Yelp User Review Predictions

Micah Okoko

2023-11-08

```
library (glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(jsonlite)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## v readr     2.1.4
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x tidyr::expand()  masks Matrix::expand()
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
## x tidyr::pack()    masks Matrix::pack()
## x tidyr::unpack()  masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tree)
library(rpart)
library(rpart.plot)
library(ipred)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

DS Methodology

Throughout this project, the General DS Methodology approach by John Rallins has been followed. This approach consists of the stages Business Understanding, Analytic Approach, Data Requirements, Collection, Understanding, and Preparation, then Modelling, Evaluation, and Deployment, supplemented by Feedback. This methodology was chosen due to its organised, yet adaptable structure making it ideal for handling the complex nature of the Yelp datasets. This was applied by initially understanding the problem and how the data can be utilised to solve it where other literature and readings were consulted. Several comprehensive datasets from Yelp were looked at with approximately 7 million reviews, and the data required was organised. Predictive models and evaluation techniques were then used and tested in order to achieve optimal results. Feedback from the evaluation is also reviewed.

Method

Introduction

There are 3 main phases of this adopted method; Problem Definition, Data Understanding & Organisation, finally Validation and Deployment.

1. Problem Definition

Problem Understanding

The problem for this paper is defined as follows: Use yelp datasets to predict user reviews.

Analytical Approach

During this stage the reading by Hastie et al. (2021) on Statistical Learning was looked at. This paper highlights that decision trees are robust to overfitting, capable of handling comprehensive datasets, and predictive performance can be enhanced using tree ensembles. Given this information and understanding of the problem, the best statistical approach decided was to utilise classification trees. Star ratings given by user i to business j is the target variable, where the user star ratings 1, 2, 3, 4, and 5 are considered discrete categories. Relevant features from the datasets will also be selected and used in tree models.

2. Data Understanding & Organisation

Data Requirements and Collection

The first step of this phase is understanding what data is required and deciding the best variables to use as predictors. The data is collected from yelp and can be found via this link, simplified versions can also be found here. This data is loaded into the local environment.

```
load("~/Desktop/Yelp User Review Predictions/Datasets/yelp_review_small.Rda")
load("~/Desktop/Yelp User Review Predictions/Datasets/yelp_user_small.Rda")
business_data <- stream_in(file("~/Desktop/Yelp User Review Predictions/Datasets/yelp_academic_dataset_
```

```
## opening file input connection.
```

```
##  Found 500 records... Found 1000 records... Found 1500 records... Found 2000 records... Found 2500 re
```

```
## closing file input connection.
```

After gaining an detailed understanding of the datasets, the 'review_data_small', 'user_data_small', and 'business_data' datasets were seen as the most relevant and merged together.

```
data_initial_merge <- merge(review_data_small, user_data_small, by = "user_id")
data_merged <- merge(data_initial_merge, business_data, by = "business_id")
```

In this dataset stars.x represents our target variable.

Data Understanding

Relevant features were then selected and put into a single dataset.

```
data_final <- data_merged %>% select(stars.x, funny.x, cool.x, review_count.x, useful.y,
                                     funny.y, cool.y, fans, average_stars, stars.y, review_count.y,
                                     is_open, starts_with('compliment_'))
```

These features were selected using careful analysis. The decision to focus on various metrics was based on the belief that these variables collectively offer a broad overview of user behavior and preferences on yelp. Features like 'funny', 'cool', 'useful' and the 'compliment_' variables directly reflect user interaction and satisfaction helping to understand the overall thought of the reviews. 'Review_count' and 'fans' indicate the level of user engagement and influence on the yelp platform, which can significantly impact how a business is perceived. 'Average_stars' from users provides a strong indication of general rating behavior, and 'is_open' tells us the business's current status.

Data Preparation

The target variable is transformed into a categorical variable.

```
data_final$stars.x <- as.factor(data_final$stars.x)
```

This is done so the tree models can efficiently handle the predictive task as a classification problem.

Finally, the data is split into training and testing sets with 10000 observations in the testing set.

```
set.seed(1)
test_set_size <- 10000
test <- sample(1:nrow(data_final), test_set_size)
data_final_test <- data_final[test, ]
data_final_train <- data_final[-test, ]
```

3. Validation & Deployment

Modelling

The random forest model, which applies the idea of bagging with random trees, was chosen for this problem as it is effective for categorical variables, can handle large numbers of features, and is robust against overfitting.

The data was modelled with random forest.

```
model_RF<-randomForest(stars.x ~ ., data=data_final_train, ntree=50, importance = TRUE, nodesize = 5)
model_RF2<-randomForest(stars.x ~ ., data=data_final_train, ntree=50, importance = TRUE, nodesize = 1)
```

The parameter 'nodesize' was adjusted where this is the minimum size of the terminal node. A smaller node size tends to create more complex trees, but if this is too small it can lead to over fitting, this means increasing the node size from 1 to 5 can be seen as a form of regularization.

Evaluation

Node size = 5:

```
model_RF
```

```
##
## Call:
##  randomForest(formula = stars.x ~ ., data = data_final_train,      ntree = 50, importance = TRUE, no
##                Type of random forest: classification
##                      Number of trees: 50
```

```
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 42.58%
## Confusion matrix:
##       1    2    3     4      5 class.error
## 1 28155 1331 1082  2196   8513   0.3179010
## 2  6275 2467 1685  3447   7302   0.8835002
## 3  3318 1232 3824  7994  10472   0.8575261
## 4  3367 1320 4104 18471  28810   0.6705842
## 5  5638 1099 2361 13379 102036   0.1805193
```

Node size = 1:

```
model_RF2
```

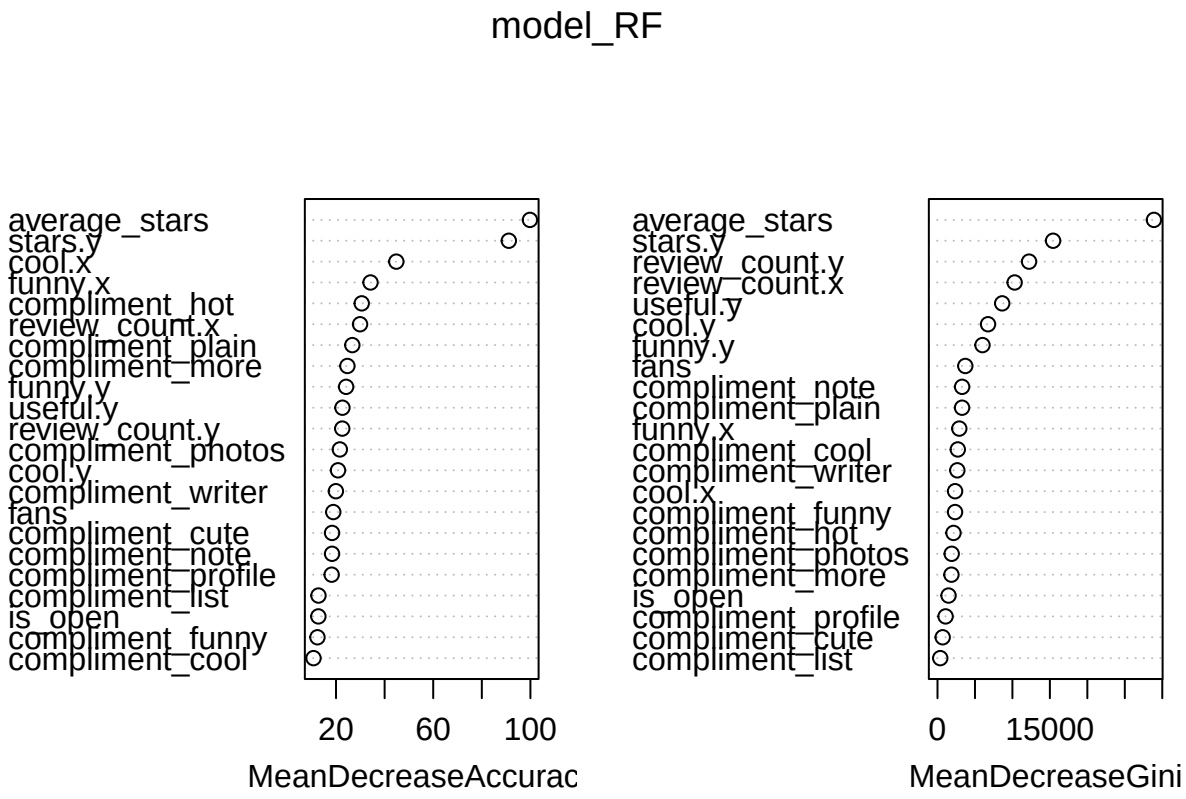
```
##
## Call:
##  randomForest(formula = stars.x ~ ., data = data_final_train,      ntree = 50, importance = TRUE, noc
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 42.88%
## Confusion matrix:
##       1    2    3     4      5 class.error
## 1 27812 1459 1161  2246   8599   0.3262107
## 2  6127 2669 1814  3414   7152   0.8739611
## 3  3284 1377 3968  7697  10514   0.8521610
## 4  3381 1464 4515 18121  28591   0.6768262
## 5  5660 1334 2668 13274 101577   0.1842057
```

The Out Of Bag (OOB) error is an estimate of the error rate that the model has for unseen data. According to Hastie et al. (2021), OOB observations are observations not used to fit a given tree. Predictions for these OOB observations are made using trees that were not fit using that observation. This provides a way to evaluate the model without a test set and tends to work best with a large numbers of trees. Since the OOB slightly decreases when node size increases from 1 to 5, it suggests overfitting has been slightly reduced.

The confusion matrix describes the performance of a classification model, each row represents predicted classes, and each column represents actual classes. The class error rates shows how the model performs for each individual class.

Setting importance = 'TRUE' in the random forest model parameters allows a measure of variable importance to be determined.

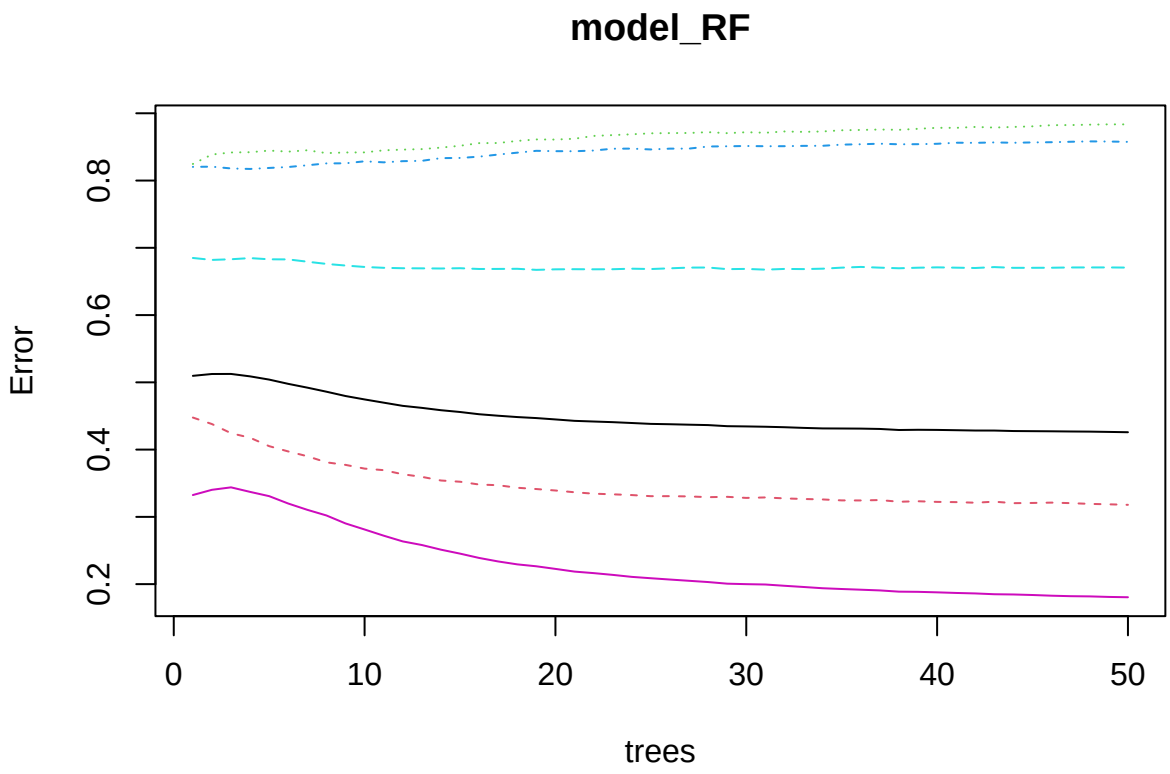
```
varImpPlot(model_RF)
```



The Mean Decrease Accuracy shows the decrease in model accuracy when a variable is excluded, the accuracy decreases more if the excluded variable is more important. The Mean Decrease Gini measures the contribution of each variable to the homogeneity of the nodes and leaves, the more a variable decreases the Gini impurity, the more it contributes to creating homogeneous nodes in the decision tree (Fernando et al, 2020). From these plots we can conclude that 'average_stars' and 'stars.y' are the most significant features in the model.

For further insight we can plot the random forest model.

```
plot(model_RF)
```



From this plot we can observe the general error rate trend as the number of trees increases. We see that the error rate generally decreases as the number of trees increases, as random forest models tend to perform better with a larger number of trees. The error rate also seems to plateau at around 30 - 40 trees indicating that adding more trees beyond this point may not improve the performance of the model.

To fully evaluate the model it is also tested against the test set and the accuracy is calculated:

```
true_values <- data_final_test$stars.x
predictions_rf <- predict(model_RF, data_final_test)
accuracy_rf <- sum(predictions_rf == true_values) / length(true_values)
print(paste("RF Accuracy:", accuracy_rf))
```

```
## [1] "RF Accuracy: 0.5775"
```

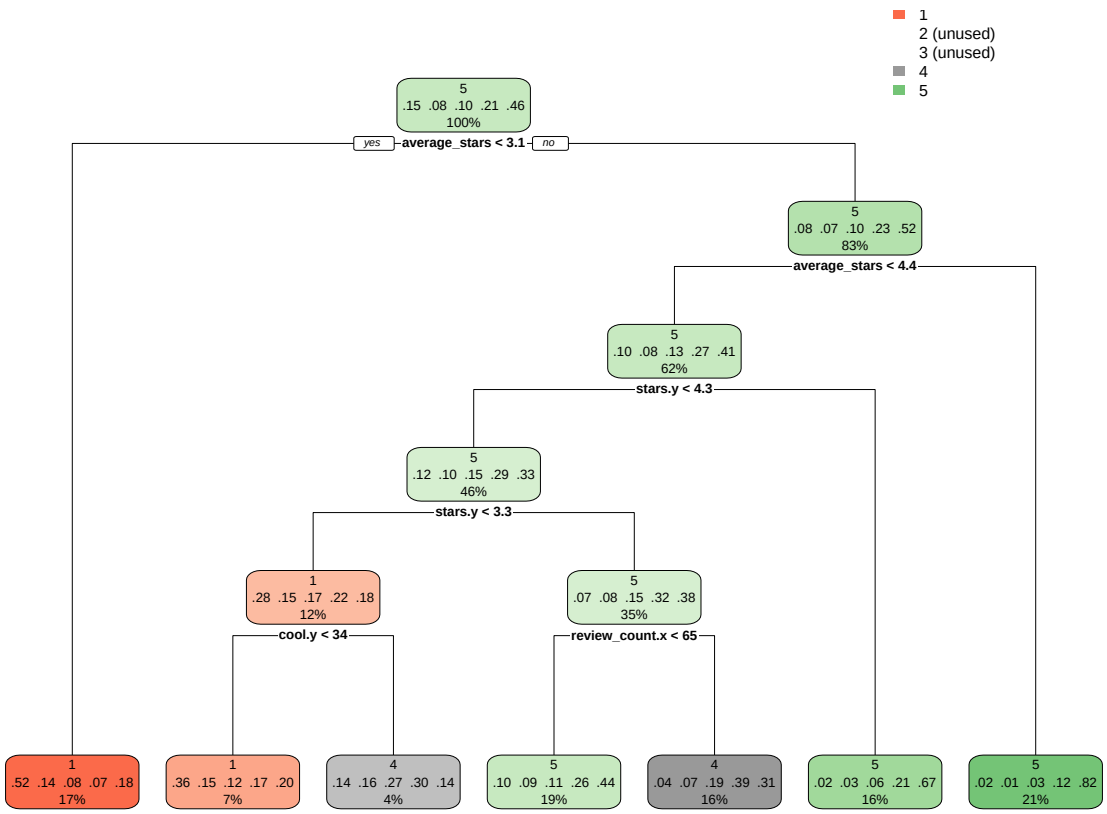
A 57.75% accuracy, which is the total number of correct predictions divided by the total number of predictions, suggests the model performs reasonably well and has learned patterns in the data.

Other Models

A partitioned classification tree was also tested.

```
rpart_tree <- rpart(stars.x ~ ., data = data_final_train, method = "class")
```

```
rpart.plot(rpart_tree)
```



This graph shows that the 'average_stars' feature is significant in determining the class of a user review. The structure of the tree also highlights that some paths result in high confidence class predictions, whilst others suggest more ambiguity with mixed class distributions. Overall it suggests that the tree captures patterns in the data, but there is variability in the confidence of predictions across different branches.

The accuracy of the model was also determined.

```
predictions_rpart <- predict(rpart_tree, data_final_test, type = "class")
accuracy_rpart <- sum(predictions_rpart == true_values) / length(true_values)
print(paste("Accuracy:", accuracy_rpart))
```

```
## [1] "Accuracy: 0.5446"
```

Although a 54.46% accuracy indicates it performs better than random guessing, it still under performs against the random forest model.

Deployment and Feedback

The feedback from the evaluation indicates further model improvements should be done before being practically deployed. This can be include further tuning the model parameters, optimising features, and using a cross validation set to check the models ability to generalise a pattern.

Challenges and Solutions

The biggest challenge was handling the large size of the datasets and generating complex models which presented computational limitations. My initial attempt to use random forest proved unfeasible given the limitations of the hardware. To overcome this, I used the smaller datasets provided, and the parameters of the random forest model was adjusted decreasing the number of trees to 50. Although these adjustments can lead to underfitting and may limit the model robustness, this approach was effective in reducing the computational load.

References:

James, G.; Witten, D.; Hastie, T.; & Tibshirani, R,. 2021 (2nd Ed.). An Introduction to Statistical Learning with Applications in R. Springer.

Martinez-Taboada, Fernando; Redondo, Jose Ignacio (2020). Variable importance plot (mean decrease accuracy and mean decrease Gini).