# Project Report

Robin Leuridan

2023-12-02

Github link:https://github.com/Trisrobi/EC349-Project.git (https://github.com/Trisrobi/EC349-Project.git)

## Data Science Methodology

We implemented the CRISP-DM methodology, the most common one for data science projects (see Schröer et. al. (2021)). It enabled us to make sure this project was reproducible for other prediction models. Though it had disadvantages, those were limited in our context.

Following this methodology, we analysed the data going through each of the different stages: business understanding, data understanding, data preparation, modelling and evaluation. The structure of this final report also follows from this methodology choice. Furthermore, we kept documentation at each stage, indicating any changes made.

## Business Understanding

In this project, the main issue concerned classification and prediction, as the variable we sought to predict (the number of stars) was discrete and we strove to create a model to predict the values obtained for each observation.

## Data Understanding

We used three different datasets: tip data, reviews_data and business_data. The reviews data were used to try to predict the variable stars (which indicates the number of stars a specific review assigns), the business data to access a multitude of variables indicating each businesses different attributes(such as whether the business has a TV) and the tip_data to retrieve compliment_count data, indicating how many compliments were given in the tip. We can see using the table below that the number of stars is skewed towards 5. Summary statistics for the other variables can be found in the appendix.

```
## [1] "##frequency table of stars##"
```

```
##
##      1      2      3      4      5
## 187161 104099 134309 281607 592575
```

## Data Preparation

To analyse the data, we merged all three datasets together by the business id variable to group the reviews according to businesses. Furthermore, we transformed the type of the character variables to factor variables so as to run a random Tree model. We included the missing variables by allowing a level of the factors to represent the value as missing. Finally, we removed some of the data with only few observations such as Open24Hours and AgesAllowed to reduce the likelihood of overfitting.

## Modeling

In order to train our data intensively, we split the data in a 90-10 split, allowing for 90% of the observations to be in the training dataset, and testing our model on the remaining 10%.

```
# createDataPartition() function from the caret package to split the original dataset into a
training and testing set and split data into training (90%) and testing set (10%)
parts = createDataPartition(data$stars.x, p = 0.9, list = F)
train = data[parts, ]
test = data[-parts, ]
```

We first used a logistic regression to model this problem. As we wanted to classify observations, this choice enabled us to keep the predictions between the values needed and not exceed them. For a binary problem, this would have limited the values obtained between 0 and 1. However, as our aim was to predict the number of stars a review might have, we also used a multinomial, ordered logistic regression. This made it possible for the dependent variables to have multiple values outside 0 and 1, which was necessary since values between 1 and 5 were needed. Though some predictors such as the review_count were to have a high impact on the predictions, the impact of others such as the variable HasTV was questionable. For this reason, we decided to adapt our logistic model using LASSO regression, in order to allow some variables to be removed entirely from the model.

As the model did not perform well in predictions as could be attested using a measure of accuracy, measured through the number of observations correctly specified, we then switched to a decision tree-based model to better represent both the data and the problem as well as make our model more flexible. We still used a classification tree to analyse the data, due to the aforementioned reasons of classification problem. To decrease the inherent variance associated with decision trees, we employed a bagging method to average the observations and thus reduce overall variance. However, following our analysis using the LASSO model, it became clear that some of the variables were vastly more important than others. Thus, we found it relevant to use a Random Forest method. As some variables were more inclined to appear in decision trees depending on their importance, risk of high correlation between different trees generated in the bagging process was increased. By using a RandomForest, the trees generated through the bagging process would be uncorrelated to each other as the variables in each tree would be randomly selected, so it would reduce the variance more than normal bagging.

```
### Random Forest ###
set.seed(1312)
model_RF<-randomForest(stars.x~.,data=train, ntree=100)
pred_RF_test = predict(model_RF, test, type="prob")
```

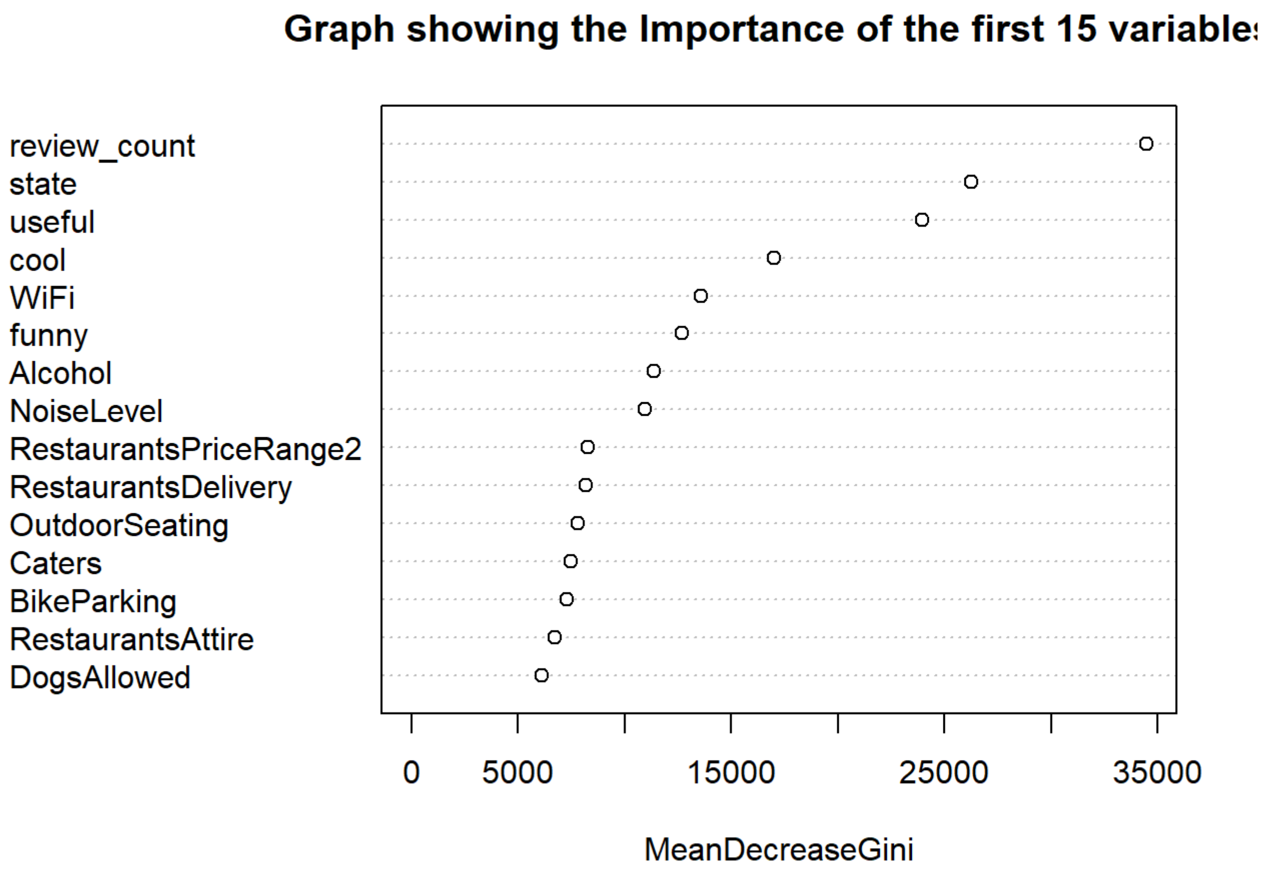
As an afterthought, a boosting method would have probably worked even better than what was currently implemented as it would have allowed the tree to average over different predictors. With such a method, the predictors which were not performing well could have been fine tuned to fit the residuals better. This would have been even more useful in our specific problem due to the predictors not fitting the data very well. Boosting would therefore allow those predictors to evolve and gain different weights to better fit the data. Due to time and technological constraints, we were however unable to implement this specific method in our project.

## Evaluation

```
## [1] "##confusionMatrix##"
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction     1     2     3     4     5
##          1  7102  2397  1748  1821  2829
##          2   658   447   384   460   516
##          3   534   485   592   769   764
##          4  1496  1468  2388  5238  5354
##          5  8926  5612  8318 19872 49794
## 
## Overall Statistics
## 
##                Accuracy : 0.4861
##                  95% CI : (0.4833, 0.4888)
##     No Information Rate : 0.4559
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.1806
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.37946 0.042944 0.044080   0.1860   0.8403
## Specificity           0.92095 0.983122 0.978102   0.8948   0.3958
## Pos Pred Value        0.44675 0.181339 0.188295   0.3285   0.5382
## Neg Pred Value        0.89819 0.921871 0.898776   0.7990   0.7473
## Prevalence            0.14400 0.080086 0.103330   0.2167   0.4559
## Detection Rate        0.05464 0.003439 0.004555   0.0403   0.3831
## Detection Prevalence  0.12231 0.018966 0.024190   0.1227   0.7119
## Balanced Accuracy     0.65020 0.513033 0.511091   0.5404   0.6180
```

While training our model, we observed that it did not work very well in the training set. Specifically, the model reported an error rate averaging at around 67%, which meant that 67% of the predictions on the data were actually false. This result maybe due to the fact that the training dataset was a lot larger than the testing dataset. Considering that there were three main variables, the testing data may have by chance been easier to categorise with those specific variables.

We could further note, using the graph below, that the most important variables in the dataset were the review counts variable as well as the states variable, representing which state each business was located in. This might be explained by a cultural difference between states impacting the amount of stars. Hypothetically, this might have been the result of citizens in that given state being meaner or more prone to give bad reviews. Furthermore, it was noteworthy that variables such as the usefulness and the coolness of the review had a high importance on the model. This meant that further work on the topic involving sentiment analysis could be useful and might greatly improve the model.



When testing the model on the test data, the accuracy was estimated to be around 48%, indicating that about 48% of our test data was correctly predicted. This accuracy was better than the one observed in the training dataset. It is interesting to note that the model predicted extreme classes such as classes 1 and 5 better than the average classes 2,3 and 4, as seen in the covariance matrix below. This could be because the data is heavily skewed towards 5, thus leading the model to classify those observations more accurately.

## Most Difficult Challenge

Our most difficult challenge was to clean up and transform the data. The datasets themselves were very large and contained a lot of different data, in numeric form and in string variable form. We encountered a problem while trying to merge the different datasets provided into one single dataset containing all the variables because some datasets were correlated in a many-to-many manner. One variable in one dataset, such as the compliment_count variable, corresponded to many different observations in another(the business_data), and as such, it was impossible to produce a code to merge the different datasets into one with the basic packages involved in R. Thus, we had to find a solution in other packages found online. In the end, we used the dplyr package which made it possible to account for many to many problems.

## Appendix

```
##      state        is_open      ByAppointmentOnly BikeParking
##  PA     :296776   0 : 227685   False:291115      False:194714
##  FL     :215693   1 :1072066   None :   117      None :   254
##  LA     :148099   NA:      0   True : 53080      True :860634
##  TN     :114652                NA   :955439      NA   :244149
##  MO     : 94541
##  IN     : 91237
##  (Other):338753
##  RestaurantsPriceRange2 CoatCheck       RestaurantsTakeOut RestaurantsDelivery
##  1   :238338            False: 166991   False: 48808       False:290112
##  2   :797327            None :    160   None : 54101       None :109128
##  3   : 81170            True :  18699   True :894078       True :575988
##  4   :  9513            NA   :1113901   NA   :302764       NA   :324523
##  None:    67
##  NA  :173336
##
##    Caters       WheelchairAccessible HappyHour      OutdoorSeating
##  False:393781   False: 32878         False:196868   False:305139
##  None :   393   None :   164         None :    62   None : 90008
##  True :505138   True :471975         True :345603   True :557603
##  NA   :400439   NA   :794734         NA   :757218   NA   :347001
##
##
##
##    HasTV        RestaurantsReservations DogsAllowed
##  False:249487   False:490726            False:431289
##  None :    56   None :  7005            None :   263
##  True :671478   True :415695            True :147316
##  NA   :378730   NA   :386325            NA   :720883
##
##
##
##              Alcohol       GoodForKids    RestaurantsAttire
##  NA               :382561   False:181091   u'casual':490080
##  u'full_bar'      :367259   None :    77   NA       :437203
##  u'none'          :205574   True :776361   'casual' :341382
##  u'beer_and_wine':122772   NA   :342222   u'dressy': 18978
##  'full_bar'       :116271                  'dressy' : 10848
##  'none'           : 66251                  None     :   720
##  (Other)          : 39063                  (Other)  :   540
##  RestaurantsTableService RestaurantsGoodForGroups DriveThru
##  False:142944            False: 74372             False:  93985
##  None :    46            None :    45             None :  25874
##  True :445649            True :833150             True :  36569
##  NA   :711112            NA   :392184             NA   :1143323
##
##
##
##       NoiseLevel     BusinessAcceptsBitcoin AcceptsInsurance    BYOB
##  u'average':631478   False: 276308          False:   7235    False: 131480
##  NA        :395702   None :     51          None :     78    None :    151
##  'average' :105319   True :   4658          True :   9396    True :  30754
##  u'quiet'  : 72586   NA   :1018734          NA   :1283042    NA   :1137366
##  u'loud'   : 60748
##  'quiet'   : 12009
##  (Other)   : 21909
##   Corkage                 BYOBCorkage      stars.x        WiFi
##  False:  82421   NA             :1210859   1:187161   u'free':454353
##  None :    316   'no'           :  46574   2:104099   u'no'  :307683
##  True :  61426   'yes_free'     :  33283   3:134309   NA     :286584
##  NA   :1155588   'yes_corkage' :   7502   4:281607   'free' :143341
##                  u'yes_free'   :    643   5:592575   'no'   : 99353
##                  u'yes_corkage':    419              u'paid':  6566
##                  (Other)       :    471              (Other):  1871
##   review_count      funny              cool              useful
##  Min.   :   5   Min.   :  0.000   Min.   :  0.0000   Min.   :  0.000
##  1st Qu.:  56   1st Qu.:  0.000   1st Qu.:  0.0000   1st Qu.:  0.000
##  Median : 154   Median :  0.000   Median :  0.0000   Median :  0.000
##  Mean   : 397   Mean   :  0.331   Mean   :  0.5088   Mean   :  1.161
##  3rd Qu.: 394   3rd Qu.:  0.000   3rd Qu.:  0.0000   3rd Qu.:  1.000
##  Max.   :7568   Max.   :346.000   Max.   :400.0000   Max.   :400.000
##
##  compliment_count
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.01376
##  3rd Qu.:0.00000
##  Max.   :5.00000
##
```

## References

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.

Christoph Schröer et al., Procedia Computer Science 181 (2021) ,526–534