DS4100 Spring 2019 : Yelp Help

Vivian Lee and Kristy Chen

2019-04-17

Problem Statement

Many people use Yelp to find new restaurants and try the latest food trends. In fact, Yelp's mission statement is "to connect people with great local businesses." While the search service has certainly succeeded in allowing users to find highly rated businesses tailored to their needs, it often hurts businesses that are assigned lower ratings and consequently do not appear in the first few returned search result pages. These businesses are often written off as bad options by potential customers based on their poor ratings or ignored by users who refuse to give them a chance.

Our goal was to use data collected from Yelp to help restaurants determine actionable changes they can implement to improve their Yelp ratings, which would have a direct influence on their sales. We collected data from restaurants within different niches, where a niche was defined by the restaurants returned for a specific keyword, location and price range search combination.

For our project, we analyzed pizza places with a price range of $$ in Boston and pizza places with a price range of $$ in New York City. Our goals were to define which features were most important for restaurants to attain a high Yelp rating for these individual niches, then compare the differences between desired features for pizza places in Boston vs New York.

Data Collection, Analysis, and Storage Process

Create the Yelp Help SQLite database

```
file_name <- "YelpHelp.sqlite"
db <- dbConnect(SQLite(), dbname=file_name)
dbSendQuery(conn = db, "pragma foreign_keys=on;")
```

```
## <SQLiteResult>
##   SQL  pragma foreign_keys=on;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

Business Info Table

The naming convention for the business info table of a given niche is its keyword, location, and price level, separated by underscores. For example, for pizza places in Boston with a price level of $$, the business info table would be named pizza_Boston_2.

```
# create and store the given business information to a table for the specified niche
add_business_info_sqlite <- function(niche, df){
 sqlStatement <- paste0("CREATE TABLE ", niche, " (
            business_id TEXT PRIMARY KEY,
            alias TEXT, name TEXT, rating INTEGER, review_count INTEGER, url TEXT,
            category_aliases TEXT, category_titles TEXT, transactions TEXT,
            Liked_by_Vegans TEXT, Liked_by_Vegetarians TEXT, Takes_Reservations TEXT,
            Delivery TEXT, Take_out TEXT, Accepts_Credit_Cards TEXT, Good_for_Kids TEXT,
            Good_for_Groups TEXT, Outdoor_Seating TEXT, Wi_Fi TEXT, Caters TEXT,
```

```
            Has_TV TEXT, Ambience TEXT, Noise_Level TEXT, Has_menu_link TEXT)
            WITHOUT ROWID")
 dbSendQuery(conn = db,  sqlStatement)
 dbWriteTable(conn = db, name = niche, value = df, row.names=FALSE, append=TRUE)
}
```

Business Customer Reviews Table

The naming convention for the customer reviews table of a given niche is its name for the business info table, concatenated with the word 'reviews'. Continuing off of the previous example, the table would be named pizza_Boston_2_reviews.

```
# create the reviews table for the specified niche
# review_date will be stored as the number of days since January 1, 1970
create_reviews_table_sqlite <- function(niche) {
  table_name <- str_c(niche, "reviews", sep="_")
  sql_statement <- sprintf("CREATE TABLE %s(
                           business_id TEXT, review_id INT, review_date TEXT,
                           review_stars INT, review_text TEXT, user_num_reviews INT,
                           user_elite_status INT,
                           PRIMARY KEY(business_id, review_id),
                           FOREIGN KEY(business_id) REFERENCES %s(business_id))
                           WITHOUT ROWID", table_name, niche)

  sql_statement <- sql_statement %>%
    str_replace_all("\n", "") %>%
    str_replace_all("\\s{2,}", "")

  dbSendQuery(conn = db, sql_statement)
}
```

Web Scraping

Retrieve and Store Business Info for a Given Niche

We retrieved business information through Yelp API calls and web scraping.

Using the business_search function from the richierocks/yelp R package that connects to the Yelp API only returns a limited subset of business features. This is a restriction enforced by Yelp's API, not the R package itself. Therefore, we used the rvest package to scrape additional business info that we felt would be relevant to our analysis. This additional information was initially hidden from view when the Yelp page was loaded via its url. To retrieve this information, we needed to click a button that would expand the info and dynamically load it to the page.

Rvest does not have the capabilities of clicking the button to show more data. Further research into this problem led us to use RSelenium. RSelenium is a package that can be used to interact with the website, including selecting page elements which enable us to scrape dynamically loaded data. Using RSelenium requires `library(RSelenium)` and chrome browser version 73, to use the chrome webdriver.

We decided to retrieve information for only the first 800 restaurants for each niche for 3 mains reasons: because we believe 800 is a sizeable amount of restaurants to examine, it takes time to scrape information for each restaurant, and trying to scrape too many restaurants at once could lead to complications with Yelp for web scraping data from their site.

Moreover, there were many additional business features for each restaurant and not every restaurant had the same features listed. We chose to limit our scraping to the following features: Liked by Vegans, Liked by Vegetarians, Takes Reservations, Delivery, Take-out, Accepts Credit Cards, Good for Kids, Good for Groups, Outdoor Seating, Wi-Fi, Caters, Has TV, Ambience, Noise Level, Has menu link. We chose these features because we believe they are most popular and present among the majority of restaurants on Yelp.

Originally, we planned to use Rvest (without RSelenium) if the additional business info column appeared on the page and did not require us to click the button to view the additional feature values. This would have sped up the scraping process so we would not need to visit every page with RSelenium, which can be rather slow.

However, since read_html parses the html page before it completely loads, sometimes the additional business info column would initially show up but be missing information that depended on dynamic page scripts. Therefore, we decided to use RSelenium on each page search to avoid the risk of missing data that would need additional time to load.

```
# use the Yelp Fusion API to get business information by keyword, location, price level
# of total_amount of businesses
# total_amount parameter is a multiple of 50.
get_business_info <- function(keyword, location, price_point, total_amount) {
  result <- data.frame(matrix(ncol = 25, nrow = 0))
  for(result_offset in seq(50,total_amount,50)) {
    Sys.sleep(0.5) # wait
    result <- rbind(result,
                    business_search(keyword, location, limit=50,
                                    price=price_point, offset=result_offset))
  }
  result <- result %>% dplyr::select(business_id, alias, name, rating, review_count, url,
                                     category_aliases, category_titles, transactions)
  result$url <- str_extract(result$url, "[^?]*")
  #collapse vectors into a single string separated by commas to store in sqlite
  # sqlite does not stores lists
  result$category_aliases <- sapply(result$category_aliases, paste0, collapse=",")
  result$category_titles <- sapply(result$category_titles, paste0, collapse=",")
  result$transactions <- sapply(result$transactions, paste0, collapse=",")

  return(result)
}

# use revest and rselenium to scrape more business info that wasn't provided through
# Yelp Fusion API call
get_more_business_info <- function(df) {
  info_list <- c("Liked by Vegans", "Liked by Vegetarians", "Takes Reservations",
                 "Delivery", "Take-out", "Accepts Credit Cards", "Good for Kids",
                 "Good for Groups", "Outdoor Seating", "Wi-Fi", "Caters", "Has TV",
                 "Ambience", "Noise Level", "Has menu link")

  #selenium set up
  rD <- rsDriver(port=8087L,browser="chrome", check = FALSE, chromever = "73.0.3683.68")
  remDr <- rD[["client"]]

  # dataframe for more business info
  more_info <- data.frame(matrix(ncol = 16, nrow = 0))
  colnames(more_info) <- c("business_id", str_replace_all(info_list,'[\\s-]+','_'))
```

```
for(business in seq_along(df$url)){
  remDr$navigate(df$url[business])
  Sys.sleep(5) # wait for page to load
  page <- read_html(remDr$getPageSource()[[1]]) %>% html_node('body')
  # if expand button exists then it requires click, otherwise scrape normally
  expand_button_xpath <-
    "//div[contains(@data-hypernova-key, 'yelp_main__MoreBusinessInfoV2')]//a"
  expand_button <- page %>%
    html_nodes(xpath=expand_button_xpath)
  if(length(expand_button) <= 0) {
    info_categories <- page %>%
      html_nodes('.short-def-list dl dt') %>%
      html_text(trim=T)
    info_has <- page %>%
      html_nodes('.short-def-list dl dd') %>%
      html_text(trim=T)
    } else {
      if(remDr$findElement("xpath", expand_button_xpath)$isElementDisplayed()[[1]]) {
        remDr$findElement("xpath",expand_button_xpath)$clickElement()
        Sys.sleep(0.5) # wait for items to show
      }
      page <- read_html(remDr$getPageSource()[[1]]) %>% html_node('body')
      info_categories <- page %>%
        html_nodes(xpath='//div[contains(@data-hypernova-key,"yelp_main__MoreBusinessInfoV2")]
                  //span[contains(@class,"text__373c0__2pB8f")][1]') %>%
        html_text(trim=T)
      info_has <- page %>%
        html_nodes(xpath='//div[contains(@data-hypernova-key,"yelp_main__MoreBusinessInfoV2")]
                  //span[contains(@class,"text__373c0__2pB8f")][2]') %>%
        html_text(trim=T)
    }
  # at times some businesses are missing extra info
  if(length(info_categories) > 0) {
    result <- c()
    for(info in seq_along(info_list)-1) {
      result[info] <- info_has[match(info_list[info], info_categories)]
    }
    result<- c(df$business_id[business], result)
    # check for menu link
    has_menu <- page %>%
      html_nodes('.menu-explore, .biz-contact-info_view-menu, .external-menu') %>%
      html_text(trim=T)
    result[16] <- ifelse(length(has_menu) != 0, "Yes", "No")

    more_info[business,] <- result
  }
}

# shutdown chromedriver
remDr$close()
rD[["server"]]$stop()

return(more_info)
```

```
}

# retrieve the business info for each restaurant in the specified niche combination,
# create a SQLite table for the specified niche, and store the retrieved info
get_save_business_info <- function(keyword, location, price_point){
  name <- str_c(keyword, str_extract(location, "([^,]+)"), price_point, sep="_") %>%
     str_replace_all('[\\s-]+','_')
  info <<- get_business_info(keyword, location, price_point, 800)
  more_info <<- get_more_business_info(info)
  complete_info <<- left_join(info, more_info, by="business_id")
  add_business_info_sqlite(name, complete_info)
}
```

Retrieve and Store Business Reviews for all Businesses in a Given Niche

We limited our retrieval of the reviews data to the reviews on the first page of each business since Yelp has preventative measures in place to block site visitors from web scraping all of the reviews data from many businesses at once.

Initially, we attempted to scrape all of the reviews data for 800 businesses at once. Unfortunately, since some restaurants had over a thousand reviews, we were flagged by Yelp after scraping for 2 hours and collecting 12,000 reviews (48 restaurants). We were temporarily banned from their site. After communicating with a Yelp representative, and lifting the ban, we decided to limit our scraping to the first page of reviews per business to avoid being permanantly blocked from the site.

```
# returns whether the given node contain the specified css class
node_contains_class <- function(node, class) {
  !is.na(node %>%
    html_node(css = class))
}

# returns a dataframe of review data for the given business on the given review page url
get_reviews_for_business <- function(yelp_url, business_id) {

  page <- read_html(yelp_url)
  root <- html_node(page, 'body')
  root_descendants <- html_children(root)
  review_nodes <- root_descendants %>%
    html_nodes(css = ".reviews > li + li")

  review_text <- review_nodes %>%
    html_nodes(css = ".review-content p") %>%
    html_text()

  review_date <- review_nodes %>%
    html_node(css = ".rating-qualifier") %>%
    html_text() %>%
    str_trim() %>%
    str_extract("[\\d/]*") %>%
    mdy()

  review_stars <- review_nodes %>%
    html_node(css = ".rating-large") %>%
    html_attr("title") %>%
```

```
    str_extract("[\\d]") %>%
    as.numeric()

  user_nodes <- review_nodes %>%
    html_nodes(css = ".user-passport-stats")

  user_num_reviews <- user_nodes %>%
    html_nodes(css = ".review-count") %>%
    html_text() %>%
    str_trim() %>%
    str_extract("[\\d]*") %>%
    as.numeric()

  # binarizes user elite status
  user_elite_status <- map_lgl(user_nodes, node_contains_class, class=".is-elite")

  # returns df of review data for the given review page url
  results <- distinct(as.data.frame(cbind(review_date, review_stars, review_text,
                                  user_num_reviews, user_elite_status)))
  results <- distinct(results)
  num_results <- results %>% nrow()

  # check if business has no reviews
  if (num_results > 0) {
    business_id_col <- rep(business_id, num_results)
    review_id_col <- 1:num_results
    reviews <- cbind(results, business_id = business_id_col, review_id = review_id_col)
  } else {
    data.frame()
  }
}

# get and store reviews for all businesses in the given niche to SQLite database
get_save_business_reviews <- function(keyword, location, price_point) {

  # create reviews table for the given niche if it does not exist
  niche <- str_c(keyword, str_extract(location, "([^,]+)"), price_point, sep="_") %>%
    str_replace_all('[\\s-]+','_')
  if (!dbExistsTable(db, niche)) create_reviews_table_sqlite(niche)

  # get restaurant urls and business_ids
  get_bid_urls_query <- sprintf("SELECT business_id, url FROM %s", niche) %>% str_trim()
  restaurant_bid_urls_df <- dbGetQuery(conn = db, get_bid_urls_query)

  # create dataframe of results for all restaurants in the niche
  urls <- restaurant_bid_urls_df$url
  business_ids <- restaurant_bid_urls_df$business_id
  n <- nrow(restaurant_bid_urls_df)
  range <- 1:n

  # retrieve and store up to 20 reviews for each business in the niche
  # limited to 20 reviews per business due to Yelp API web scraping restrictions
  for (i in range) {
```

```
    results <- get_reviews_for_business(urls[i], business_ids[i])

    # save reviews data for the business to niche reviews SQLite table
    dbWriteTable(conn = db,
                 name = str_c(niche, "reviews", sep="_"),
                 value = results, row.names=FALSE, append=TRUE)

    # add wait time in between API calls for each business
    Sys.sleep(2)
  }
}

# retrieves relevant reviews data from the reviews table to perform additional analyses
# for individual restaurants and create new features to add to our model
get_analyses_relevant_reviews_data <- function(reviews_table_name) {
  select_query <- sprintf("SELECT business_id, review_id, review_date,
                          user_elite_status FROM %s",
                          reviews_table_name)
  dbGetQuery(conn = db, select_query)
}

# save the new calculated/analyzed data to the niche business_info SQLite table
save_business_review_analyses_cols <- function(keyword, location, price_point) {

  # niche table name
  niche_table_name <- str_c(keyword, str_extract(location, "([^,]+)"),
                            price_point, sep="_") %>% str_replace_all('[\\s-]+','_')

  # reviews table name
  reviews_table_name <- str_c(niche_table_name, "reviews", sep = "_")

  # retrieve the reviews table data
  reviews <- get_analyses_relevant_reviews_data(reviews_table_name)

  base_year = mdy("01-01-1970")
  one_year_ago = today() - 365

  # calculate percent elite reviewers and percent of reviews written in past year
  # per restaurant
  businesses_with_review_analyses_cols <- reviews %>%
    group_by(business_id) %>%
    summarise(
      perc_elite_reviewers = round(
        sum(user_elite_status %>% as.logical()) / n(), digits = 2),
      perc_reviews_in_past_year = round(
        sum(review_date %>% as.numeric() + base_year > one_year_ago) / n(),
        digits = 2)
    )

  businesses_perc_elite_df <- businesses_with_review_analyses_cols %>%
    dplyr::select(business_id, perc_elite_reviewers)

  businesses_perc_recent_reviews_df <- businesses_with_review_analyses_cols %>%
```

```
    dplyr::select(business_id, perc_reviews_in_past_year)

  # add perc_elite_reviewers to sqlite table

  # create new column in niche sqlite table for perc_elite_reviewers if it doesn't exist
  query_elite <- sprintf("ALTER TABLE %s ADD perc_elite_reviewers REAL",
                         niche_table_name)
  if (!("perc_elite_reviewers" %in% dbListFields(db, niche_table_name))) {
    dbSendQuery(conn = db, query_elite)
  }

  # update perc_elite_reviewers for each business in the sqlite db
  apply(businesses_perc_elite_df, 1,
        update_business_perc_elite, table_name = niche_table_name)


  # add perc_reviews_in_past_year to sqlite table

  # create new column in niche sqlite table for perc_reviews_in_past_year
  # if it doesn't exist
  query_recent_reviews <- sprintf("ALTER TABLE %s ADD perc_reviews_in_past_year REAL",
                         niche_table_name)
  if (!("perc_reviews_in_past_year" %in% dbListFields(db, niche_table_name))) {
    dbSendQuery(conn = db, query_recent_reviews)
  }

  # update perc_reviews_in_past_year for each business in the sqlite db
  apply(businesses_perc_recent_reviews_df, 1,
        update_business_perc_recent_reviews, table_name = niche_table_name)
}

# update default column value of perc_elite_reviewers in business_info table
update_business_perc_elite <- function(x, table_name) {
  business_id <- x[1]
  perc_elite_reviewers <- x[2]

  if (is.na(perc_elite_reviewers)) {
    # don't need to update if there is no value available for perc_elite_reviewers
  } else {
    update_query <-
        sprintf("UPDATE %s SET perc_elite_reviewers = %s WHERE business_id == '%s'",
                table_name, perc_elite_reviewers, business_id)
      result <- dbSendQuery(conn = db, update_query)
      dbClearResult(result)
  }
}

# update default column value of perc_reviews_in_past_year in business_info table
update_business_perc_recent_reviews <- function(x, table_name) {
  business_id <- x[1]
  perc_recent_reviews <- x[2]

  if (is.na(perc_recent_reviews)) {
```

```
    # don't need to update if there is no value available for perc_recent_reviews
  } else {
    update_query <-
        sprintf("UPDATE %s SET perc_reviews_in_past_year = %s
                WHERE business_id == '%s'",
                table_name, perc_recent_reviews, business_id)
      result <- dbSendQuery(conn = db, update_query)
      dbClearResult(result)
  }
}
```

Sentiment Analysis of Business Reviews Text

We analyzed the text from each of the restaurant reviews we collected and performed sentiment analysis to calculate two new features:

1. avg_perc_positive_sentiment

This feature represents a sentiment score based on unigrams, where each word is analyzed individually. For this feature, we broke up the text of each customer review into its individual word components, removed the stopwords, and calculated the percentage of the remaining words that had a positive sentiment according to the Bing lexicon. Then we grouped the percent positive sentiment score for each review by the restaurant the review was written for (the business_id) and calculated the average percent positive sentiment score for each restaurant.

2. avg_bigrams_sentiment_score

Sentiment analysis on unigrams (individual words) can sometimes lead to inaccurate sentiment scores. For example, a review that states "This pizza place is not bad" is actually positive. However, splitting up the sentence into its individual components and removing the stopwords would leave us with the word "bad", and the review would be interpreted as having a negative sentiment. Therefore, we also analyze bigrams (pairs of words) in the reviews text to catch this error.

We gave scores to adverbs such as ("not", "very", "extremely", and "quite") which, as stopwords, would normally be removed when performing sentiment analysis. We kept these words because of the strong influence they can have on the sentiment of a review. When an adverb was paired with a word having positive (+1) or negative sentiment (-1) (according to the Bing lexicon), we multiplied the scores to account for the effect these adverbs have on the review sentiment.

For example, the review "This pizza place is not bad" would have the bigram ("not", "bad") with scores (-1, -1) and report a sentiment score of +1. Conversely, the review "This pizza place is not good" would have the bigram ("not", "good") with scores (-1, 1) and report a sentiment score of -1. Lastly, the review "The food here is extremely delicious" would have the bigram ("extremely", "delicious") with scores (2, 1) and report a sentiment score of +2.

We calculate the score for each bigram in the review, where the first word is either an adverb or a non-stopword and the second word is a non-stopword. The sum of these scores is the bigrams sentiment score for the review and we take the average of the bigrams review scores to get the average bigrams sentiment score for each restaurant. Since the largest avg_bigrams_sentiment_score had a very small value relative to the other numeric feature values, we normalize the avg_bigrams_sentiment_score using min-max normalization.

```
# retrieve the reviews table data to analyze sentiment for the specified
# restaurant niche
get_sentiment_relevant_reviews_data <- function(reviews_table_name) {
  select_query <- sprintf("SELECT business_id, review_id, review_text, review_stars
                          FROM %s",
                          reviews_table_name)
```

```
  dbGetQuery(conn = db, select_query)
}
```

Sentiment Analysis based on Unigrams

```
# input: reviews dataframe (business_id, review_id, review_text, review_star)
# output: dataframe containing row per non-stopword of each review
clean_reviews_sentiment_analysis_unigrams <- function(reviews) {
  # tidy reviews text
  (cleaned_reviews_unigrams <- reviews %>%
    unnest_tokens(word, review_text) %>%
    filter(!word %in% stop_words$word))
}
```

```
# input: reviews dataframe where each row contains a business_id, review_id,
#        review_stars, and word from the review text
# output: dataframe containing pos_word_count, neg_word_count, normalized
#         perc_positive_sentiment per review
get_sentiments_per_review_unigrams <- function(reviews) {

  # get sentiment words
  sentiments <- get_sentiments("bing")
  positive_words <- sentiments %>% filter(sentiment == "positive") %>% unlist()
  negative_words <- sentiments %>% filter(sentiment == "negative") %>% unlist()

  cleaned_reviews <- clean_reviews_sentiment_analysis_unigrams(reviews)

  # calculate sentiment metrics per review
  cleaned_reviews %>%
    mutate(pos = word %in% positive_words) %>%
    mutate(neg = word %in% negative_words) %>%
    group_by(business_id, review_id) %>%
    summarise(
      pos_count = sum(pos),
      neg_count = sum(neg)
    ) %>%
    mutate(perc_positive_sentiment = round(pos_count / (pos_count + neg_count),
                                           digits = 2))
}
```

```
# input: (business id, review id, perc positive sentiment score), table name,
# result: updates the perc_positive_sentiment value in the db for the given
#         business review observation x
update_review_sentiment_unigrams <- function(x, table_name) {

  business_id <- x[1]
  review_id <- x[2]
  perc <- x[3]

  if (is.na(perc)) {
    # don't need to update if there is no value available for perc
  } else {
      update_query <-
```

```
        sprintf("UPDATE %s SET perc_positive_sentiment = %s WHERE business_id == '%s'
                AND review_id == %s",
                table_name, perc, business_id, review_id)
      result <- dbSendQuery(conn = db, update_query)
      dbClearResult(result)
  }
}
```

```
# input: (business_id, avg sentiment), table name
# result: updates the avg_perc_positive_sentiment value in the db for the
#         given business observation x
update_business_avg_sentiment <- function(x, table_name) {
  business_id <- x[1]
  avg_sentiment <- x[2]

  if (is.na(avg_sentiment)) {
    # don't need to update if there is no value available for avg_sentiment
  } else {
    update_query <-
        sprintf("UPDATE %s SET avg_perc_positive_sentiment = %s WHERE business_id == '%s'",
                table_name, avg_sentiment, business_id)
      result <- dbSendQuery(conn = db, update_query)
      dbClearResult(result)
  }
}
```

```
# save calculated sentiment data by unigrams from the given sentiments_per_review
# df to the specified reviews table in sqlite
save_sentiment_per_review_unigrams <- function(sentiments_per_review,
                                               reviews_table_name) {

  # create new column in reviews sqlite table for perc_positive_sentiment
  # if it doesn't exist
  alter_table_query <- sprintf("ALTER TABLE %s ADD perc_positive_sentiment REAL",
                               reviews_table_name)
  if (!("perc_positive_sentiment" %in% dbListFields(conn = db, reviews_table_name))) {
    dbSendQuery(conn = db, alter_table_query)
  }

  # update sentiments_per_review data for each review in the sqlite db
  reviews_sentiments <- sentiments_per_review %>%
    dplyr::select(business_id, review_id, perc_positive_sentiment)

  apply(reviews_sentiments, 1, update_review_sentiment_unigrams,
        table_name = reviews_table_name)
}

# save the average perc_positive_sentiment per business in sqlite
save_avg_sentiment_per_business_unigrams <- function(avg_sentiment_per_business,
                                                     niche_table_name) {

  # create new column in niche sqlite table for avg_perc_positive_sentiment if it
  # doesn't exist
  alter_table_query <- sprintf("ALTER TABLE %s ADD avg_perc_positive_sentiment REAL",
```

```
                                niche_table_name)
  if (!("avg_perc_positive_sentiment" %in% dbListFields(db, niche_table_name))) {
    dbSendQuery(conn = db, alter_table_query)
  }

  # update avg sentiment for each business in the sqlite db
  apply(avg_sentiment_per_business, 1, update_business_avg_sentiment,
        table_name = niche_table_name)
}
```

```
calculate_save_unigrams_sentiments <- function(reviews_data, reviews_table,
                                               niche_table) {
  # calculate sentiment metrics per review based on unigrams
  sentiments_per_review_unigrams <- get_sentiments_per_review_unigrams(reviews_data)

  # store perc_positive_sentiment per review to sqlite reviews table
  save_sentiment_per_review_unigrams(sentiments_per_review_unigrams, reviews_table)

  # calculate average pos perc sentiment score (unigrams) per business
  avg_sentiment_per_business <- sentiments_per_review_unigrams %>%
    group_by(business_id) %>%
    summarise(
      avg_perc_positive_sentiment = round(mean(perc_positive_sentiment, na.rm=TRUE),
                                          digits = 2)
    )

  # store avg_sentiment per business to sqlite niche table
  save_avg_sentiment_per_business_unigrams(avg_sentiment_per_business, niche_table)
}
```

Sentiment Analysis based on Bigrams

```
clean_reviews_sentiment_analysis_bigrams <- function(reviews) {
  (cleaned_reviews_bigrams <- reviews %>%
     unnest_tokens(bigram, review_text, token="ngrams", n=2) %>%
     separate(bigram, c("word1","word2"), sep = " ") %>%
     filter(!word1 %in% stop_words$word |
              word1 %in% c("not", "very", "really", "extremely", "quite")) %>%
     filter(!word2 %in% stop_words$word))
}
```

```
# input: reviews dataframe where each row contains a business_id, review_id,
#        review_stars, and word from the review text
# output: dataframe containing normalized bigrams_sentiment score per review
get_sentiments_per_review_bigrams <- function(reviews) {

  # get sentiment words
  sentiments <- get_sentiments("bing")
  positive_words <- sentiments %>% filter(sentiment == "positive") %>% unlist()
  negative_words <- sentiments %>% filter(sentiment == "negative") %>% unlist()

  # calculate sentiment for each bigram (pair of words)
  cleaned_reviews <- clean_reviews_sentiment_analysis_bigrams(reviews) %>%
    mutate(sent1 = map_dbl(word1, get_sentiment_score,
```

```
                        pos = positive_words, neg = negative_words)) %>%
    mutate(sent2 = map_dbl(word2, get_sentiment_score,
                        pos = positive_words, neg = negative_words)) %>%
    mutate(sent = sent1 * sent2)

  # calculate sentiment metrics based on all bigrams per review
  cleaned_reviews %>%
    group_by(business_id, review_id) %>%
    summarise (
      bigrams_sentiment_score = round(sum(sent) / n(), digits = 2)
    ) %>%
    mutate(bigrams_sentiment_score =
             round((bigrams_sentiment_score - min(bigrams_sentiment_score)) /
                     (max(bigrams_sentiment_score) - min(bigrams_sentiment_score)),
                   digits = 2))
}

get_sentiment_score <- function(word, pos, neg) {
  if (word %in% c("very", "really", "extremely", "quite")) {
    2
  }
  else if (word %in% positive_words) {
    1
  } else if (word %in% negative_words | word == "not") {
    -1
  } else {
    0
  }
}
```

```
# input: (business id, review id, bigrams sentiment score), table name,
# result: updates the bigrams_sentiment_score value in the db for the given
#         business review observation x
update_review_sentiment_bigrams <- function(x, table_name) {

  business_id <- x[1]
  review_id <- x[2]
  score <- x[3]

  if (is.na(score)) {
    # don't need to update if there is no value available for score
  } else {
      update_query <-
        sprintf("UPDATE %s SET bigrams_sentiment_score = %s WHERE business_id == '%s'
                AND review_id == %s",
                table_name, score, business_id, review_id)
      result <- dbSendQuery(conn = db, update_query)
      dbClearResult(result)
  }
}
```

```
# input: (business_id, avg bigrams sentiment), table name
# result: updates the  value in the db for the given business
# observation x
```

```r
update_business_avg_bigrams_sentiment <- function(x, table_name) {
  business_id <- x[1]
  avg_bigrams_sentiment_score <- x[2]

  if (is.na(avg_bigrams_sentiment_score)) {
    # don't need to update if there is no value available for avg_bigrams_sentiment_score
  } else {
    update_query <-
        sprintf("UPDATE %s SET avg_bigrams_sentiment_score = %s
                WHERE business_id == '%s'",
                table_name, avg_bigrams_sentiment_score, business_id)
      result <- dbSendQuery(conn = db, update_query)
      dbClearResult(result)
  }
}
```

```r
# save calculated sentiment data by bigrams from the given sentiments_per_review
# df to the specified reviews table in sqlite
save_sentiments_per_review_bigrams <- function(sentiments_per_review,
                                               reviews_table_name) {

  # create new column in reviews sqlite table for bigrams_sentiment_score if it
  # doesn't exist
  alter_table_query <- sprintf("ALTER TABLE %s ADD bigrams_sentiment_score REAL",
                               reviews_table_name)
  if (!("bigrams_sentiment_score" %in% dbListFields(conn = db, reviews_table_name))) {
    dbSendQuery(conn = db, alter_table_query)
  }

  # update bigrams_sentiment_score data for each review in the sqlite db
  reviews_sentiments <- sentiments_per_review %>%
    dplyr::select(business_id, review_id, bigrams_sentiment_score)

  apply(reviews_sentiments, 1, update_review_sentiment_bigrams,
        table_name = reviews_table_name)

}

# save the average bigrams_sentiment_score per business in sqlite
save_avg_sentiment_per_business_bigrams <- function(avg_bigrams_score_per_business,
                                                    niche_table_name) {

  # create new column in niche sqlite table for avg_bigrams_ if it doesn't exist
  alter_table_query <- sprintf("ALTER TABLE %s ADD avg_bigrams_sentiment_score REAL",
                               niche_table_name)
  if (!("avg_bigrams_sentiment_score" %in% dbListFields(db, niche_table_name))) {
    dbSendQuery(conn = db, alter_table_query)
  }

  # update avg sentiment for each business in the sqlite db
  apply(avg_bigrams_score_per_business, 1,
        update_business_avg_bigrams_sentiment, table_name = niche_table_name)
}
```

```
calculate_save_bigrams_sentiments <- function(reviews_data, reviews_table,
                                              niche_table) {

  # calculate sentiment metrics per review based on bigrams
  sentiments_per_review_bigrams <- get_sentiments_per_review_bigrams(reviews_data)

  # store bigrams_sentiment_score per review to sqlite reviews table
  save_sentiments_per_review_bigrams(sentiments_per_review_bigrams, reviews_table)

  # calculate average bigrams sentiment score per business
  avg_bigrams_score_per_business <- sentiments_per_review_bigrams %>%
    group_by(business_id) %>%
    summarise (
      avg_bigrams_sentiment_score = round(mean(bigrams_sentiment_score, na.rm=TRUE),
                                          digits = 2)
    )

  # store avg_bigrams_sentiment_score per business to sqlite niche table
  save_avg_sentiment_per_business_bigrams(avg_bigrams_score_per_business, niche_table)
}
```

Calculate and Save Business Review Sentiment Scores for Unigrams and Bigrams

```
# analyze all stored reviews for each business in the specified niche,
# calculate sentiment metrics, and save to sqlite reviews table
analyze_save_business_review_sentiments <- function(keyword, location, price_point){

  # niche table name
  niche_table_name <- str_c(keyword, str_extract(location, "([^,]+)"),
                            price_point, sep="_") %>% str_replace_all('[\\s-]+','_')

  # reviews table name
  reviews_table_name <- str_c(niche_table_name, "reviews", sep = "_")

  # retrieve the reviews table data
  reviews <- get_sentiment_relevant_reviews_data(reviews_table_name)

  # calculate and save avg sentiments per restaurant based on unigrams
  calculate_save_unigrams_sentiments(reviews, reviews_table_name, niche_table_name)

  # calculate and save avg sentiments per restaurant based on bigrams
  calculate_save_bigrams_sentiments(reviews, reviews_table_name, niche_table_name)
}
```

Additional Sentiment Analysis Based on Unigrams

Here we provide some additional sentiment analysis based on unigrams (individual word components in each reviews text).

New information we learned from these analyses include:

- word frequency counts for both positive and negative words in reviews for each rating group (1-5)

- most common positive/negative words used in reviews, searched from the top 100 words in each rating group
- comparisons of positive/negative words found in high rating reviews vs low rating reviews

```
sentiments <- get_sentiments("bing")
positive_words <- sentiments %>% filter(sentiment == "positive") %>% unlist()
negative_words <- sentiments %>% filter(sentiment == "negative") %>% unlist()

clean_reviews <- get_sentiment_relevant_reviews_data("pizza_Boston_2_reviews") %>%
  clean_reviews_sentiment_analysis_unigrams()
```

```
## Warning: Closing open result set, pending rows
```

```
positive_words_reviews <- clean_reviews %>%
  filter(word %in% positive_words)

negative_words_reviews <- clean_reviews %>%
  filter(word %in% negative_words)
```

```
# returns all words from the given clean_df associated with the specified review rating
get_words_from_review_with_rating <- function(clean_df, rating) {
  clean_df %>%
    filter(review_stars == rating) %>%
    dplyr::select(word) %>%
    unlist()
}

# get all positive and negative words used in reviews
all_pos_words <- unique(positive_words_reviews$word)
all_neg_words <- unique(negative_words_reviews$word)

# get all positive and negative words used in reviews with the specified rating
pos_words_1 <- get_words_from_review_with_rating(positive_words_reviews, 1)
pos_words_2 <- get_words_from_review_with_rating(positive_words_reviews, 2)
pos_words_3 <- get_words_from_review_with_rating(positive_words_reviews, 3)
pos_words_4 <- get_words_from_review_with_rating(positive_words_reviews, 4)
pos_words_5 <- get_words_from_review_with_rating(positive_words_reviews, 5)

neg_words_1 <- get_words_from_review_with_rating(negative_words_reviews, 1)
neg_words_2 <- get_words_from_review_with_rating(negative_words_reviews, 2)
neg_words_3 <- get_words_from_review_with_rating(negative_words_reviews, 3)
neg_words_4 <- get_words_from_review_with_rating(negative_words_reviews, 4)
neg_words_5 <- get_words_from_review_with_rating(negative_words_reviews, 5)

pos_words_rating_df <- tibble(word = all_pos_words)

neg_words_rating_df <- tibble(word = all_neg_words)

# dataframe tracking presence of the word in reviews with ratings 1-5
pos_words_rating_df <- pos_words_rating_df %>%
  mutate(in_rating_1 = word %in% pos_words_1) %>%
  mutate(in_rating_2 = word %in% pos_words_2) %>%
  mutate(in_rating_3 = word %in% pos_words_3) %>%
  mutate(in_rating_4 = word %in% pos_words_4) %>%
  mutate(in_rating_5 = word %in% pos_words_5)
```

```r
neg_words_rating_df <- neg_words_rating_df %>%
  mutate(in_rating_1 = word %in% neg_words_1) %>%
  mutate(in_rating_2 = word %in% neg_words_2) %>%
  mutate(in_rating_3 = word %in% neg_words_3) %>%
  mutate(in_rating_4 = word %in% neg_words_4) %>%
  mutate(in_rating_5 = word %in% neg_words_5)
```

```r
# returns a dataframe where each row contains (word, frequency), considering all reviews
# with the given rating
get_word_counts_for_reviews_with_rating <- function(clean_reviews_df, rating) {
  clean_reviews_df %>%
    filter(review_stars == rating) %>%
    group_by(word) %>%
    summarise(
      n = n()
    ) %>%
    arrange(desc(n))
}

# returns first n values of the specified column x from the given dataframe df
get_top_n_of_x <- function(df, n, x) {
  df %>% head(n) %>% dplyr::select(x) %>% unlist()
}

# get word frequency counts for positive words in reviews for each rating group
pos_wc_1 <- get_word_counts_for_reviews_with_rating(positive_words_reviews, 1)
pos_wc_2 <- get_word_counts_for_reviews_with_rating(positive_words_reviews, 2)
pos_wc_3 <- get_word_counts_for_reviews_with_rating(positive_words_reviews, 3)
pos_wc_4 <- get_word_counts_for_reviews_with_rating(positive_words_reviews, 4)
pos_wc_5 <- get_word_counts_for_reviews_with_rating(positive_words_reviews, 5)

# get word frequency counts for negative words in reviews for each rating group
neg_wc_1 <- get_word_counts_for_reviews_with_rating(negative_words_reviews, 1)
neg_wc_2 <- get_word_counts_for_reviews_with_rating(negative_words_reviews, 2)
neg_wc_3 <- get_word_counts_for_reviews_with_rating(negative_words_reviews, 3)
neg_wc_4 <- get_word_counts_for_reviews_with_rating(negative_words_reviews, 4)
neg_wc_5 <- get_word_counts_for_reviews_with_rating(negative_words_reviews, 5)

# most common positive words used in reviews, search from top 100 in each rating group
(get_top_n_of_x(pos_wc_1, 100, "word") %>%
  intersect(get_top_n_of_x(pos_wc_2, 100, "word")) %>%
  intersect(get_top_n_of_x(pos_wc_3, 100, "word")) %>%
  intersect(get_top_n_of_x(pos_wc_4, 100, "word")) %>%
  intersect(get_top_n_of_x(pos_wc_5, 100, "word")))
```

```
##  [1] "nice"        "pretty"      "recommend"   "worth"       "top"
##  [6] "love"        "hot"         "decent"      "fresh"       "free"
## [11] "ready"       "fine"        "super"       "friendly"    "happy"
## [16] "warm"        "amazing"     "enjoy"       "delicious"   "sweet"
## [21] "clean"       "fast"        "wow"         "excited"     "favorite"
## [26] "recommended" "polite"      "loved"       "awesome"     "fair"
## [31] "excellent"   "impressed"   "cool"        "enjoyed"     "fun"
## [36] "easy"        "pleasant"    "seasoned"    "soft"        "strong"
```

```
## [41] "convenient"  "tender"      "lucky"       "perfect"     "wonderful"
## [46] "attentive"   "authentic"   "beautiful"   "classic"     "fancy"
## [51] "lovely"      "fairly"      "glad"        "solid"       "fantastic"
## [56] "helped"      "popular"     "prefer"      "comfortable" "crisp"
## [61] "reasonable"
```

```
# most common negative words used in reviews, search from top 100 in each rating group
(get_top_n_of_x(neg_wc_1, 100, "word") %>%
  intersect(get_top_n_of_x(neg_wc_2, 100, "word")) %>%
  intersect(get_top_n_of_x(neg_wc_3, 100, "word")) %>%
  intersect(get_top_n_of_x(neg_wc_4, 100, "word")) %>%
  intersect(get_top_n_of_x(neg_wc_5, 100, "word")))
```

```
##  [1] "bad"          "cold"         "terrible"     "wrong"
##  [5] "disappointed" "hard"         "poor"         "frozen"
##  [9] "bland"        "dirty"        "fried"        "slow"
## [13] "overpriced"   "mess"         "sad"          "mistake"
## [17] "issue"        "negative"     "complain"     "expensive"
## [21] "greasy"       "cheap"        "fault"        "hate"
## [25] "issues"       "lack"         "loud"         "smell"
## [29] "weird"        "crowded"      "dark"         "sour"
## [33] "break"        "hang"         "lemon"
```

```
# compare words that appear in high rating reviews vs low rating reviews
pos_words_high_rating_reviews <- union(pos_wc_5$word, pos_wc_4$word)
pos_words_low_rating_reviews <- union(pos_wc_3$word,
                                      union(pos_wc_2$word, pos_wc_1$word))
neg_words_high_rating_reviews <- union(neg_wc_5$word, neg_wc_4$word)
neg_words_low_rating_reviews <- union(neg_wc_3$word,
                                      union(neg_wc_2$word,neg_wc_1$word))

# positive words found in high rating reviews and not low rating reviews
unique_pos_words_high_rating <- setdiff(pos_words_high_rating_reviews,
                                        pos_words_low_rating_reviews)

# positive words found in low rating reviews and not high rating reviews
unique_pos_words_low_rating <-setdiff(pos_words_low_rating_reviews,
                                      pos_words_high_rating_reviews)

# negative words found in high rating reviews and not low rating reviews
unique_neg_words_high_rating <-setdiff(neg_words_high_rating_reviews,
                                       neg_words_low_rating_reviews)

# negative words found in low rating reviews and not high rating reviews
unique_neg_words_low_rating <-setdiff(neg_words_low_rating_reviews,
                                      neg_words_high_rating_reviews)
```

Define Test Niches:

1. Pizza in Boston, MA with Price Level $$
2. Pizza in New York, NY with Price Level $$

Adding niches to sqlite database

Web Scraping Function Call	Avg Runtime
get_save_business_info(keyword, location, price_point)	2.5 hr
get_save_business_reviews(keyword, location, price_point)	1.5 hr

Therefore, 8 hours to scrape both niches.

Note: the web scraping function calls take a few hours to run and save to the SQLite database, therefore we include the completed sqlite database file in the zipped file along with our report.

Analysis Function Call	Avg Runtime
save_business_review_analyses_cols(keyword, location, price_point)	0.5 hr
analyze_save_business_review_sentiments(keyword, location, price_point)	0.5 hr

The function calls to analyze the collected data, calculate new features, and store the calculated values into the database also take a decent amount of time.

These function calls mentioned above are commented out and left here to show the order they were called in for our analyses of the test niches.

```
# get_save_business_info("pizza", "Boston, MA", "2")
# get_save_business_reviews("pizza", "Boston, MA", "2")
# save_business_review_analyses_cols("pizza", "Boston, MA", 2)
# analyze_save_business_review_sentiments("pizza", "Boston, MA", 2)

# get_save_business_info("pizza", "New York, NY", "2")
# get_save_business_reviews("pizza", "New York, NY", "2")
# save_business_review_analyses_cols("pizza", "New York, NY", 2)
# analyze_save_business_review_sentiments("pizza", "New York, NY", 2)
```

Data Imputation and Cleaning for the Model

- If a column contains NA and the feature is a binary value, then we assume the restaurant does not have that feature
- If Noise Level is NA, we impute 'Average' because 'Average' is the most frequent Noise Level value among the 800 observations and we assume that if a restaurant's Noise Level is not explicitly specified to be another value, it is average
- If Ambience is NA, we impute 'Casual' because it is the most frequent Ambience value among the 800 restaurant observations we analyzed

Some restaurants have more than one term for Ambience, such as "Casual, Trendy". We consider such values to be their own factor instead of separating the value into "Casual" and "Trendy" because we interpret "Casual, Trendy" to have a different meaning than just "Casual" or "Trendy" alone.

```
impute_data <- function(keyword, location, price_point) {

  table_name <- str_c(keyword, str_extract(location, "([^,]+)"),
                      price_point, sep="_") %>%
    str_replace_all('[\\s-]+','_')
  table_data <- dbGetQuery(db, sprintf("SELECT * FROM %s", table_name))
```

```
  # transform columns: Yes/No to 1/0 and impute NA to 0. If NA, then assume restaurant
  # does not have the feature
  list_logical <- c("Liked_by_Vegans", "Liked_by_Vegetarians", "Takes_Reservations",
                    "Delivery", "Take_out", "Accepts_Credit_Cards", "Good_for_Kids",
                    "Good_for_Groups","Outdoor_Seating", "Wi_Fi", "Caters", "Has_TV",
                    "Has_menu_link")
  for(col in list_logical) {
     table_data[,col] <- ifelse(is.na(table_data[,col]) |
                                table_data[,col] == "No", 0, 1)
  }
  # transform Noise_Level and Ambience
  table_data[,"Noise_Level"][is.na(table_data[,"Noise_Level"])] <- "Average"
  table_data[,"Ambience"][is.na(table_data[,"Ambience"])] <- "Casual"

  # change columns to factor
  table_data[10:24] <- lapply(table_data[10:24], factor)
  assign(table_name, table_data, envir = .GlobalEnv)
}
```

```
impute_data("pizza", "Boston, MA", "2")
impute_data("pizza", "New York, NY", "2")
```

Close Connection and Disconnect Database

```
dbListTables(db)
```

```
## [1] "pizza_Boston_2"          "pizza_Boston_2_reviews"
## [3] "pizza_New_York_2"        "pizza_New_York_2_reviews"
```

```
dbDisconnect(db)
```

Create Visualizations Function

```
get_visualization <- function(keyword, location, price_point) {

  table_name <- str_c(keyword, str_extract(location, "([^,]+)"),
                      price_point, sep="_") %>%
    str_replace_all('[\\s-]+','_')
  table_data <- get(table_name)
  table_data$rating <- as.numeric(pizza_New_York_2$rating)
  visualize_plot <-  function(feature){
    # bar plot for ambience
    if(feature=="Ambience") {
      ggplot(table_data, aes(x = Ambience, y = rating)) + geom_bar(stat="identity") +
        theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
        ggtitle(str_c(table_name, feature, sep=" "))
    } else {
      ggplot(table_data, aes(x = table_data[, feature], y = rating)) +
        geom_boxplot(size = .75) + geom_jitter(alpha = .5) +
        labs(x= feature) + ggtitle(str_c(table_name, feature, sep=" "))
    }
  }
```

```
  lapply(names(table_data[c(5,10:28)]), visualize_plot)
}
```

Visualization Plots and Hypothesis - pizza_Boston_2

Notable Observations from Distribution Plots:

From the distribution plot of the Ambience feature, we see that an overwhelming majority of the restaurant observations have Ambience = 'Casual'. Therefore, we exclude the Ambience feature from our initial model.

An overwhelming majority of restaurant observations have Noise Level = 'Average', but we also see a sizeable number of observations with other Noise Level values in the plot. Therefore, we do not exclude the Noise Level feature from our initial model, but hypothesize that Noise Level will ultimately not be a significant feature.

There are dark, denser clusters of data points with ratings above 3.5 on the 1-box side of the Take Out, Has_TV, Accepts Credit Cards, review_count, has_menu_link, good for groups distribution plots.

There are dark, denser clusters of data points with ratings above 3.5 on the 0-box side of the Outdoor seating, Liked by Vegans, and Liked by Vegetarians distribution plots.

The Caters, WiFi, Takes Reservations, Delivery, perc_elite_reviewers plots are fairly evenly distributed between the option value ranges.

Though a majority of restaurants are Good for Kids, the distribution of the ratings for the observations in the two groups (Good and Not Good for Kids) are fairly equal. Both groups have a larger proportion of observations with ratings above 4.0 and a sizeable proportion of ratings below 4.0.

There are dark, denser clusters of data points with ratings > 3.5 for restaurants with 75-100% reviews published in the past year.

There is a dense cluster of data points with the median avg_bigrams_sentiment_score and the data points are fairly evenly distributed between ratings of 1-5.

There is a strong positive correlation between avg_perc_positive_sentiment and rating.

Hypothesis for Pizza in Boston $$ Model

See Appendix A at the end of the report for the visualization distribution plots of the features in the Pizza Boston $$ niche

Hypothesized Influence of Significance of Features in Model Based on Notable Observations:

- Little to no influence: Caters, Wifi, Takes Reservations, Delivery, perc_elite_reviewers, Good for Kids, avg_bigrams_sentiment
- Significant negative influence: Outdoor seating, Liked by Vegans, Liked by Vegetarians
- Significant positive influence: avg_perc_positive_sentiment, Take Out, Has_TV, Accepts Credit Cards, review_count, has_menu_link, good for groups, perc_reviews_in_past_year

Stratified Sampling: Create Training and Test Sets - pizza_Boston_2

We split our training data (the restaurant observations in the pizza_Boston_2 table) into 5 folds and chose to perform 5-fold cross validation to construct our final model. This decision was made after we noticed that different randomly selected training sets led to different features being deemed significant when we performed backwards feature elimination to construct our model.

To address this, we set a seed to control the randomness of the observations selected from the sample() function call, then randomly selected observations for each fold, ensuring that each fold had an approximately even proportion of observations from each rating class.

We constructed 5 ordered logistic regression (polr) models, where we alternated using each fold as the test set while the remaining 4 folds served as the training set. Then we took the most common significant features identified by the 5 models to construct our final model.

This process helped us to identify which features would have the highest likelihood of being significant to the model for any randomly selected training set. We validated this method and the features selected for our final model by using a random subset of 70% of the restaurant observations as our training set and testing the model's accuracy on the remaining 30% of the observations.

```
# function for stratified sampling
get_5_groups_stratified_sampling <- function(keyword, location, price_point) {
  set.seed(250)
  table_name <- str_c(keyword, str_extract(location, "([^,]+)"),
                      price_point, sep="_") %>% str_replace_all('[\\s-]+','_')
  table_data <- get(table_name)

  unique_ratings <<- unique(table_data$rating)

  group1 <<- tibble()
  group2 <<- tibble()
  group3 <<- tibble()
  group4 <<- tibble()
  group5 <<- tibble()

  # split reviews into 5 groups, ensure each group has fairly even proportion of
  # samples from each rating group
  for (r in unique_ratings) {
    samples_with_rating_r <- table_data %>%
      filter(rating == r)

    groups <- split(samples_with_rating_r, sample(1:5, nrow(samples_with_rating_r), replace=T))

    if (length(groups) > 0) {
      for (i in 1:length(groups)) {
        group_name <- str_c("group", i)
        assign(group_name, rbind(get(group_name), groups[[i]]), envir = .GlobalEnv)
      }
    }
  }
}
```

```
get_5_groups_stratified_sampling("pizza", "Boston, MA", "2")
```

Create the Model: Ordered Logistic Regression - pizza_Boston_2

Given that the dependent variable, rating, is an ordered factor, we chose to build an ordered logistic regression model. Therefore, we used the polr model to achieve an ordered logistic regression. The polr model provides t-value statistics and does not provide p-value, so we had to calculate the p-value from the given t-value. The model also has a Hess argument that is set to True to return the observed information matrix from optimization (called the Hessian) to get the standard errors.

We performed backwards feature elimination, where we removed features one by one until all features had

a p-value >= 0.05, to construct our models. We constructed a model where each of the 5 folds alternated serving as the test set, while the remaining folds served as the training set.

Then we selected the features that were deemed significant in each of these models to determine which features are most influential to a restaurant's Yelp rating among the restaurants in the specified niche.

Our models report a low exact accuracy rate of ~50%. However, upon further review of our model results, we noticed that ordinal logistic regression makes predictions based on the calculated probability of the sample belonging to each class.

When we change our method of calculating accuracy to be "accurate within the top 2 most likely predictions," the accuracy rates of our models was much higher (~75-85% accurate).

```
# calculate p-value for given model based on t-value
calculate_pvalue <- function(given_model) {
  ctable <- coef(summary(given_model))
  p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
  ctable <- cbind(ctable, "p-value" = p)
  ctable
}

# function to determine exact accuracy
calculate_exact_accuracy <- function(given_model, set) {
  pred <- factor(predict(given_model, set), levels=sort(unique_ratings), ordered=TRUE)
  accuracy <- ifelse(pred == set$rating, 1, 0)
  return(mean(na.omit(accuracy))*100)
}

# function to determine the accuracy within the top 2 most likely predictions
calculate_top2_accuracy <- function(given_model, set) {
  # predict probablility
  pred_prob <- predict(given_model, set, type="probs")
  # accuracy for range
  rank_pred_prob <- t(apply(-pred_prob, 1, rank, ties.method='max'))
  rank_1 <- apply(rank_pred_prob,1,function(x) match(1,x))
  rank_2 <- apply(rank_pred_prob,1,function(x) match(2,x))
  rank_1 <- colnames(rank_pred_prob[,rank_1])
  rank_2 <- colnames(rank_pred_prob[,rank_2])
  accuracy_prob <- ifelse(rank_1 == set$rating | rank_2 == set$rating, 1, 0)
  return(mean(na.omit(accuracy_prob))*100)
}
```

Model: pizza_Boston_2, Fold 1

Significant Features:
- Takes_Reservations
- Delivery
- Take_out
- Accepts_Credit_Cards
- Caters
- avg_perc_positive_sentiment
- perc_elite_reviewers

exact accuracy rate: 47.85276%
within top 2 predictions accuracy rate: 79.1411%

```
training_set <- rbind(group2, group3, group4, group5)
test_set <- group1

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link feature
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level feature
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + avg_perc_positive_sentiment + perc_elite_reviewers +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating feature
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                Has_TV + avg_perc_positive_sentiment + perc_elite_reviewers +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups feature
model4 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_reviews_in_past_year feature
model5 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_elite_reviewers + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
```

```r
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians feature
model6 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                 perc_elite_reviewers + avg_bigrams_sentiment_score,
               data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count feature
model7 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery + Take_out +
                 Accepts_Credit_Cards + Good_for_Kids + Wi_Fi + Caters + Has_TV +
                 avg_perc_positive_sentiment + perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids feature
model8 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                 Take_out + Accepts_Credit_Cards + Wi_Fi + Caters + Has_TV +
                 avg_perc_positive_sentiment + perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score feature
model9 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                 Take_out + Accepts_Credit_Cards + Wi_Fi + Caters + Has_TV +
                 avg_perc_positive_sentiment + perc_elite_reviewers,
               data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Wi_Fi feature
model10 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                 Take_out + Accepts_Credit_Cards + Caters + Has_TV +
                 avg_perc_positive_sentiment + perc_elite_reviewers,
               data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans feature
model11 <- polr(as.factor(rating)~Takes_Reservations + Delivery +
                 Take_out + Accepts_Credit_Cards + Caters + Has_TV +
                 avg_perc_positive_sentiment + perc_elite_reviewers,
               data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_Tv feature
model12 <- polr(as.factor(rating)~Takes_Reservations + Delivery +
                 Take_out + Accepts_Credit_Cards + Caters +
                 avg_perc_positive_sentiment + perc_elite_reviewers,
               data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# confidence intervals for model parameters
confint.default(model12)
```

```
##                                   2.5 %      97.5 %
## Takes_Reservations1         -1.0594487 -0.41083368
## Delivery1                   -0.7530194 -0.09765120
## Take_out1                    0.1505598  1.58513796
## Accepts_Credit_Cards1       -1.8537488 -0.09208791
## Caters1                      0.1538625  0.78922169
## avg_perc_positive_sentiment 14.0457234 17.63317520
## perc_elite_reviewers         0.1052426  1.68686418
```

```r
# predict exact rating and accuracy
calculate_exact_accuracy(model12, test_set) #47.85276
```

```
## [1] 47.85276
```

```r
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model12, test_set) #79.1411
```

```
## [1] 79.1411
```

Model: pizza_Boston_2, Fold 2

Significant Features:
- Takes_Reservations
- Delivery
- Take_out
- Wi_Fi
- Caters
- avg_perc_positive_sentiment
- perc_reviews_in_past_year

exact accuracy rate: 48.10127%
within top 2 predicitons accuracy rate: 79.74684%

```r
training_set <- rbind(group1, group3, group4, group5)
test_set <- group2

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link feature
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# remove review_count feature
model2 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids feature
model3 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating feature
model4 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters + Has_TV + Noise_Level +
                avg_perc_positive_sentiment + perc_elite_reviewers +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians feature
model5 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                Take_out + Accepts_Credit_Cards + Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + avg_perc_positive_sentiment + perc_elite_reviewers +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups feature
model6 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                Take_out + Accepts_Credit_Cards + Wi_Fi + Caters +
                Has_TV + Noise_Level + avg_perc_positive_sentiment + perc_elite_reviewers +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level feature
model7 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                Take_out + Accepts_Credit_Cards + Wi_Fi + Caters +
                Has_TV + avg_perc_positive_sentiment + perc_elite_reviewers +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_elite_reviewers feature
model8 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
```

```
                Take_out + Accepts_Credit_Cards + Wi_Fi + Caters +
                Has_TV + avg_perc_positive_sentiment + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards feature
model9 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                Take_out + Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year + avg_bigrams_sentiment_score,
              data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score feature
model10 <- polr(as.factor(rating)~Liked_by_Vegans + Takes_Reservations + Delivery +
                Take_out + Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year,
              data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans feature
model11 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Wi_Fi +
                 Caters + Has_TV + avg_perc_positive_sentiment + perc_reviews_in_past_year,
              data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV feature
model12 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Wi_Fi +
                 Caters + avg_perc_positive_sentiment + perc_reviews_in_past_year,
              data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model12)
```

```
##                                   2.5 %     97.5 %
## Takes_Reservations1         -1.18084835 -0.4896296
## Delivery1                   -0.90375604 -0.2493809
## Take_out1                    0.05819461  1.5111099
## Wi_Fi1                      -0.79994804 -0.1776730
## Caters1                      0.09279357  0.7198538
## avg_perc_positive_sentiment 14.51100055 18.0690009
## perc_reviews_in_past_year    0.10499850  1.2702375
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model12, test_set) #48.10127
```

```
## [1] 48.10127
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model12, test_set) #79.74684
```

```
## [1] 79.74684
```

Model: pizza_Boston_2, Fold 3

Significant Features:

- review_count
- Takes_Reservations
- Delivery
- Take_out
- Good_for_Groups
- avg_perc_positive_sentiment

exact accuracy rate: 50.83799%
within top 2 predicitons accuracy rate: 78.77095%

```
training_set <- rbind(group1, group2, group4, group5)
test_set <- group3

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV feature
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians feature
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating feature
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                 Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link feature
model4 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                 Noise_Level + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
```

```
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level feature
model5 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                avg_perc_positive_sentiment + perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards feature
model6 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                avg_perc_positive_sentiment + perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_reviews_in_past_year feature
model7 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                avg_perc_positive_sentiment + perc_elite_reviewers +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans feature
model8 <- polr(as.factor(rating)~review_count +
                Takes_Reservations + Delivery + Take_out +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                avg_perc_positive_sentiment + perc_elite_reviewers +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids feature
model9 <- polr(as.factor(rating)~review_count +
                Takes_Reservations + Delivery + Take_out +
                Good_for_Groups + Wi_Fi + Caters +
                avg_perc_positive_sentiment + perc_elite_reviewers +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_elite_reviewers feature
model10 <- polr(as.factor(rating)~review_count +
                Takes_Reservations + Delivery + Take_out +
                Good_for_Groups + Wi_Fi + Caters +
                avg_perc_positive_sentiment +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score feature
model11 <- polr(as.factor(rating)~review_count +
```

```
                  Takes_Reservations + Delivery + Take_out +
                  Good_for_Groups + Wi_Fi + Caters +
                  avg_perc_positive_sentiment, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Caters feature
model12 <- polr(as.factor(rating)~review_count +
                  Takes_Reservations + Delivery + Take_out +
                  Good_for_Groups + Wi_Fi + avg_perc_positive_sentiment, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Wi_Fi feature
model13 <- polr(as.factor(rating)~review_count +
                  Takes_Reservations + Delivery + Take_out +
                  Good_for_Groups + avg_perc_positive_sentiment, data= training_set, Hess=TRUE)
calculate_pvalue(model13)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model13)
```

```
##                                    2.5 %       97.5 %
## review_count                 0.000210408  0.001275993
## Takes_Reservations1         -0.989483039 -0.317529748
## Delivery1                   -0.788483314 -0.141625456
## Take_out1                    0.303078526  1.684967752
## Good_for_Groups1            -0.938643313 -0.086785386
## avg_perc_positive_sentiment 14.534606207 18.135813214
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model13, test_set) #50.83799
```

```
## [1] 50.83799
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model13, test_set) #78.77095
```

```
## [1] 78.77095
```

Model: pizza_Boston_2, Fold 4

Significant Features:
- Takes_Reservations
- Delivery
- Caters
- Has_TV
- avg_perc_positive_sentiment
- perc_reviews_in_past_year

exact accuracy rate: 51.33333%
within top 2 predicitons accuracy rate: 74%

```
training_set <- rbind(group1, group2, group3, group5)
test_set <- group4

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                  Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
```

```
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating feature
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count feature
model2 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_elite_reviewers
model3 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids feature
model4 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise Level feature
model5 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups feature
model6 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
```

```
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Wi_Fi + Caters + Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians feature
model7 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Wi_Fi + Caters + Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link feature
model8 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Take_out feature
model9 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery + Accepts_Credit_Cards +
                Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards feature
model10 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery +
                Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score feature
model11 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery +
                Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans feature
model11 <- polr(as.factor(rating)~Takes_Reservations + Delivery +
                Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Wi_Fi feature
model12 <- polr(as.factor(rating)~Takes_Reservations + Delivery +
```

```
                Caters + Has_TV + avg_perc_positive_sentiment +
                perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model12)
```

```
##                                   2.5 %     97.5 %
## Takes_Reservations1         -1.03714937 -0.3641767
## Delivery1                   -0.97078049 -0.3108585
## Caters1                      0.14615868  0.7758971
## Has_TV1                     -0.88542685 -0.1632337
## avg_perc_positive_sentiment 14.25724218 17.7852029
## perc_reviews_in_past_year    0.09396852  1.1923462
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model12, test_set) #51.33333
```

```
## [1] 51.33333
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model12, test_set) #74
```

```
## [1] 74
```

Model: pizza_Boston_2, Fold 5

Significant Features:
- Takes_Reservations
- Delivery
- Take_out
- Accepts_Credit_Cards
- Wi_Fi
- Caters
- Has_TV
- avg_perc_positive_sentiment
- perc_reviews_in_past_year
- avg_bigrams_sentiment_score

exact accuracy rate: 54%
within top 2 predicitons accuracy rate: 82%

```
training_set <- rbind(group1, group2, group3, group4)
test_set <- group5

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians feature
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
```

```
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating feature
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids feature
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count feature
model4 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans feature
model5 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups feature
model6 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Wi_Fi + Caters + Has_TV + Noise_Level + Has_menu_link +
                avg_perc_positive_sentiment + perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_elite_reviewers feature
model7 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Wi_Fi + Caters + Has_TV + Noise_Level + Has_menu_link +
                avg_perc_positive_sentiment + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
```

```r
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level feature
model8 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Wi_Fi + Caters + Has_TV + Has_menu_link +
                 avg_perc_positive_sentiment + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link feature
model9 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Wi_Fi + Caters + Has_TV + avg_perc_positive_sentiment + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# confidence intervals for model parameters
confint.default(model9)
```

```
##                                   2.5 %      97.5 %
## Takes_Reservations1         -0.92912320 -0.25048222
## Delivery1                   -0.83210420 -0.17358576
## Take_out1                    0.04758004  1.50280843
## Accepts_Credit_Cards1       -1.88666738 -0.10321859
## Wi_Fi1                      -0.65172498 -0.02637615
## Caters1                      0.03592939  0.67061323
## Has_TV1                     -0.84403242 -0.12532267
## avg_perc_positive_sentiment 14.66921121 18.30497211
## perc_reviews_in_past_year    0.13525414  1.27762341
## avg_bigrams_sentiment_score -2.19118626 -0.46847796
```

```r
# predict exact rating and accuracy
calculate_exact_accuracy(model9, test_set) #54
```

```
## [1] 54
```

```r
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model9, test_set) #82
```

```
## [1] 82
```

Cross Validation Results Analysis - Pizza Boston $$

From the 5 models we constructed using 5-fold cross validation, every model deemed the features: Takes Reservations, Delivery, and avg_perc_positive_sentiment as statistically significant. 4 of the 5 models also deemed Take Out to be an important feature.

Therefore, we conclude (and verify with a model using 70% of the total observations as training data and the remaining 30% as test data) that the most important features for pizza places in Boston with a price level $$ are:
- Takes Reservations
- Offers Delivery
- High percentage of positive sentiment words in customer reviews
- Offers Takeout

We were correct in our prediction that Take Out and avg_perc_positive_sentiment have significant positive relationships with a restaurant's Yelp rating. However, we initially hypothesized that Takes Reservations and Delivery would not have a significant influence on rating but discovered that they did.

Final Model Accuracy Metrics:
- Exact Accuracy Rate: 50.21277%
- Within Top 2 Predictions Accuracy Rate: 82.12766%

```
training_set <- tibble()
test_set <- tibble()

pizza_Boston_2$rating <- factor(pizza_Boston_2$rating,
                                levels=sort(unique_ratings), ordered=TRUE)

for (r in unique_ratings) {
  samples_with_rating_r <- pizza_Boston_2 %>%
    filter(rating == r)

  new_training_samples <- sample_n(samples_with_rating_r,
                                   ceiling(0.70 * nrow(samples_with_rating_r)),
                                   replace=FALSE)

  new_test_samples <- anti_join(samples_with_rating_r, new_training_samples)

  training_set <- rbind(training_set, new_training_samples)

  test_set <- rbind(test_set, new_test_samples)
}

# create initial model
model <- polr(as.factor(rating)~Takes_Reservations + Delivery +
                Take_out +  avg_perc_positive_sentiment, data= training_set, Hess=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model)
```

```
##                                  2.5 %     97.5 %
## Takes_Reservations1         -0.9321124 -0.2705270
## Delivery1                   -0.8677153 -0.2055742
## Take_out1                    0.1386354  1.6266802
## avg_perc_positive_sentiment 14.1085345 17.7719772
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model, test_set) #50.21277
```

```
## [1] 50.21277
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model, test_set) #82.12766
```

```
## [1] 82.12766
```

Models Summary: Pizza Boston $$

Model	Exact Accuracy	Within Top 2 Accuracy
Fold 1	47.85%	79.14%
Fold 2	48.10%	79.75%
Fold 3	50.84%	78.77%
Fold 4	51.33%	74.00%
Fold 5	54.00%	82.00%

Model	Exact Accuracy	Within Top 2 Accuracy
Final	50.21%	82.13%

Visualization Plots and Hypothesis - pizza_New_York_2

Notable Observations from Distribution Plots:

Examining the Ambience values in the feature distribution plot reveals that a majority of the restaurant observations have Ambience = "Casual". Therefore, we exclude the Ambience feature from our model.

avg_perc_pos_sentiment has a positive correlation with Yelp rating and we see denser clusters of observations with high avg_perc_pos_sentiment around a rating of 4.0.

The distribution of avg_bigrams_sentiment_score distr is scattered within Q1 and Q3, but there are slightly denser clusters around the median and Q3.

perc_reviews_in_past_year and perc_elite_reviewers are evenly distributed between the rating classes. Whereas a majority of restaurants have perc_reviews_in_past_year $> 50\%$, the majority of restaurants have perc_elite_reviewers $< 50\%$.

Review count also illustrates that there are less than 1000 reviews for almost all of the pizza shops in New York that have a price level of 2 ($$).

The distribution plots of Good for Kids, Good for Groups, Has Menu Link, and Delivery have large clusters around the 1-option, indicating that a majority of restaurants have these features. Liked by Vegetarians and Liked by Vegans are clustered around the 0-option; the majority of restaurants do not have the features. However, the ratings of restaurant observations for all of these features are distributed fairly evenly between low and high ratings, suggesting that these features may not have a strong correlation with rating.

The majority of restaurants have a Noise Level = 'Average', but there is a dense cluster of observations around a rating of 4.0 with 'Average' noise level and a cluster of observations with 'Very Loud' noise level that have lower ratings. Therefore, we hypothesize that a 'Very Loud' noise level has a negative relationship with Yelp rating.

The distribution plots for Caters, Has TV, WiFi, Outdoor Seating, and Takes Reservations are fairly evenly distributed between the two options (0 and 1) and the ratings for the observations in the plots are also fairly distributed between high and low ratings. This suggests that these features also may not correlate with a Yelp rating.

The distribution plots for Take Out and Accepts Credit Cards are clustered around the 1-option. There is a dark, dense cluster of observations with ratings above 3.5, which suggests that having these options can correlate to a higher rating.

Hypothesis for Pizza in New York $$ Model

See Appendix B at the end of the report for the visualization distribution plots of the features in the Pizza New York $$ niche

Hypothesized Influence of Significance of Features in Model Based on Notable Observations:

- Little to no influence: Good for Kids, Good for Groups, Has Menu Link, Delivery Liked by Vegetarians, Liked by Vegans, noise level, Caters, Has TV, WiFi, Outdoor Seating, Takes Reservations
- Significant positive influence: Take Out, Accepts Credit Cards, Average positive sentiment, Review count, percent elite reviewers, Percent reviews in the past year, Average bigrams

Stratified Sampling: Create Training and Test Sets - pizza_New_York_2

We split our data into 5 folds with each fold having an equal distribution of observations from each rating class and performed 5-fold cross validation to construct our final polr model for the pizza_New_York_2 niche, using the same process and reasoning as mentioned above for our pizza_Boston_2 model.

```
get_5_groups_stratified_sampling("pizza", "New York, NY", "2")
```

Create the Model: Ordered Logistic Regression - pizza_New_York_2

Here, we built polr models for the pizza_New_York_2 niche and performed backwards feature elimination like we did for pizza_Boston_2. As before, we selected the most common features that were significant in each of the 5 models to construct our final model. We also report the exact accuracy rate and the accuracy rate within the top 2 most likely predictions for each of the constructed models again.

Model: pizza_New_York_2, Fold 1

Significant Features:
- Takes Reservations
- avg_perc_positive_sentiment
- perc_elite_reviewers
- avg_bigrams_sentiment_score

exact accuracy rate: 53.41615%
within top 2 predictions accuracy rate: 84.47205%

```
training_set <- rbind(group2, group3, group4, group5)
test_set <- group1

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# remove Good_for_Kids
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Caters
model4 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Outdoor_Seating + Wi_Fi +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV
model5 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Outdoor_Seating + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating
model6 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Take_out
model7 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Accepts_Credit_Cards +
                 Good_for_Groups + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards
model8 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery +
                 Good_for_Groups + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# remove Good_for_Groups
model9 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Delivery
model10 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count
model11 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_reviews_in_past_year
model12 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans
model13 <- polr(as.factor(rating)~Takes_Reservations + Wi_Fi +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model13)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link
model14 <- polr(as.factor(rating)~Takes_Reservations + Wi_Fi + avg_perc_positive_sentiment +
                 perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model14)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Wi_Fi
model15 <- polr(as.factor(rating)~Takes_Reservations + avg_perc_positive_sentiment +
                 perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model15)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# confidence intervals for model parameters
confint.default(model15)
```

```
##                                   2.5 %      97.5 %
## Takes_Reservations1         -0.8550918 -0.22658188
## avg_perc_positive_sentiment 12.3933333 16.24038983
## perc_elite_reviewers         0.7539968  2.27324552
## avg_bigrams_sentiment_score -1.7434106 -0.01832481
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model15, test_set) #53.41615
```

```
## [1] 53.41615
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model15, test_set) #84.47205
```

```
## [1] 84.47205
```

Model: pizza_New_York_2, Fold 2

Significant Features:
- Takes_Reservations
- Take_out
- avg_perc_positive_sentiment
- perc_reviews_in_past_year

exact accuracy rate: 51.57233%
within top 2 predictions accuracy rate: 85.53459%

```
training_set <- rbind(group1, group3, group4, group5)
test_set <- group2

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count
model1 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level
model2 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# remove Good_for_Groups
model3 <- polr(as.factor(rating)~Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians
model4 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating
model5 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans
model6 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link
model7 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Wi_Fi + Caters +
                 Has_TV + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids
model8 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Wi_Fi + Caters +
                 Has_TV + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Caters
model9 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
```

```
                Wi_Fi + Has_TV + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Delivery
model10 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Wi_Fi + Has_TV + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV
model11 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Wi_Fi
model12 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                avg_perc_positive_sentiment + perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score
model13 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                avg_perc_positive_sentiment + perc_elite_reviewers +
                 perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model13)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_elite_reviewers
model14 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                avg_perc_positive_sentiment + perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model14)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards
model15 <- polr(as.factor(rating)~Takes_Reservations + Take_out +
                avg_perc_positive_sentiment + perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model15)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model15)
```

```
##                                 2.5 %     97.5 %
## Takes_Reservations1         -0.8996011 -0.2569091
## Take_out1                   -3.1981817 -0.4676119
## avg_perc_positive_sentiment 13.0277208 16.7858152
## perc_reviews_in_past_year    0.4752566  1.6485843
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model15, test_set) #51.57233
```

```
## [1] 51.57233
```

```r
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model15, test_set) #85.53459
```

```
## [1] 85.53459
```

Model: pizza_New_York_2, Fold 3

Significant Features:
- Takes_Reservations
- Take_out
- Wi_Fi
- Has_menu_link
- avg_perc_positive_sentiment
- perc_elite_reviewers

exact accuracy rate: 53.63128%
within top 2 predictions accuracy rate: 84.35754%

```r
training_set <- rbind(group1, group2, group4, group5)
test_set <- group3

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
```

```
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count
model4 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Delivery
model5 <- polr(as.factor(rating)~Liked_by_Vegans +
                Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans
model6 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating
model7 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV
model8 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Caters
model9 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups
model10 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
```

```
                  Wi_Fi + Has_menu_link + avg_perc_positive_sentiment +
                  perc_elite_reviewers + perc_reviews_in_past_year +
                  avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_reviews_in_past_year
model11 <- polr(as.factor(rating)~Takes_Reservations + Take_out + Accepts_Credit_Cards +
                  Wi_Fi + Has_menu_link + avg_perc_positive_sentiment +
                  perc_elite_reviewers +
                  avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards
model12 <- polr(as.factor(rating)~Takes_Reservations + Take_out +
                  Wi_Fi + Has_menu_link + avg_perc_positive_sentiment +
                  perc_elite_reviewers +
                  avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score
model13 <- polr(as.factor(rating)~Takes_Reservations + Take_out +
                  Wi_Fi + Has_menu_link + avg_perc_positive_sentiment +
                  perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model13)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model13)
```

```
##                                    2.5 %      97.5 %
## Takes_Reservations1         -0.74509597 -0.10008291
## Take_out1                   -3.84183140 -0.64302009
## Wi_Fi1                       0.08208234  0.72210171
## Has_menu_link1              -0.96089275 -0.01064476
## avg_perc_positive_sentiment 13.33067256 17.23762411
## perc_elite_reviewers         0.56554677  2.12542976
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model13, test_set) #53.63128
```

```
## [1] 53.63128
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model13, test_set) #84.35754
```

```
## [1] 84.35754
```

Model: pizza_New_York_2, Fold 4

Significant Features:
- Takes_Reservations
- Delivery
- avg_perc_positive_sentiment
- perc_elite_reviewers

exact accuracy rate: 50%
within top 2 predictions accuracy rate: 82.89474%

```
training_set <- rbind(group1, group2, group3, group5)
test_set <- group4

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count
model4 <- polr(as.factor(rating)~Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level
model5 <- polr(as.factor(rating)~Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters +
                Has_TV + Has_menu_link + avg_perc_positive_sentiment +
```

```
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV
model6 <- polr(as.factor(rating)~Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_menu_link
model7 <- polr(as.factor(rating)~Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians
model8 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + Caters + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Caters
model9 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score
model10 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_reviews_in_past_year
model11 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Groups + Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups
model12 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards
```

```
model13 <- polr(as.factor(rating)~Takes_Reservations + Delivery + Take_out +
                Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model13)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Take_out
model14 <- polr(as.factor(rating)~Takes_Reservations + Delivery +
                Wi_Fi + avg_perc_positive_sentiment +
                perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model14)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Wi_Fi
model15 <- polr(as.factor(rating)~Takes_Reservations + Delivery + avg_perc_positive_sentiment +
                perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model15)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model15)
```

```
##                                  2.5 %     97.5 %
## Takes_Reservations1         -0.7690912 -0.1418461
## Delivery1                   -1.0157764 -0.1414982
## avg_perc_positive_sentiment 12.8822356 16.6232437
## perc_elite_reviewers         0.4977838  2.0306701
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model15, test_set) #50
```

```
## [1] 50
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model15, test_set) #82.89474
```

```
## [1] 82.89474
```

Model: pizza_New_York_2, Fold 5

Significant Features:
- Takes_Reservations
- Wi_Fi
- avg_perc_positive_sentiment
- perc_elite_reviewers

exact accuracy rate: 55.7047%
within top 2 predictions accuracy rate: 84.56376%

```
training_set <- rbind(group1, group2, group3, group4)
test_set <- group5

# create initial model
model0 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                Good_for_Kids + Good_for_Groups + Outdoor_Seating + Wi_Fi + Caters +
                Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                perc_elite_reviewers + perc_reviews_in_past_year +
                avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
```

```r
calculate_pvalue(model0)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Outdoor_Seating
model1 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Accepts_Credit_Cards +
                 Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model1)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Accepts_Credit_Cards
model2 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out +
                 Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                 Has_TV + Noise_Level + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model2)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Noise_Level
model3 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out +
                 Good_for_Kids + Good_for_Groups + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model3)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Kids
model4 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Good_for_Groups + Wi_Fi + Caters +
                 Has_TV + Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model4)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Has_TV
model5 <- polr(as.factor(rating)~review_count + Liked_by_Vegans + Liked_by_Vegetarians +
                 Takes_Reservations + Delivery + Take_out + Good_for_Groups + Wi_Fi + Caters +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model5)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegetarians
model6 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Good_for_Groups + Wi_Fi + Caters +
                 Has_menu_link + avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model6)[,"p-value"] %>% sort(decreasing=TRUE)
```

```r
# remove Has_menu_link
model7 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Delivery + Take_out + Good_for_Groups + Wi_Fi + Caters +
                 avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model7)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Delivery
model8 <- polr(as.factor(rating)~review_count + Liked_by_Vegans +
                 Takes_Reservations + Take_out + Good_for_Groups + Wi_Fi + Caters +
                 avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model8)[,"p-value"] %>% sort(decreasing=TRUE)

# remove review_count
model9 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Take_out + Good_for_Groups + Wi_Fi + Caters +
                 avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model9)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Good_for_Groups
model10 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Take_out + Wi_Fi + Caters +
                 avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model10)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Caters
model11 <- polr(as.factor(rating)~Liked_by_Vegans +
                 Takes_Reservations + Take_out + Wi_Fi +
                 avg_perc_positive_sentiment +
                 perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model11)[,"p-value"] %>% sort(decreasing=TRUE)

# remove Liked_by_Vegans
model12 <- polr(as.factor(rating)~ Takes_Reservations + Take_out + Wi_Fi +
                 avg_perc_positive_sentiment + perc_elite_reviewers + perc_reviews_in_past_year +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model12)[,"p-value"] %>% sort(decreasing=TRUE)

# remove perc_reviews_in_past_year
model13 <- polr(as.factor(rating)~ Takes_Reservations + Take_out + Wi_Fi +
                 avg_perc_positive_sentiment + perc_elite_reviewers +
                 avg_bigrams_sentiment_score, data= training_set, Hess=TRUE)
calculate_pvalue(model13)[,"p-value"] %>% sort(decreasing=TRUE)

# remove avg_bigrams_sentiment_score
```

```
model14 <- polr(as.factor(rating)~ Takes_Reservations + Take_out + Wi_Fi +
                  avg_perc_positive_sentiment + perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model14)[,"p-value"] %>% sort(decreasing=TRUE)


# remove Take_out
model15 <- polr(as.factor(rating)~ Takes_Reservations + Wi_Fi + avg_perc_positive_sentiment
                + perc_elite_reviewers, data= training_set, Hess=TRUE)
calculate_pvalue(model15)[,"p-value"] %>% sort(decreasing=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model15)
```

```
##                                   2.5 %     97.5 %
## Takes_Reservations1         -0.95079843 -0.3166562
## Wi_Fi1                       0.06682933  0.6905924
## avg_perc_positive_sentiment 13.07544210 16.7915695
## perc_elite_reviewers         0.52462465  2.0273891
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model15, test_set) #55.7047
```

```
## [1] 55.7047
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model15, test_set) #84.56376
```

```
## [1] 84.56376
```

Cross Validation Results Analysis - Pizza New York $$

From the 5 models built using 5-fold cross validation, every model deemed the features: Takes Reservations and avg_perc_positive_sentiment as statistically significant. 4 of the 5 models also deemed perc_elite_reviewers to be an important feature.

Therefore, we conclude (and verify with a model using 70% of the total observations as training data and the remaining 30% as test data) that the most important features for pizza places in NY, New York with a price level $$ are:
- Takes Reservations
- High percentage of positive sentiment words in customer reviews
- High percentage of elite reviewers

We were correct in our prediction that percent elite reviewers and average positive sentiment have significant positive relationships with rating. However, we initially hypothesized, based on our visualization distribution plots, that Takes Reservations would not have a significant influence on rating but discovered that it did.

Final Model Accuracy Metrics:
- Exact Accuracy Rate: 51.89873%
- Within Top 2 Predictions Accuracy Rate: 84.38819%

```
training_set <- tibble()
test_set <- tibble()

pizza_New_York_2$rating <- factor(pizza_New_York_2$rating,
                                  levels=sort(unique_ratings), ordered=TRUE)

for (r in unique_ratings) {
```

```
  samples_with_rating_r <- pizza_New_York_2 %>%
    filter(rating == r)

  new_training_samples <- sample_n(samples_with_rating_r,
                                   ceiling(0.70 * nrow(samples_with_rating_r)),
                                   replace=FALSE)

  new_test_samples <- anti_join(samples_with_rating_r, new_training_samples)

  training_set <- rbind(training_set, new_training_samples)

  test_set <- rbind(test_set, new_test_samples)
}

# create initial model
model_pizza_New_York_2 <- polr(as.factor(rating)~Takes_Reservations + perc_elite_reviewers +
                avg_perc_positive_sentiment, data= training_set, Hess=TRUE)
```

```
# confidence intervals for model parameters
confint.default(model_pizza_New_York_2)
```

```
##                                  2.5 %    97.5 %
## Takes_Reservations1         -0.79581085 -0.129578
## perc_elite_reviewers         0.01907328  1.659977
## avg_perc_positive_sentiment 13.55949314 17.636056
```

```
# predict exact rating and accuracy
calculate_exact_accuracy(model_pizza_New_York_2, test_set) #51.89873
```

```
## [1] 48.52321
```

```
# predict probablility and top 2 accuracy
calculate_top2_accuracy(model_pizza_New_York_2, test_set) #84.38819
```

```
## [1] 84.38819
```

Models Summary: Pizza New York $$

Model	Exact Accuracy	Within Top 2 Accuracy
Fold 1	53.42%	84.47%
Fold 2	51.57%	85.53%
Fold 3	53.63%	84.36%
Fold 4	50.00%	82.89%
Fold 5	55.70%	84.56%
Final	51.90%	84.39%

Comparison of Pizza Boston $$ and Pizza New York $$ Niches

Our final model for pizza_New_York_2 had a higher exact accuracy rate and accurate within top 2 predictions accuracy rate than our final model for pizza_Boston_2. We believe this is most likely because the 800 restaurants we examined for the pizza New York niche were better representations of restaurants within the niche itself.

A comparison of the features important across the two niches shows that having a high percentage of elite

reviewers is an important feature for pizza restaurants in NY. This was not deemed a significant feature for pizza restaurants in Boston. From this we conclude New York has a higher proportion of Yelp elite members that provide well-written reviews and accurate ratings whose reviews have a strong influence on other regular Yelp users in New York. Pizza places in New York with a $$ price level on Yelp can seek to improve their Yelp rating by getting more positive reviews from Yelp elite members.

Unlike pizza_New_York_2, the pizza_Boston_2 niche deems offering delivery and takeout options to be important features that are correlated to higher Yelp ratings. This suggests that more people in Boston prefer to eat their pizza outside of the restaurants that serve them and will be more likely to try and offer higher ratings to restaurants that provide these options.

Both pizza places at a price level of $$ in Boston and New York consider taking reservations and restaurants with a high percentage of positive Yelp reviews to be significant features correlated to high Yelp ratings. This implies that people in both of these cities prefer to avoid long wait times and to be immediately seated on arrival. Restaurants that do not take reservations or that have slow service are more likely to receive lower ratings. The positive correlation with the average percentage of positive sentiment indicates that customers are more likely to give higher ratings to restaurants if they had a positive experience there, which is naturally expected.

Future Steps

Our next steps would include analyzing other popular cities, such as pizza Chicago $$, to determine what features are important to pizza places there, to look for similarities with pizze places in Boston or New York, and to gain further insight into what Chicagoans value in contrast to Bostonians or New Yorkers.

We also want to examine different niches at different price levels. For example, analyzing Seafood Boston $$$ vs Seafood Boston $$ could help us determine if restaurants at price levels have different features that are significant to their ratings.

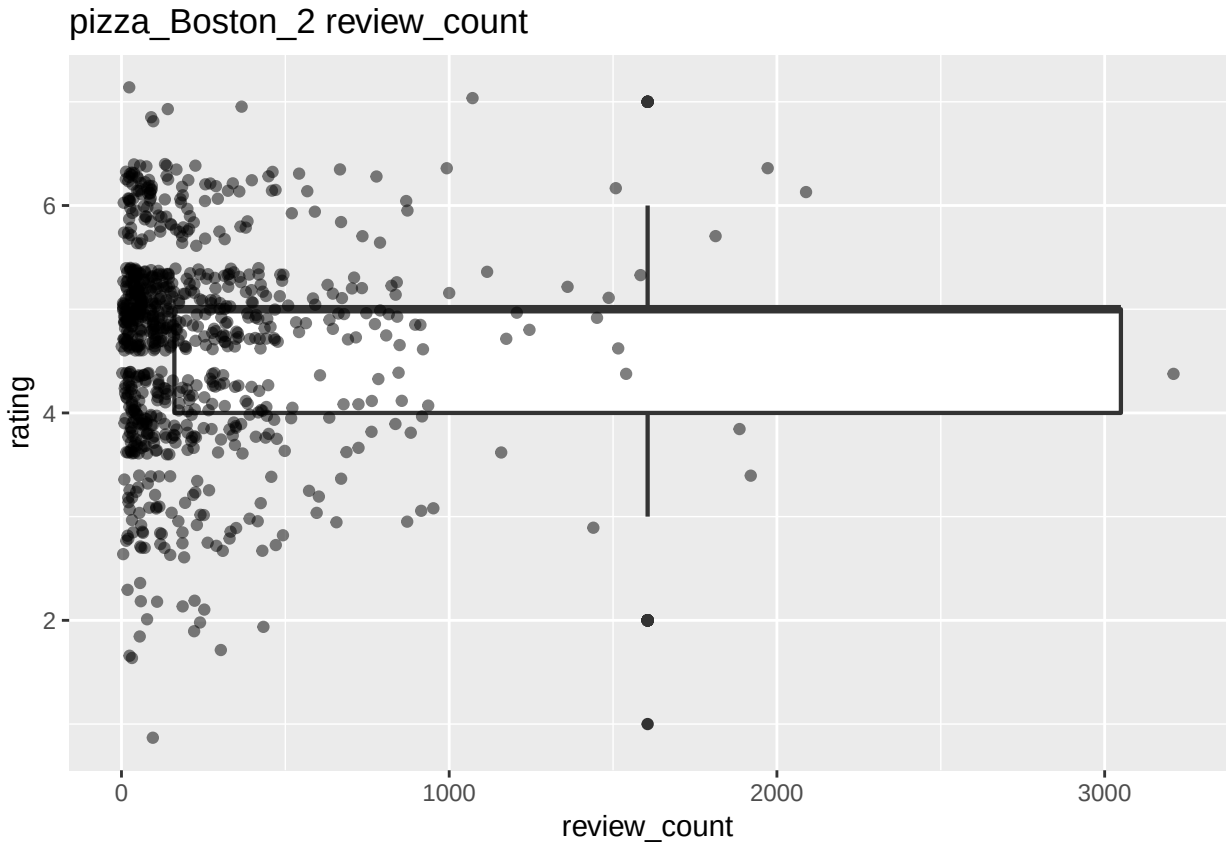
In summary, we can extend our analysis to identify which features are important in achieving better Yelp ratings across various restaurant niches through new combinations of keywords, locations, and price levels.

Appendix A: Visualization Plots for pizza_Boston_2

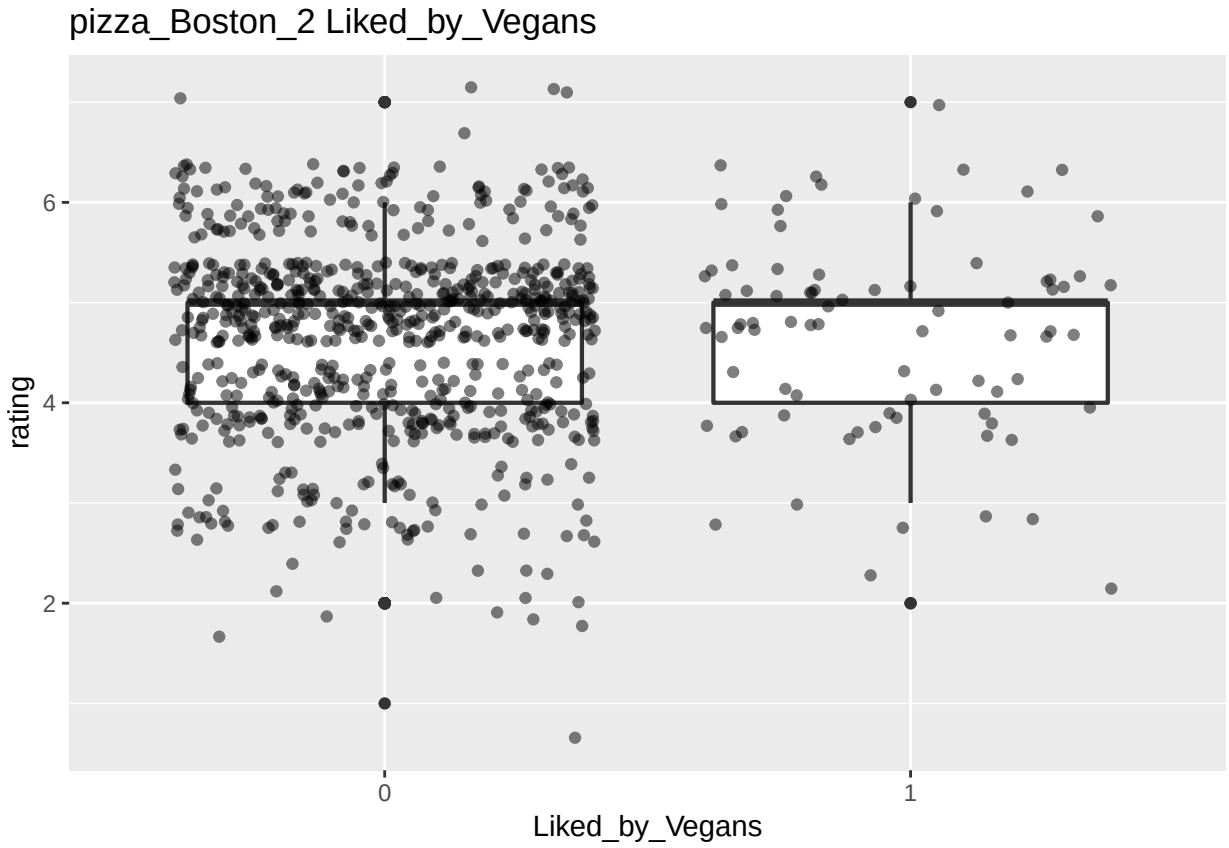
```
get_visualization("pizza", "Boston, MA", "2")
```

```
## [[1]]
```

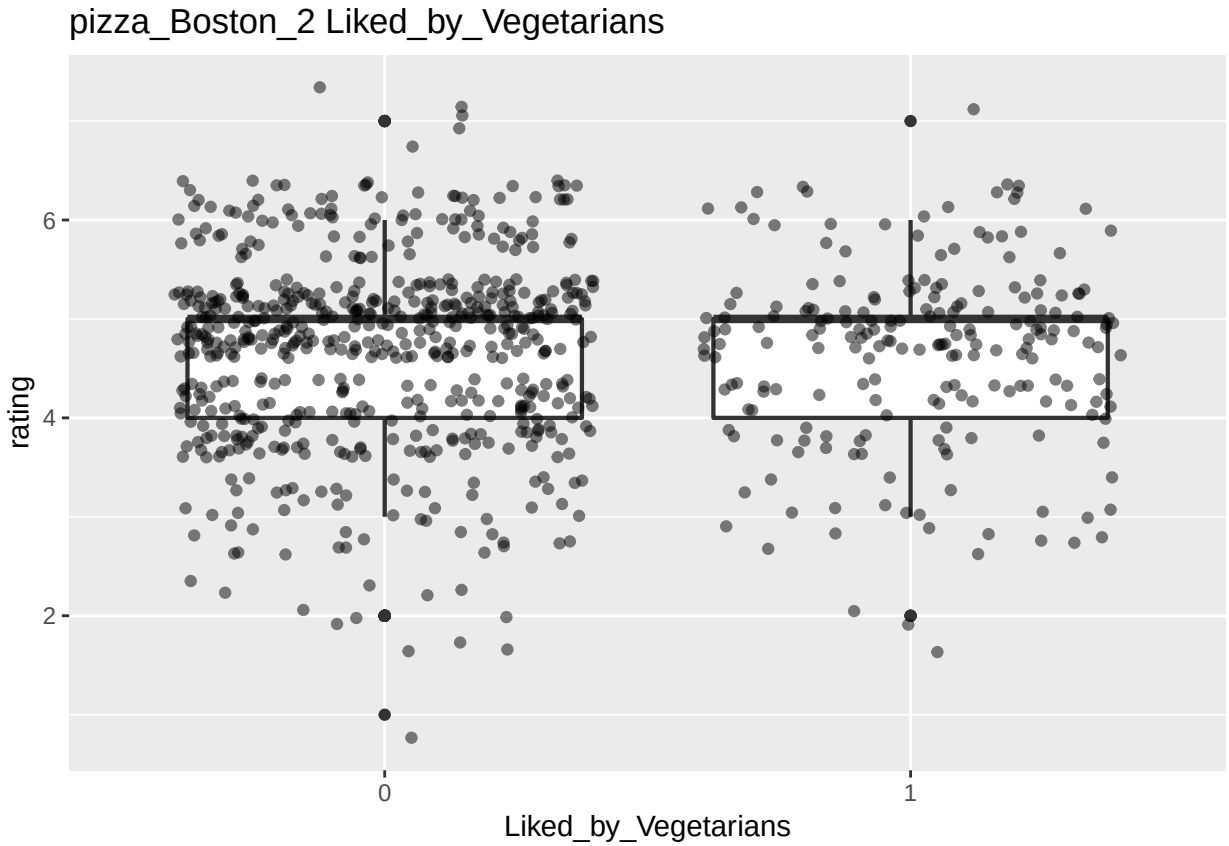
```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```


pizza_Boston_2 review_count
rating
6
4
2
0
1000
2000
3000
review_count

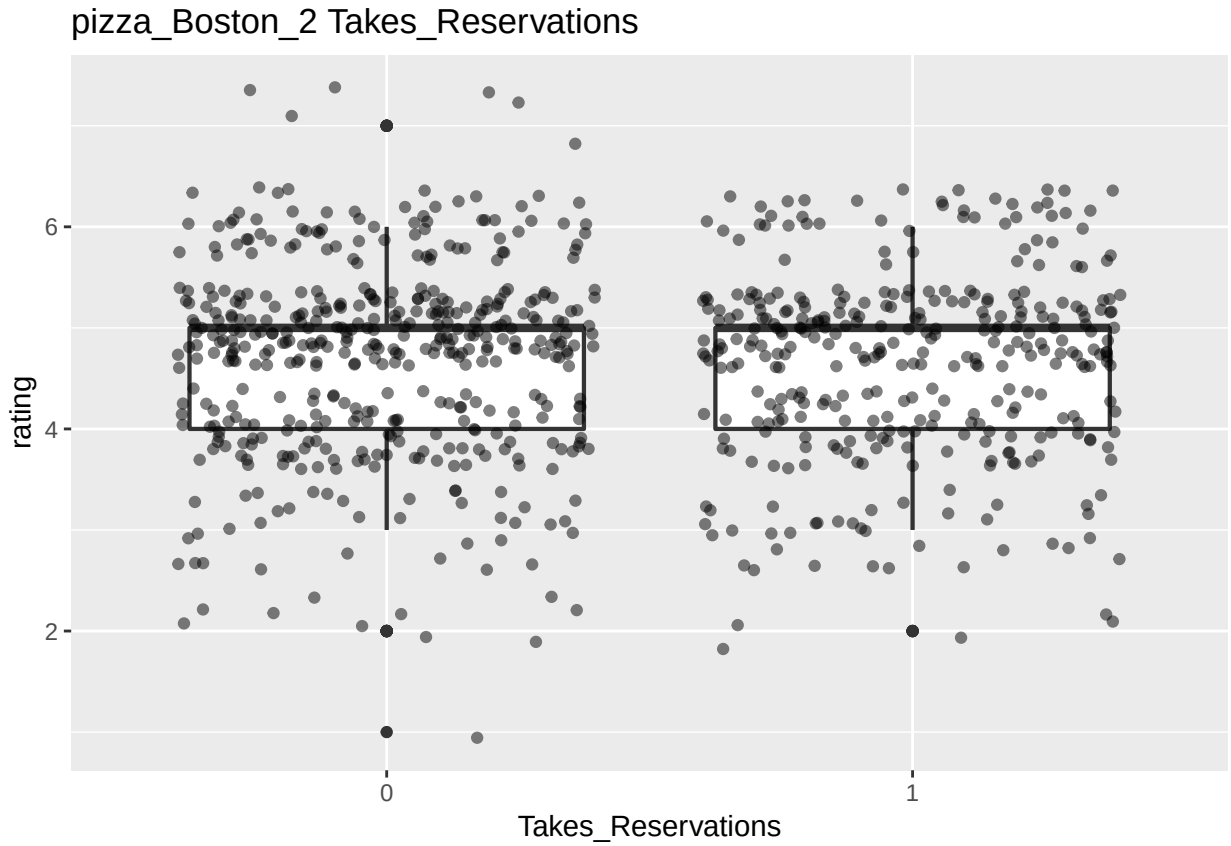
```
##
## [[2]]
```




```
##
## [[3]]
```

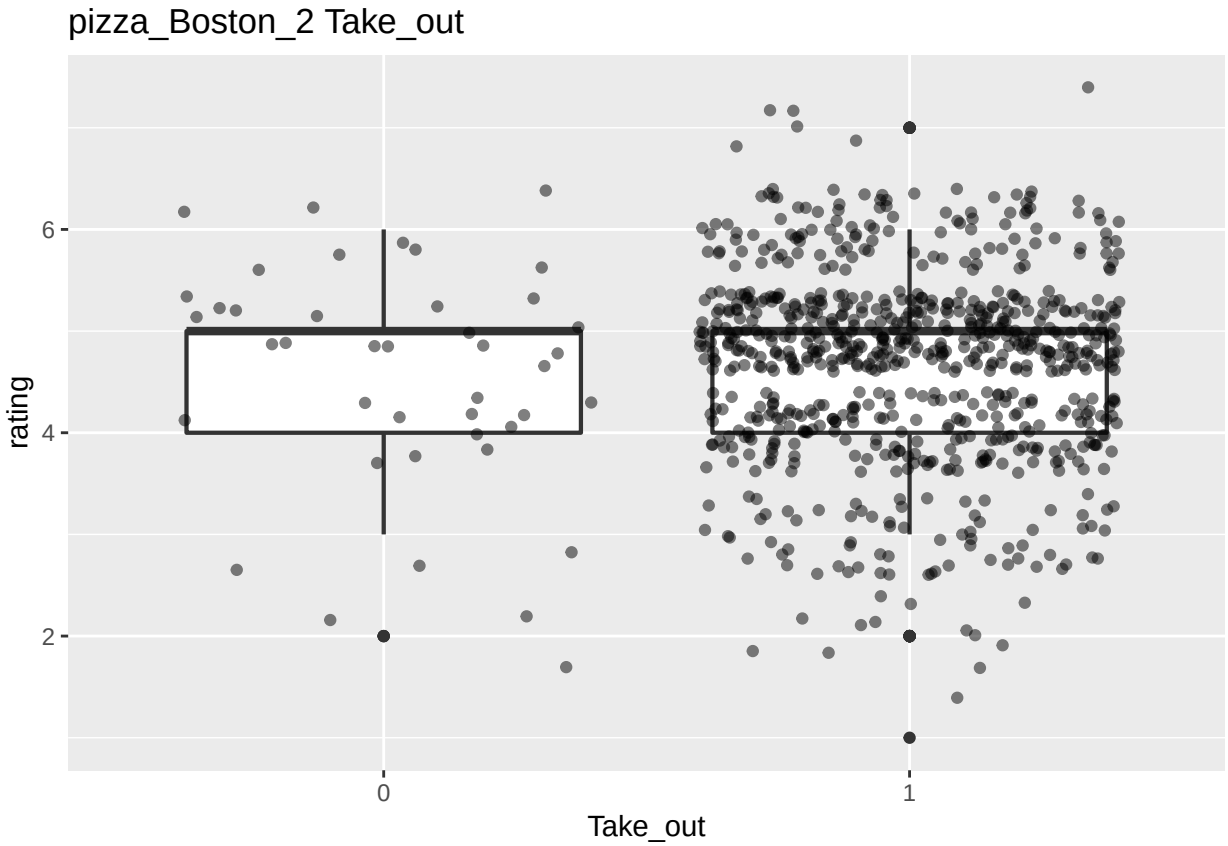

pizza_Boston_2 Liked_by_Vegetarians
rating
6
4
2
0
1
Liked_by_Vegetarians

```
##
## [[4]]
```

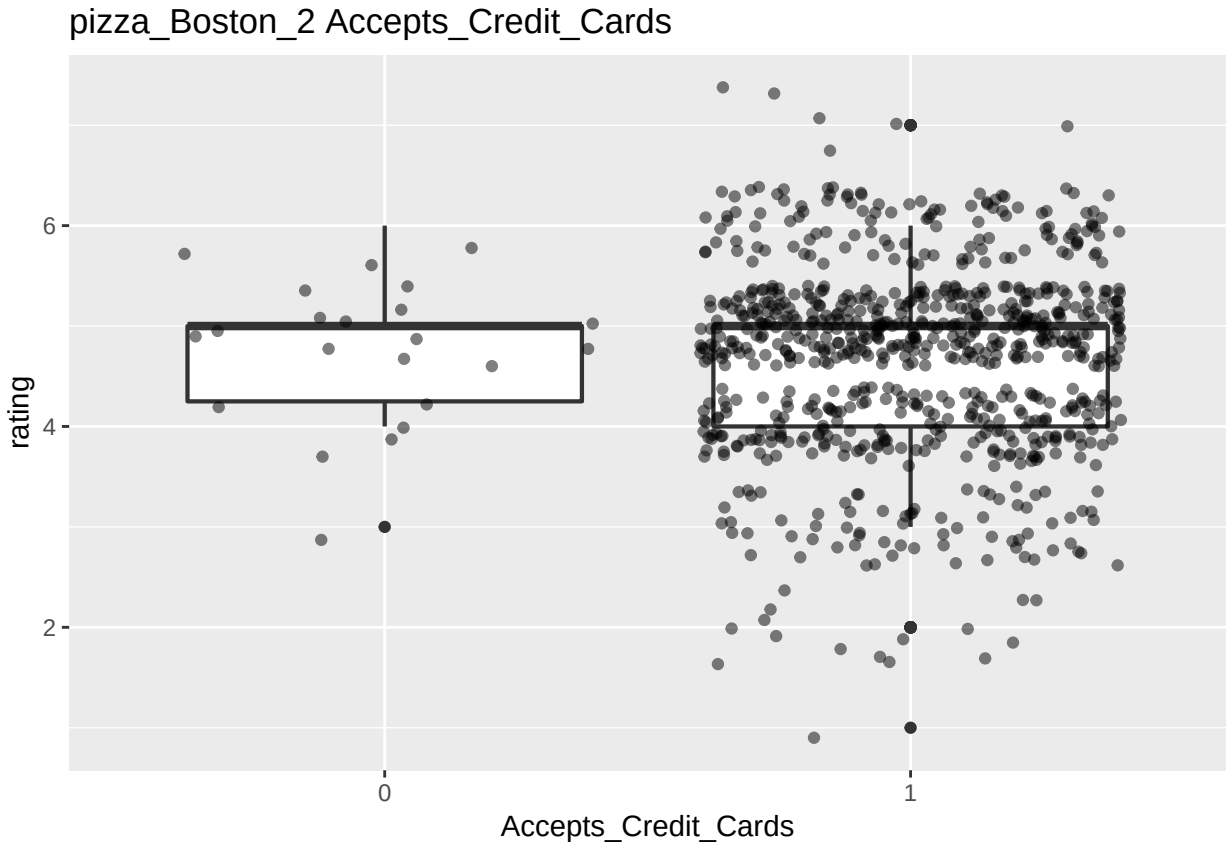

pizza_Boston_2 Takes_Reservations
rating
6
4
2
0
1
Takes_Reservations

```
##
## [[5]]
```

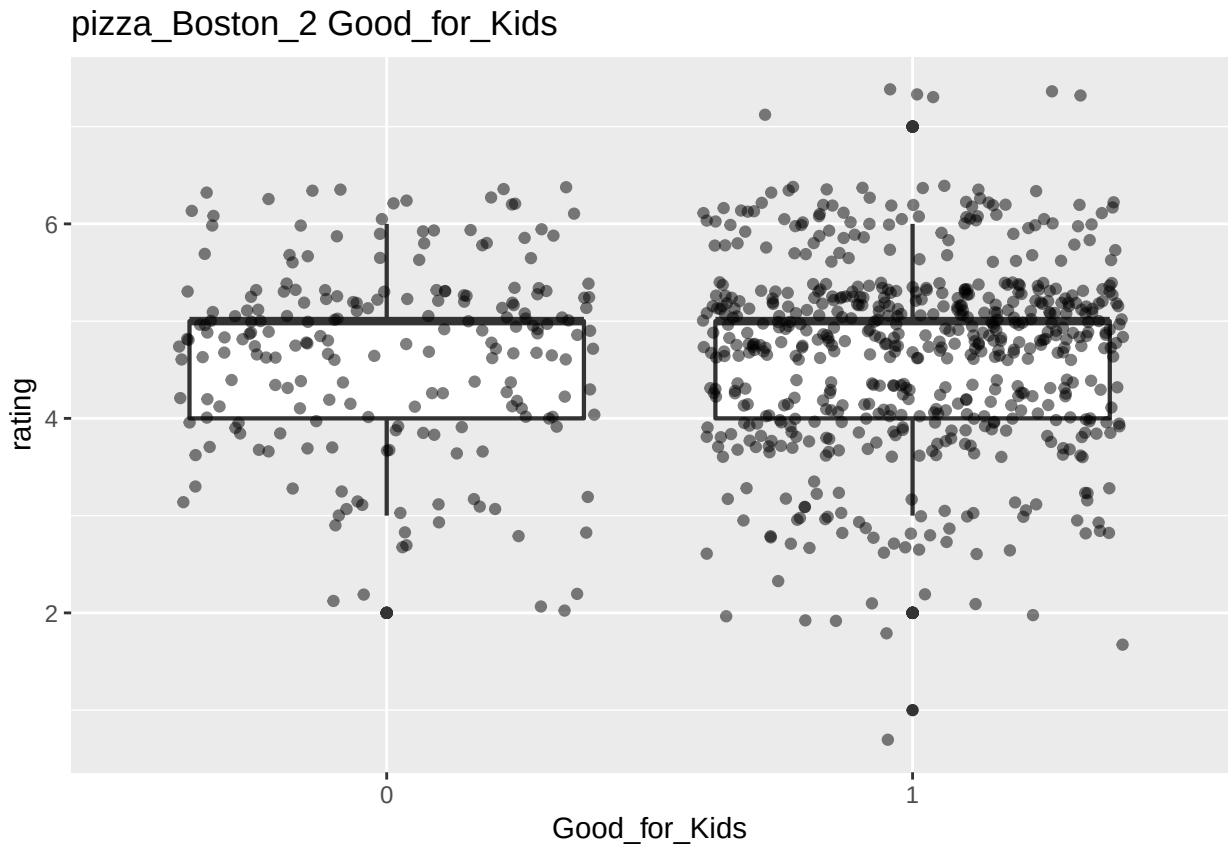

```
##
## [[6]]
```



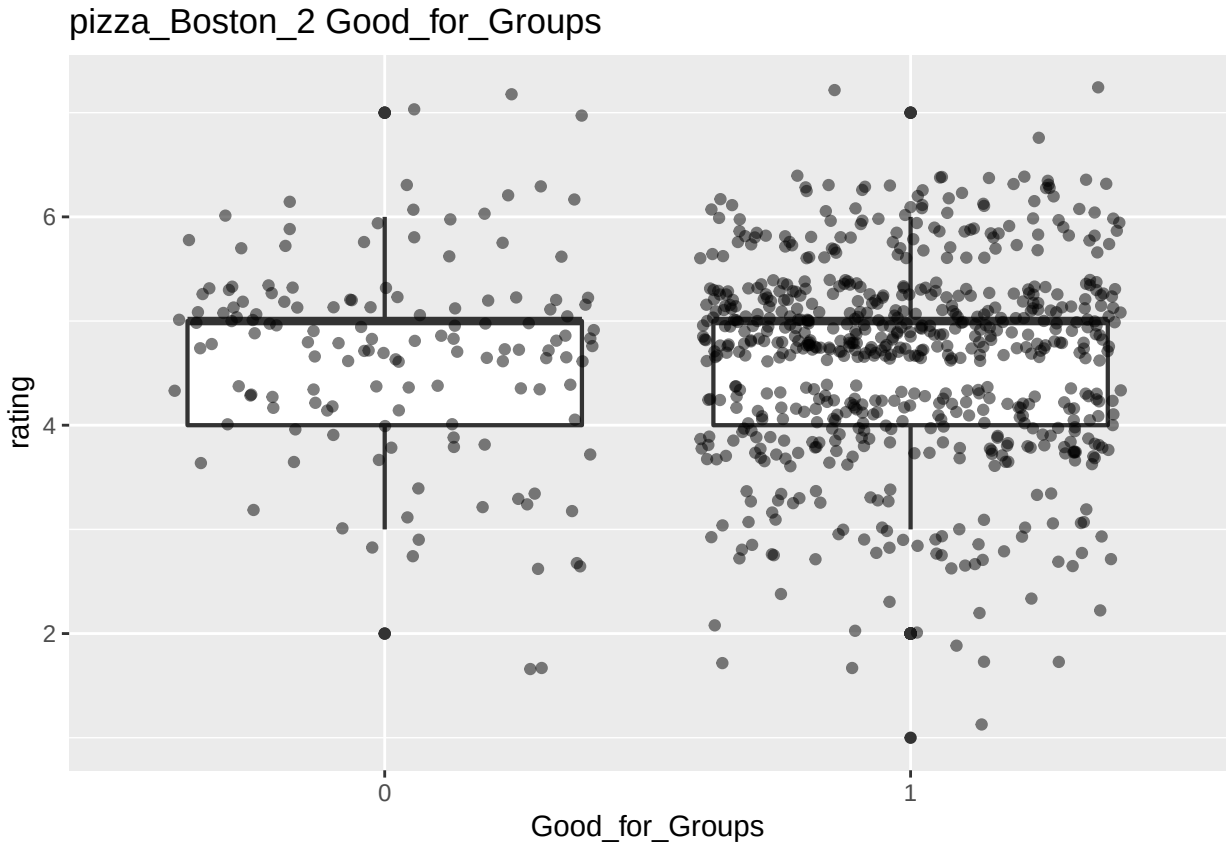

```
##
## [[7]]
```


pizza_Boston_2 Accepts_Credit_Cards
rating
6
4
2
0
1
Accepts_Credit_Cards

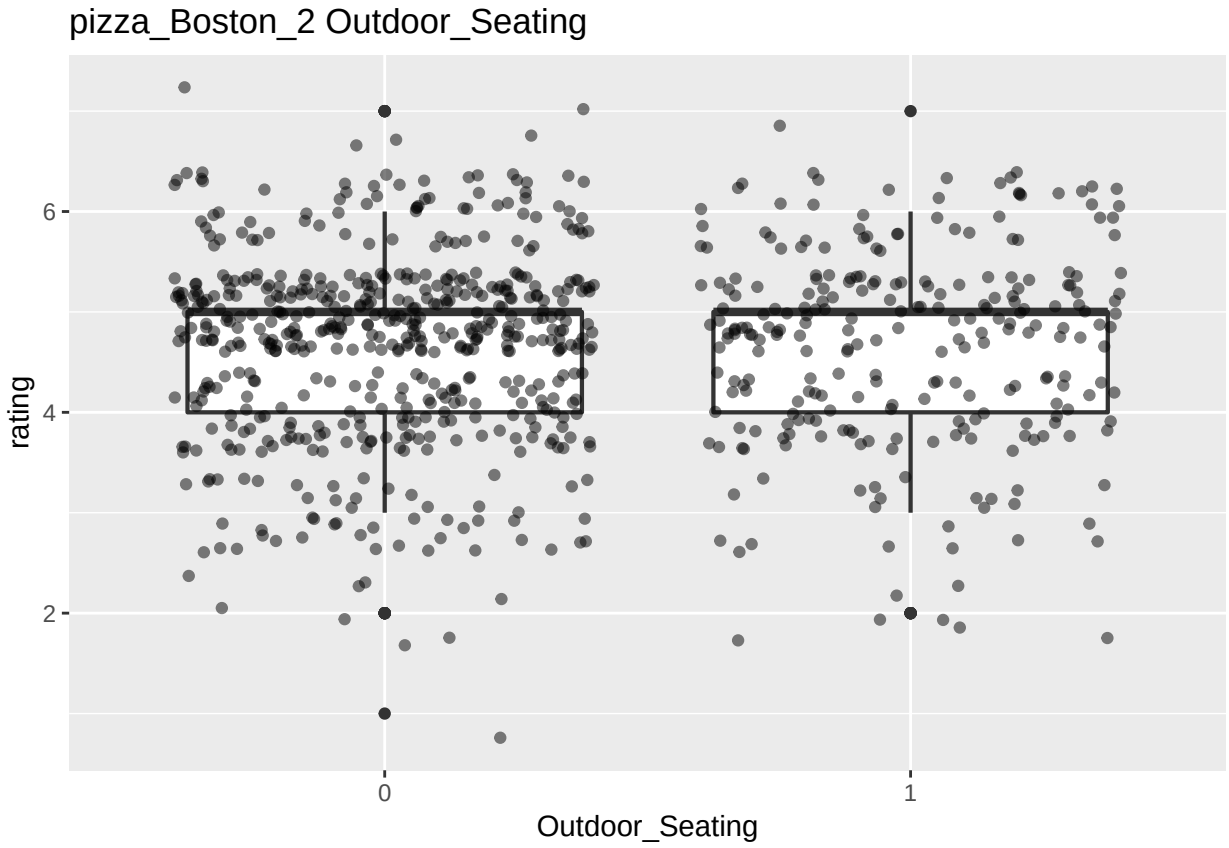
```
##
## [[8]]
```



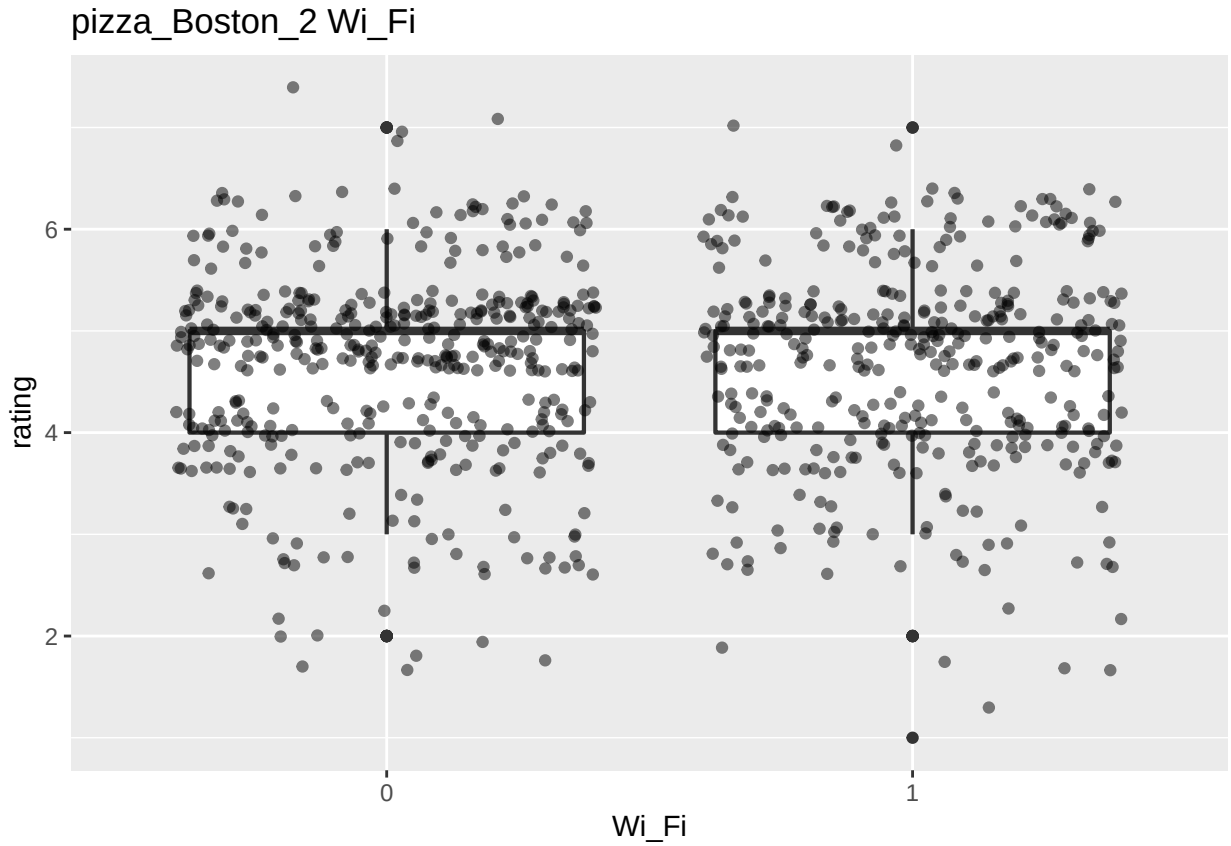

```
##
## [[9]]
```



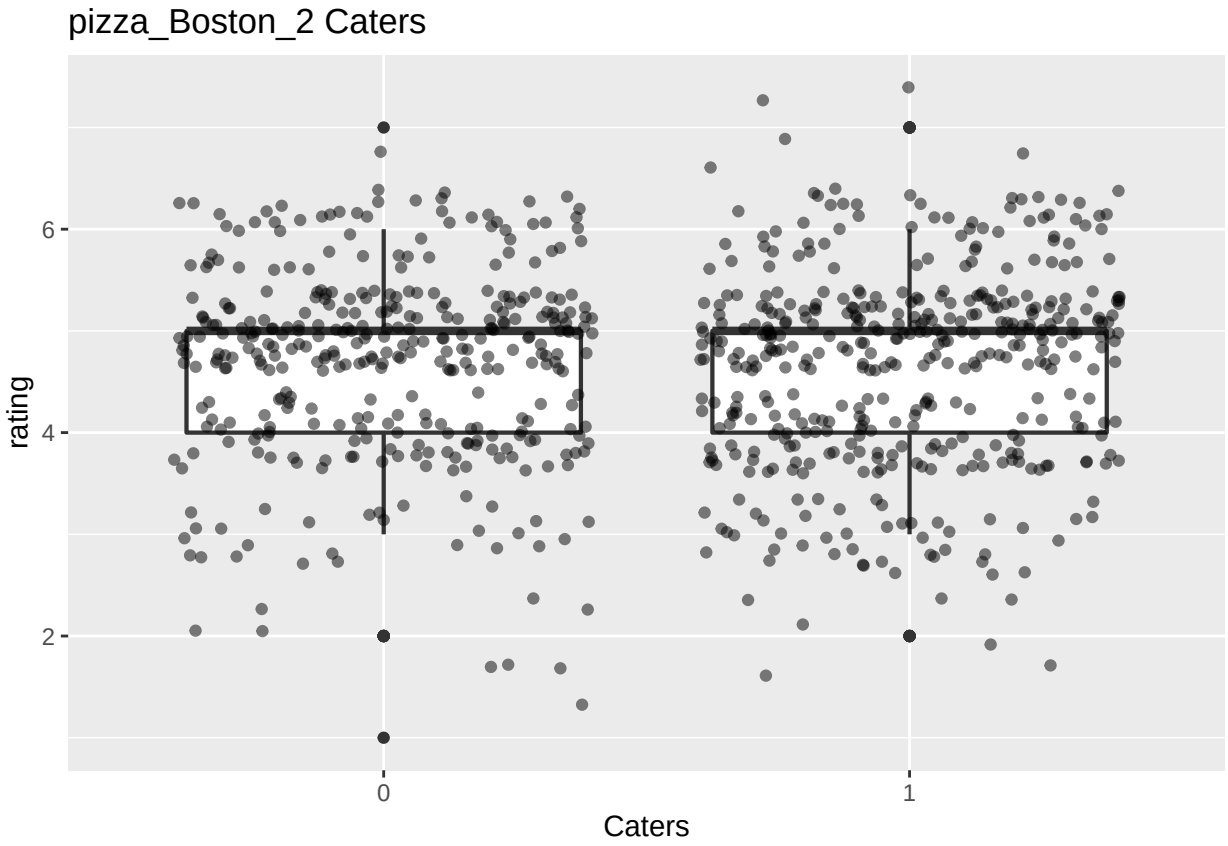

```
##
## [[10]]
```


pizza_Boston_2 Outdoor_Seating
rating
6
4
2
0
1
Outdoor_Seating

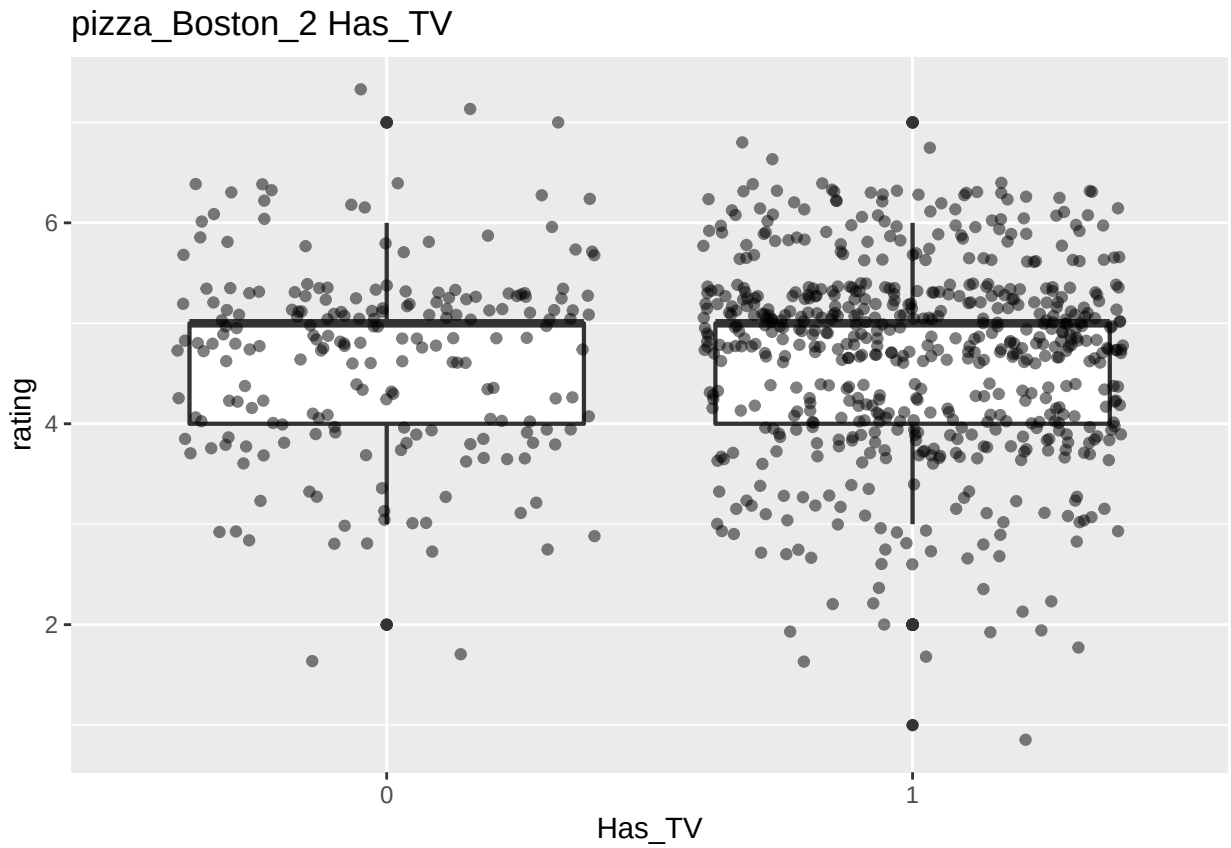
```
##
## [[11]]
```



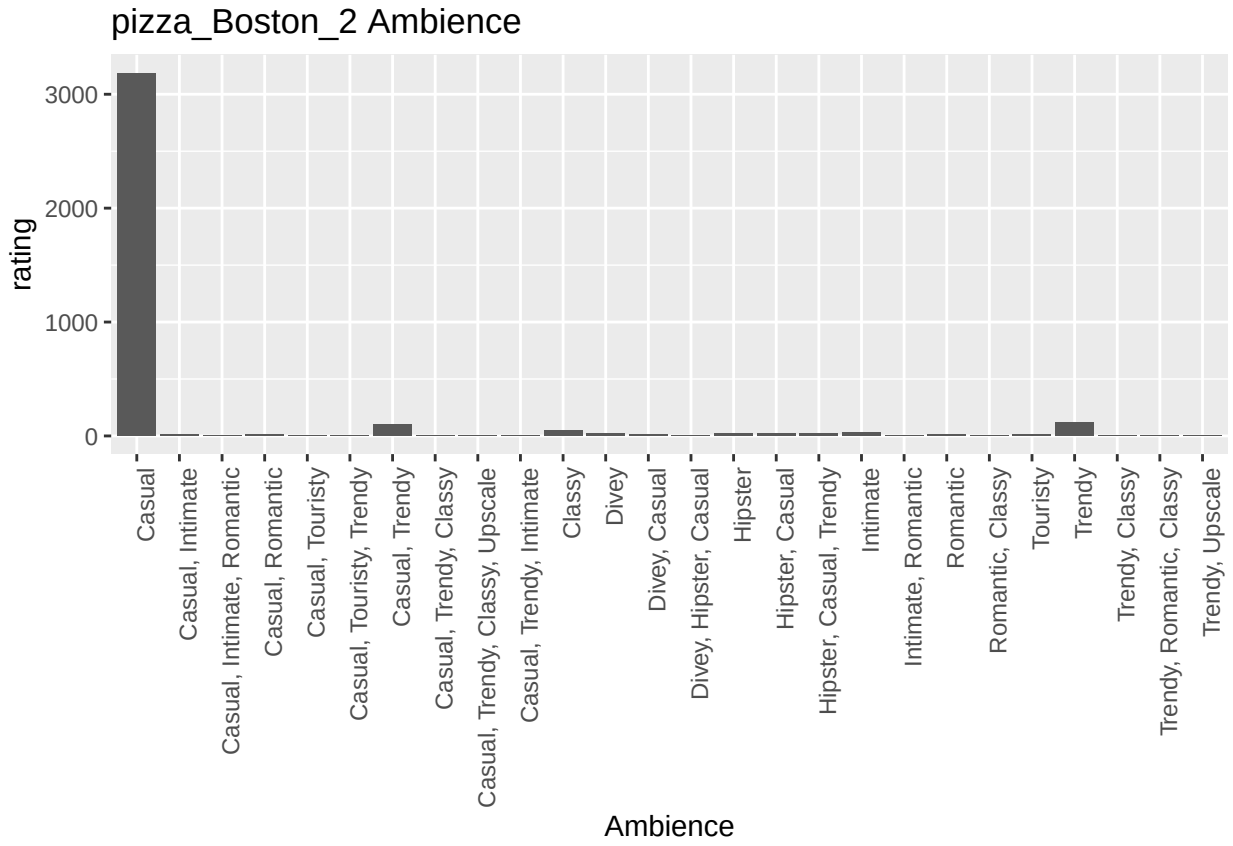

```
##
## [[12]]
```



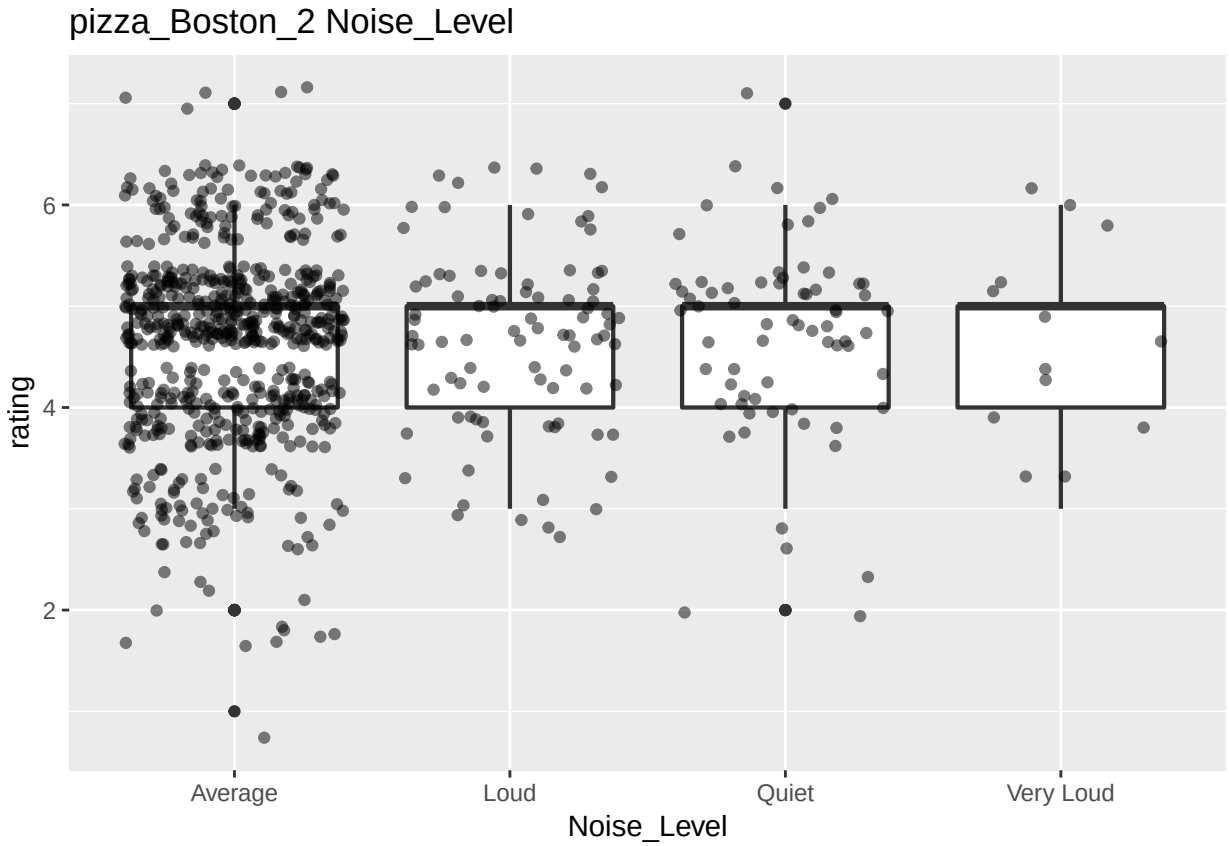

```
##
## [[13]]
```



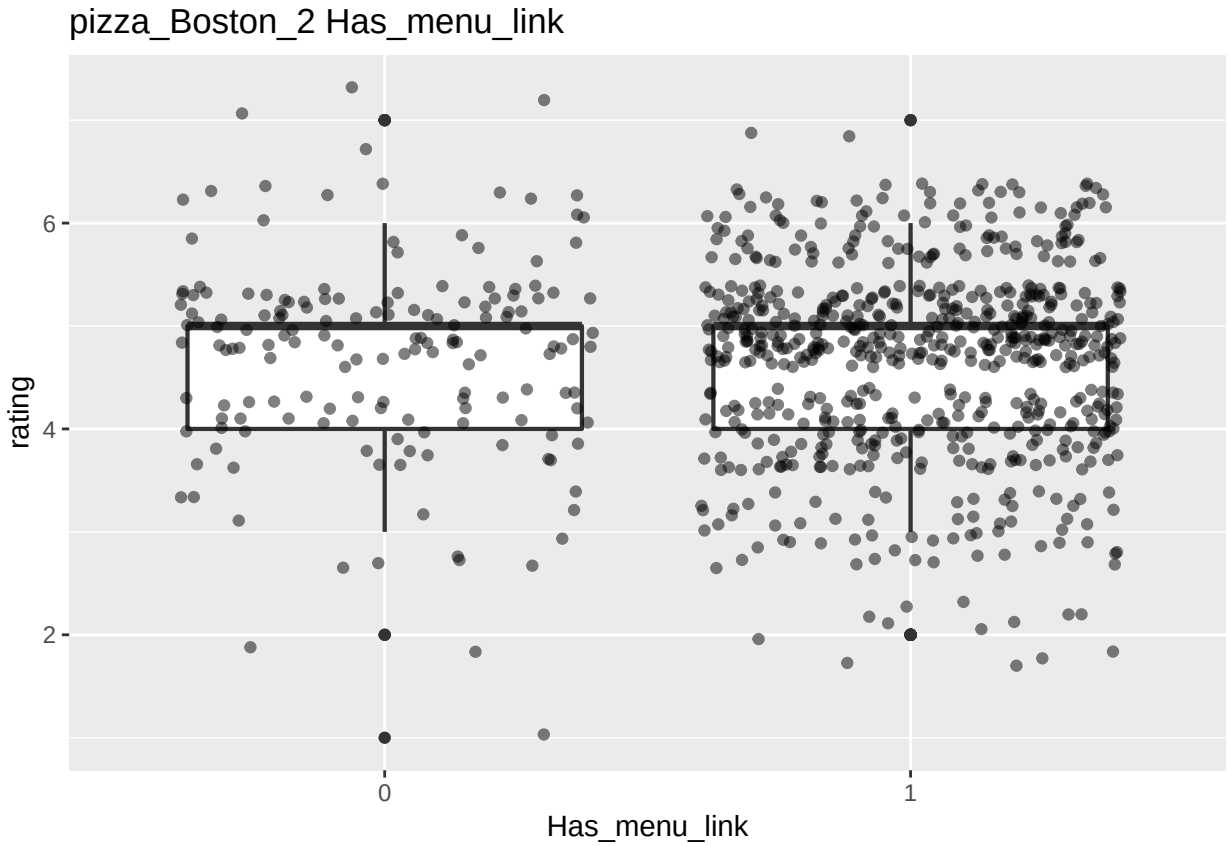

```
##
## [[14]]
```


pizza_Boston_2 Ambience
rating
3000
2000
1000
0
Casual
Casual, Intimate
Casual, Intimate, Romantic
Casual, Romantic
Casual, Touristy
Casual, Touristy, Trendy
Casual, Trendy
Casual, Trendy, Classy
Casual, Trendy, Classy, Upscale
Casual, Trendy, Intimate
Classy
Divey
Divey, Casual
Divey, Hipster, Casual
Hipster
Hipster, Casual
Hipster, Casual, Trendy
Intimate
Intimate, Romantic
Romantic
Romantic, Classy
Touristy
Trendy
Trendy, Classy
Trendy, Romantic, Classy
Trendy, Upscale
Ambience

```
##
## [[15]]
```

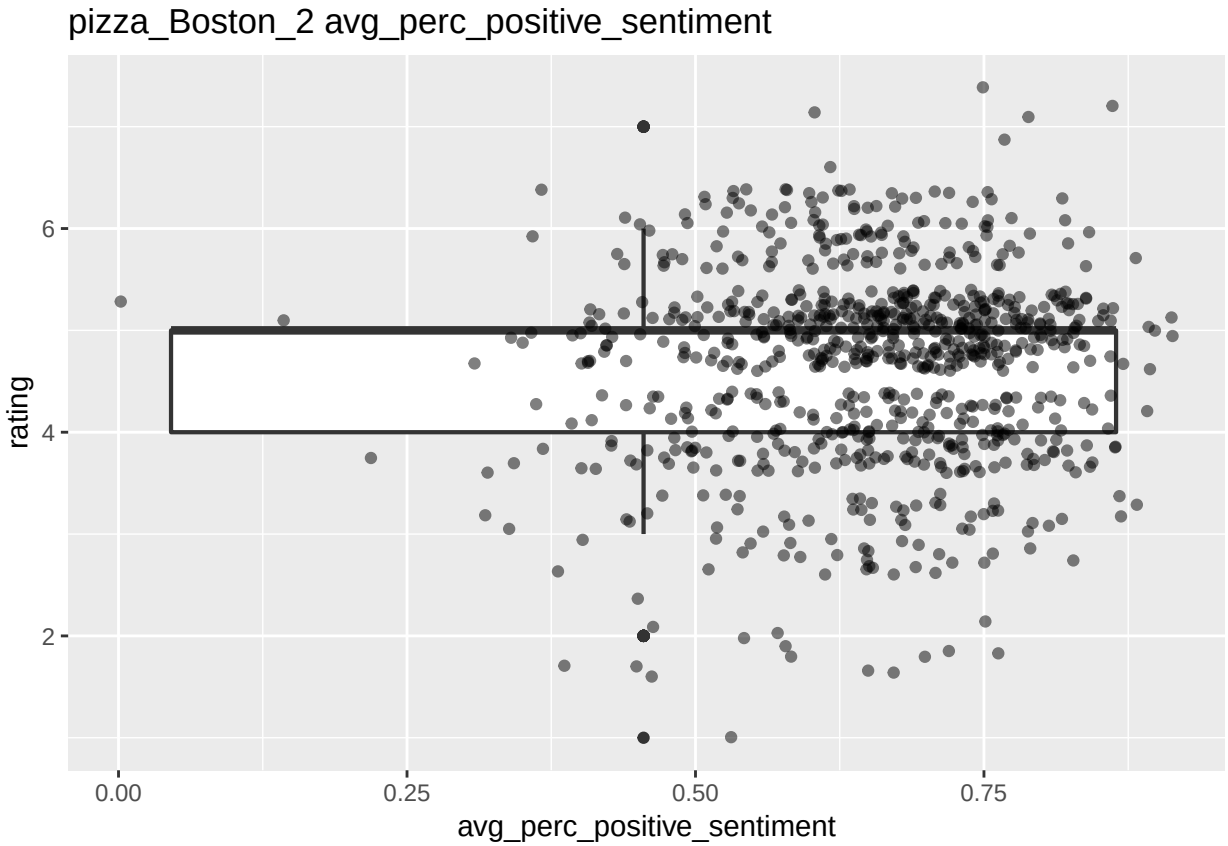

pizza_Boston_2 Noise_Level
rating
6
4
2
Average
Loud
Quiet
Very Loud
Noise_Level

```
##
## [[16]]
```



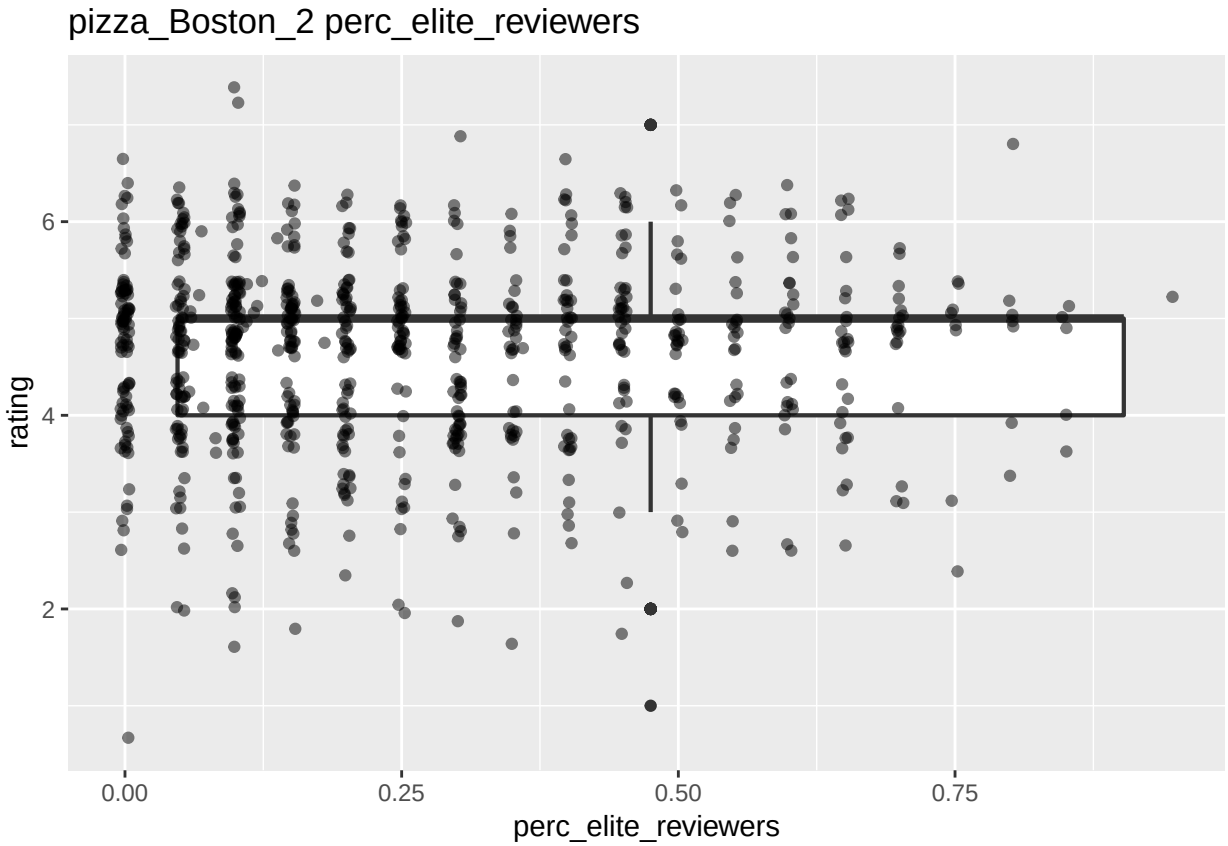

```
##
## [[17]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



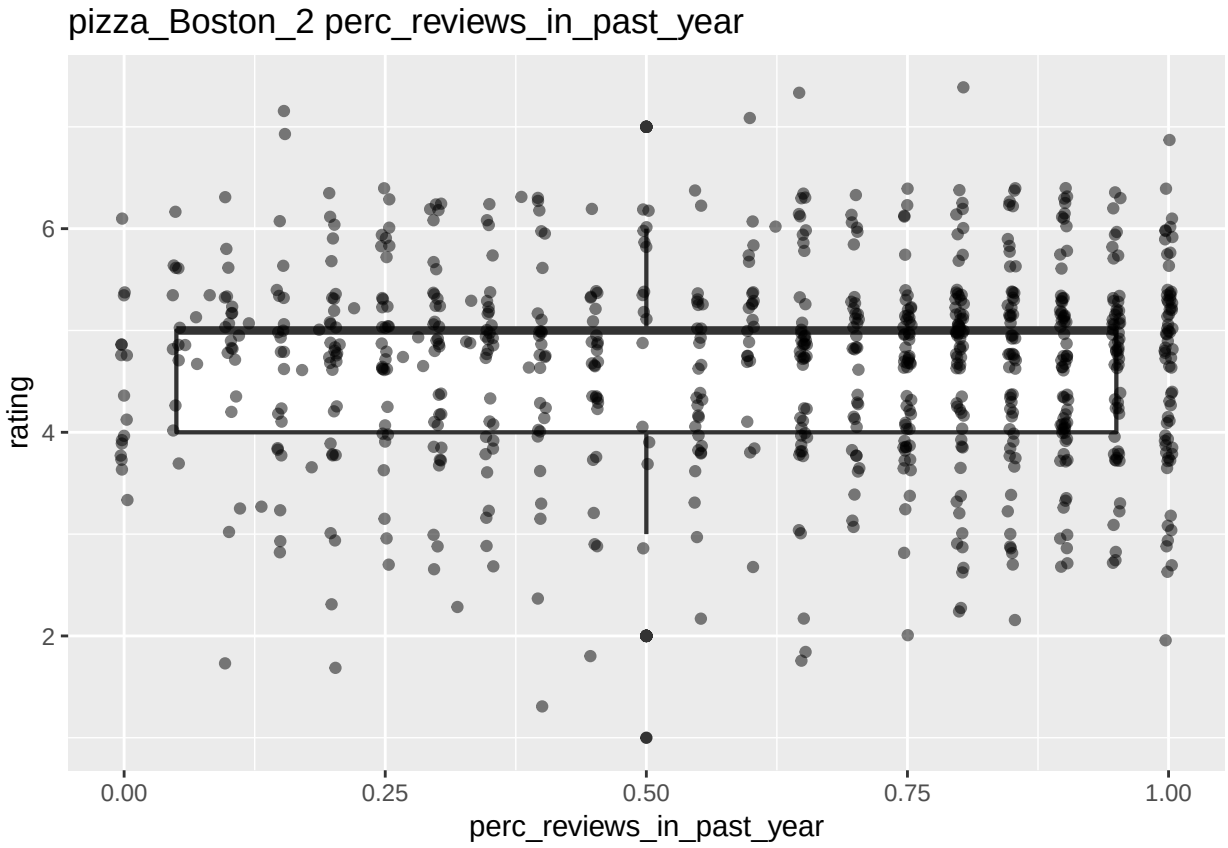

```
##
## [[18]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



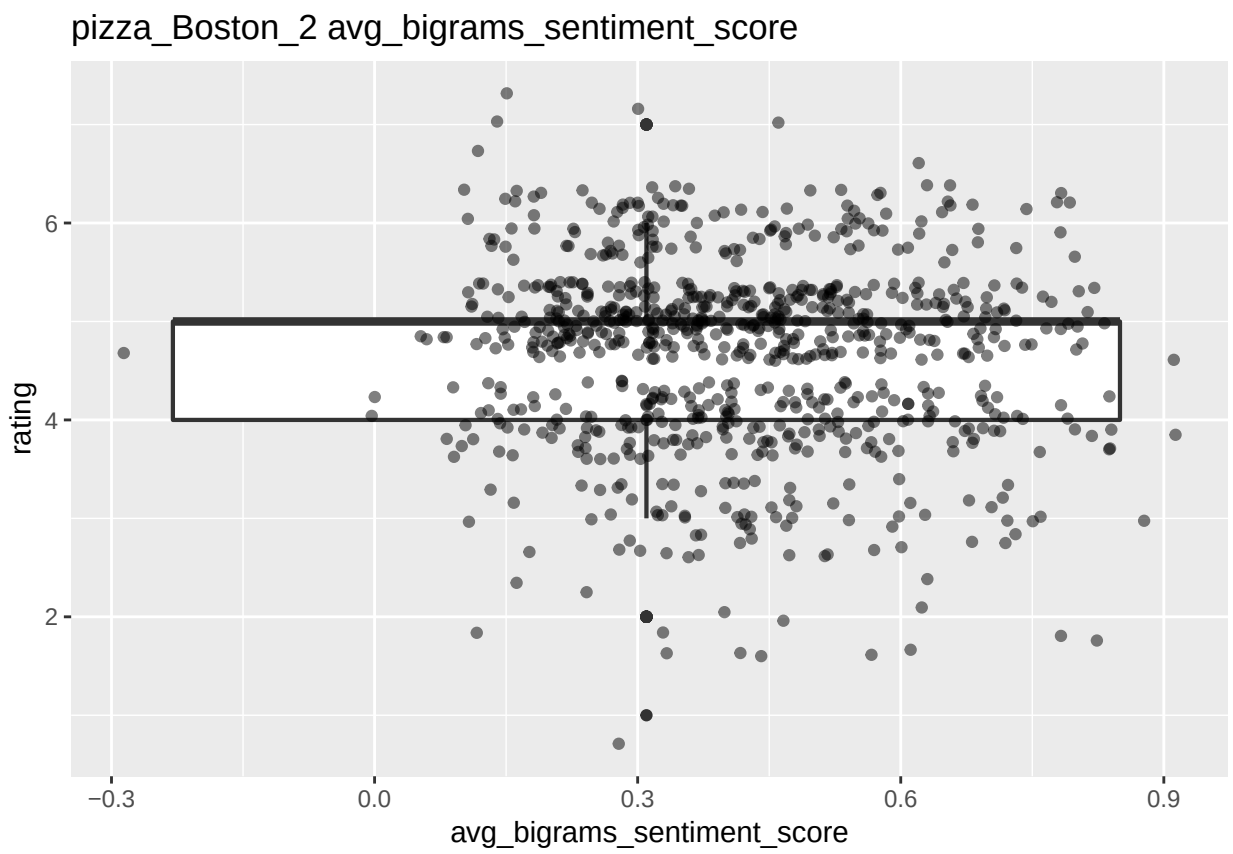

```
##
## [[19]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```




```
##
## [[20]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```

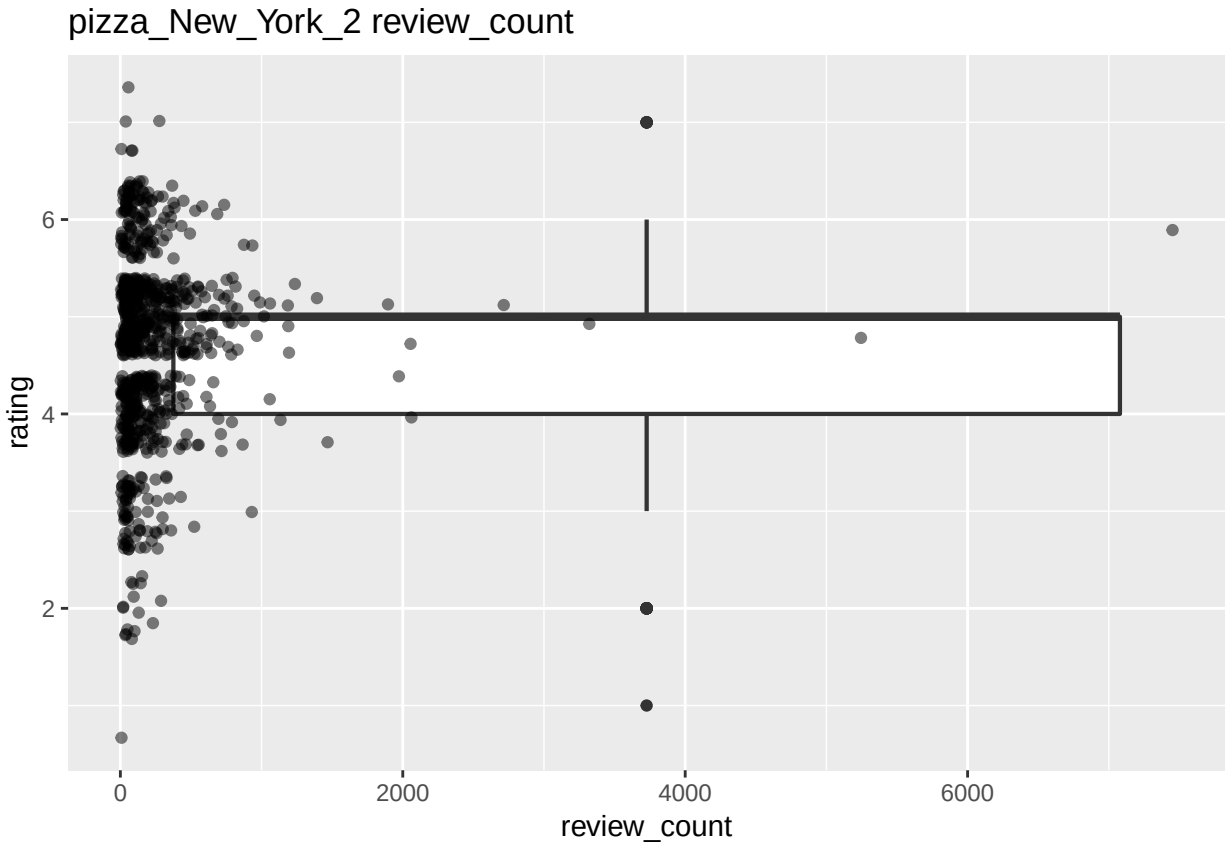


Appendix B: Visualization Plots for pizza_New_York_2

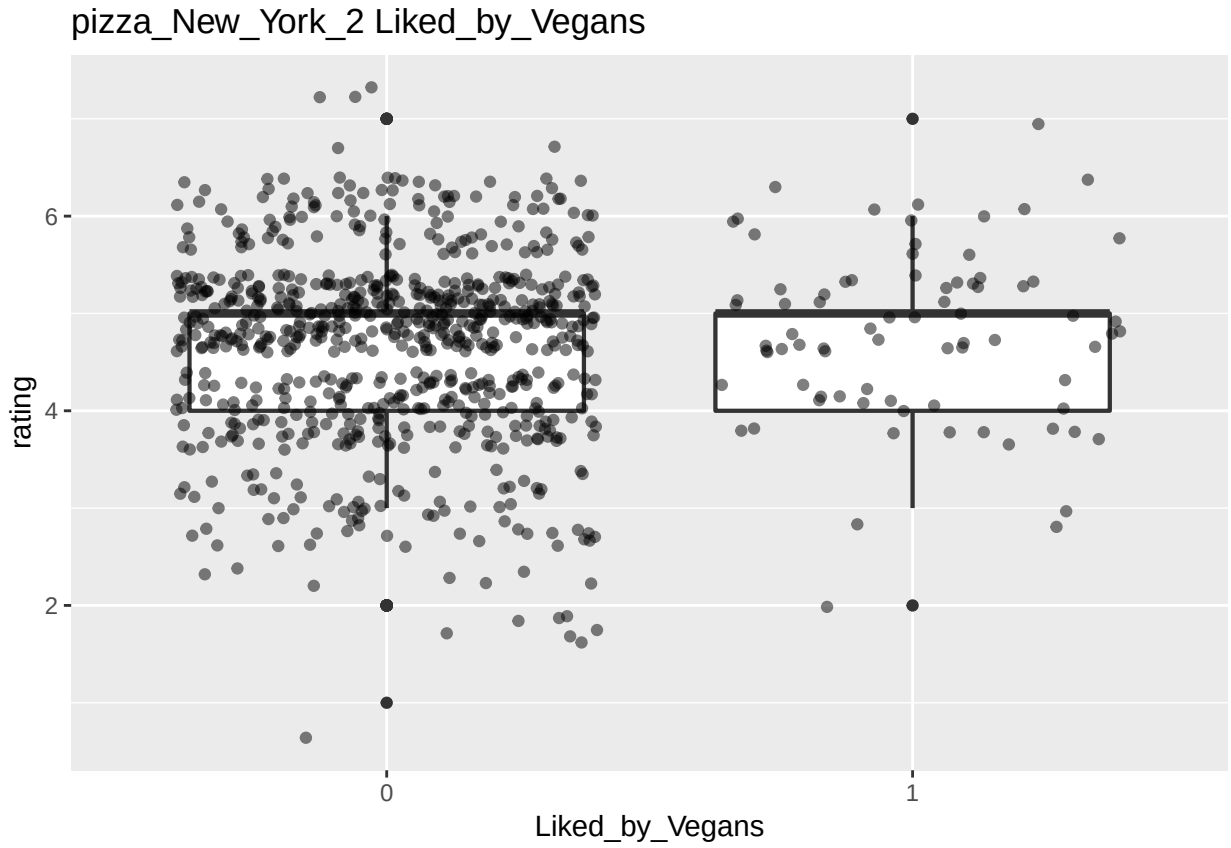
```
get_visualization("pizza", "New York, NY", "2")
```

```
## [[1]]
```

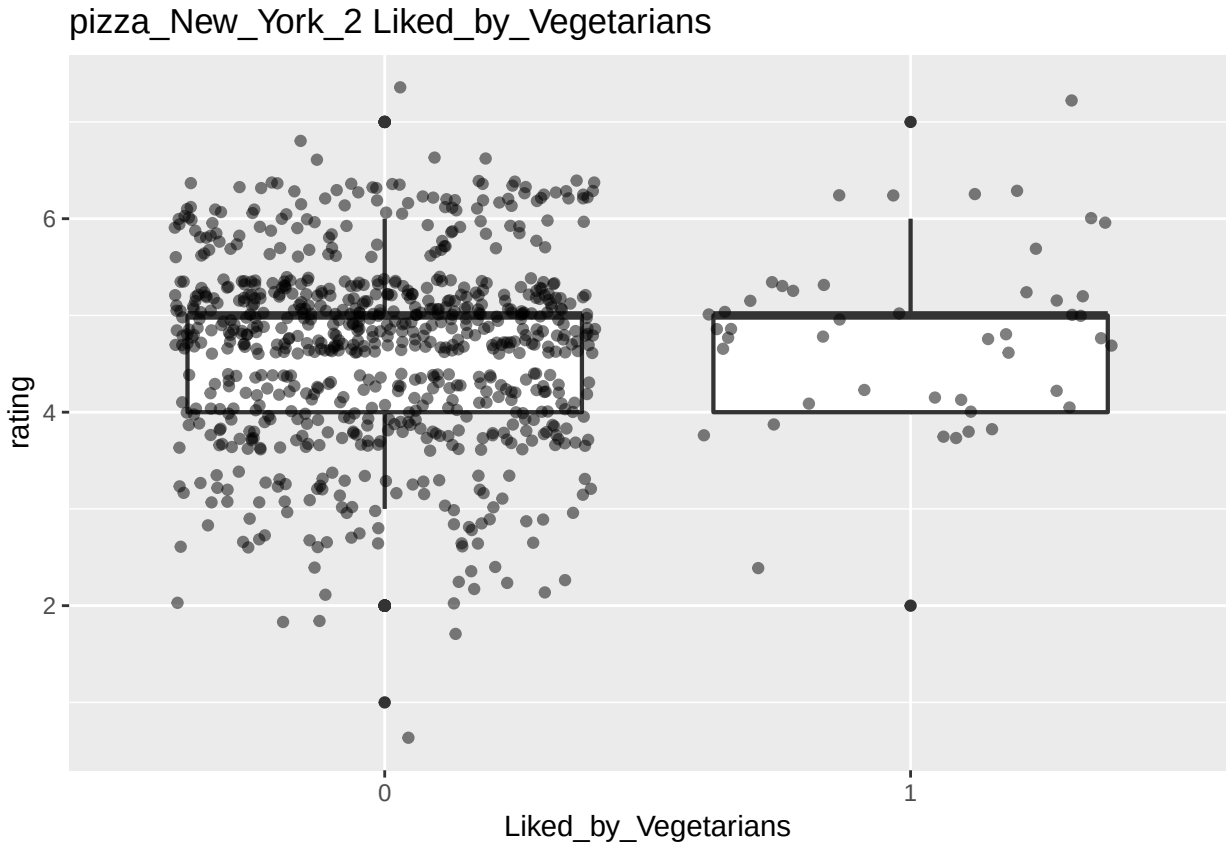
```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```


pizza_New_York_2 review_count
rating
6
4
2
0
2000
4000
6000
review_count

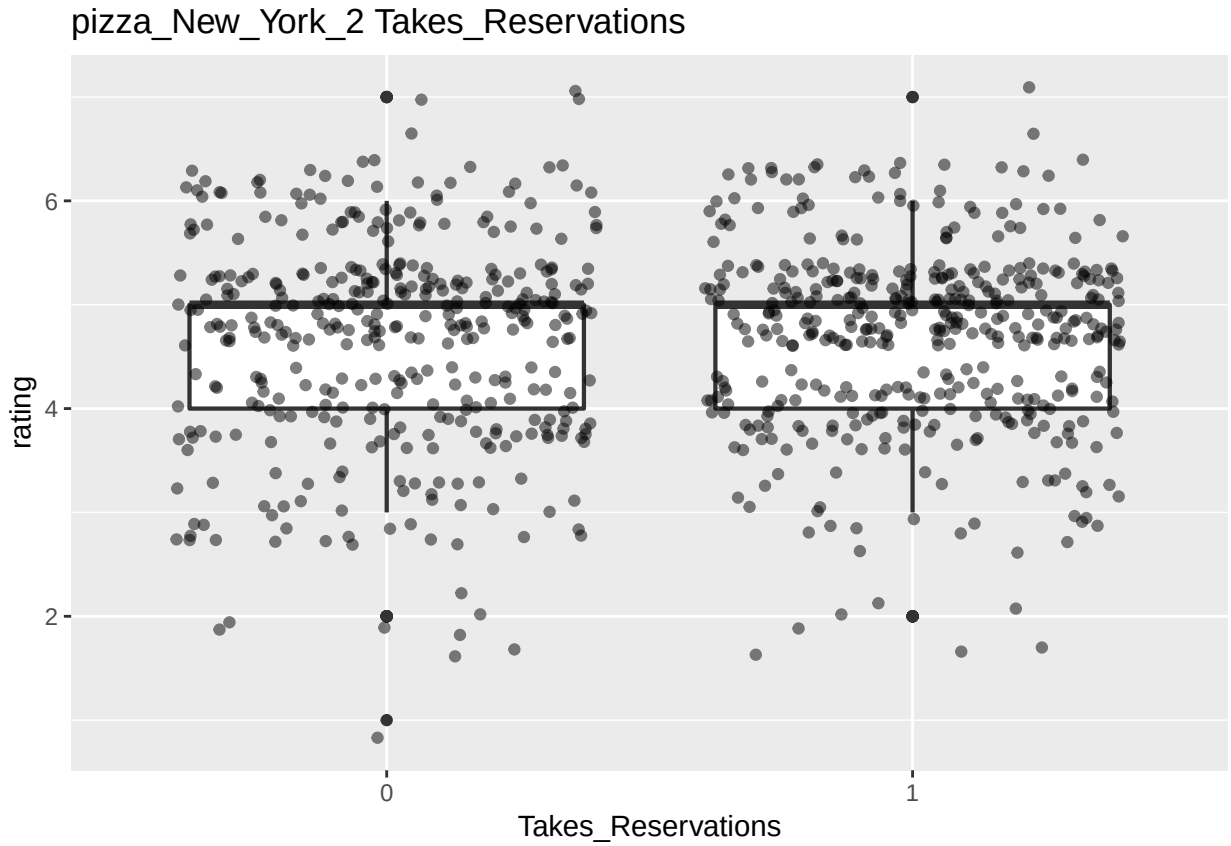
```
##
## [[2]]
```


pizza_New_York_2 Liked_by_Vegans
rating
6
4
2
0
1
Liked_by_Vegans

```
##
## [[3]]
```

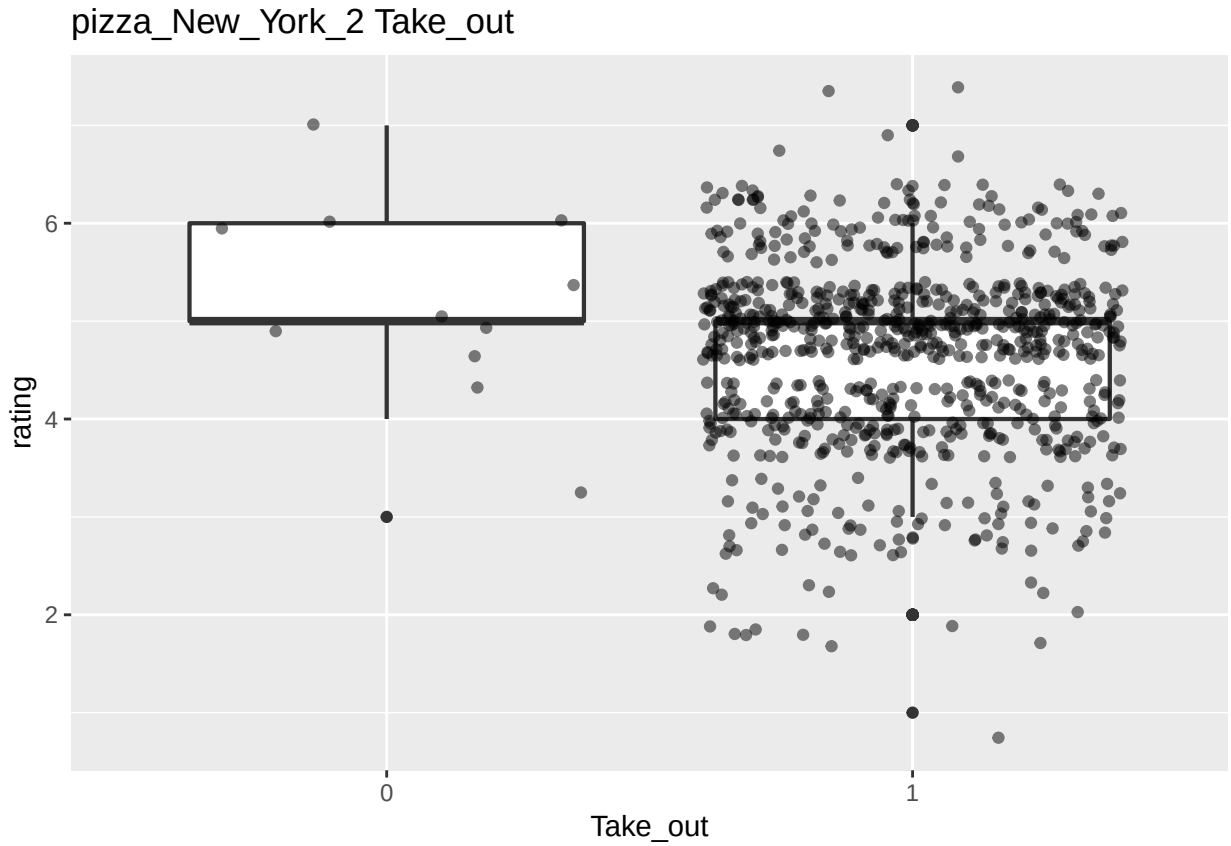

pizza_New_York_2 Liked_by_Vegetarians
rating
6
4
2
0
1
Liked_by_Vegetarians

```
##
## [[4]]
```

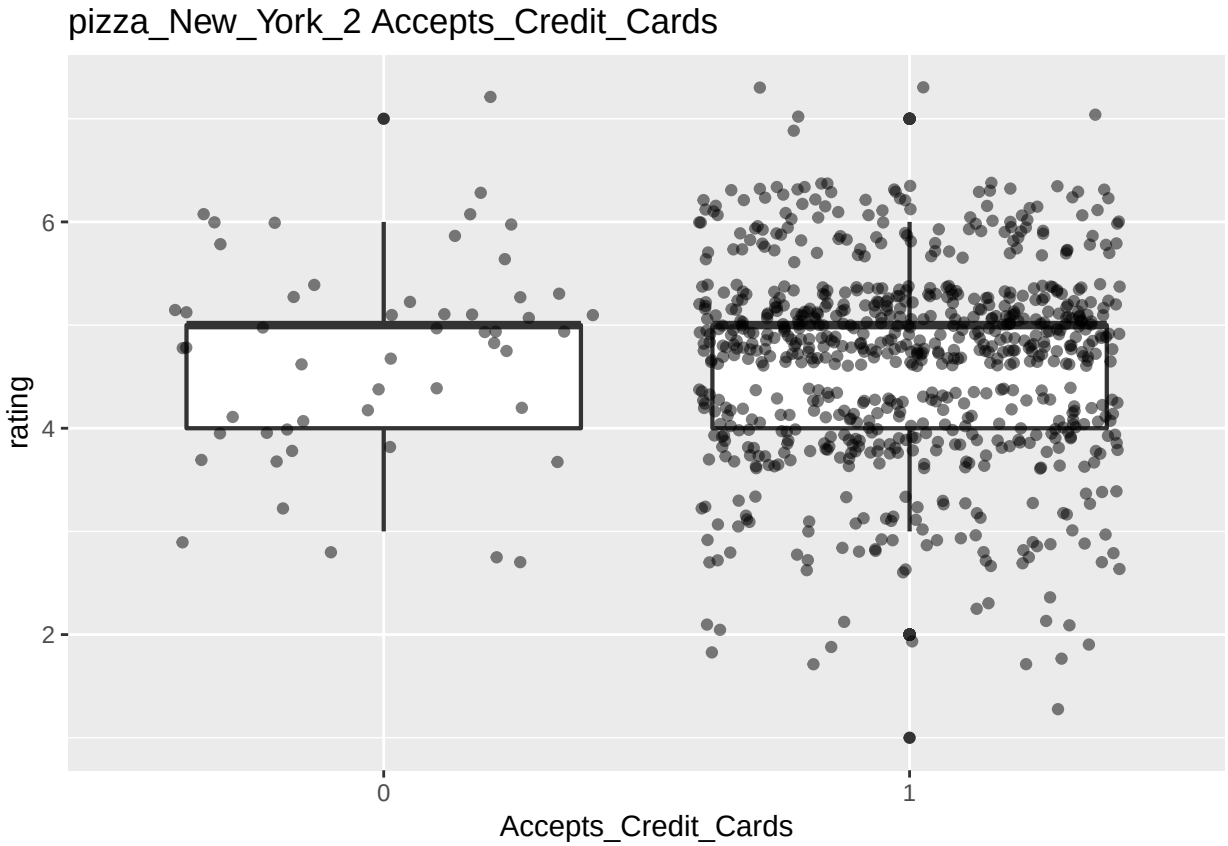

pizza_New_York_2 Takes_Reservations
rating
6
4
2
0
1
Takes_Reservations

```
##
## [[5]]
```

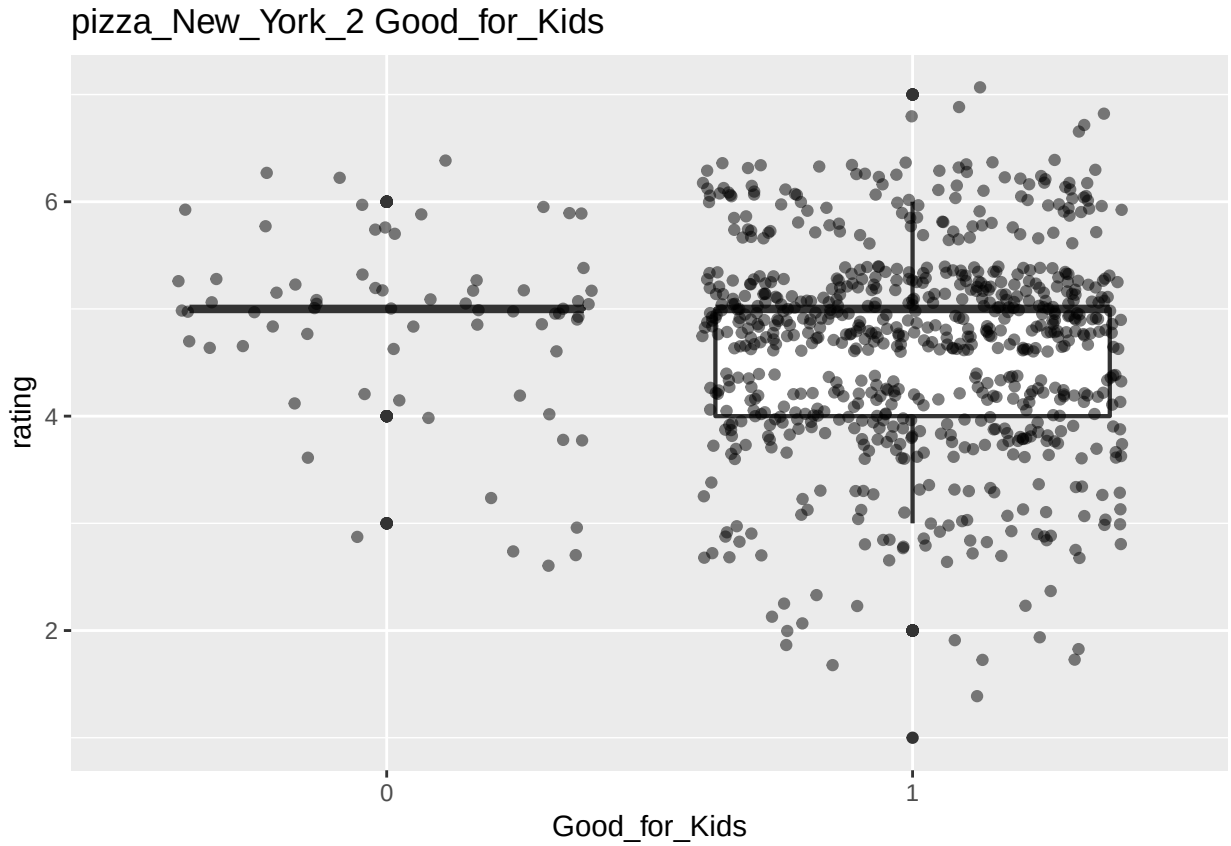

```
##
## [[6]]
```



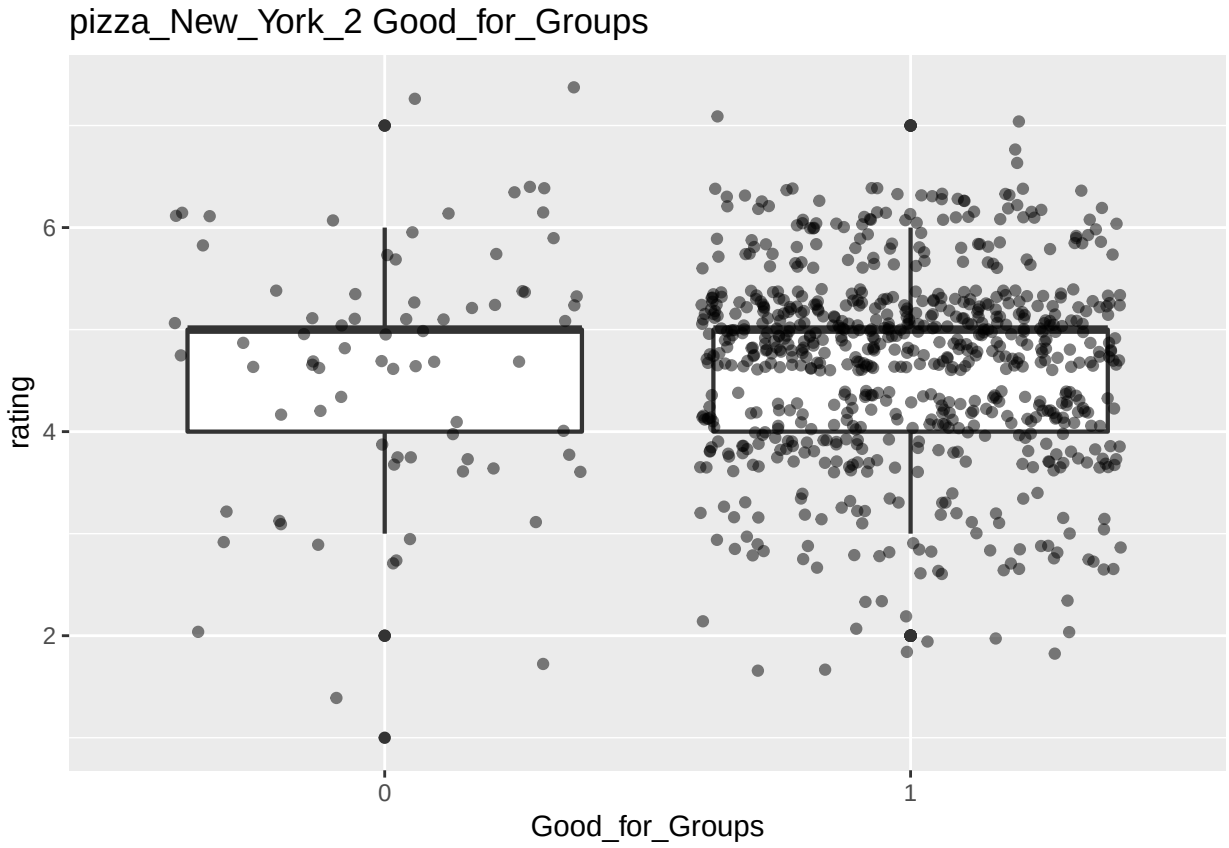

```
##
## [[7]]
```


pizza_New_York_2 Accepts_Credit_Cards
rating
6
4
2
0
1
Accepts_Credit_Cards

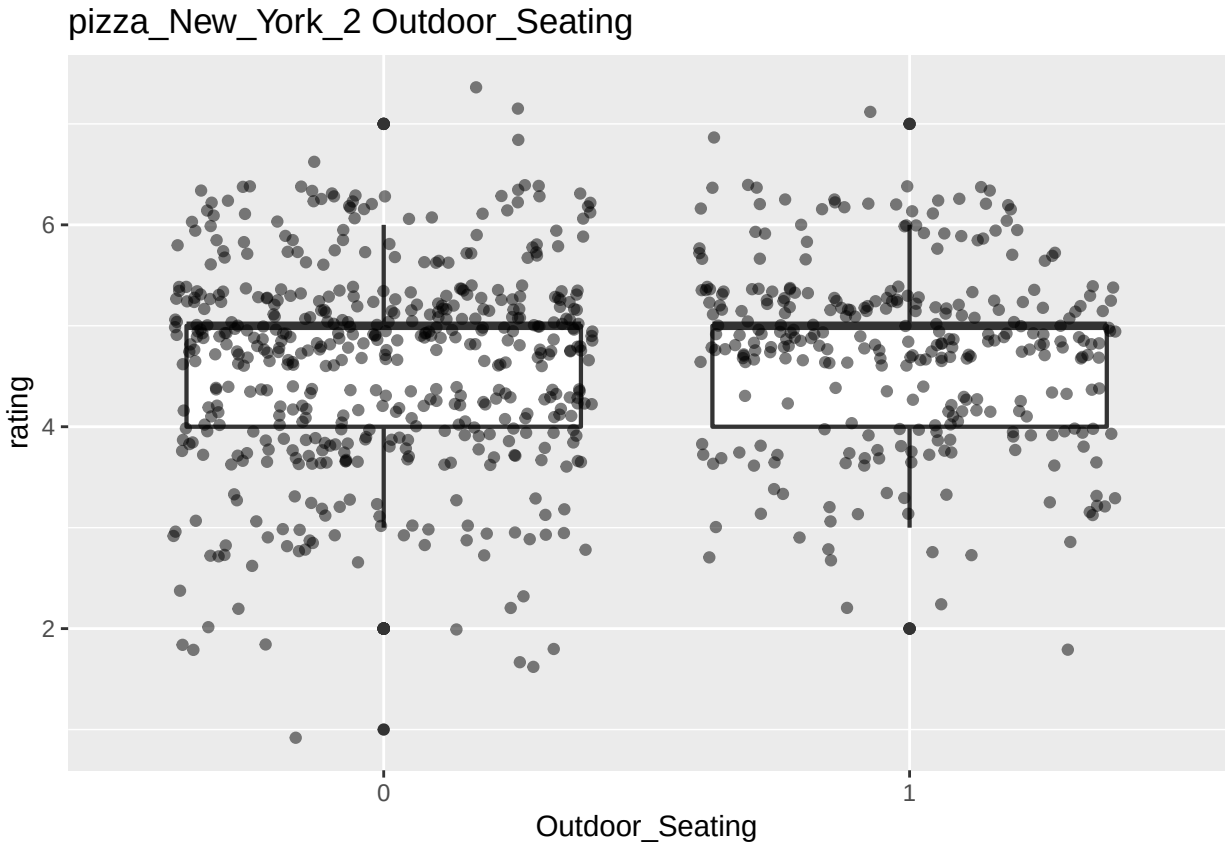
```
##
## [[8]]
```



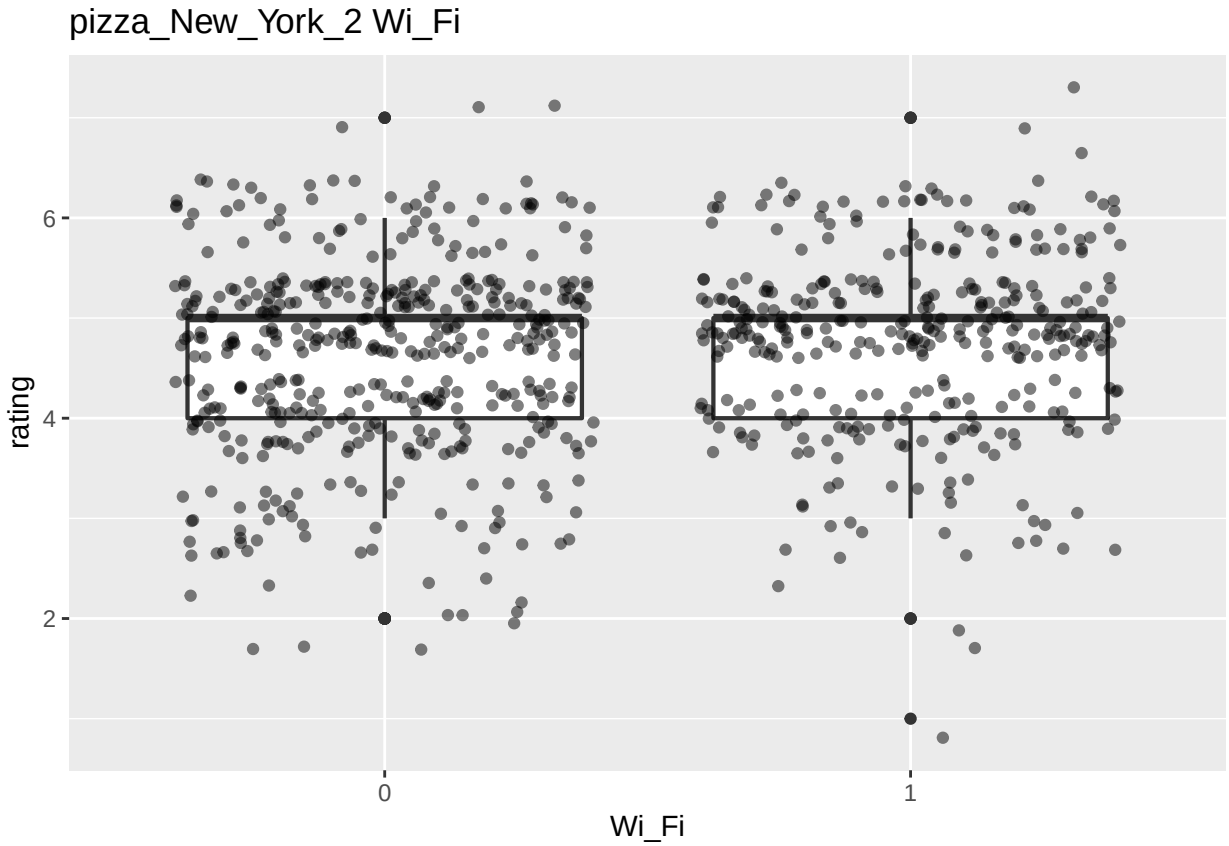

```
##
## [[9]]
```


pizza_New_York_2 Good_for_Groups
rating
6
4
2
0
1
Good_for_Groups

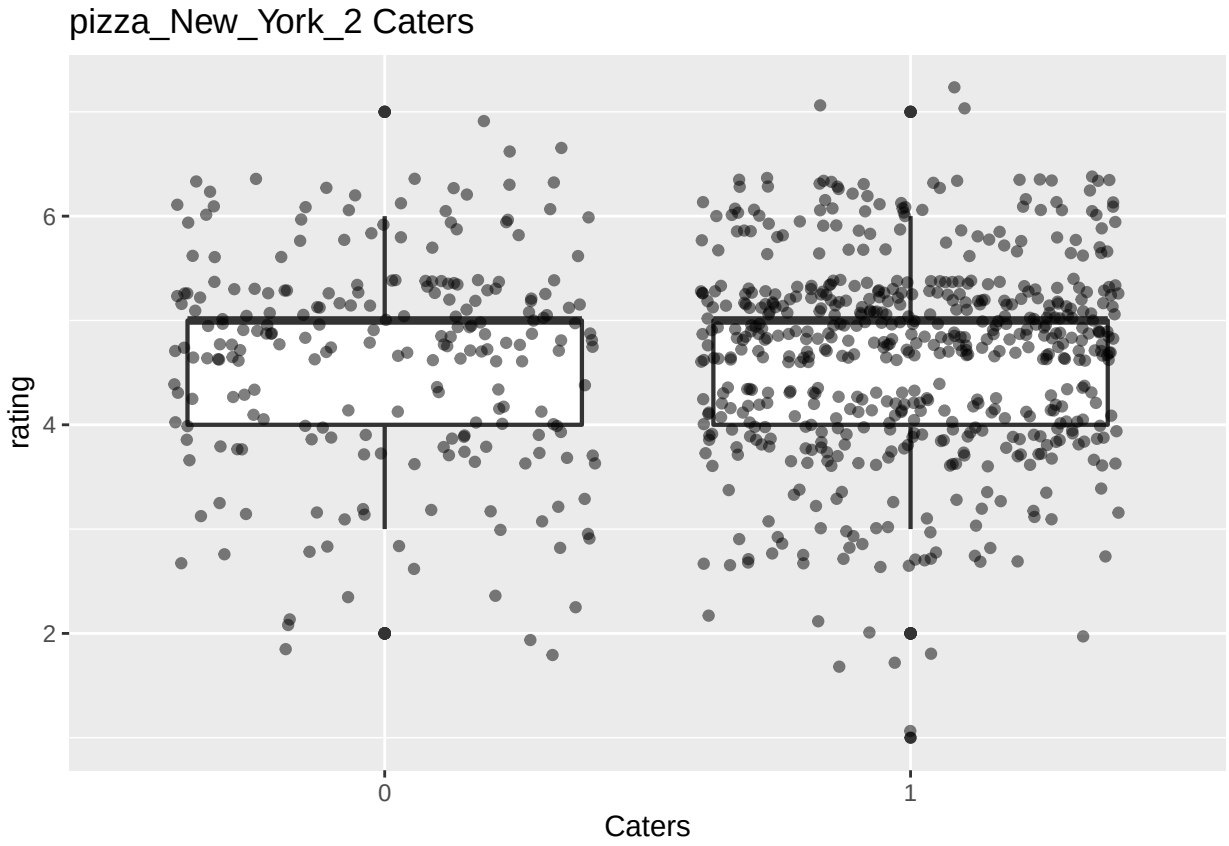
```
##
## [[10]]
```


pizza_New_York_2 Outdoor_Seating
rating
6
4
2
0
1
Outdoor_Seating

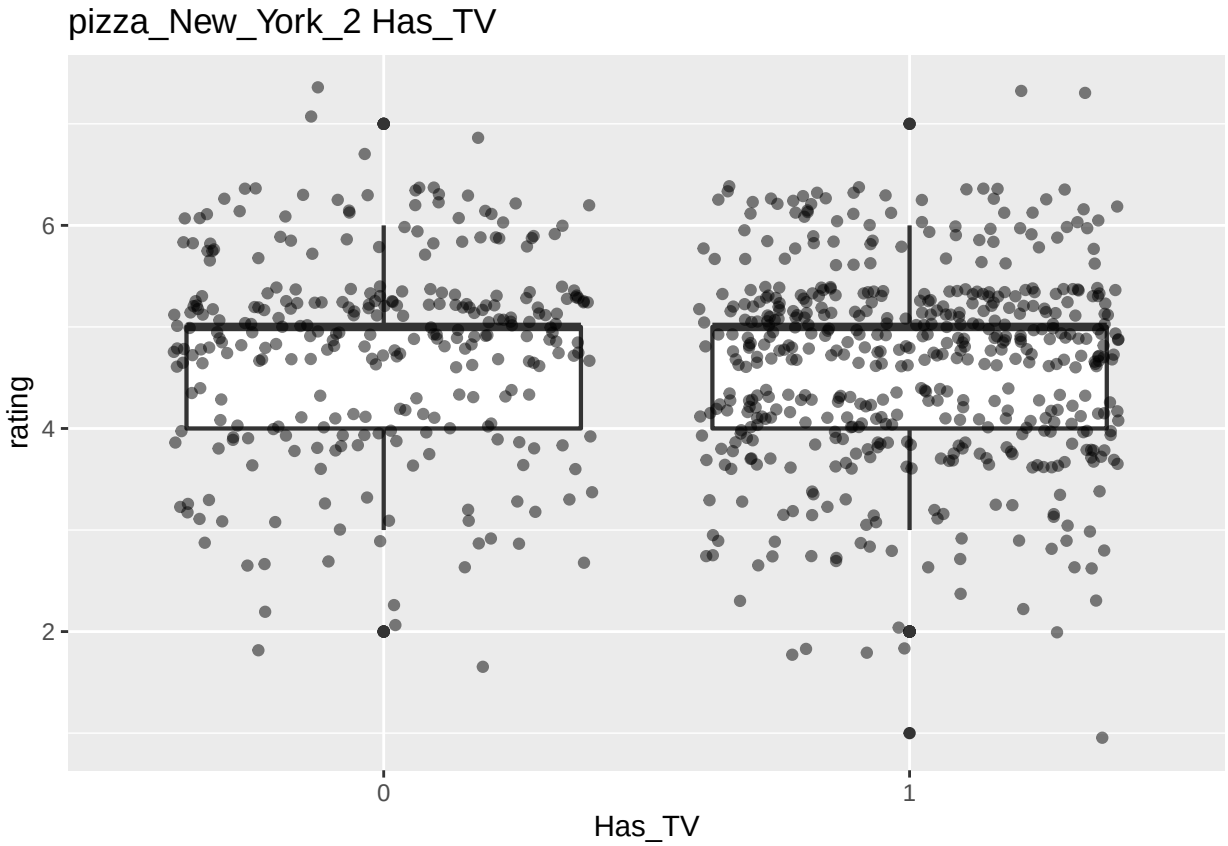
```
##
## [[11]]
```



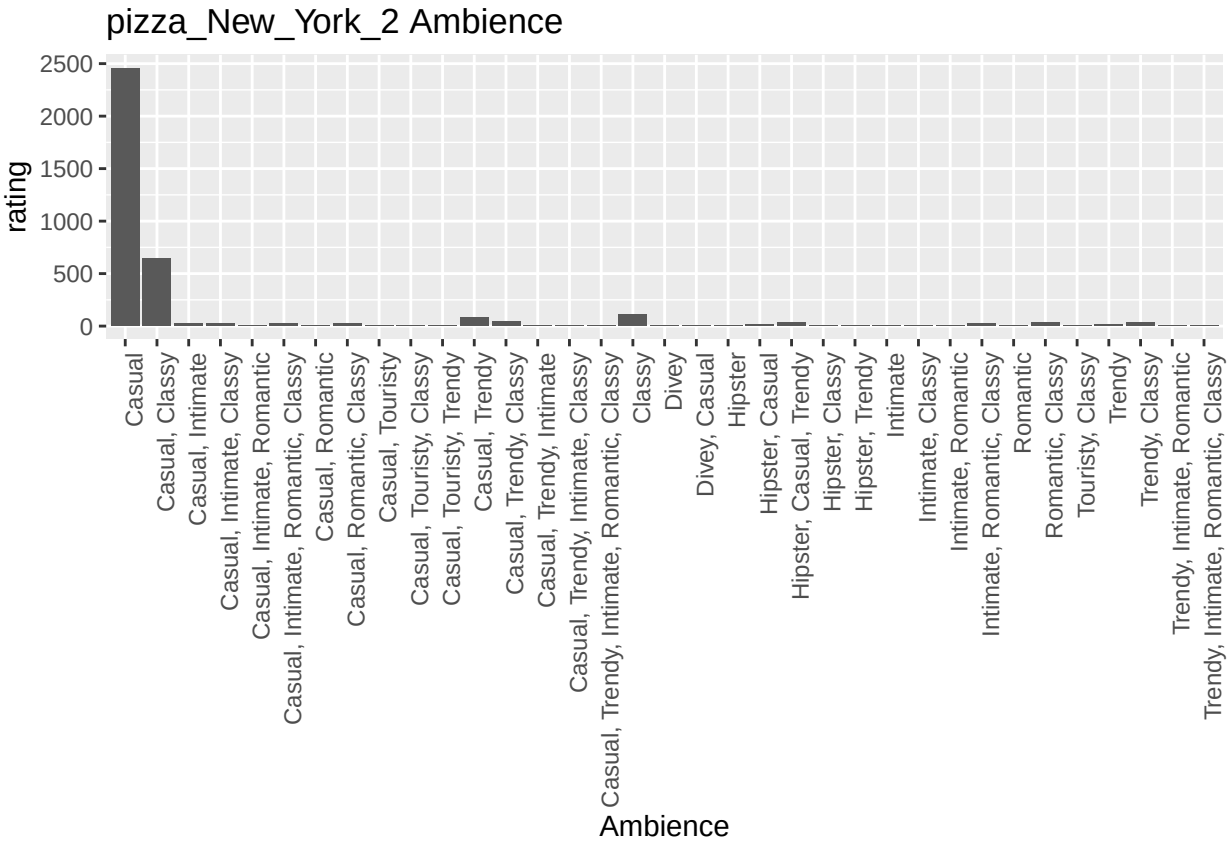

```
##
## [[12]]
```



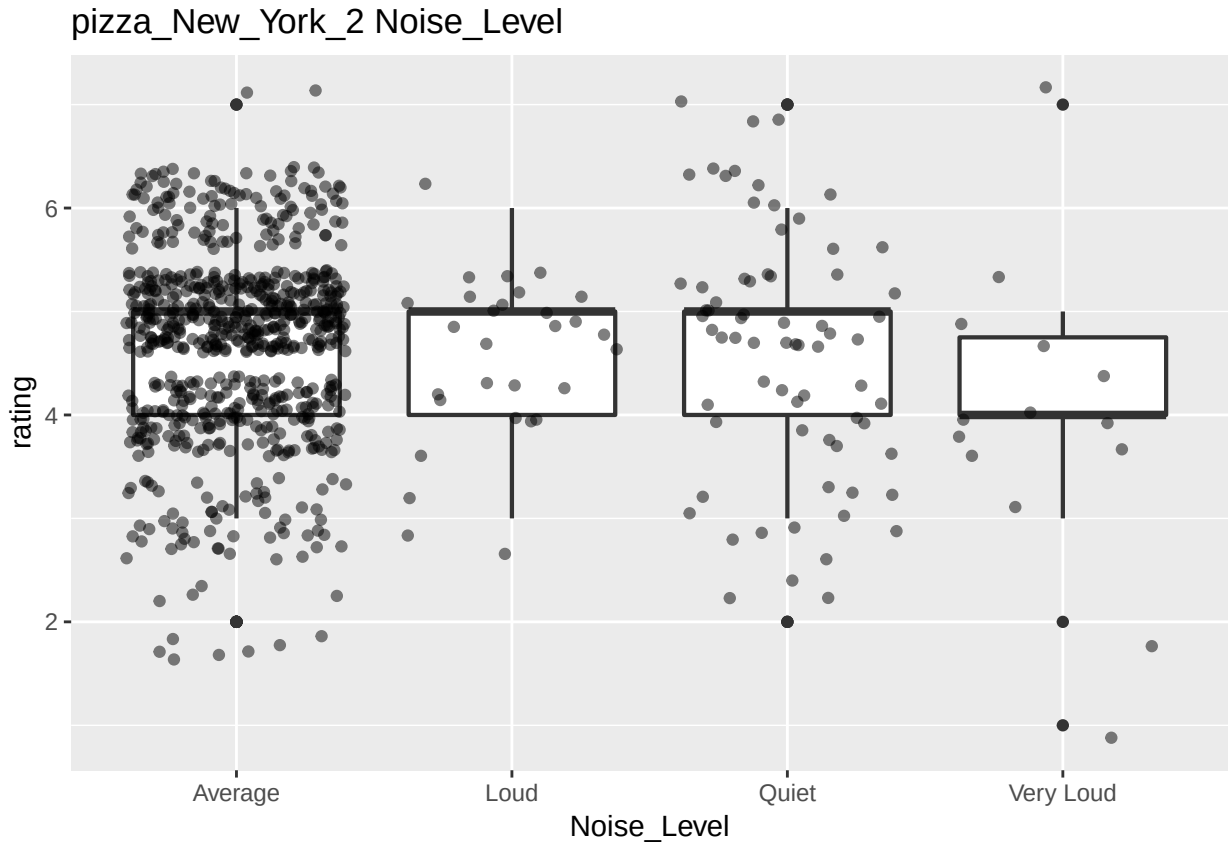

```
##
## [[13]]
```



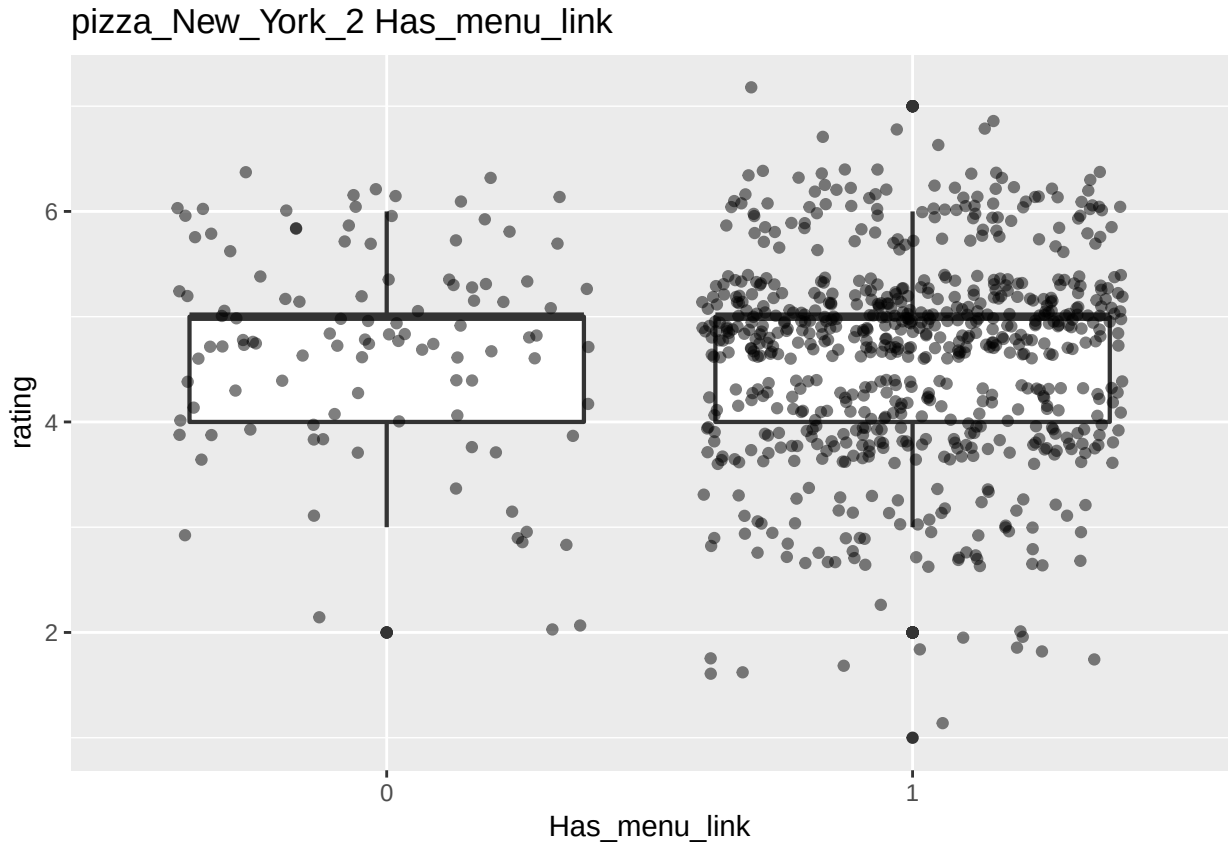

```
##
## [[14]]
```


pizza_New_York_2 Ambience
rating
2500
2000
1500
1000
500
0
Casual
Casual, Classy
Casual, Intimate
Casual, Intimate, Classy
Casual, Intimate, Romantic
Casual, Intimate, Romantic, Classy
Casual, Romantic
Casual, Romantic, Classy
Casual, Touristy
Casual, Touristy, Classy
Casual, Touristy, Trendy
Casual, Trendy
Casual, Trendy, Classy
Casual, Trendy, Intimate
Casual, Trendy, Intimate, Classy
Casual, Trendy, Intimate, Romantic, Classy
Classy
Divey
Divey, Casual
Hipster
Hipster, Casual
Hipster, Casual, Trendy
Hipster, Classy
Hipster, Trendy
Intimate
Intimate, Classy
Intimate, Romantic
Intimate, Romantic, Classy
Romantic
Romantic, Classy
Touristy, Classy
Trendy
Trendy, Classy
Trendy, Intimate, Romantic
Trendy, Intimate, Romantic, Classy
Ambience

```
##
## [[15]]
```

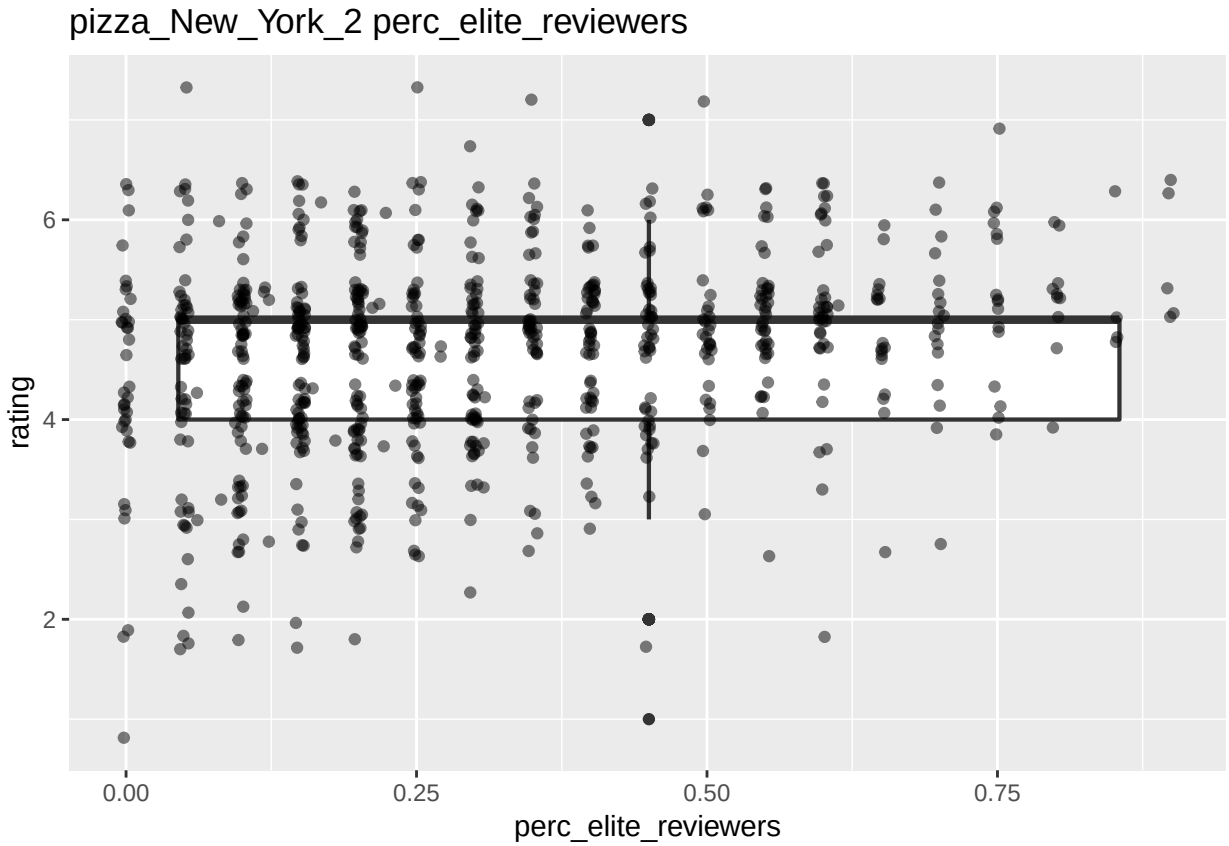

pizza_New_York_2 Noise_Level
rating
6
4
2
Average
Loud
Quiet
Very Loud
Noise_Level

```
##
## [[16]]
```



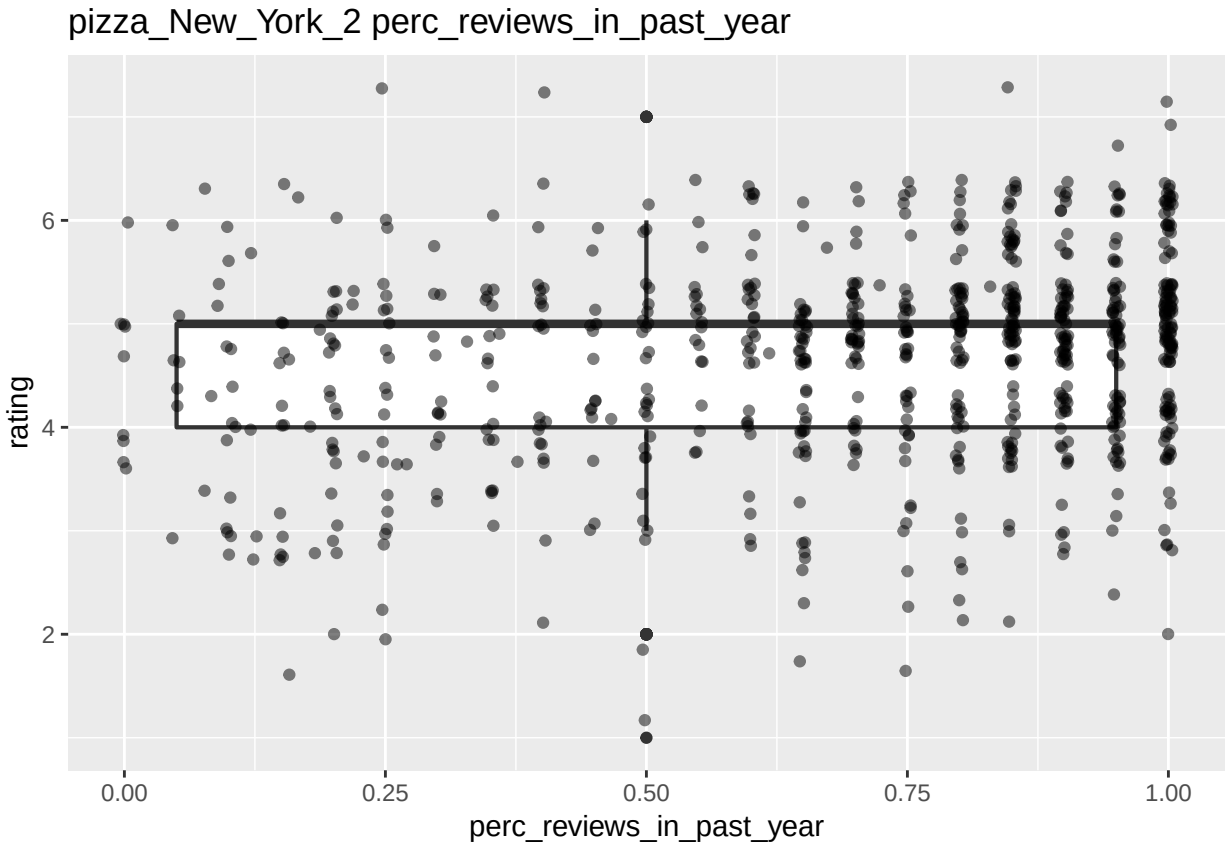

```
##
## [[17]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



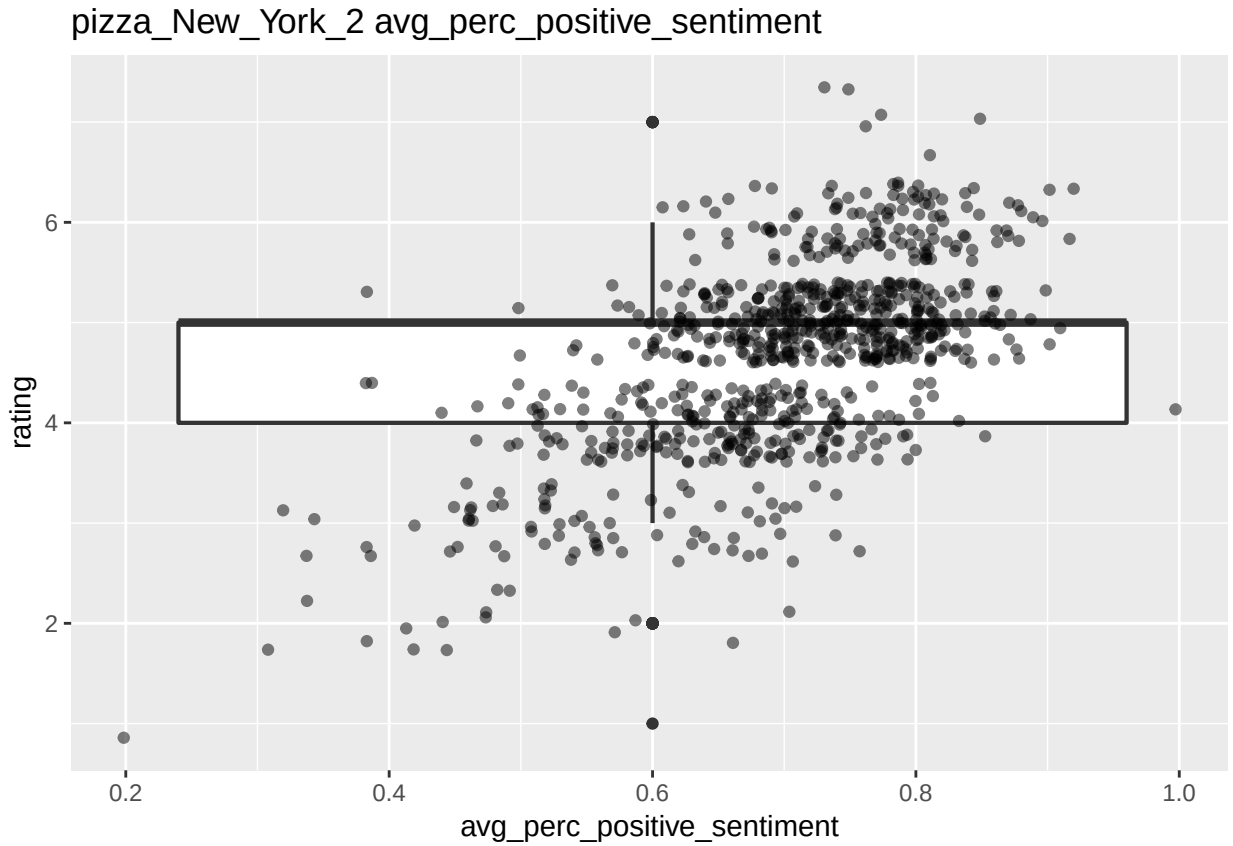

```
##
## [[18]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



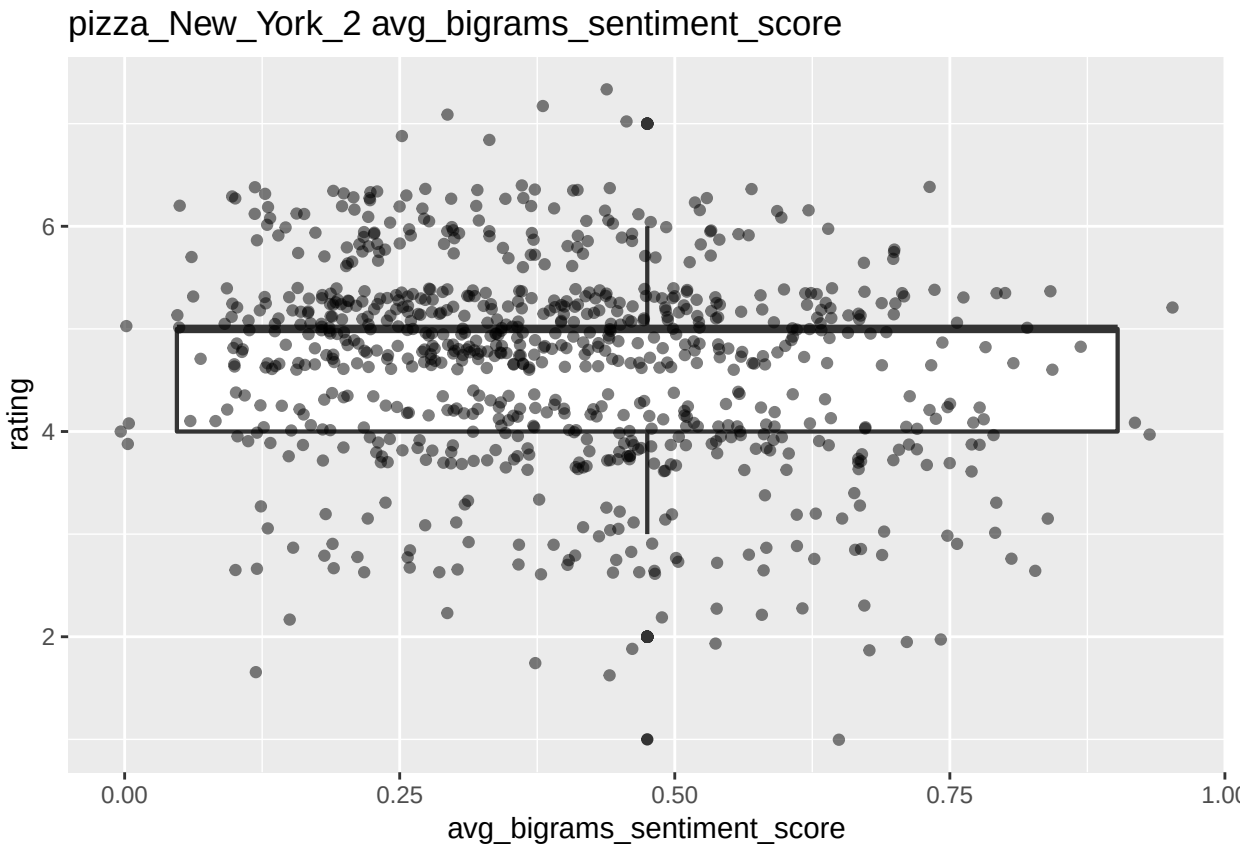

```
##
## [[19]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```




```
##
## [[20]]
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```


pizza_New_York_2 avg_bigrams_sentiment_score
rating
6
4
2
0.00
0.25
0.50
0.75
avg_bigrams_sentiment_score