What makes a 'good' Yelp rating?

Adam, Bharathrham, Mo

12/14/2019

Our Inspiration & Objective

Dining out is a hallmark of human culture. It is a hobby, recreation, and a vessel that fosters community amongst friends and family. In addition, the restaurant industry plays a major role in the US economy. This year, in 2019, there are over 1 million U.S. restaurants which combined are expected to produce over $863 billion in revenue. Over the next ten years, the restaurant industry is expected to produce over 1.6 million new jobs and employ 10% of the overall U.S. workforce. So what is associated with a good restaurant?

There is an obvious answer to this question. Good food is associated with a good restaurant. But there is more to a restaurant than the quality of its food. Restaurants differ in the cuisines they offer, the number of dietary restrictions they accommodates, and the amenities they offer. For the purposes of this project, we were interested in building an application that would allow a soon-to-be restaurant owner to predict the yelp score of their future restaurant. An interesting secondary use of our application would be for existing restaurant owners to tinker with inputs to determine what changes are likely to improve or hurt their yelp rating.

To build this application, the first step is to create a predictive model that uses restaurant attributes as input and predicts yelp scores.

Our Process

To build this predictive model, we utilized the largest restaurant review dataset: Yelp. The specific dataset we downloaded can be found here: https://www.dropbox.com/s/gd1k41y9gbpfwq3/yelp_academic_dataset_business.json?dl=0.

This dropbox file links to an older Yelp dataset challenge. For each data challenge, Yelp modifies what data is available to view. For our particular problem statement, we wanted as many attributes regarding restaurants as possible. Therefore we decided to use an older data challenge data set. *Before running our rmd file it is important to download the file from the link above and drag it into the project directory (when it asks you to replace file, hit okay). Because this file is too large for github to handle it will not be fully pulled into the relevant directory.*

```
yelp <- stream_in(file("yelp_academic_dataset_business.json"))
```

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

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

The dataset available to us is a nested JSON file which is a bit tricky to work with. Thankfully Kan Nishida provided wonderful, easy to follow instructions on how to access and clean up the Yelp dataset. We will link his blog post below: https://blog.exploratory.io/working-with-json-data-in-very-simple-way-ad7ebcc0bb89

```
#Flatten and reformat to make the data easier to manipulate
yelp_flat <- flatten(yelp)
yelp_tbl <- as_data_frame(yelp_flat)
yelp_modified <- yelp_tbl %>% mutate(categories = as.character(categories))
yelp_modified <- as_data_frame(yelp_modified)
```

After following Nishida's instructions, we still had a bit of data cleaning to do.

From this massive list of businesses, we needed to create a dataset containing only restaurants, their ratings, and variables describing their attributes. To ensure that star ratings are well-founded and trustworthy, we only looked at restaurants with many ratings (> 10) which are still currently in operation. For each restaurant, our dataset includes 122 variables describing attributes like ambience, cuisine, specialties, alcohol options, and a plethora of additional characteristics like whether it offers happy hour, takes reservations, is good for kids, etc.. Our work creating this dataset accounts for the majority of our time spent bringing this project to life. The following sections describe this lengthy process of wrangling, cleaning, and imputing data ahead of building a random forest model. We then use this model to analyze variable importance and generate predictions for user input in an interactive Shiny application.

Since we're only interested in restaurants and their attributes, we needed to remove irrelevant variables/cases that applied to other types of businesses. We filtered out any hair salons, night clubs, or other businesses that would contaminate our model and analysis. Next on our data wrangling crusade, we removed variables that didn't apply to restaurants leaving only those germane to our project.

```
# Filter data so we're only working with restaurants
yelp_modified <- yelp_modified %>% filter(is.atomic(yelp_modified$categories))
yelp_modified <- yelp_modified %>% filter(str_detect(categories, "Restaurant"))

#These are all variables we do not want to include in our final model
nixed_variables <- c("type", "latitude", "longitude",
    "neighborhoods",
    # We will have already filtered review count > 10, so this is unnecessary
    "review_count",
    # These are unique to each business, so unnecessary for model
    "name", "business_id",
    "full_address",
    "attributes.Smoking",
    "attributes.By Appointment Only",
    "attributes.Accepts Credit Cards",
    "attributes.Payment Types.amex",
    "attributes.Payment Types.mastercard",
    "attributes.Payment Types.visa",
    "attributes.Payment Types.discover",
    "attributes.Attire",
    "attributes.Ages Allowed",
    "attributes.Parking.street",
    "attributes.Wheelchair Accessible",
    "attributes.Good For Dancing",
    "attributes.Coat Check",
    "attributes.Waiter Service",
    "attributes.BYOB",
    "attributes.Corkage",
    "attributes.BYOB/Corkage",
    "attributes.Order at Counter",
    "attributes.Accepts Insurance",
    "city")

# Filter data and select only relevant columns
yelp_modified <- yelp_modified %>% filter(open == TRUE) %>%
  filter(review_count >= 10) %>%
  select(-starts_with("hours")) %>%
  #We have filtered for restaurants, so we don't want music related columns
```

```
  select(-starts_with("attributes.Music.")) %>%
  #We have filtered for restaurants, so we don't want hair related columns
  select(-starts_with("attributes.Hair")) %>%
  #This data will be used as a 'category' instead of an 'attribute'
  select(-starts_with("attributes.Dietary")) %>%
  select(-nixed_variables) %>%
  #We need review data for all restaurants
  filter(!is.na(stars))
```

The categories column has some very interesting data but is hard to access due to its format. Each element of the column is a vector of strings, and each string was a possible category, like Italian, Chinese, or Vegetarian. We wanted our model to split on whether or not a restaurant had these categories, which meant splitting on whether or not the string existed in the given vector.

A Random Forest would not automatically do this and even if it did, the output and variable importance would be very hard to interpret. Our solution was to add a boolean column for each possible category. Set the value to 'TRUE' if the category existed in the vector and 'FALSE' otherwise.

```
yelp_modified <- yelp_modified %>%
    mutate(american = (str_detect(categories, "American"))) %>%
    mutate(asian.fusion = (str_detect(categories, "Asian Fusion"))) %>%
    mutate(bagels = (str_detect(categories, "Bagels"))) %>%
    mutate(bakeries = (str_detect(categories, "Bakeries"))) %>%
    mutate(barbecue = (str_detect(categories, "Barbeque"))) %>%
    mutate(bars = (str_detect(categories, "Bars"))) %>%
    mutate(beer = (str_detect(categories, "Beer|Beer Bar|Beer Garden|Beer Gardens"))) %>%
    mutate(bistros = (str_detect(categories, "Bistros")))
    ... # many similar lines below
```

There were around 280 categories to start with, so we did not keep all categories. This was for two reasons. First, having too many columns increases computation time greatly, therefore we removed any variables that rarely occurred. For example, there was only 1 wok restaurant in our data set so keeping that variable would be useless. Second, a lot of the variables seemed irrelevant, such as "accepts insurance." Much like earlier, we didn't want our model to split on these variables as they would likely be random, rather than meaningful splits.

A balance was struck so we did not remove too many columns that our model would be worthless, but also did not keep too many that our model would take hours to build.

Another major issue we had was missing data. Trees and Random Forests do not handle NA values at all, so we had to decide what to do. Our two options were replacing all NA values with 'unknown', essentially allowing it as a new value that our random forest could split on, or imputing the data with some classification model. Imputing data is when you predict your missing values using all the available data in the dataset.

The idea of using 'unknown' as a value was not exciting, since it presented the problem that the tree could split on a given factor being unknown instead of being TRUE or FALSE. This would be problematic because then our model would be implying that leaving a data field blank on yelp can improve or hurt your rating. This is an unsubstantiated stretch and so we decided that we would use an imputation model as long as the imputation model did not hurt our test error rate.

After running two models, one with unknown and one imputed, we learned that the imputed models had a slightly lower Root Mean Squared Error than the unknown model (0.49 vs 0.52). We therefore went with the imputed model as our final model.

The imputation model we used was k Nearest-Neighbors. This itself was a struggle, as each package we tried had its own intricacies and problems that led to more errors and more headaches. We eventually settled on

the VIM package to impute data (link to the package documentation: https://www.rdocumentation.org/packages/VIM/versions/4.8.0/topics/kNN)

```
#value of k chosen arbitrarily, based on recommendations in the package documentation
yelp_imputed <- kNN(yelp_cleaned, k = 11)

#the VIM package's kNN imputation adds a new column that ends with _imp for every column
#This column simply says whether the data at that location was imputed
# or not (whether it was originally NA or not)
#This is not useful information for our model, so we remove it.
yelp_imputed <- yelp_imputed %>% select(-ends_with("_imp"))
```

Explaining our Model

One of the issues with Yelp's data is that the star ratings are always discrete. A restaurant can only have a rating in 0.5 increments, so 0.5, 1.0, 1.5, 2.0 up to 5.0. This is not realistic when it comes to averaging out several hundred user reviews (each of which can have values in 0.5 increments), which means that Yelp is processing these reviews and rounding them to the nearest 0.5 increment. This presents a problem, since a lot of values that would have been very different, i.e. 3.26 and 3.74 would both be collapsed down to 3.5.

When making our model, we had the option to treat the rating as a factor (can only take the prescribed discrete values) or as a numeric (can take on any decimal value). We felt it was more appropriate to treat it as a numeric, since that reflects the smaller changes that come from changing various factors. It also more closely reflects how Yelp would see the data before rounding it to the closest 0.5 increment, which is more interesting.

After tuning our model, we learned that our ideal mtry was 20 and minnode was 5. We used these values as is in our final model to avoid excess computation.

```
#Create test and training samples
samples <- createDataPartition(y = yelp_imputed$stars, p = .8, list = FALSE)

train_imp <- yelp_imputed[samples,]
test_imp <- yelp_imputed[-samples,]

#Both mtry and min.node.size were previously tuned. final, optimal values only
# used for the sake of compuatation time
yelp_forest <- train(stars ~ .,
                 data = train_imp,
                 method = "ranger",
                 trControl = trainControl(method="oob"),
                 num.trees = 1000,
                 importance = "permutation",
                 tuneGrid= data.frame(mtry = 20, splitrule = "variance", min.node.size = 5))
```

```
predictions <- predict(yelp_forest, test_imp)
postResample(pred = predictions, obs = test_imp$stars)
```

```
##      RMSE  Rsquared       MAE
## 0.5046990 0.3362043 0.3664466
```

Root Mean Squared Error = 0.504. Mean Absolute Error = 0.366. R-squared = 0.336

We can interpret the R-squared value as the amount of variation in the yelp scores thats explained in our model. We did significantly better than setting all predicted yelp scores to the sample mean, but this is definitely not an amazing model. This essentially means that over half of the variation in yelp scores is not explained by our model. This is expected, since our data doesn't include any information on food quality or service quality, which, in our personal experiences, influences how we review restaurants the most.

The Root Mean Squared Error and Mean Absolute Error both measure the same thing in different ways. Both show how different our predicted scores were from the true yelp score for that restaurant. Our average margin of error was about 0.75. This is good, because stars scale from 0.5 to 5.0, and our error was around 0.36, meaning that when rounded to the nearest 0.5, our model would likely capture the truth anyways.

The Shiny App

Displaying the relationships between restaurant attributes and their varying effects on star rating in an interesting way was our project's central goal. Therefore, we built an interactive Shiny App where users can design their own restaurant and see what predicted star rating it would receive in real life. Every variable used in building the random forest is available for customization by the user, including all restaurant amenities and descriptors of style, mood, and cuisine. A panel in the bottom-right region of the window displays our prediction of the user's restaurant's average star rating, which updates each time a box is un/checked or an attribute changed. In playing with the application ourselves, we've found that users can design restaurants to receive ratings in the region between 2.8 to 4.1. Since the overwhelming majority of restaurants in our training dataset had either 3.0, 3.5, or 4.0 star ratings, our model always predicts values in this interval.

To use the shiny app yourself: (1) clone our GitHub repository, (2) open the directory yelp-forest in RMarkdown, and (3) open the app.R file inside. (4) Press the "Run App" button, wait a minute for the data preprocessing and model building, then a window should pop open and you're good to go! Note: make sure the file "fuoy.png" is in the yelp-forest directory and "yelp_academic_dataset_business.json" is in it's parent directory. Both of these can be found on the project GitHub and may need to be explicitly downloaded and moved.

Trying to Better Understand our Model by Analyzing Variable Importance

Our top ten most important variables were the following:

1. Drive Thru
2. Price Range
3. Alcohol Full
4. Dogs Allowed
5. Cates
6. Has TV (After this point, variables have comparatively negligible impact and are far more difficult to analyze)
7. Good for lunch
8. Takes Reservations
9. Alcohol none
10. Casual Ambiance

```r
# Finding the importance of varables in terms of the random forest.
yelp_importance <- sort(importance(yelp_forest$finalModel), decreasing = TRUE)
head(yelp_importance, 6)
```

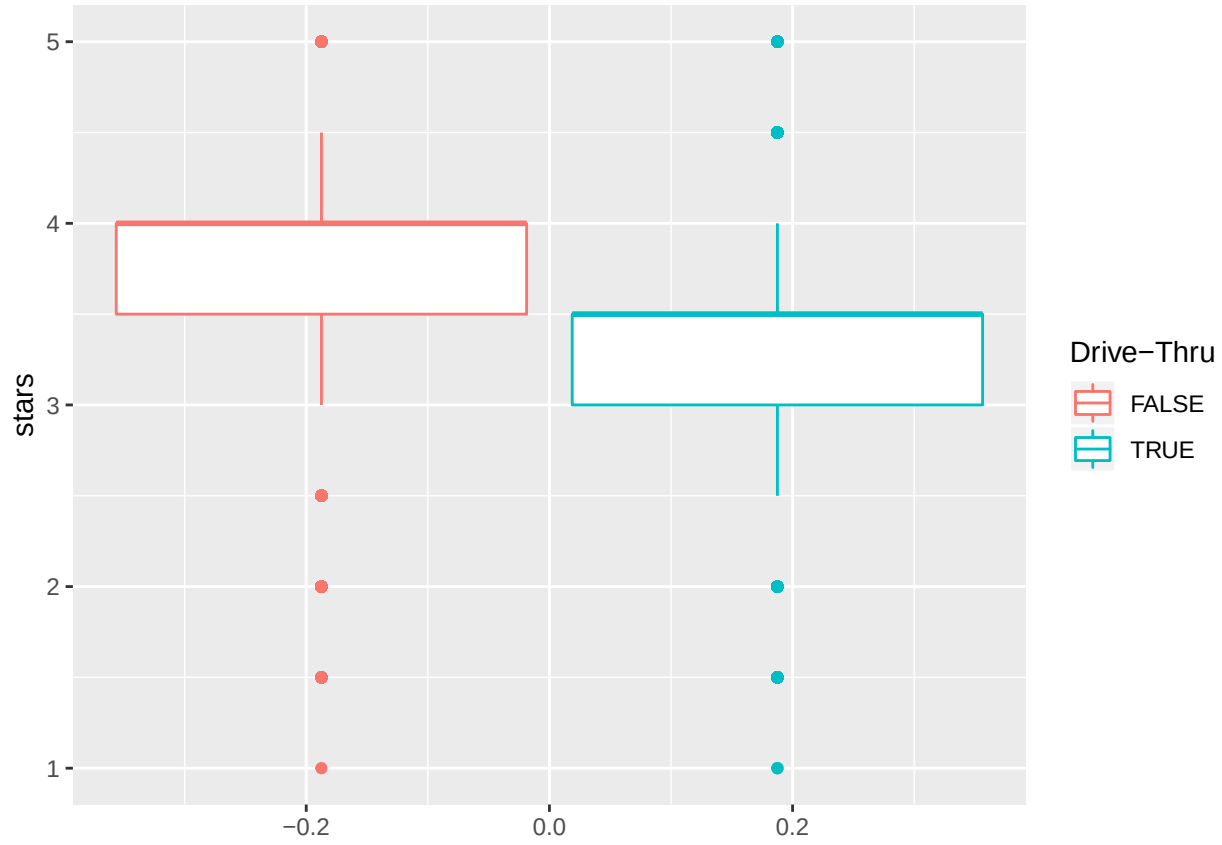
```
##   `Drive-Thru`TRUE      `Price Range`    Alcoholfull_bar `Dogs Allowed`TRUE
##         0.06840658         0.04738930         0.03418821         0.02711939
##         CatersTRUE       `Has TV`TRUE
##         0.02543481         0.02092735
```

Variable importance only tells a fraction of the story. While we know which variables are important, we do not know how these variables impact restaurant ratings. Do they improve or hurt overall restaurant score and can we extrapolate or infer any meaning from this? To find out we created boxplots for each variable. We shall analyze the results below.

Drive Through (Having Drive Correlated with Lower Ratings)

Our box plot shows that having drive through is associated with a lower rating. This is an interesting result as you would expect drive through to be a convenient amenity. However drive throughs are typically associated with cheap, low quality, fast-food joints. Because of this restaurants without a drive through option might have better quality food or other desirable amenities and therefore it makes sense that they have a superior star rating

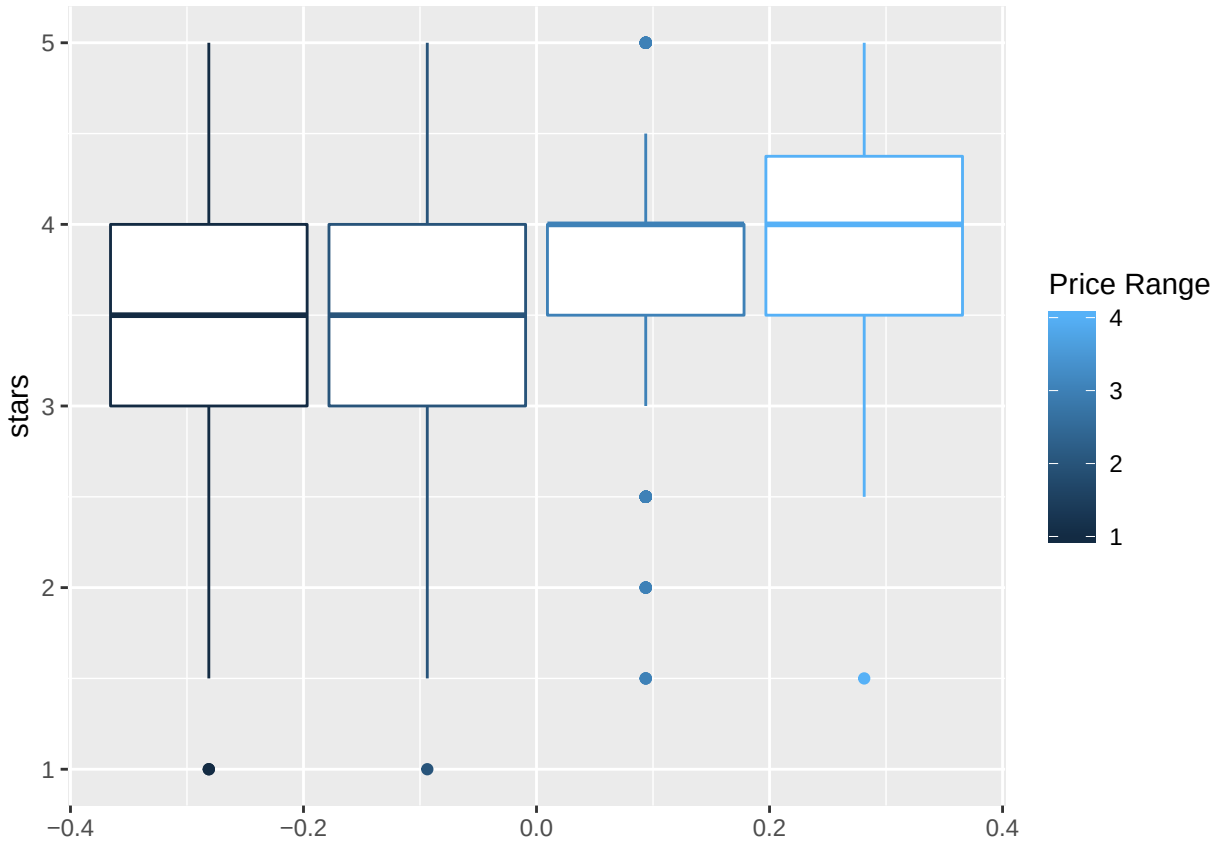
```
# Creating Boxplots Comparing Drive Through Variable
ggplot() + geom_boxplot(data = yelp_imputed, aes(group =`Drive-Thru`, y = stars,
                                                 color = `Drive-Thru`))
```



Price Range (Higher Price Correlated with Higher Ratings)

This makes a lot of sense. Higher priced restaurants tend to be better in quality. They likely have better chefs and better wait staff. Therefore higher priced restaurants will tend to have higher average ratings.

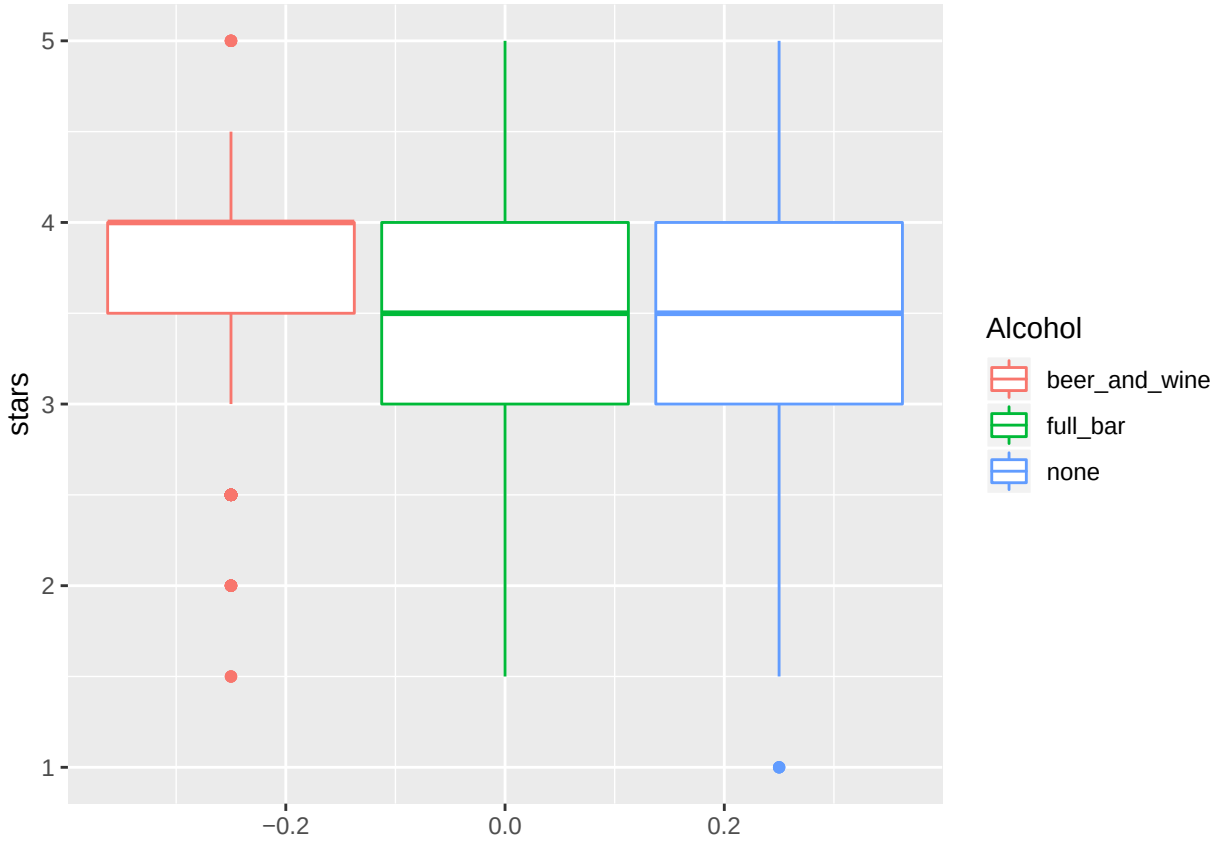
```
# Creating Boxplots Comparing Price Range Variable
ggplot() + geom_boxplot(data = yelp_imputed, aes(group =`Price Range`, y = stars,
                                                 color = `Price Range`))
```



Alcohol (Beer & Wine Correlated with Higher Ratings)

Full Bars and No Bars have identical box plots. However, restaurants that serve beer and wine are correlated with higher ratings. This could be for similar reasons as the fast food and price variables. Restaurants that serve wine could have a higher price range and will likely be "higher quality" restaurants. Therefore it is unsurprising that these three variables are all correlated with higher ratings.

```
# Creating Boxplots Comparing Alcohol Variable
ggplot() + geom_boxplot(data = yelp_imputed, aes(group =`Alcohol`, y = stars,
                                                 color = `Alcohol`))
```

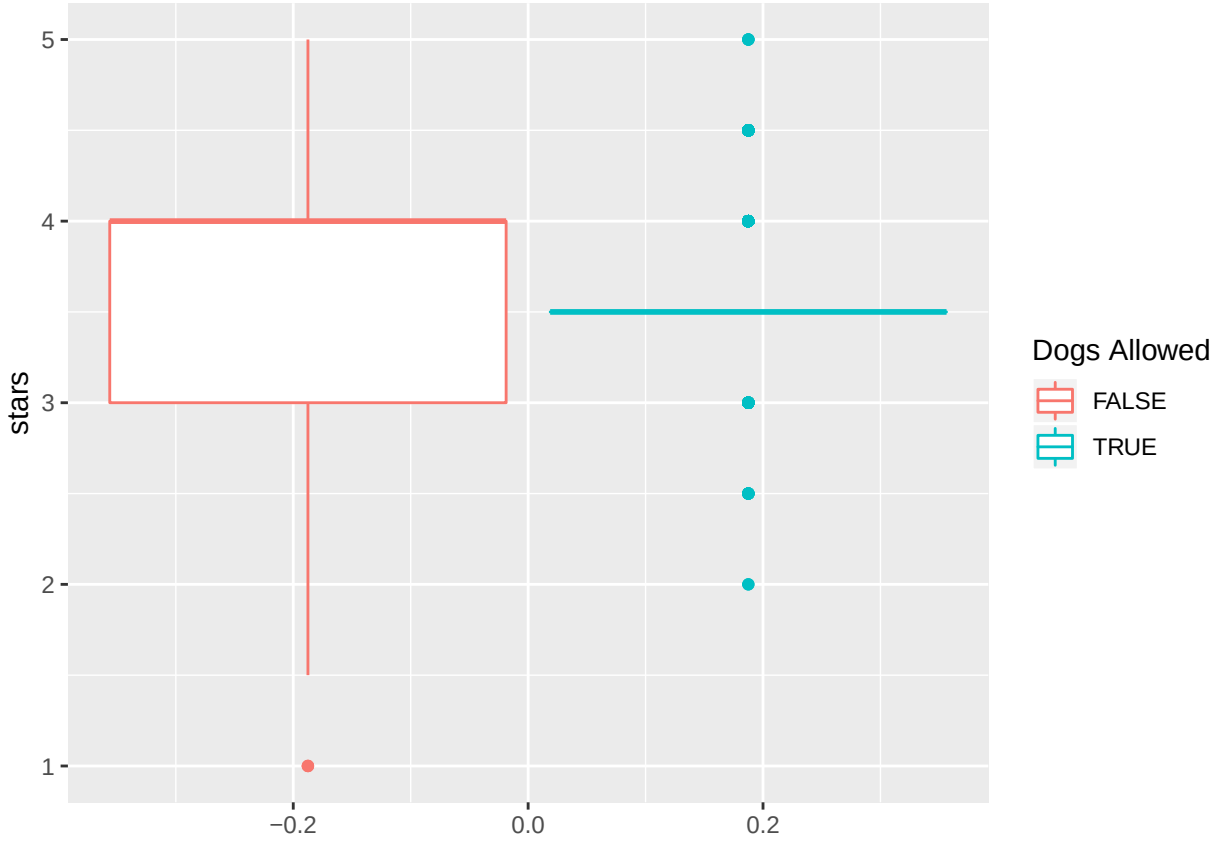


Dogs Allowed (Allowing Dogs Correlated with Lower Ratings)

The box plot for Dogs Allowed == TRUE looks strange. The 25th percentile, 75th percentile, and median are basically the exact same point. At first, we suspected that there were rather few data points where Dogs Allowed == TRUE causing the graph to look this way. However, upon further inspection, there are over 2,000 data points in which Dogs Allowed == TRUE. This confirms that allowing dogs is correlated with lower overall ratings, but more consistent overall ratings. Perhaps many individuals consider pets to be unhygienic and therefore do not wish to eat around them, thus hurting the ratings. Another explanation would be that the Random Forest split on Dogs Allowed due to its consistent ratings being really efficient at reducing impurity of nodes.

Another interesting observation is that when filtering by Dog Friendly restaurants around Claremont, a majority of our results had ratings of 4 and higher. According to the box and whisker plot, those ratings would be an outlier. This could be due to two reasons. Maybe Claremont has a more pet friendly culture than the states we analyzed, or maybe Claremont simply has better restaurants in general.

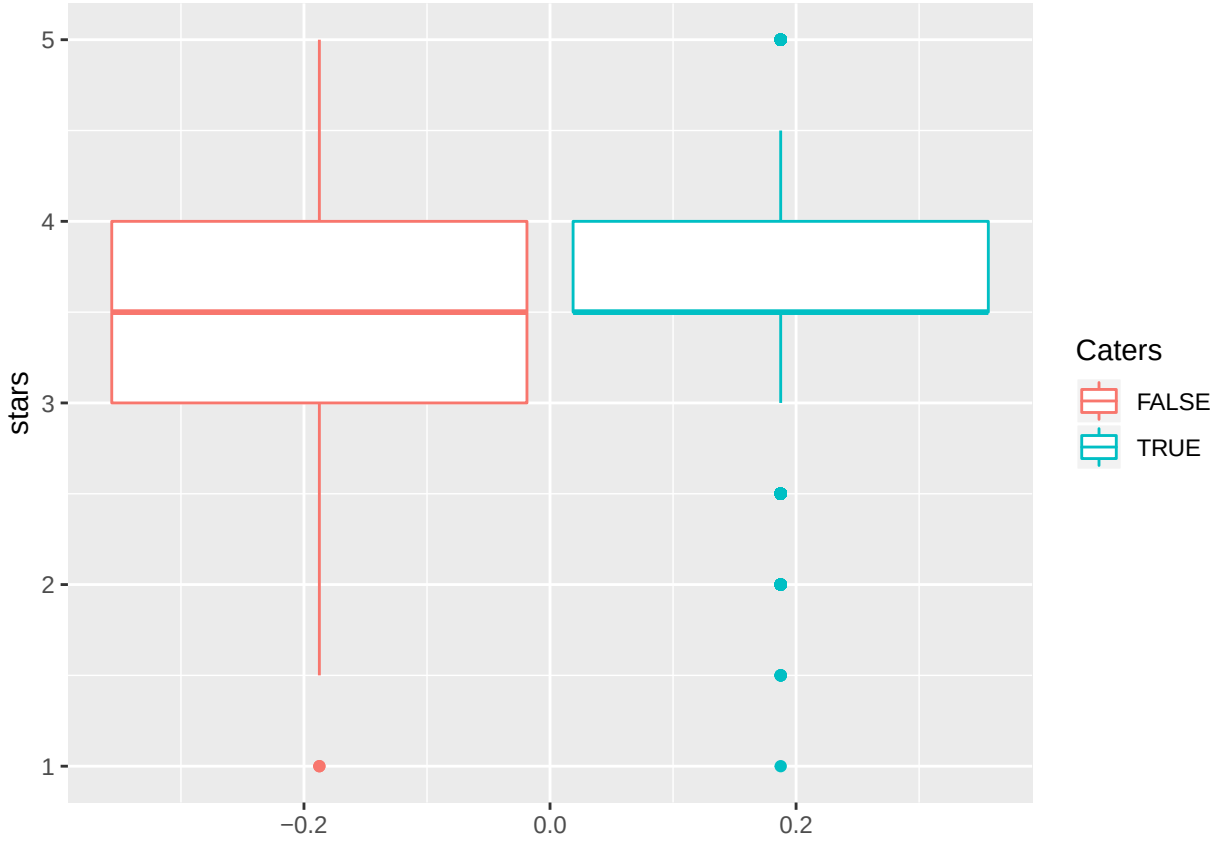
```
# Creating Boxplots Comparing Dogs Allowed Variable
ggplot() + geom_boxplot(data = yelp_imputed, aes(group =`Dogs Allowed`, y = stars,
                                                 color = `Dogs Allowed`))
```



Caters (Having Catering Increases the lower bound and Decreases the Upper Bound of Ratings)

The median and 75th percentiles for having and not having catering are roughly the same. However, the 25th percentile and lower bound for having catering is higher than not having catering. Meanwhile, the upper bound for having catering is lower than for not having catering. On average, having catering makes your restaurant's Yelp score less variable, which could make it a good split for our model.

```
# Creating Boxplots Comparing Caters Variable
ggplot() + geom_boxplot(data = yelp_imputed, aes(group =`Caters`, y = stars,
                                                 color = `Caters`))
```

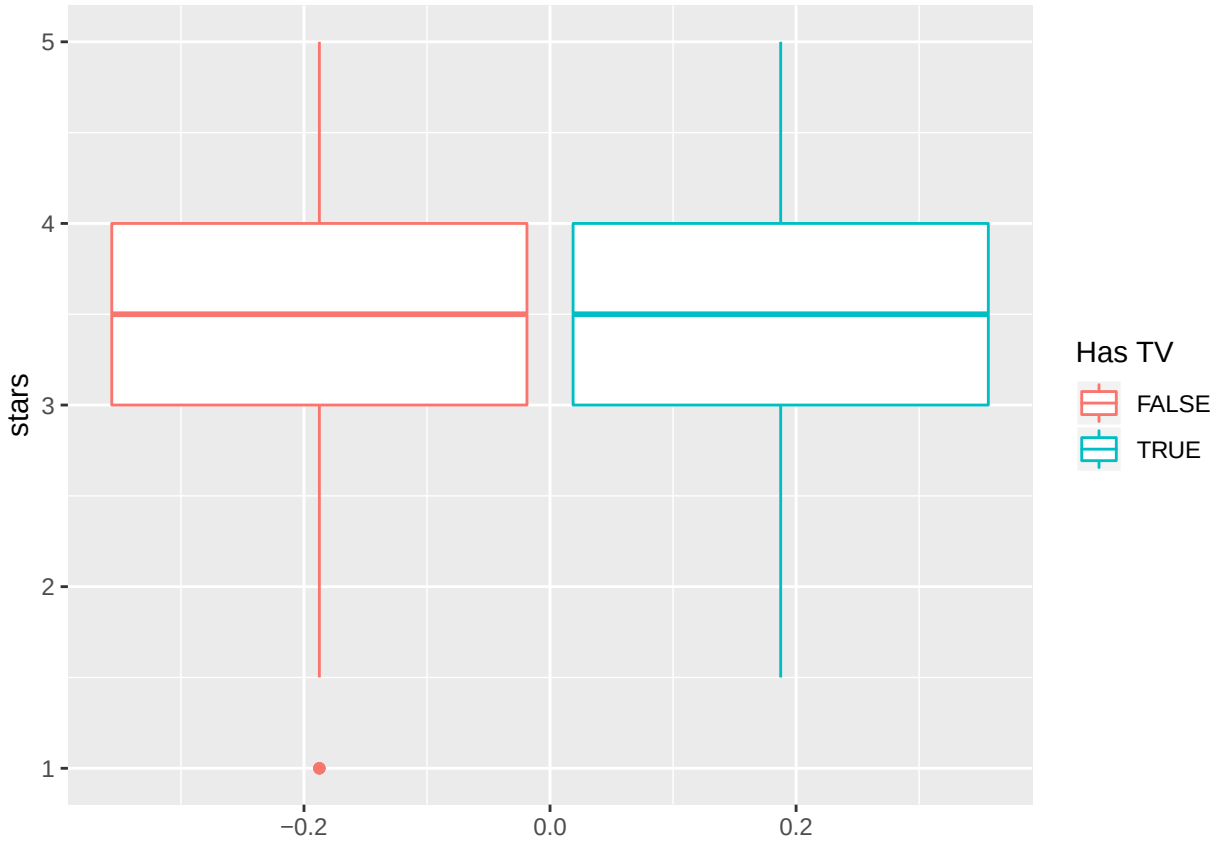


Has TV (Negligible Correlation)

The boxplots for restaurants with and without TVs look identical. There is negligible correlation for having a TV and overall restaurant rating. This could be due to the way our Yelp dataset is rounding to increments of 0.5. There might be a correlation between having a TV and higher/lower ratings but if there is one it is masked by Yelp's rounding of the data.

From this variable and onwards, because the variable importance is low and the box plots start to look similar, our random forest trees are probably using these variables as subsplits on existing splits... ie if a tree is already split by "Price Range" perhaps "Has TV" can distinguish between higher/lower ratings amongst restaurants within the same price range. Because of this it is difficult to analyze the implications of the variables from "Has TV" onwards as it would require us to closely analyze the trees in our random forests which is a task we did not have time to engage in for the purposes of this project (our primary goal was to build the shiny app referenced in the previous section). In addition, the next most important variable after "Has TV" is ~30% less important than "Has TV" and so we will end our variable analysis here.

```
# Creating Boxplots Comparing Has TV Variable
ggplot() + geom_boxplot(data = yelp_imputed, aes(group =`Has TV`, y = stars,
                                                 color = `Has TV`))
```



Limitations

While informative and interesting, our project's model and resulting analysis are slightly tainted by the dataset we used, and we must explicitly note these drawbacks to avoid misleading readers. Our dataset is large, comprised of more than 11,000 restaurants, but it's lacking in geographic diversity. We worked with data points from thirteen states (7 in America, 2 in Canada, 2 in the UK, and 2 in germany) and around 160 distinct cities between those states. Some regions had significantly more observations than others (Rhineland-Palatinate contained only two). We cannot, in good faith, extend our conclusions about variable effects and relationships to broader swaths of these countries. Instead, readers should consider our results as applicable only to the states for which we have restaurant data.

Since Yelp only provided this dataset for academic projects and their Yelp Dataset Challenge, it is not intended to include all restaurant data, even in the states and cities for which we have lots of data. Ideally, this dataset would've been a random sample from Yelp's full database, but we cannot know how Yelp chose the data and therefore cannot know the implicit biases with its selection. To combat this, we have to be careful in how we present our conclusions.

With this in mind, our population would be 'Restaurants in PA, NC, SC, WI, IL, AZ, NV, QC, ON, EDH, MLN, BW and RP' and our conclusions would not generalize well to other populations. WIth more robust data and computational power, we could create a far more accurate, trustworthy, and interesting model. But this is the world we live in, and we made do with the best available dataset and tools. We considered partitioning our data by region and analyzing each separately. It's reasonable to think a restaurant in Mexico City would need different factors and amenities than a restaurant in Budapest. In the end, we ruled against this idea because we didn't have enough data points for each region and learned that our RF rarely split on location anyways.

Our initial question was "'what attributes outside of food quality contribute to a restaurant's Yelp rating?" Our model, while useful in determining what factors drive a positive Yelp rating, cannot incorporate all the mixed effects to answer that question. Having a TV in a bar or a pub is usually a good thing, but it doesn't mix well with having a 'classy' ambience. Using a different statistical learning technique (perhaps Bayesian Mixed Effects) could've been better for answering this question but would've required knowledge and expertise beyond our own. However, we still wanted to communicate the complexity of these relationships to people interested in our project, so we built an interactive Shiny application where users can tweak over 100 variables and look at the nuanced, sometimes unexpected ways they affect restaurant rating.

Ethical Considerations

Yelp has, in the past, been accused of artificially lowering restaurants scores if they do not pay Yelp advertising fees. We are not here to argue whether this is true or still occurs. If true, then this policy could lead to lower restaurant scores in low-income neighborhoods where businesses do not have the money or influence to combat Yelp's score deflation.

Another consideration is the subjectivity of some of our model choices. In an ideal world, we would be able to run our model on all the data and use all available factors, but this is simply not feasible with the computation power available to us. We tried to trim variables based on relevance to restaurants as well as popularity (there were only two Curry Sausage restaurants, so the factor was discarded for our model) but our implicit biases definitely played a part in variable choice and the final model.

What we learned

During the course of this project, we gained exposure to several valuable tools and skills not taught in this class. In our earliest attempts to collect relevant restaurant data, we established a Yelp App account and scraped data from the Yelp API. We quickly realized that this approach didn't give us enough data points for each restaurant. After poking around on the internet, we found the yelp academic challenge JSON file

that would become the centerpiece of our project. To our dismay, this file wasn't a conventional JSON but a nested JSON where each business entry appears as its own separate JSON element. Reading this data into an R dataframe proved incredibly difficult until we found a tutorial that walked us through the steps needed to deconstruct a nested JSON and access its valuable data. After cleaning and wrangling the data using methods taught in class, we needed to find some way to replace NA values before training our random forest model. Outside research and a conversation with professor harden led us to the idea of using KNN to impute missing values from similar observations with valid entries. Finding a good R package for our particular use case was difficult and time consuming, but we eventually stumbled upon "VIM" which perfectly accomplished our goals and was fairly easy to use. After model building, we decided to take our project above and beyond by developing an interactive Shiny application. We used R's online tutorials, youtube videos, and stack exchange forums to learn how to develop a clean UI and effective backend server

Our team particularly enjoyed this project because it gave us the freedom to investigate and use new statistical methods and software tools that we were excited about. It was both encouraging and motivating to see how easily one can independently acquire new skills like Shiny app development. We all plan to use these new skilled acquired during this project as we move forward into our next classes, outside projects, and professional careers.

Citations

Alexander Kowarik, Matthias Templ (2016). Imputation with the R Package VIM. Journal of Statistical Software, 74(7), 1-16. doi:10.18637/jss.v074.i07

Grace-Martin, K. (2018, May 9). Assessing the Fit of Regression Models. Retrieved December 17, 2019, from https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/.

Hadley Wickham (2019). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.0. https://CRAN.R-project.org/package=stringr

Hadley Wickham (2019). httr: Tools for Working with URLs and HTTP. R package version 1.4.1. https://CRAN.R-project.org/package=httr

Jeroen Ooms (2019). curl: A Modern and Flexible Web Client for R. R package version 4.2. https://CRAN.R-project.org/package=curl

Jeroen Ooms (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. arXiv:1403.2805 [stat.CO] URL https://arxiv.org/abs/1403.2805.

Kirill Müller and Hadley Wickham (2019). tibble: Simple Data Frames. R package version 2.1.3. https://CRAN.R-project.org/package=tibble

Marvin N. Wright, Andreas Ziegler (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, 77(1), 1-17. doi:10.18637/jss.v077.i01

Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2019). caret: Classification and Regression Training. R package version 6.0-84. https://CRAN.R-project.org/package=caret

National Statistics. (n.d.). Retrieved from https://restaurant.org/research/restaurant-statistics/restaurant-industry-facts-at-a-glance.

Nishida, K. (2017, December 25). Working with JSON data in very simple way. Retrieved from https://blog.exploratory.io/working-with-json-data-in-very-simple-way-ad7ebcc0bb89.

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Yelp Data Challenge Dataset. (2016).