EC349 – Assignment 1 Docs.

GitHub Link:

https://github.com/4ad1tyaa/EC349-Data-Science-Project.git

R Script:

```
# ASSIGNMENT 1 - EC349

# Part 1: "You must split the User Reviews data into a training and a test dataset. The test
dataset must contain 10,000 randomly drawn observations using the "caret" package in R.

# Loading Small User Reviews Dataset File
load("/Users/god/Desktop/God/3rdYearUni/DataScience/RStuff/Term1Assignment/Provided
Material/yelp_review_small.Rda")

# Installing Relevant Packages and Loading (if needed)
options(repos = c(CRAN = "https://cloud.r-project.org/"))

if (!require("caret", character.only = TRUE)) {
  install.packages("caret")
}
library(caret)

if (!require("randomForest", character.only = TRUE)) {
  install.packages("randomForest")
}
library(randomForest)

# Creating Training and Test Datasets
total_observations <- nrow(review_data_small)
test_size <- 10000
test_fraction <- test_size / total_observations

split_sets <- createDataPartition(review_data_small$stars, p = test_fraction, list = FALSE)

training_dataset <- review_data_small[-split_sets, ]
test_dataset <- review_data_small[split_sets, ]

# Seeing 10,002 observations in the test dataset, as such will be randomly selecting 10,000
observations from the test dataset
sampling_test_dataset <- sample(nrow(test_dataset), 10000)
new_test_dataset <- test_dataset[sampling_test_dataset, ]

# Performing check to see if there are the correct number of observations in both datasets
cat("Training Dataset Observations: ", nrow(training_dataset), "\n")
cat("Test Dataset Observations: ", nrow(new_test_dataset), "\n")

# ---------------------------------------------------------------

# Part 2: Predicting User Reviews for those 10,000 observations - opting for RandomForest

# Loading Small User Data Dataset File
load("/Users/god/Desktop/God/3rdYearUni/DataScience/RStuff/Term1Assignment/Provided
Material/yelp_user_small.Rda")

# Merging User Data Dataset with User Reviews Dataset
combined_data <- merge(training_dataset, user_data_small, by = "user_id")

# Cleaning data to make it easier to manipulate in future (handling missing values,
categorical variables etc.)
combined_data[is.na(combined_data)] <- apply(combined_data, 2, function(x)
ifelse(is.numeric(x), median(x, na.rm = TRUE), x))
combined_data$yelping_since <- as.Date(combined_data$yelping_since)
combined_data$years_yelping <- as.numeric(format(Sys.Date(), "%Y")) -
as.numeric(format(combined_data$yelping_since, "%Y"))
combined_data <- combined_data[, !(names(combined_data) %in% c("name",
"yelping_since", "elite", "friends", "text", "date"))]


# Taking a subset for faster processing
set.seed(123)
sample_index <- sample(1:nrow(combined_data), size = 0.5 * nrow(combined_data))
subset_data <- combined_data[sample_index, ]

# Building a Random Forest model with adjusted parameters to train
my_model <- randomForest(stars ~ ., data = subset_data, ntree = 200, mtry =
round(sqrt(ncol(subset_data))), do.trace = 10)

# Evaluation on training dataset
print(my_model)

# Predictions + MSE on training dataset
predictions <- predict(my_model, subset_data)
mse <- mean((subset_data$stars - predictions)^2)
cat("The Mean Squared Error is: ", mse, "\n")

# Preparing test dataset
test_combined_data <- merge(new_test_dataset, user_data_small, by = "user_id")
test_combined_data$yelping_since <- as.Date(test_combined_data$yelping_since)
test_combined_data$years_yelping <- as.numeric(format(Sys.Date(), "%Y")) -
as.numeric(format(test_combined_data$yelping_since, "%Y"))
```

```r
test_combined_data <- test_combined_data[, !(names(test_combined_data) %in%
c("name", "yelping_since", "elite", "friends", "text", "date"))]

# Making Predictions
test_dataset_predictions <- predict(my_model, test_combined_data)

# Examining Predictions + Calculating MSE on test dataset
test_mse <- mean((test_combined_data$stars - test_dataset_predictions)^2)
cat("Test Mean Squared Error is: ", test_mse, "\n")

head(test_dataset_predictions)
comparison_df <- data.frame(Actual = test_combined_data$stars, Predicted =
test_dataset_predictions)
head(comparison_df)
summary(test_dataset_predictions)
plot(comparison_df$Actual, comparison_df$Predicted, main = "Predicted vs Actual Stars",
xlab = "Actual Stars", ylab = "Predicted Stars", pch = 19)
abline(0, 1, col = "red")
```

Markdown File:

~~~
---
title: "Data Science Assignment 1"
author: "Aadi Kannan - u2005706"
date: "`r Sys.Date()`"
output:
  html_document: default
  pdf_document: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r, include=FALSE}
source("SmallDatasetAssignment1.R", local = knitr::knit_global())
```

## **Introduction**

In this assignment, I have attempted to predict the User Reviews for **~10,000** test
observations on Yelp.

## **DS Methodology + Biggest Challenge**

I utilised a DS Methodology of **CRISP-DM**, owing to its more flexible framework, so I
could check that the analysis I have conducted is aligned accurately with the objective. For
~~~

the first stage of business understanding, I made sure to understand the exact goal to be achieved - predicting Yelp user ratings. This, in turn, helped to guide the data understanding, exploration and preparation, whereby I had merged the datasets and processed the data. I followed an iterative process, using a subset of data (i.e. 50% of test data) in order to expedite and adjust anything before I made my final predictions. I repeatedly compared my training and test results - resulting in thorough evaluation.

My most difficult challenge was running my model on the test data - it often took up a lot of time and was computationally expensive, but I had initally chosen this route in pursuit of a more "accurate" model. However, I eventually decided to create a subset of the test data (~50%), and used 200 trees. I am fully aware and accept that this may have an impact on my results, whether that be through a lower percentage of variance explained, or a higher MSE, or lower accuracy. I was also uncomfortable with the uncertain timeframes presented when running my model on the test data - to solve this issue, I enlisted the help of the *do.trace* function, which enabled me to more effectively keep track of how far the code had been compiled, and of how many trees had been run.

Pre-Model Preparation

I used the small datasets provided, given the lack of computational power on my personal desktop. This project utilised two datasets: 'user_data_small" - providing user information, and "review_data_small" - providing user reviews. After noticing the variable "user_id" was present in both datasets, they were merged to create a new merged comprehensive dataset, ready for future analysis.

The script below is the code used to merge the training user reviews dataset with the user information parent dataset...
```{r, eval=FALSE}
combined_data <- merge(training_dataset, user_data_small, by = "user_id")
```

And the script below is the code used to merge the test user reviews dataset with the user information parent dataset.
```{r, eval=FALSE}
test_combined_data <- merge(new_test_dataset, user_data_small, by = "user_id")
```

## **Post-Model Analysis**

### **Training the Model**

I've chosen to select the randomForest algorithm, as it not only achieves what bootstrap aggregation aims to do, but applies that concept to random trees, thereby reducing correlations across predictions, but also the variance in prediction.

Please find below the code that was written in order to train my model. As mentioned previously, the number of trees was set to 200, and 50% of the test dataset was utilised, in order to reach a balance between model accuracy and computational efficiency.
```{r, EVAL=FALSE}
# Taking a subset for faster processing
set.seed(123)
sample_index <- sample(1:nrow(combined_data), size = 0.5 * nrow(combined_data))
subset_data <- combined_data[sample_index, ]

# Building a Random Forest model with adjusted parameters to train
my_model <- randomForest(stars ~ ., data = subset_data, ntree = 200, mtry =
round(sqrt(ncol(subset_data))), do.trace = 10)
```

Evaluating the Model

In order to measure the model's performance, I used a combination of the Mean of Squared Residuals, the Percentage of Variance Explained, and Mean Squared Error, as can be seen from the code below. The initial training on the subset showed promise, as can be seen from the results from the code (results will also be discussed in depth in the following section.)
```{r, echo=TRUE}
# Predictions + MSE on training dataset
predictions <- predict(my_model, subset_data)
mse <- mean((subset_data$stars - predictions)^2)
cat("The Mean Squared Error is: ", mse, "\n")
```

### **Running Model on Test Dataset**

I then fed the model the Test Dataset in order to evaluate its 'final' results - the code and results are below, but once again, the results will be discussed in the next section in further detail).
```{r, echo=TRUE}
# Preparing test dataset
test_combined_data <- merge(new_test_dataset, user_data_small, by = "user_id")
test_combined_data$yelping_since <- as.Date(test_combined_data$yelping_since)
test_combined_data$years_yelping <- as.numeric(format(Sys.Date(), "%Y")) -
as.numeric(format(test_combined_data$yelping_since, "%Y"))
test_combined_data <- test_combined_data[, !(names(test_combined_data) %in%
c("name", "yelping_since", "elite", "friends", "text", "date"))]

# Making Predictions
test_dataset_predictions <- predict(my_model, test_combined_data)

# Examining Predictions + Calculating MSE on test dataset
test_mse <- mean((test_combined_data$stars - test_dataset_predictions)^2)
cat("Test Mean Squared Error is: ", test_mse, "\n")
```

```
head(test_dataset_predictions)
comparison_df <- data.frame(Actual = test_combined_data$stars, Predicted =
test_dataset_predictions)
head(comparison_df)
summary(test_dataset_predictions)
plot(comparison_df$Actual, comparison_df$Predicted, main = "Predicted vs Actual Stars",
xlab = "Actual Stars", ylab = "Predicted Stars", pch = 19)
abline(0, 1, col = "red")
```

## **Analysing Results**

### **Training Dataset Results**

When fed the training dataset, the model achieved (to 3 significant figures)...
1) Mean of Squared Residuals = 1.35
2) Mean Squared Error = 0.370
3) Percentage of Variance Explained = 38.310

I initially had a much higher MSE and lower % of Variance explained, so decided to add the categorical variables that I had initially discarded, to help improve the accuracy of the model, whcih resulted in the values shown above,

### **Test Dataset Results**

When fed the test dataset, the model achieved (to 3 significant figures)...
1) Mean Squared Error = 1.386
2) Prediction Range = 1.158 - 4.978

Interpretation of Results

### **Discussion of Results**

The MSE is 1.386, meaning that on average, the squared difference between the predicted stars and the actual stars is about 1.39. Since the MSE isn't far from 1, this suggests that the model's predictions are 'reasonably' close to the actual values, although it needs to be said that there's definitely room for improvement, especially provided with more computational power (and perhaps more advanced coding knowledge!).

The predictions range from a minimum of about 1.16 to a maximum of approximately 4.98; this suggests that my model is perhaps behaving conservatively - it seems to avoid predicting extreme ratings.

The scatter plot is showcasing the relationship between actual and predicted stars. The red line informs us of a perfect prediction. Ideally, we would like the points to be closer to the red line, indicating better prediction accuracy. We can see noticeable vertical dispersion for each star category, highlighting the variance in the predictions. Again, the predictions seem more concentrated around the mean, suggesting the model doesn't tend to predict extreme values very often.

Interpretation Summary

Overall, the mode has a moderate level of prediction accuracy, with a tendency to avoid extreme results, and perhaps under-predicting them also (as can be seen from the head of my comparison dataframe). There is a consistent spread in predictions across the range of actual star values, indicating some level of prediction error across the board. The model's conservative nature in predicting extremes may be due to a lack of distinctive features that differentiate extreme ratings from more moderate ones, or perhaps due to the very nature of the randomForest algorithm, which can average out predictions.

## **Improvement for Future Reference**

One immediate way to improve the model that comes to mind is to increase the complexity of the model, which may enable us to capiture more variance. However, I ultimately chose not pursue this route due to a risk of overfitting.

## **Final Comments**

Thanks for reading - hope you had as much fun as I had writing the code!

Data Science Assignment 1

Aadi Kannan - u2005706

2023-12-05

Introduction

In this assignment, I have attempted to predict the User Reviews for **~10,000** test observations on Yelp.

DS Methodology + Biggest Challenge

I utilised a DS Methodology of **CRISP-DM**, owing to its more flexible framework, so I could check that the analysis I have conducted is aligned accurately with the objective. For the first stage of business understanding, I made sure to understand the exact goal to be achieved - predicting Yelp user ratings. This, in turn, helped to guide the data understanding, exploration and preparation, whereby I had merged the datasets and processed the data. I followed an iterative process, using a subset of data (i.e. 50% of test data) in order to expedite and adjust anything before I made my final predictions. I repeatedly compared my training and test results - resulting in thorough evaluation.

My most difficult challenge was running my model on the test data - it often took up a lot of time and was computationally expensive, but I had initally chosen this route in pursuit of a more “accurate” model. However, I eventually decided to create a subset of the test data (~50%), and used 200 trees. I am fully aware and accept that this may have an impact on my results, whether that be through a lower percentage of variance explained, or a higher MSE, or lower accuracy. I was also uncomfortable with the uncertain timeframes presented when running my model on the test data - to solve this issue, I enlisted the help of the *do.trace* function, which enabled me to more effectively keep track of how far the code had been compiled, and of how many trees had been run.

Pre-Model Preparation

I used the small datasets provided, given the lack of computational power on my personal desktop. This project utilised two datasets: ’user_data_small” - providing user information, and “review_data_small” - providing user reviews. After noticing the variable “user_id” was present in both datasets, they were merged to create a new merged comprehensive dataset, ready for future analysis.

The script below is the code used to merge the training user reviews dataset with the user information parent dataset…

```
combined_data <- merge(training_dataset, user_data_small, by = "user_id")
```

And the script below is the code used to merge the test user reviews dataset with the user information parent dataset.

```
test_combined_data <- merge(new_test_dataset, user_data_small, by = "user_id")
```

Post-Model Analysis

Training the Model

I’ve chosen to select the randomForest algorithm, as it not only achieves what bootstrap aggregation aims to do, but applies that concept to random trees, thereby reducing correlations across predictions, but also the variance in prediction.

Please find below the code that was written in order to train my model. As mentioned previously, the number of trees was set to 200, and 50% of the test dataset was utilised, in order to reach a balance between model accuracy and computational efficiency.

```
# Taking a subset for faster processing
set.seed(123)
sample_index <- sample(1:nrow(combined_data), size = 0.5 * nrow(combined_data))
subset_data <- combined_data[sample_index, ]

# Building a Random Forest model with adjusted parameters to train
my_model <- randomForest(stars ~ ., data = subset_data, ntree = 200, mtry = round(sqrt(ncol(subset_data))), do.tr
ace = 10)
```

Evaluating the Model

In order to measure the model’s performance, I used a combination of the Mean of Squared Residuals, the Percentage of Variance Explained, and Mean Squared Error, as can be seen from the code below. The initial training on the subset showed promise, as can be seen from the results from the code (results will also be discussed in depth in the following section.)

```
# Predictions + MSE on training dataset
predictions <- predict(my_model, subset_data)
mse <- mean((subset_data$stars - predictions)^2)
cat("The Mean Squared Error is: ", mse, "\n")
```

```
## The Mean Squared Error is:  0.3680304
```

Running Model on Test Dataset

I then fed the model the Test Dataset in order to evaluate its ‘final’ results - the code and results are below, but once again, the results will be discussed in the next section in further detail).

```
# Preparing test dataset
test_combined_data <- merge(new_test_dataset, user_data_small, by = "user_id")
test_combined_data$yelping_since <- as.Date(test_combined_data$yelping_since)
test_combined_data$years_yelping <- as.numeric(format(Sys.Date(), "%Y")) - as.numeric(format(test_combined_data$y
elping_since, "%Y"))
test_combined_data <- test_combined_data[, !(names(test_combined_data) %in% c("name", "yelping_since", "elite",
"friends", "text", "date"))]

# Making Predictions
test_dataset_predictions <- predict(my_model, test_combined_data)

# Examining Predictions + Calculating MSE on test dataset
test_mse <- mean((test_combined_data$stars - test_dataset_predictions)^2)
cat("Test Mean Squared Error is: ", test_mse, "\n")
```

```
## Test Mean Squared Error is:  1.376604
```

```
head(test_dataset_predictions)
```

```
##        1        2        3        4        5        6
## 4.901218 4.289869 4.113917 3.631539 3.541831 3.835250
```

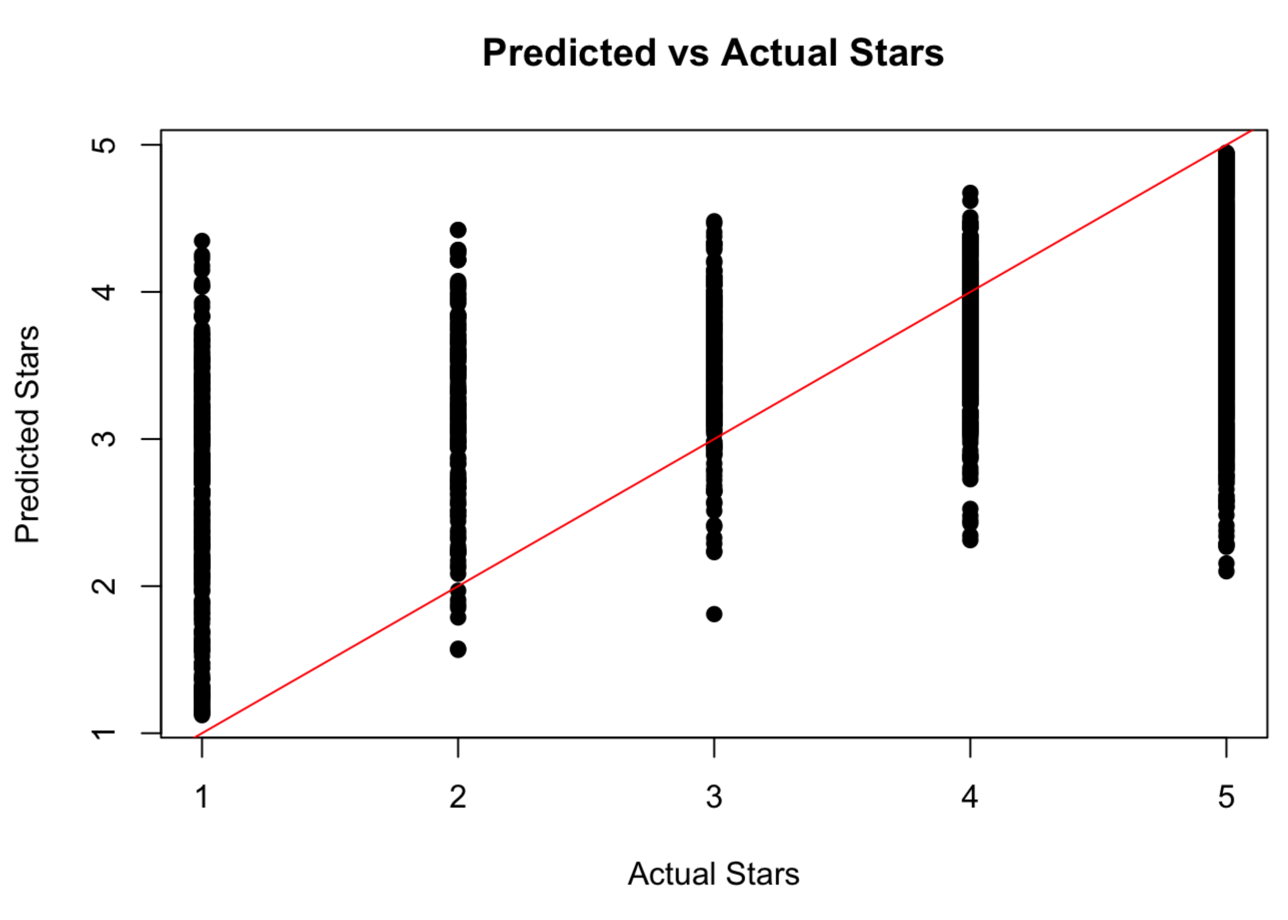
```
comparison_df <- data.frame(Actual = test_combined_data$stars, Predicted = test_dataset_predictions)
head(comparison_df)
```

```
##   Actual Predicted
## 1      5  4.901218
## 2      5  4.289869
## 3      4  4.113917
## 4      3  3.631539
## 5      5  3.541831
## 6      2  3.835250
```

```
summary(test_dataset_predictions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.123   3.201   3.677   3.584   4.076   4.947
```

```
plot(comparison_df$Actual, comparison_df$Predicted, main = "Predicted vs Actual Stars", xlab = "Actual Stars", yl
ab = "Predicted Stars", pch = 19)
abline(0, 1, col = "red")
```



Analysing Results

Training Dataset Results

When fed the training dataset, the model achieved (to 3 significant figures)… 1) Mean of Squared Residuals = 1.35 2) Mean Squared Error = 0.370 3) Percentage of Variance Explained = 38.310

I initially had a much higher MSE and lower % of Variance explained, so decided to add the categorical variables that I had initially discarded, to help improve the accuracy of the model, whcih resulted in the values shown above,

Test Dataset Results

When fed the test dataset, the model achieved (to 3 significant figures)… 1) Mean Squared Error = 1.386 2) Prediction Range = 1.158 - 4.978

Interpretation of Results

Discussion of Results

The MSE is 1.386, meaning that on average, the squared difference between the predicted stars and the actual stars is about 1.39. Since the MSE isn’t far from 1, this suggests that the model’s predictions are ‘reasonably’ close to the actual values, although it needs to be said that there’s definitely room for improvement, especially provided with more computational power (and perhaps more advanced coding knowledge!).

The predictions range from a minimum of about 1.16 to a maximum of approximately 4.98; this suggests that my model is perhaps behaving conservatively - it seems to avoid predicting extreme ratings.

The scatter plot is showcasing the relationship between actual and predicted stars. The red line informs us of a perfect prediction. Ideally, we would like the points to be closer to the red line, indicating better prediction accuracy. We can see noticeable vertical dispersion for each star category, highlighting the variance in the predictions. Again, the predictions seem more concentrated around the mean, suggesting the model doesn’t tend to predict extreme values very often.

Interpretation Summary

Overall, the mode has a moderate level of prediction accuracy, with a tendency to avoid extreme results, and perhaps under-predicting them also (as can be seen from the head of my comparison dataframe). There is a consistent spread in predictions across the range of actual star values, indicating some level of prediction error across the board. The model’s conservative nature in predicting extremes may be due to a lack of distinctive features that differentiate extreme ratings from more moderate ones, or perhaps due to the very nature of the randomForest algorithm, which can average out predictions.

Improvement for Future Reference

One immediate way to improve the model that comes to mind is to increase the complexity of the model, which may enable us to capiture more variance. However, I ultimately chose not pursue this route due to a risk of overfitting.

Final Comments

Thanks for reading - hope you had as much fun as I had writing the code!