
Bakeries	0.962630365	7.85E-01	1.23E+00	0.22	
cityLas Vegas	0.289945841	2.54E-01	1.14E+00	0.25	
\`Wine & Spirits\`	0.937195141	8.16E-01	1.15E+00	0.25	
monday.hr.open	0.090352151	1.02E-01	8.84E-01	0.38	
tuesday.hr.open	-0.150969104	1.78E-01	-8.46E-01	0.4	
cityMarkham	0.763986621	9.28E-01	8.23E-01	0.41	
cityChandler	0.493402881	7.06E-01	6.99E-01	0.48	
review_count	-0.000233109	4.36E-04	-5.34E-01	0.59	
cityGilbert	-0.398771669	7.75E-01	-5.15E-01	0.61	
checkins	-0.128514033	2.54E-01	-5.06E-01	0.61	
Buffets	-0.373391087	7.45E-01	-5.01E-01	0.62	

cityMesa	0.358790357	7.49E-01	4.79E-01	0.63	
cityCleveland	0.340831792	7.52E-01	4.54E-01	0.65	
`Event Planning & Services`	-0.188462696	4.21E-01	-4.48E-01	0.65	
Nightlife	0.104125274	2.80E-01	3.72E-01	0.71	
cityCharlotte	0.202301319	5.63E-01	3.59E-01	0.72	
cityHenderson	0.276058411	8.08E-01	3.42E-01	0.73	
`Home Services`	0.250446636	7.27E-01	3.44E-01	0.73	
`Coffee & Tea`	0.158070347	4.82E-01	3.28E-01	0.74	
`Fast Food`	0.225375044	6.84E-01	3.30E-01	0.74	
cityTempe	-0.170103415	5.86E-01	-2.90E-01	0.77	
Restaurants	0.085377481	2.89E-01	2.96E-01	0.77	
`Active Life`	0.201188832	6.85E-01	2.94E-01	0.77	
cityPhoenix	0.089746352	3.36E-01	2.67E-01	0.79	
cityEdinburgh	0.161438787	6.60E-01	2.44E-01	0.81	
Fashion	0.089429747	4.23E-01	2.11E-01	0.83	
cityScottsdale	-0.098143577	4.96E-01	-1.98E-01	0.84	
is_open	-0.050607872	4.02E-01	-1.26E-01	0.9	
friday.hr.open	0.008166812	7.76E-02	1.05E-01	0.92	
`Arts & Entertainment`	0.044581803	5.33E-01	8.36E-02	0.93	
`Hotels & Travel`	0.039042079	4.68E-01	8.34E-02	0.93	
`Local Services`	-0.001032196	6.51E-01	-1.59E-03	1	

Table 1 - Dependent variable: star rating of individual reviews

Variables	Value	Std.Error	t value	p value	sig
stars.x3	1.82E+00	0.429728189	4.23492567	0.00	*
stars.x4	1.39E+00	0.388369953	3.56984289	0.00	*

stars.x5	1.75E+00	0.378846358	4.62923148	0.00	*
cityAurora	1.35E+00	0.268832168	5.02167467	0.00	*
cityGilbert	2.67E+00	0.691005593	3.8610022	0.00	*
cityMarkham	1.34E+00	0.41358364	3.24830961	0.00	*
cityNew Kensington	1.36E+00	0.129186726	10.50656878	0.00	*
cityNorth Las Vegas	1.51E+00	0.322973921	4.68593376	0.00	*
cityNorth York	-2.20E+00	0.094829256	-23.1785671	0.00	*
cityPeoria	3.66E+00	0.281313098	13.00935296	0.00	*
citySurprise	-1.42E+00	0.08988269	-15.75055903	0.00	*
sunday.hr.open	1.78E-01	0.06155951	2.8899106	0.00	*
Shopping	-1.45E+00	0.459023754	-3.1505366	0.00	*
Automotive	-2.05E+00	0.305529257	-6.71369336	0.00	*
is_open	-9.67E-01	0.387645101	-2.49454876	0.01	*
cityMississauga	6.26E-01	0.27022272	2.31484721	0.02	*
word.count	4.35E-03	0.001803719	2.41149906	0.02	*
`Hotels & Travel`	-1.13E+00	0.470023021	-2.41324376	0.02	*
stars.x2	1.04E+00	0.471376899	2.20925377	0.03	*
Fashion	9.42E-01	0.434235964	2.17045801	0.03	*
cityChandler	-1.11E+00	0.595739643	-1.8669691	0.06	
friday.hr.open	1.09E-01	0.058879351	1.8517017	0.06	
saturday.hr.open	-1.23E-01	0.064957989	-1.89209744	0.06	
cityCleveland	-1.01E+00	0.586518706	-1.72806956	0.08	
monday.hr.open	-1.54E-01	0.09070915	-1.69599275	0.09	
cityMadison	-1.11E+00	0.671445755	-1.65402501	0.1	
thursday.hr.open	-2.69E-01	0.165486559	-1.62755311	0.1	

`Home Services`	-1.12E+00	0.688767658	-1.62154023	0.1	
`Local Services`	-9.84E-01	0.660209475	-1.49073337	0.14	
`Beauty & Spas`	-7.57E-01	0.580363551	-1.30435324	0.19	
wednesday.hr.open	2.50E-01	0.209469107	1.19462384	0.23	
`Arts & Entertainment`	-6.07E-01	0.505984263	-1.19946455	0.23	
cityConcord	-5.31E-01	0.448774914	-1.18275862	0.24	
`Event Planning & Services`	4.43E-01	0.382379298	1.15940003	0.25	
Restaurants	3.12E-01	0.284081417	1.09918402	0.27	
`Fast Food`	-6.48E-01	0.60886836	-1.06362339	0.29	
cityMesa	6.70E-01	0.661624331	1.01237649	0.31	
cityPhoenix	2.25E-01	0.351144257	0.63980501	0.52	
Bakeries	-4.34E-01	0.675848719	-0.64283598	0.52	
checkins	-1.29E-01	0.214734427	-0.60173606	0.55	
cityGoodyear	-4.67E-02	0.083141779	-0.56132257	0.57	
Nightlife	1.25E-01	0.266704164	0.46819189	0.64	
tuesday.hr.open	-5.11E-02	0.117661169	-0.43416947	0.66	
cityScottsdale	1.98E-01	0.468690373	0.42162257	0.67	
cityMontrÃ©al	-2.25E-01	0.614865806	-0.36624735	0.71	
`Wine & Spirits`	2.13E-01	0.594527698	0.35832678	0.72	
Casinos	1.42E-01	0.481548991	0.29570634	0.77	
cityTempe	-1.41E-01	0.536311515	-0.26382554	0.79	
`Active Life`	1.43E-01	0.545469897	0.26150195	0.79	
cityGlendale	9.35E-02	0.368373608	0.25391406	0.8	
Buffets	1.79E-01	0.735709321	0.24305227	0.81	
cityLas Vegas	-5.31E-02	0.24062987	-0.220771	0.83	

cityPittsburgh	1.10E-01	0.496920333	0.22046463	0.83	
\`Coffee & Tea\`	9.97E-02	0.465379661	0.21419006	0.83	
cityCharlotte	7.07E-02	0.48376921	0.1462201	0.88	
review_count	5.71E-05	0.000394127	0.14493625	0.88	
Desserts	-5.77E-02	0.4747386	-0.12162596	0.9	
cityHenderson	-7.37E-02	0.632405798	-0.11658985	0.91	
cityEdinburgh	3.01E-02	0.680235588	0.04428689	0.96	

Table 2 - Dependent variable: sentiment score of review

Appendix 2 - Full Code

```
 ### S D S S  E X A M I N A T I O N
######################################################################################
# Pierre Michel B. Hardy
# I6117019

### P A C K A G E S
#############################################################################################
########
library(tm)
library(SnowballC)
library(dplyr)
library(sentimentr)
library(reshape2)
library(fastDummies)
library(tidyr)
library(MASS)
library(imager)
library(colordistance)
library(caret)
library(e1071)
library(xgboost)
library(MLmetrics)
library(DMwR)

### D A T A
#############################################################################################
################
load("Reviews_5.RData")
load("Business information.RData")
load("Labels_test_set_for_students.RData")
load("Labels_photos_5.RData")
```

Part One

Question A

```
### P A R T  O N E
#############################################################################################
#########

### QUESTION A ###
```

```
# extract the first review

  reviews_sample$text[1]
```

Question B

```
### QUESTION B ###

# Preprocessing the review data before analysis.

# Remove the punctuation and keeping the first review in each step so we can see changes.

  reviews_sample$Clean_Text <- removePunctuation(reviews_sample$text)
  cleaning.steps.1 <- reviews_sample$Clean_Text[1]

# Remove the numbers

  reviews_sample$Clean_Text <- removeNumbers(reviews_sample$Clean_Text)
  cleaning.steps.2 <- reviews_sample$Clean_Text[1]

# Lowercase everything

  reviews_sample$Clean_Text <- tolower(reviews_sample$Clean_Text)
  cleaning.steps.3 <- reviews_sample$Clean_Text[1]

# Remove whitespaces

  reviews_sample$Clean_Text <- stripWhitespace(reviews_sample$Clean_Text)
  cleaning.steps.4 <- reviews_sample$Clean_Text[1]

# Remove common words

  reviews_sample$Clean_Text <- removeWords(reviews_sample$Clean_Text, stopwords("English"))
  cleaning.steps.5 <- reviews_sample$Clean_Text[1]

# Remove special characters

  reviews_sample$Clean_Text <- gsub("[:punct:]", "", reviews_sample$Clean_Text)
  cleaning.steps.6 <- reviews_sample$Clean_Text[1]

# Stem the word

  reviews_sample$Clean_Text <- stemDocument(reviews_sample$Clean_Text)
  cleaning.steps.7 <- reviews_sample$Clean_Text[1]
```

Question C

```
### QUESTION C ###

# First, we transform it into a corpus and create a document term matrix

  review.corpus <- Corpus(VectorSource(reviews_sample$Clean_Text))
  review.corpus.tf <- DocumentTermMatrix(review.corpus)
  inspect(review.corpus.tf[1:10,1:10])

# We create an TF-IDF matrix

  review.corpus.tfidf <- weightTfIdf(review.corpus.tf)
  inspect(review.corpus.tfidf[1:10,1:10])

# In order for me to interpret the output better, I decided to transform it into a matrix
since I am more acquainted
# with it.

  tfidf.matrix <- as.matrix(review.corpus.tfidf)

# Take the average per column (per term) in order quantify which terms are most important

  ave.term <- colMeans(tfidf.matrix)
```

```
# Take the top ten and bottom ten terms

  ave.term.most <- order(ave.term, decreasing = TRUE)
  ave.term.least <- order(ave.term, decreasing = FALSE)
  ave.term[ave.term.most[1:10]]
  Ave.term[ave.term.least[1:36]]
```

Question D

```
### QUESTION D ###

# Calculate the sentiment score of each review

  reviews_sample$Sentiment <- sentiment(reviews_sample$Clean_Text)
```

Question E

```
### QUESTION E ###

# Create a scatterplot between the star ratings and polarity scores

  plot(reviews_sample$stars, reviews_sample$Sentiment$sentiment, main="Star Reviews vs
Polarity Score", xlab="Stars",
       ylab="Polarity")

# Fit a regression line to clearly see the relationship

  reg.line <- lm(reviews_sample$Sentiment$sentiment~reviews_sample$stars)
  abline(reg.line)
  abline(h=0, col="red")

# Let's see if the line is significant (it is)
  summary(reg.line)
```

Question F

```
### QUESTION F ###

# Let's join the reviews and business datasets

  combined <- left_join(reviews_sample, business, by="business_id")

# I will select the columns I find relevant for the analysis and would give informative
value

  combined <- dplyr::select(combined, c("stars.x", "date", "useful","funny", "cool",
"Sentiment", "city", "stars.y",
                                 "review_count", "is_open",
"categories","monday","tuesday","wednesday","thursday",
                              "friday","saturday","sunday", "checkins"))
  combined$polarity <- combined$Sentiment$sentiment
  combined$word.count <- combined$Sentiment$word_count
  combined <- combined[,-6]

# Next, I transform the dates of the review into seasons. I later re-read the question and
realized that only the
# firm level characteristics are asked. I will keep this part of the code anyways.

  combined$date <- substr(combined$date, 6, 7)
  combined$date <- as.numeric(combined$date)
  combined$date <- lapply(combined$date, function(x){
    if(x<3){
      x<-"Winter"
       } else if (x>2 && x < 6) {
      x <- "Spring"
       } else if (x>5 && x < 9){
      x <- "Summer"
       } else if (x > 8 && x < 12) {
```

```
      x <- "Autumn"
        } else {
      x <- "Winter"
    }
  })

# Next, I decided to separate the opening and closing times of the establishment per day

# monday
  combined$monday <- as.character(combined$monday)
  combined$monday.open <- substr(combined$monday, 1,2)
  combined$monday.close <-
substr(combined$monday,nchar(combined$monday)-3,nchar(combined$monday)-2)
  combined$monday.open <- gsub("[:punct:]", "", combined$monday.open)
  combined$monday.open <- gsub("No","", combined$monday.open)
  combined$monday.close <- gsub("[:punct:]", "", combined$monday.close)
  combined$monday.close <- gsub("No","", combined$monday.close)
  combined$monday.close <- gsub("-","",combined$monday.close)
  combined$monday.open <- as.numeric(combined$monday.open)
  combined$monday.close <- as.numeric(combined$monday.close)
  combined$monday.hr.open <- combined$monday.close-combined$monday.open
  combined$monday.hr.open <- lapply(combined$monday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
      x<- x+24
        } else {
      x <- x
        }
  })

# tuesday
  combined$tuesday <- as.character(combined$tuesday)
  combined$tuesday.open <- substr(combined$tuesday, 1,2)
  combined$tuesday.close <-
substr(combined$tuesday,nchar(combined$tuesday)-3,nchar(combined$tuesday)-2)
  combined$tuesday.open <- gsub("[:punct:]", "", combined$tuesday.open)
  combined$tuesday.open <- gsub("No","", combined$tuesday.open)
  combined$tuesday.close <- gsub("[:punct:]", "", combined$tuesday.close)
  combined$tuesday.close <- gsub("No","", combined$tuesday.close)
  combined$tuesday.close <- gsub("-","",combined$tuesday.close)
  combined$tuesday.open <- as.numeric(combined$tuesday.open)
  combined$tuesday.close <- as.numeric(combined$tuesday.close)
  combined$tuesday.hr.open <- combined$tuesday.close-combined$tuesday.open
  combined$tuesday.hr.open <- lapply(combined$tuesday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
      x<- x+24
        } else {
      x <- x
        }
  })

# wednesday
  combined$wednesday <- as.character(combined$wednesday)
  combined$wednesday.open <- substr(combined$wednesday, 1,2)
  combined$wednesday.close <-
substr(combined$wednesday,nchar(combined$wednesday)-3,nchar(combined$wednesday)-2)
  combined$wednesday.open <- gsub("[:punct:]", "", combined$wednesday.open)
  combined$wednesday.open <- gsub("No","", combined$wednesday.open)
  combined$wednesday.close <- gsub("[:punct:]", "", combined$wednesday.close)
  combined$wednesday.close <- gsub("No","", combined$wednesday.close)
  combined$wednesday.close <- gsub("-","",combined$wednesday.close)
  combined$wednesday.open <- as.numeric(combined$wednesday.open)
  combined$wednesday.close <- as.numeric(combined$wednesday.close)
  combined$wednesday.hr.open <- combined$wednesday.close-combined$wednesday.open
  combined$wednesday.hr.open <- lapply(combined$wednesday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
      x<- x+24
        } else {
      x <- x
        }
  })
```

```
# thursday
  combined$thursday <- as.character(combined$thursday)
  combined$thursday.open <- substr(combined$thursday, 1,2)
  combined$thursday.close <-
substr(combined$thursday,nchar(combined$thursday)-3,nchar(combined$thursday)-2)
  combined$thursday.open <- gsub("[:punct:]", "", combined$thursday.open)
  combined$thursday.open <- gsub("No","", combined$thursday.open)
  combined$thursday.close <- gsub("[:punct:]", "", combined$thursday.close)
  combined$thursday.close <- gsub("No","", combined$thursday.close)
  combined$thursday.close <- gsub("-","",combined$thursday.close)
  combined$thursday.open <- as.numeric(combined$thursday.open)
  combined$thursday.close <- as.numeric(combined$thursday.close)
  combined$thursday.hr.open <- combined$thursday.close-combined$thursday.open
  combined$thursday.hr.open <- lapply(combined$thursday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
         x<- x+24
       } else {
      x <- x
       }
  })

# friday
  combined$friday <- as.character(combined$friday)
  combined$friday.open <- substr(combined$friday, 1,2)
  combined$friday.close <-
substr(combined$friday,nchar(combined$friday)-3,nchar(combined$friday)-2)
  combined$friday.open <- gsub("[:punct:]", "", combined$friday.open)
  combined$friday.open <- gsub("No","", combined$friday.open)
  combined$friday.close <- gsub("[:punct:]", "", combined$friday.close)
  combined$friday.close <- gsub("No","", combined$friday.close)
  combined$friday.close <- gsub("-","",combined$friday.close)
  combined$friday.open <- as.numeric(combined$friday.open)
  combined$friday.close <- as.numeric(combined$friday.close)
  combined$friday.hr.open <- combined$friday.close-combined$friday.open
  combined$friday.hr.open <- lapply(combined$friday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
      x<- x+24
       } else {
      x <- x
       }
  })

# saturday
  combined$saturday <- as.character(combined$saturday)
  combined$saturday.open <- substr(combined$saturday, 1,2)
  combined$saturday.close <-
substr(combined$saturday,nchar(combined$saturday)-3,nchar(combined$saturday)-2)
  combined$saturday.open <- gsub("[:punct:]", "", combined$saturday.open)
  combined$saturday.open <- gsub("No","", combined$saturday.open)
  combined$saturday.close <- gsub("[:punct:]", "", combined$saturday.close)
  combined$saturday.close <- gsub("No","", combined$saturday.close)
  combined$saturday.close <- gsub("-","",combined$saturday.close)
  combined$saturday.open <- as.numeric(combined$saturday.open)
  combined$saturday.close <- as.numeric(combined$saturday.close)
  combined$saturday.hr.open <- combined$saturday.close-combined$saturday.open
  combined$saturday.hr.open <- lapply(combined$saturday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
      x<- x+24
       } else {
      x <- x
       }
  })

# sunday
  combined$sunday <- as.character(combined$sunday)
  combined$sunday.open <- substr(combined$sunday, 1,2)
  combined$sunday.close <-
substr(combined$sunday,nchar(combined$sunday)-3,nchar(combined$sunday)-2)
  combined$sunday.open <- gsub("[:punct:]", "", combined$sunday.open)
  combined$sunday.open <- gsub("No","", combined$sunday.open)
  combined$sunday.close <- gsub("[:punct:]", "", combined$sunday.close)
  combined$sunday.close <- gsub("No","", combined$sunday.close)
  combined$sunday.close <- gsub("-","",combined$sunday.close)
  combined$sunday.open <- as.numeric(combined$sunday.open)
  combined$sunday.close <- as.numeric(combined$sunday.close)
  combined$sunday.hr.open <- combined$sunday.close-combined$sunday.open
  combined$sunday.hr.open <- lapply(combined$sunday.hr.open, function(x){
    if (is.na(x)==FALSE && x<0){
      x<- x+24
       } else {
      x <- x
       }
  })

# I will once again select only the columns I find useful

  combined <- dplyr::select(combined, c("stars.x", "date", "useful","funny", "cool",
"city", "stars.y",
                                 "review_count", "is_open",
"categories","monday.hr.open","tuesday.hr.open",

"wednesday.hr.open","thursday.hr.open","friday.hr.open","saturday.hr.open",
                                 "sunday.hr.open", "checkins", "polarity","word.count"))

# I decided to dummify the categories variable. I start by separating the multiple
categories

  combined <- separate_rows(combined, categories, sep=";")
  combined$categories.value <- 1

# After assigning 1 to all the corresponding categories, I will spread them out so that one
category is one column

  combined <- spread(combined, categories, categories.value, fill=0)

# Since there are 334 categories, I choose only those that are relevant. I decided to
investigate the categories
# that have an above average number of shops to it. Then I subjectively chose the high
level categories (e.g.
# "restaurants" only and excluding other cuisines). I chose 20 relevant categories from the
334.

  categs <- dplyr::select(combined, 20:ncol(combined))
  categs.sum <- colSums(categs)
  categs.ordered <- order(categs.sum, decreasing = TRUE)
  categs.sum[categs.ordered[1:67]]
  combined <- dplyr::select(combined, c("stars.x", "date", "useful","funny", "cool",
"city", "stars.y",
                        "review_count", "is_open","monday.hr.open","tuesday.hr.open",
"wednesday.hr.open",

"thursday.hr.open","friday.hr.open","saturday.hr.open","sunday.hr.open", "checkins",
                         "polarity","word.count","Restaurants", "Nightlife","Event Planning
& Services",
                      "Arts & Entertainment","Shopping","Coffee & Tea","Beauty &
Spas","Desserts","Active Life",
                         "Bakeries","Local Services","Home Services","Fast
Food","Casinos","Automotive",
                      "Wine & Spirits","Hotels & Travel","Fashion","Buffets","Health &
Medical"))
```

Question G

```
### QUESTION G ###

# turn review star rating into factor and hours open into numeric

  combined$stars.x <- as.factor(combined$stars.x)
  combined$monday.hr.open <- as.numeric(combined$monday.hr.open)
  combined$tuesday.hr.open <- as.numeric(combined$tuesday.hr.open)
  combined$wednesday.hr.open <- as.numeric(combined$wednesday.hr.open)
  combined$thursday.hr.open <- as.numeric(combined$thursday.hr.open)
  combined$friday.hr.open <- as.numeric(combined$friday.hr.open)
  combined$saturday.hr.open <- as.numeric(combined$saturday.hr.open)
  combined$sunday.hr.open <- as.numeric(combined$sunday.hr.open)

# in case I wish to try different combinations, I create another data frame with my chosen
columns.

  reg1 <- dplyr::select(combined, c("stars.x", "city",
                             "review_count", "is_open","monday.hr.open","tuesday.hr.open",
"wednesday.hr.open",

"thursday.hr.open","friday.hr.open","saturday.hr.open","sunday.hr.open", "checkins",
                             "polarity", "Restaurants", "Nightlife","Event Planning &
Services", "word.count",
                              "Arts & Entertainment","Shopping","Coffee & Tea","Beauty &
Spas","Desserts","Active Life",
                             "Bakeries","Local Services","Home Services","Fast
Food","Casinos","Automotive",
                             "Wine & Spirits","Hotels & Travel","Fashion","Buffets","Health
& Medical"))

# scale checkins

  mean.checkin <- mean(reg1$checkins)
  sd.checkin <- sd(reg1$checkins)
  reg1$checkins <- (reg1$checkins-mean.checkin)/sd.checkin

# estimate the regression

  combined.reg <- polr(stars.x ~., data = reg1, method="logistic", Hess = TRUE)

# error: attempt to find suitable starting values failed

  summary(reg1$city)
  reg1 <- reg1[reg1$city %in% reg1$city[duplicated(reg1$city)],]

# retry

  combined.reg <- polr(stars.x ~., data = reg1, method="logistic", Hess = TRUE)
  summary(combined.reg)
  combined.reg$coefficients

# calculate for significance

  coef.table.star <- coef(summary(combined.reg))
  p.value <- pnorm(abs(coef.table.star[, "t value"]),lower.tail = FALSE)* 2
  coef.table.star <- cbind(coef.table.star, "p value" = round(p.value,2))
  coef.table.star <- as.data.frame(coef.table.star)
  coef.table.star$sig <- coef.table.star$`p value`
  coef.table.star$sig <- lapply(coef.table.star$sig, function(x){
    if(x<=0.05){
      coef.table.star$sig<-"*"
       } else {
      coef.table.star$sig <- ""
       }
  })
```

Question H

```
### QUESTION H ###

# I recreate another dataframe as with the previous question

  reg2 <- dplyr::select(combined, c("stars.x", "city",
                                    "review_count",
"is_open","monday.hr.open","tuesday.hr.open", "wednesday.hr.open",

"thursday.hr.open","friday.hr.open","saturday.hr.open","sunday.hr.open", "checkins",
                                  "polarity", "Restaurants", "Nightlife","Event Planning &
Services", "word.count",
                                    "Arts & Entertainment","Shopping","Coffee &
Tea","Beauty & Spas","Desserts",
                                    "Active Life", "Bakeries","Local Services","Home
Services","Fast Food","Casinos",
                                    "Automotive", "Wine & Spirits","Hotels &
Travel","Fashion","Buffets",
                                 "Health & Medical"))
  reg2$polarity <- as.factor(reg2$polarity)

# scale checkins

  mean.checkin2 <- mean(reg2$checkins)
  sd.checkin2 <- sd(reg2$checkins)
  reg2$checkins <- (reg2$checkins-mean.checkin2)/sd.checkin2

# remove variables with only one sample to avoid the error encountered earlier

  reg2 <- reg2[reg2$city %in% reg2$city[duplicated(reg2$city)],]

# estimate the model and extract the coefficients

  combined.reg2 <- polr(polarity ~., data = reg2, method="logistic", Hess = TRUE)
  summary(combined.reg2)
  combined.reg2$coefficients

# calculate for significance

  coef.table.polar <- coef(summary(combined.reg2))
  p.value <- pnorm(abs(coef.table.polar[, "t value"]),lower.tail = FALSE)* 2
  coef.table.polar <- cbind(coef.table.polar, "p value" = round(p.value,2))
  coef.table.polar <- as.data.frame(coef.table.polar)
  coef.table.polar$sig <- coef.table.polar$`p value`
  coef.table.polar$sig <- lapply(coef.table.polar$sig, function(x){
    if(x<=0.05){
      coef.table.polar$sig<-"*"
       } else {
      coef.table.polar$sig <- ""
       }
  })
```

Part Two

```
### P A R T  T W O
############################################################################################
########

# Due to the the following questions' computationally intensive nature, the Global
Environment will be cleared

  remove(list=ls())

# Reload the required datasets

  load("Labels_test_set_for_students.RData")
  load("Labels_photos_5.RData")
```

Question A

```
### QUESTION A ###

# load the images
# first, we specify which destination folder

  test.images.folder <- "C:/Users/pierr/OneDrive/Documents/Maastricht
University/Masters/SDSS/Exam/stud_5/stud_5"

# second, we list the names of each file/image
```

```
  test.images.name <- list.files(test.images.folder)

# these files are being removed since the algorithm identifies them as png files masked as
jpegs

  test.images.name <- test.images.name[-23]
  test.images.name <- test.images.name[-67]

# third, we create a list of path names include the image name

  test.images.fnames <- paste(test.images.folder, "/",
test.images.name[1:length(test.images.name)],sep="")

# To save on RAM space, we won't load all the images into RStudio
```

Question B

```
### QUESTION B ###

# In this analysis, we will extract features capturing the color model and lines of an
image
# We start by capturing the number of important lines in an image
# This part is computationally intensive and is divided into four chunks.
# This results to inefficient for-loops that, while not elegant, gets the job done.
# RAM is cleared a bit in between chunks

# isolate the name of images as label

  image.names <- substr(test.images.name, 1, 22)

# declare a dataframe to collect the lines

  n.n.line <- as.data.frame(0)

# first chunk

  # each chunk takes the lines for 100 images

  for (i in 1:100){
       # load the image

      test.images <- load.image(test.images.fnames[i])

       # detect the edges

      test.image.lines <- cannyEdges(test.images)

       # from the edges, get the lines

      lines <- hough_line(test.images, ntheta = 800, data.frame=TRUE)

       # we want only the important lines based on the score

      quant <- quantile(lines$score, probs = seq(0.995,1,0.05))

       # filter out the important lines

      lines <- filter(lines, lines$score>=quant)

       # count how many important lines there are

      n.lines <- nrow(lines)

       # place this number in a dataframe alongside the name of the image

      n.n.line [i,1] <- image.names[i]
      n.n.line [i,2] <- n.lines
  }
```

```
# We remove some heavy variables in the Global Environment between chunks to free up RAM
  rm(test.images)

# second chunk. The process is the same as the first chunk.

  for (i in 101:200){
    test.images <- load.image(test.images.fnames[i])
    test.image.lines <- cannyEdges(test.images)
    lines <- hough_line(test.images, ntheta = 800, data.frame=TRUE)
    quant <- quantile(lines$score, probs = seq(0.995,1,0.05))
    lines <- filter(lines, lines$score>=quant)
    n.lines <- nrow(lines)
    n.n.line [i,1] <- image.names[i]
    n.n.line [i,2] <- n.lines
  }

# clear some space

  rm(test.images)
  rm(lines)

# third chunk

  for (i in 201:300){
    test.images <- load.image(test.images.fnames[i])
    test.image.lines <- cannyEdges(test.images)
    lines <- hough_line(test.images, ntheta = 800, data.frame=TRUE)
    quant <- quantile(lines$score, probs = seq(0.995,1,0.05))
    lines <- filter(lines, lines$score>=quant)
    n.lines <- nrow(lines)
    n.n.line [i,1] <- image.names[i]
    n.n.line [i,2] <- n.lines
  }

# clear some space
  rm(test.images)
  rm(lines)

# we remove these three images. It causes the fourth chunk to crash RStudio for some reason

  test.images.fnames <- test.images.fnames[-304]
  test.images.fnames <- test.images.fnames[-337]
  test.images.fnames <- test.images.fnames[-336]

# fourth and final chunk

  for (i in 301:length(test.images.fnames)){
    test.images <- load.image(test.images.fnames[i])
    test.image.lines <- cannyEdges(test.images)
    lines <- hough_line(test.images, ntheta = 800, data.frame=TRUE)
    quant <- quantile(lines$score, probs = seq(0.995,1,0.05))
    lines <- filter(lines, lines$score>=quant)
    n.lines <- nrow(lines)
    n.n.line [i,1] <- image.names[i]
    n.n.line [i,2] <- n.lines
  }

# in order to save time and not having to repeat this process, I will save it as a csv

  write.csv(n.n.line, file="lines.csv")

# Second feature I wish to extract is the color histogram containing the RGB

# extract the names of the image

  rgb.mat.names <- substr(test.images.name, 1, 22)

# declare the matrix to put the values of the color histogram

  rgb.mat <- matrix(0, ncol = 81, nrow = length(rgb.mat.names))
```

```
# create a loop to collect the rgb info per image

  for (i in 1:length(test.images.fnames)){

       # get the actual image histogram

      x <- getImageHist(test.images.fnames[i], bins = 3, hsv = FALSE, plotting=FALSE)

       # we get the columns containing the quantified info of R, G, & B.

      r <- x$r
      g <- x$g
      b <- x$b

       # We append them all into one row

      a <- append(r, g)
      a <- append(a, b)
      rgb.mat[i,] <- a
  }

# We combine the rgb information with the image name

  rgb.mat <- cbind(rgb.mat, rgb.mat.names)

# Third features I wish to extract is the color histogram containing the HSV
# The process is exactly the same as the RGB above except that hsv = TRUE instead of FALSE

  hsv.mat.names <- substr(test.images.name, 1, 22)
  length(hsv.mat.names)
  hsv.mat <- matrix(0, ncol = 81, nrow = length(hsv.mat.names))
  for (i in 1:length(test.images.fnames)){
       x <- getImageHist(test.images.fnames[i], bins = 3, hsv = TRUE, plotting=FALSE)
       r <- x$h
       g <- x$s
       b <- x$v
       a <- append(r, g)
       a <- append(a, b)
    hsv.mat[i,] <- a
  }
  hsv.mat <- cbind(hsv.mat, hsv.mat.names)

# Rename the columns of the rgb and hsv files

  r.names <- c(paste("r",1:27,sep=""))
  g.names <- c(paste("g",1:27,sep=""))
  b.names <- c(paste("b",1:27, sep=""))
  h.names <- c(paste("h",1:27,sep=""))
  s.names <- c(paste("s",1:27,sep=""))
  v.names <- c(paste("v",1:27,sep=""))
  colnames(rgb.mat) <- c(r.names,g.names,b.names,"photo_id")
  colnames(hsv.mat) <- c(h.names, s.names, v.names,"photo_id")
  colnames(n.n.line) <- c("photo_id","n.lines")

# turn them into data frames

  rgb.mat <- as.data.frame(rgb.mat)
  hsv.mat <- as.data.frame(hsv.mat)

# build the main data frame. I join the three data.frames with the labels to create the
dataset ready for training

  image.main <- left_join(n.n.line, photo_labels, by="photo_id")
  image.main <- left_join(image.main, rgb.mat, by="photo_id")
  image.main <- left_join(image.main, hsv.mat, by="photo_id")

# to save on time, I create a copy and save it as csv

  write.csv(image.main, file="image_main.csv")
```

Question D

```
### QUESTION D ###

# I set a seed so i can reproduce the results

  set.seed(14)

# declare the target attribute as a factor

  image.main$label <- as.factor(image.main$label)

# i create a partition of 80-20

  part <- createDataPartition(y=image.main$label, p=0.8, list=FALSE)

# I remove some unnecessary columns

  image.main2 <- image.main[-1]
  image.main2 <- image.main2[-3]
  image.main2 <- image.main2[-2]

# I create the partition: 80% training set and 20% testing set

  train.set2 <- image.main2[part,]
  test.set <- image.main2[-part,]
```

Question E

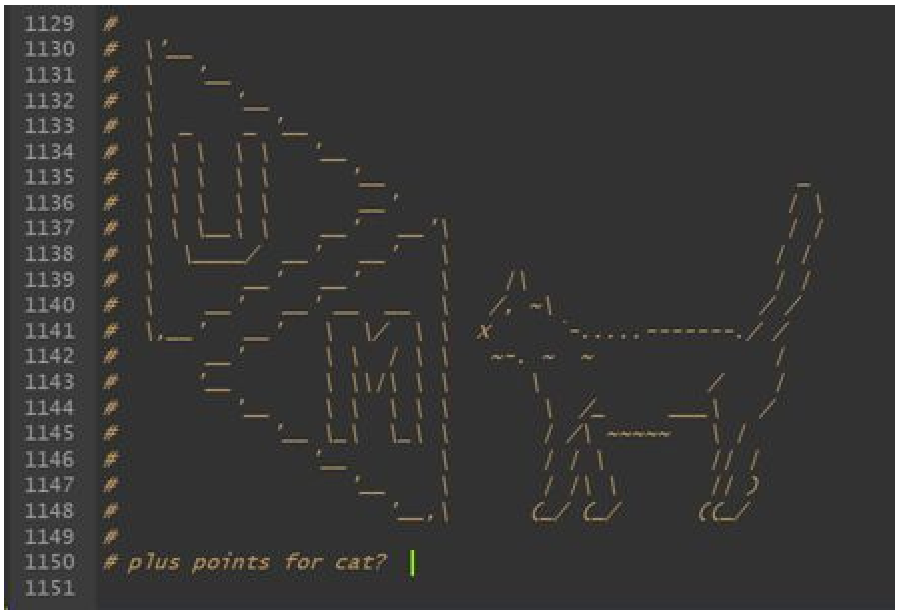
```
### QUESTION E ###

# I calculate the evaluation metrics of a baseline model
# The baseline model is is just predicting that everything "food," which is the majority
attribute

# In-Sample
# Create a dataframe with the true value and the "prediction"

  base.in <- train.set2[2]
  base.in$pred <- "food"

# compute the accuracy

  base.in.acc <- Accuracy(y_pred = base.in$pred, y_true = base.in$label)
  base.in.acc

# comput the recall

  base.in.recall <-Recall(y_pred = base.in$pred, y_true = base.in$label)
  base.in.recall

# compute the f1 score
  base.in.f1 <- F1_Score(y_true = base.in$label, y_pred = base.in$pred) #  error

# out of sample
# the process is similar to above

  base.out <- test.set[2]
  base.out$pred <- "food"
  base.out.acc <- Accuracy(y_pred = base.out$pred, y_true = base.out$label)
  base.out.acc
  base.out.recall <-Recall(y_pred = base.out$pred, y_true = base.out$label)
  base.out.recall
  base.out.f1 <- F1_Score(y_true = base.out$label, y_pred = base.out$pred)
  base.out.f1

# We train the first model: gradient boosted regression trees using the xgboost function
```

```
  train.set <- train.set2

# xgboost requires a separate dataframe for the labels

  labels <- train.set$label

# xgboost also requires the labels to be numeric

  labels <- as.numeric(labels)

# remove the labels from the train.set

  train.set <- train.set[-grep('label', colnames(train.set))]

# actually train the xgboost model. nrounds and eta is put there to prevent overfitting.
seed to make it reproducible
# subsample is retained as one to use what limited data is at hand.

  xgb <- xgboost(data=data.matrix(train.set), label=labels, nrounds=25, eta=0.1,
seed=14,subsample=1)

# a bit of insight to the resulting model. Highlights top 10 important variables

  model <- xgb.dump(xgb, with.state=T)
  model[1:10]
  names <- dimnames(data.matrix(train.set))[[2]]
  imp.mat <- xgb.importance(names, model = xgb)
  xgb.plot.importance(imp.mat[1:10,])

# evaluation of the xgboost model
# in-sample

# predict the values

  xg.pred.in <- predict(xgb, data.matrix(train.set))
  xg.pred.in <- round(xg.pred.in)

# create a dataframe with the true values and predicted values

  xg.in <- train.set2
  xg.in <- xg.in[2]
  xg.in$pred <- xg.pred.in

# transform the predicted values from numeric into its original form

  xg.in$pred <- lapply(xg.in$pred, function(x){
  if(x==1){
    x<-"drink"
  } else if (x==2) {
       x <- "food"
  } else if (x==3){
       x <- "inside"
  } else {
       x <- "outside"
  }
  })

# calculate evaluation metrics: accuracy, recall, and f1-score

  xg.in$pred <- unlist(xg.in$pred)
  xg.in.acc <- Accuracy(y_pred = xg.in$pred, y_true = xg.in$label)
  xg.in.acc
  xg.in.recall <-Recall(y_pred = xg.in$pred, y_true = xg.in$label)
  xg.in.recall
  xg.in.f1 <- F1_Score(y_true = xg.in$label, y_pred = xg.in$pred)
  xg.in.f1

# out of sample
# the entire process is similar to the insample evaluation

  xg.pred.out <- predict (xgb, data.matrix(test.set[,-2]))
  xg.pred.out <- round(xg.pred.out)
  xg.out <- test.set
  xg.out <- xg.out[2]
  xg.out$pred <- xg.pred.out
  xg.out$pred <- lapply(xg.out$pred, function(x){
  if(x==1){
    x<-"drink"
  } else if (x==2) {
      x <- "food"
  } else if (x==3){
      x <- "inside"
  } else {
      x <- "outside"
  }
  })
  xg.out$pred <- unlist(xg.out$pred)
  xg.out.acc <- Accuracy(y_pred = xg.out$pred, y_true = xg.out$label)
  xg.out.acc
  xg.out.recall <-Recall(y_pred = xg.out$pred, y_true = xg.out$label)
  xg.out.recall
  xg.out.f1 <- F1_Score(y_true = xg.out$label, y_pred = xg.out$pred)
  xg.out.f1

# Next, we train the second model: SVM

# Train the model

  svm.model <- svm(label~., data=train.set2)

# evaluation of the svm model
# the whole process is similar to the evaluation of the xgboost model from lines 763-826
# in-sample

  svm.in.pred <- predict(svm.model, train.set2)
  svm.in <- train.set2
  svm.in <- svm.in[2]
  svm.in$pred <- svm.in.pred
  svm.in.acc <- Accuracy(y_pred = svm.in$pred, y_true = svm.in$label)
  svm.in.acc
  svm.in.recall <-Recall(y_pred = svm.in$pred, y_true = svm.in$label)
  svm.in.recall
  svm.in.f1 <- F1_Score(y_true = svm.in$label, y_pred = svm.in$pred)
  svm.in.f1

# out sample

  svm.out.pred <- predict(svm.model, test.set)
  svm.out <- test.set
  svm.out <- svm.out[2]
  svm.out$pred <- svm.out.pred
  svm.out.acc <- Accuracy(y_pred = svm.out$pred, y_true = svm.out$label)
  svm.out.acc
  svm.out.recall <-Recall(y_pred = svm.out$pred, y_true = svm.out$label)
  svm.out.recall
  svm.out.f1 <- F1_Score(y_true = svm.out$label, y_pred = svm.out$pred)
  svm.out.f1

# Due to disappointing results suspected to be caused by the unbalanced dataset, we will
attempt oversampling

# calculate proportion table of the original training data set

  prop.table(table(train.set2$label))

# create datasets that compensates for the underrepresented classes using SMOTE
# we also calculate the resulting dataset's proportion table

  smote.svm <- SMOTE(label~., data=train.set2, perc.over=300, seed=14)
  draw.one <- smote.svm
  prop.table(table(smote.svm$label))
  smote.svm <- SMOTE(label~., data=smote.svm, perc.over=200, seed=14)
  draw.two <- smote.svm
  prop.table(table(smote.svm$label))
  smote.svm <- SMOTE(label~., data=smote.svm,perc.over=100, seed=14)
  draw.three <- smote.svm
  prop.table(table(smote.svm$label))
  smote.svm <- SMOTE(label~., data=smote.svm,perc.over=50, seed=14)
  draw.four <- smote.svm
  prop.table(table(smote.svm$label))

# combine all the datasets made by SMOTE and the original dataset

  draw2 <- bind_rows(train.set2, draw.one)
  draw2 <- bind_rows(draw2, draw.two)
  draw2 <- bind_rows(draw2, draw.three)
  draw2 <- bind_rows(draw2, draw.four)
  train.set2 <- draw2

# the proportion table shows that the dataset is more balanced now

  prop.table(table(train.set2$label))

# for a less time consuming analysis, i have saved the balanced dataset into a csv

draw2 <- read.csv("balanced data.csv")

# prepared the balanced dataset to be the new training set

  draw2 <- draw2[-1]
  train.set2 <- draw2

# train the xgboost on the balanced dataset.
# This part is similar to the earlier training of the xgboost from line 750

  train.set <- train.set2
  labels <- train.set$label
  labels <- as.numeric(labels)
  train.set <- train.set[-grep('label', colnames(train.set))]
  xgb.bal <- xgboost(data=data.matrix(train.set), label=labels, nrounds=25, eta=0.1,
seed=14,subsample=1)
  model <- xgb.dump(xgb.bal, with.state=T)
  model[1:10]
  names <- dimnames(data.matrix(train.set))[[2]]
  imp.mat <- xgb.importance(names, model = xgb)
  xgb.plot.importance(imp.mat[1:10,])

# evaluate the new xgboost
# this part is similar to the earlier evaluation of a xgboost from line 763

# in sample

  xg.pred.in <- predict(xgb.bal, data.matrix(train.set))
  xg.pred.in <- round(xg.pred.in)
  xg.in <- train.set2
  xg.in <- xg.in[2]
  xg.in$pred <- xg.pred.in
  xg.in$pred <- lapply(xg.in$pred, function(x){
  if(x==1){
    x<-"drink"
  } else if (x==2) {
       x <- "food"
  } else if (x==3){
       x <- "inside"
  } else {
       x <- "outside"
  }
  })
  xg.in$pred <- unlist(xg.in$pred)
  xg.in.acc <- Accuracy(y_pred = xg.in$pred, y_true = xg.in$label)
  xg.in.acc
  xg.in.recall <-Recall(y_pred = xg.in$pred, y_true = xg.in$label)
  xg.in.recall
  xg.in.f1 <- F1_Score(y_true = xg.in$label, y_pred = xg.in$pred)
  xg.in.f1

# out of sample

  xg.pred.out <- predict (xgb.bal, data.matrix(test.set[,-2]))
  xg.pred.out <- round(xg.pred.out)
  xg.out <- test.set
  xg.out <- xg.out[2]
  xg.out$pred <- xg.pred.out
  xg.out$pred <- lapply(xg.out$pred, function(x){
  if(x==1){
    x<-"drink"
  } else if (x==2) {
       x <- "food"
  } else if (x==3){
       x <- "inside"
  } else {
       x <- "outside"
  }
  })
  xg.out$pred <- unlist(xg.out$pred)
  xg.out.acc <- Accuracy(y_pred = xg.out$pred, y_true = xg.out$label)
  xg.out.acc
  xg.out.recall <-Recall(y_pred = xg.out$pred, y_true = xg.out$label)
  xg.out.recall
  xg.out.f1 <- F1_Score(y_true = xg.out$label, y_pred = xg.out$pred)
  xg.out.f1


# re-train the svm model on the balanced dataset
# this process, even the evaluation, is similar to the earlier svm training from line 830

  svm.model <- svm(label~., data=train.set2)

# in sample

  svm.in.pred <- predict(svm.model, train.set2)
  svm.in <- train.set2
  svm.in <- svm.in[2]
  svm.in$pred <- svm.in.pred
  svm.in.acc <- Accuracy(y_pred = svm.in$pred, y_true = svm.in$label)
  svm.in.acc
  svm.in.recall <-Recall(y_pred = svm.in$pred, y_true = svm.in$label)
  svm.in.recall
  svm.in.f1 <- F1_Score(y_true = svm.in$label, y_pred = svm.in$pred)
  svm.in.f1

# out of sample

  svm.out.pred <- predict(svm.model, test.set)
  svm.out <- test.set
  svm.out <- svm.out[2]
  svm.out$pred <- svm.out.pred
  svm.out.acc <- Accuracy(y_pred = svm.out$pred, y_true = svm.out$label)
  svm.out.acc
  svm.out.recall <-Recall(y_pred = svm.out$pred, y_true = svm.out$label)
  svm.out.recall
  svm.out.f1 <- F1_Score(y_true = svm.out$label, y_pred = svm.out$pred)
  Svm.out.f1
```

Question G

```
### QUESTION G ###

# Based on the results, the chosen model to predict the 100 test images is the xgboost
training on the unbalanced
# dataset
# The 100 teest images will now undergo the same feature extraction earlier. For a more
in-depth detailing of the
# process, please refer from line 459

# extract file path

  test.images.folder2 <- "C:/Users/pierr/OneDrive/Documents/Maastricht
University/Masters/SDSS/Exam/test/test"

# list names

  test.images.name2 <- list.files(test.images.folder2)

# two images removed. One is png and one causes RStudio to crash

  test.images.name2 <- test.images.name2[-20]
  test.images.name2 <- test.images.name2[-93]

# create a list of path names include the image name

test.images.fnames2 <- paste(test.images.folder2, "/",
test.images.name2[1:length(test.images.name2)],sep="")

# get lines

  image.names2 <- substr(test.images.name2, 1, 22)
  n.n.line2 <- as.data.frame(0)
  for (i in 1:length(test.images.fnames2)){
    test.images2 <- load.image(test.images.fnames2[i])
    test.image.lines2 <- cannyEdges(test.images2)
    lines2 <- hough_line(test.images2, ntheta = 800, data.frame=TRUE)
    quant2 <- quantile(lines2$score, probs = seq(0.995,1,0.05))
    lines2 <- filter(lines2, lines2$score>=quant2)
    n.lines2 <- nrow(lines2)
    n.n.line2 [i,1] <- image.names2[i]
    n.n.line2 [i,2] <- n.lines2
  }

# get rgb

  rgb.mat.names2 <- substr(test.images.name2, 1, 22)
  length(rgb.mat.names2)
  rgb.mat2 <- matrix(0, ncol = 81, nrow = length(rgb.mat.names2))
  for (i in 1:length(test.images.fnames2)){
       x <- getImageHist(test.images.fnames2[i], bins = 3, hsv = FALSE, plotting=FALSE)
       r <- x$r
       g <- x$g
       b <- x$b
       a <- append(r, g)
       a <- append(a, b)
    rgb.mat2[i,] <- a
  }
  rgb.mat2 <- cbind(rgb.mat2, rgb.mat.names2)

# get hsv

  hsv.mat.names2 <- substr(test.images.name2, 1, 22)
  length(hsv.mat.names2)
  hsv.mat2 <- matrix(0, ncol = 81, nrow = length(hsv.mat.names2))
  for (i in 1:length(test.images.fnames2)){
       x <- getImageHist(test.images.fnames2[i], bins = 3, hsv = TRUE, plotting=FALSE)
       r <- x$h
       g <- x$s
    b <- x$v
       a <- append(r, g)
       a <- append(a, b)
    hsv.mat2[i,] <- a
  }
  hsv.mat2 <- cbind(hsv.mat2, hsv.mat.names2)

# rename the columns
```

```
  colnames(rgb.mat2) <- c(r.names,g.names,b.names,"photo_id")
  colnames(hsv.mat2) <- c(h.names, s.names, v.names,"photo_id")
  colnames(n.n.line2) <- c("photo_id","n.lines")
  rgb.mat2 <- as.data.frame(rgb.mat2)
  hsv.mat2 <- as.data.frame(hsv.mat2)

# build the main data frame

  image.main2 <- left_join(n.n.line2, photos.test, by="photo_id")
  image.main2 <- left_join(image.main2, rgb.mat2, by="photo_id")
  image.main2 <- left_join(image.main2, hsv.mat2, by="photo_id")
  write.csv(image.main2, file="image_main_test.csv")

# remove unnecessary columns

  image.main3 <- image.main2
  image.main3 <- image.main3[-1]
  image.main3 <- image.main3[-2]
  image.main3 <- image.main3[-2]

# Chosen classifier: XGBoost trained on unbalanced dataset. Make predictions

  xg.pred.test <- predict(xgb, data.matrix(image.main3))

# transform predictions

  xg.pred.test <- round(xg.pred.test)
  xg.test <- image.main2
  xg.test <- xg.test[1]
  xg.test$pred <- xg.pred.test
  xg.test$pred <- lapply(xg.test$pred, function(x){
  if(x==1){
    x<-"drink"
  } else if (x==2) {
       x <- "food"
  } else if (x==3){
       x <- "inside"
  } else {
       x <- "outside"
  }
  })

# save into RData file

  save(xg.test, file="Predictions_Hardy_PM_i6117019.RData")

# done!
```



```
#
#  \'__
#  \   '__
#  \      '__
#  \  _    _ '__
#  \ \ \  \ \   '__
#  \ \ \  \ \      '__                                _
#  \ \ \  \ \       __'                              / \
#  \ \ \__\ \    __'   __'\                          / /
#  \  \____/  __'   __'   \                         / /
#  \       __'   __'      \    /\                   / /
#  \    __'   __' __  __  \   /, ~\ .              / /
#  \.__'   __'   \  \/  \ \  x      -.....-------./ /
#       __'      \ \  / \ \   ~-. ~  ~              /
#      '__       \ \\/\ \ \      \             /    /
#         '__    \ \  \ \ \       \  /_     ___\   /
#            '__ \_\  \_\ \       / /\ ~~~~~   \ /
#               '__       \       / / \        // /
#                  '__    \       / /\ \       // )
#                     '__.\      (_/ (_/      ((_/
#
# plus points for cat?
```