Text Analytics Notes - Using Sentiment Analysis for Regression

Promothesh Chatterjee*

Using Sentiment Analysis for Regression

Creating the dataset

Regression analysis (in various forms) is a workhorse of data analysis projects. One major problem with text data is how to use it as an independent variable in regression analysis. While there are many ways that we will discuss later in the course, here we will use sentiment scores as a predictor in a regression model. Let's say we want to predict ratings provided by these consumers based on the text reviews and other variables such as browser used etc.

Let's first load the data and create a column specifying the names of the hotels in the reviews.

```
library(tidyverse)
library(sentimentr)

# Load up the .CSV data and explore in RStudio.
hotel_raw <- read_csv("C:/Users/u0474728/Dropbox/Utah Department Stuff/Teaching/Text Analysis/Summer 20

hotel_raw <- hotel_raw %>%
    mutate(hotel_name = case_when(str_detect(Description, "Hilton")~ "Hilton",
                                  str_detect(Description, "Hyatt")~ "Hyatt",
                                  str_detect(Description, "Marriott")~ "Marriott"))
hotel_sub <- hotel_raw %>% filter(hotel_name=="Hilton"|hotel_name=="Hyatt"|hotel_name=="Marriott")
```

Adding a simulated variable- rating

Next, let us simulate some ratings for these hotels. In a real dataset, we will have ratings variable but we are creating the ratings variable synthetically to suit out purpose.

```
# Create a new column rating based on a string condition
hotel_sub <- hotel_sub %>%
        mutate(rating = ifelse(str_detect(hotel_name, "Marriott"),
        round(rnorm(nrow(filter(hotel_sub, str_detect(hotel_name, "Marriott"))),
                    mean = 4.2, sd = .25)), NA)) %>%
        mutate(rating = ifelse(str_detect(hotel_name, "Hilton"),
        round(rnorm(nrow(filter(hotel_sub, str_detect(hotel_name, "Hilton"))),
                    mean = 3.55, sd = .33)), rating)) %>%
        mutate(rating = ifelse(str_detect(hotel_name, "Hyatt"),
        round(rnorm(nrow(filter(hotel_sub, str_detect(hotel_name, "Hyatt"))),
                    mean = 3.25, sd = .35)), rating))
```

This R code is using the `dplyr` and `stringr` packages to manipulate `hotel_sub` data-frame. It's creating a new column `rating` in the data frame, where the rating is a random number that depends on the hotel name. Here's a breakdown of what the code does:

1. `hotel_sub <- hotel_sub %>%`: This is the start of a pipeline operation using the pipe operator (`%>%`). The data frame `hotel_sub` is being passed to the next function in the pipeline.

2. `mutate(rating = ifelse(str_detect(hotel_name, "Marriott"), round(rnorm(nrow(filter(hotel_sub, str_detect(hotel_name, "Marriott"))), mean = 4.2, sd = .25)), NA))`: The `mutate` function is used to create a new column or modify an existing one in a data frame. Here, a new column called `rating` is being created. The `ifelse` function checks if the `hotel_name` contains "Marriott". If it does, the `rating` is assigned a rounded normally distributed random number with a mean of 4.2 and a standard deviation of 0.25. The number of random numbers generated is equal to the number of rows in `hotel_sub` where `hotel_name` contains "Marriott". If `hotel_name` does not contain "Marriott", `NA` is assigned to `rating`.

3. `mutate(rating = ifelse(str_detect(hotel_name, "Hilton"), round(rnorm(nrow(filter(hotel_sub, str_detect(hotel_name, "Hilton"))), mean = 3.55, sd = .33)), rating))`: This line is similar to the previous one, but it checks if `hotel_name` contains "Hilton". If it does, the `rating` is assigned a rounded normally distributed random number with a mean of 3.55 and a standard deviation of 0.33. If `hotel_name` does not contain "Hilton", the existing `rating` is kept.

4. `mutate(rating = ifelse(str_detect(hotel_name, "Hyatt"), round(rnorm(nrow(filter(hotel_sub, str_detect(hotel_name, "Hyatt"))), mean = 3.25, sd = .35)), rating))`: This line checks if `hotel_name` contains "Hyatt". If it does, the `rating` is assigned a rounded normally distributed random number with a mean of 3.25 and a standard deviation of 0.35. If `hotel_name` does not contain "Hyatt", the existing `rating` is kept.

In summary, this code is creating a new `rating` column in the `hotel_sub` data frame. The rating is a random number that depends on the hotel name: if the name contains "Marriott", "Hilton", or "Hyatt", the rating is a normally distributed random number with different means and standard deviations for each hotel chain. If a row doesn't match any of the conditions, its rating will be `NA`.

Using sentimentr library to get sentiment scores

Now, we will use library sentimentr to get a sentiment score for each of the reviews. For how it generates the sentiment score, I will direct you to the author's github page: https://github.com/trinker/sentimentr

```r
abc <- hotel_sub %>%
    get_sentences('Description') %>%
    sentiment_by(by = 'User_ID')
```

Here's a breakdown of what the code does:

1. `abc <- hotel_sub %>%`: This is the start of a pipeline operation using the pipe operator (`%>%`). The data frame `hotel_sub` is being passed to the next function in the pipeline. The result of the pipeline will be stored in the `abc` variable.

2. `get_sentences('Description') %>%`: The `get_sentences()` function is used to split the text in the 'Description' column of the `hotel_sub` data frame into individual sentences. This is done because sentiment analysis is often more accurate when performed at the sentence level rather than on larger chunks of text.
3. `sentiment_by(by = 'User_ID')`: The `sentiment_by()` function is used to calculate sentiment scores for each 'User_ID' in the data frame. This function returns a data frame with one row for each 'User_ID', and columns for the average sentiment score, standard deviation of the sentiment scores, and count of sentences.

In summary, this code is performing sentiment analysis on the 'Description' column of the `hotel_sub` data frame, and calculating average sentiment scores for each 'User_ID'. The result is a data frame with sentiment scores that is stored in the `abc` variable.

Creating final dataset for regression analysis

Now, we need to combine the two datasets so that we can use the predictors for regression analysis.

```
joined_data <- inner_join(hotel_sub, abc, by="User_ID")
```

We are using the inner_join function from dplyr to combine the two datasets using the common key "User_ID". We will perform our regression analysis on the dataset joined_data.

Analysis

Let's first perform an ANOVA analysis to see if the dataset that we have created is accurate. Recall we had created the ratings variable such that Marriott had the highest rating, followed by Hilton and then Hyatt.

```
# Perform an ANOVA
model <- aov(rating ~ as.factor(hotel_name), data = joined_data)

# Print the summary of the model
summary(model)
```

```
##                         Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(hotel_name)    2  363.6  181.81   963.3 <2e-16 ***
## Residuals             3251  613.6    0.19
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Calculate the means for each group
group_means <- aggregate(rating ~ as.factor(hotel_name), hotel_sub, mean)

# Print the group means
print(group_means)
```

```
##   as.factor(hotel_name)   rating
## 1                Hilton 3.564482
## 2                 Hyatt 3.236074
## 3              Marriott 4.106383
```

```r
# Perform a Tukey's HSD test
posthoc <- TukeyHSD(model)

# Print the results of the post-hoc test
print(posthoc)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = rating ~ as.factor(hotel_name), data = joined_data)
##
## $`as.factor(hotel_name)`
##                        diff        lwr        upr p adj
## Hyatt-Hilton    -0.3284078 -0.3743156 -0.2824999     0
## Marriott-Hilton  0.5419009  0.5007765  0.5830254     0
## Marriott-Hyatt   0.8703087  0.8219746  0.9186428     0
```

This R code is performing an Analysis of Variance (ANOVA) to test if there are significant differences in the `rating` between different `hotel_name` groups. Here's a breakdown of what the code does:

1. `model <- aov(rating ~ as.factor(hotel_name), data = joined_data)`: This line fits an ANOVA model using the `aov()` function. The model is predicting `rating` based on `hotel_name`. The `as.factor(hotel_name)` part is converting `hotel_name` to a factor variable, which is necessary because ANOVA is used to compare means across different groups or categories. The fitted model is stored in the `model` variable.

2. `summary(model)`: This line prints a summary of the fitted ANOVA model, which includes the degrees of freedom, sum of squares, mean square, F statistic, and p-value for each factor in the model.

3. `group_means <- aggregate(rating ~ as.factor(hotel_name), hotel_sub, mean)`: This line calculates the mean `rating` for each `hotel_name` group. The `aggregate()` function is used to apply a function (in this case, `mean`) to subsets of the data frame `hotel_sub` defined by `hotel_name`.

4. `print(group_means)`: This line prints the calculated group means. Obviously, the group means are similar to what we created.

5. `posthoc <- TukeyHSD(model)`: This line performs a Tukey's Honestly Significant Difference (HSD) test on the fitted ANOVA model. This is a post-hoc test that is used to determine which specific groups' means (if any) are statistically significantly different from each other.

6. `print(posthoc)`: This line prints the results of the Tukey's HSD test, which includes the differences in means, lower and upper bounds of the confidence intervals, and adjusted p-values for each pair of groups.

In summary, this code is testing if there are significant differences in the `rating` between different `hotel_name` groups, and if so, which specific groups are significantly different from each other.

Linear Regression without sentiment score First, let's check if predictors such as Is_Response (happy vs. unhappy), hotel_name, word_count(word counts in the reviews) have any influence on rating. Remember, this is a synthetic dataset for illustration of regression using sentiment scores, hence the results may not make much sense.

```r
# Run a linear regression
model1 <- lm(rating ~ Is_Response+ hotel_name +word_count, data = joined_data)

# Print the summary of the model
summary(model1)
```

```
##
## Call:
## lm(formula = rating ~ Is_Response + hotel_name + word_count,
##     data = joined_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2613 -0.2425 -0.1098  0.4295  1.4557
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.561e+00  1.590e-02 223.895   <2e-16 ***
## Is_Responsenot happy -2.722e-02  1.594e-02  -1.707   0.0878 .
## hotel_nameHyatt      -3.296e-01  1.958e-02 -16.833   <2e-16 ***
## hotel_nameMarriott    5.414e-01  1.755e-02  30.855   <2e-16 ***
## word_count            6.706e-05  4.658e-05   1.439   0.1501
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4343 on 3249 degrees of freedom
## Multiple R-squared:  0.3729, Adjusted R-squared:  0.3721
## F-statistic:   483 on 4 and 3249 DF,  p-value: < 2.2e-16
```

Here's a breakdown of what each part does:

- `lm()` is the function used to fit linear models in R.

- `rating ~ Is_Response + hotel_name + word_count` is the model formula. It specifies that the dependent variable (the variable we're trying to predict) is `rating`, and the independent variables (the variables we're using to make the prediction) are `Is_Response`, `hotel_name`, and `word_count`. The `~` symbol can be read as "is modeled as a function of".
- `model1 <- lm(...)` saves the result of the `lm()` function (i.e., the fitted model) to the variable `model1`.

After fitting the model, `summary(model1)` is then used to print a summary of the model. The summary includes:

- The coefficients of the model, which are the estimated parameters of the model. For each independent variable, the coefficient represents the expected change in the dependent variable for a one-unit change in the independent variable, assuming all other variables are held constant.
- The standard errors of the coefficients, which measure the variability in the coefficient estimates.
- The t-values and p-values for the hypothesis tests on the coefficients. The null hypothesis is that the true coefficient is zero (i.e., the independent variable has no effect on the dependent variable). If the p-value is less than your chosen significance level (commonly 0.05), you can reject the null hypothesis and conclude that the independent variable has a statistically significant effect on the dependent variable.
- The residuals, which are the differences between the observed and predicted values of the dependent variable.
- Various statistics that can be used to assess the fit of the model, such as the R-squared and F-statistic.

Please note that since `hotel_name` is a categorical variable, R will automatically create dummy variables for each level of the variable. A dummy variable is a binary variable that indicates whether a certain level is present.

The output reveals that hotel_nameHyatt and hotel_Marriott are significant predictors of rating. The Adjusted R-squared: 0.3938 tells us that almost 40% of variance in the DV is explained by the model.

Linear Regression with sentiment score

We will run the same analysis now including the sentiment score as a predictor. If this were a real dataset we probably would have seen an improvement in the model Adjusted R-squared.

```r
# Run a linear regression
model2 <- lm(rating ~ Is_Response+ hotel_name+word_count +ave_sentiment, data = joined_data)

# Print the summary of the model
summary(model2)
```

```
##
## Call:
```

```
## lm(formula = rating ~ Is_Response + hotel_name + word_count +
##     ave_sentiment, data = joined_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2605 -0.2443 -0.1105  0.4281  1.4546
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.547e+00  2.309e-02 153.625   <2e-16 ***
## Is_Responsenot happy -1.787e-02  1.940e-02  -0.921    0.357
## hotel_nameHyatt      -3.297e-01  1.958e-02 -16.835   <2e-16 ***
## hotel_nameMarriott    5.414e-01  1.755e-02  30.851   <2e-16 ***
## word_count            7.262e-05  4.705e-05   1.544    0.123
## ave_sentiment         5.045e-02  5.958e-02   0.847    0.397
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4343 on 3248 degrees of freedom
## Multiple R-squared:  0.3731, Adjusted R-squared:  0.3721
## F-statistic: 386.5 on 5 and 3248 DF,  p-value: < 2.2e-16
```