Report: ECON498 Machine Learning I Final
Carly Bouts

**Data Loading and Cleaning**

Skimming business_sample.json, I noticed that the data for restaurants and the data for non-restaurant businesses looks very different. Restaurant data includes price ranges and information about, for example, ambiance and cuisine, while non-restaurant data does not include this information. Additionally and obviously, non-restaurant data has a broader range of business categories (education, automotive, professional services, shopping, etc.) available than restaurant data. I decided that it would be best to account for these differences in forming my machines because many variables such as "price range" and "ambience" presumably have strong predictive value for restaurants but no predictive value for non-restaurant businesses. For this reason, I first created two separate runme.py files: runme_restaurants.py and runme_nonrestaurants.py.

In the first sections of both of these files, I tell the computer to read business_sample.json (or whichever json file the program is currently reading) and gather information about businesses' locations, business hours, attributes, and categories. Location variables include dummy variables for the three countries represented in the data, dummy variables for the 12 states most frequently represented in the data, and dummy variables for the 15 cities most frequently represented in the data. Business hours data includes data for each day of the week (hours, opening time, closing time, total hours open) as well as business/week-level data (the most frequent closing time for a given business, the most frequent opening time for a given business, and a dummy variable reflecting whether or not the business has weekend hours).

Business attributes for both types of business include information about parking options and whether or not the business accepts credit cards. Business attributes in runme_restaurants include information such as restaurant ambiance, noise level, and take out/delivery options. Attributes specific to non-restaurant data include whether or not the business accepts insurance and whether or not the business is run on an appointment-only basis. Business category information for restaurants includes dummy variables for various cuisines, while the same information for non-restaurant businesses includes dummy variables indicating business sector (shopping, automotive, etc.).

In cleaning my data, I took the necessary steps to prepare it for use in my random forest and linear models. This meant converting all boolean variables to integers (1 for "true" and 0 for "false"), converting other "object" data types to integers or floats, and manipulating times in the "hours" section of my code using timedeltas and converting the times to seconds. After correcting these data types and creating my dataframe, I dropped the lines that included "None" values. I also dropped the lines that contained "None" as a string. This allowed me to use my dataframe to train machines fairly seamlessly.

**My Model**

Because of the above-discussed differences in the information available for restaurants vs. non-restaurant businesses, I made two random forest models: one to predict stars of restaurants, and one to do the same for non-restaurant businesses. These models were trained in the runme files using the relevant data for restaurants and for non-restaurant businesses, respectively.

The random forest classifiers (and linear regressions) I use are supervised learning models. This means that they were trained and then tested on some data for which we already knew the true outcome (data_test was used to make predictions, which were then compared with target_test, the true outcomes). Because we had good information about target_test, it made the most sense to make use of this helpful information by using a supervised learning model.

My random forest models predict only whole-star review scores. I made it this way by restricting the training and test data to only those businesses which have whole-star scores. While this can be considered a limitation of my models (as discussed below in the "limitations" section), I consider it to be a good feature of the models because it allows for increased accuracy at the cost of some precision which may or may not have been meaningful in the first place (again, this is discussed below in the "limitations" section).

I calibrated the models using a subset of the training data (N = 25,000). I found the optimal parameters ((n_estimators=250, max_depth=23) for restaurants and (n_estimators=800, max_depth=17) for nonrestaurants) through processes of trial and error which are documented in my runme files. For example, in the process of finding the optimal specifications for restaurants, I tried the following combinations of parameters (plus many more which gave me lower accuracy scores) (I did this process for non-restaurant businesses separately):

- n_estimators=200 and max_depth=25 gave an accuracy score of .68
- n_estimators=500 and max_depth=25 gave an accuracy score of .67
- n_estimators=500 and max_depth=15 gave an accuracy score of .66
- n_estimators=200 and max_depth=35 gave an accuracy score of .717
- n_estimators=200 and max_depth=40 gave an accuracy score of .673
- n_estimators=500 and max_depth=35 gave an accuracy score of .68
- n_estimators=200 and max_depth=35 gave an accuracy score of .714
- n_estimators=300 and max_depth=35 gave an accuracy score of .706
- n_estimators=250 and max_depth=35 gave an accuracy score of .722
- n_estimators=250 and max_depth=36 gave an accuracy score of .68
- n_estimators=250 and max_depth=37 gave an accuracy score of .7
- n_estimators=300 and max_depth=37 gave an accuracy score of .68
- n_estimators=300 and max_depth=36 gave an accuracy score of .71
- n_estimators=1000 and max_depth=36 gave an accuracy score of .69
- n_estimators=260 and max_depth=36 gave an accuracy score of .69
- n_estimators=250 and max_depth=36 gave an accuracy score of .703
- n_estimators=250 and max_depth=23 gave an accuracy score of .708
- n_estimators=1250 and max_depth=23 gave an accuracy score of .716
- n_estimators=650 and max_depth=25 gave an accuracy score of .711
- n_estimators=250 and max_depth=28 gave an accuracy score of .666
- n_estimators=250 and max_depth=26 gave an accuracy score of .708
- n_estimators=250 and max_depth=24 gave an accuracy score of .738
- n_estimators=300 and max_depth=24 gave an accuracy score of .71

- n_estimators=250 and max_depth=23 gave an accuracy score of .740
- n_estimators=600 and max_depth=25 gave an accuracy score of .71

*Random Forest: Restaurants*

The random forest trained in runme_restaurants.py ultimately gave me an accuracy score of .741.
The confusion matrix for the random forest trained in runme_restaurants.py was as shown in the terminal output here:

```
Confusion matrix from restaurant random forest:
[[  0   1   0   0   0]
 [  1  10  12   2   0]
 [  0   3  64  44   0]
 [  0   0  24 206   0]
 [  0   0   0  11   0]]
Accuracy score from restaurant random forest:
0.7407407407407407
............
r2 score from restaurant linear model
0.3250948738064846
```

There are several important things to notice in this terminal output. For now, we ignore the last two lines in the output. First, we interpret the accuracy score. It means that roughly 74% of the model's predictions of target_test (the review scores for the test portion of our N=25,000 training sample) were correct. This is pretty good- nearly ¾ of our predictions about restaurants are correct. Now, we look at the confusion matrix. This is meaningful because the indexes assigned to the rows and columns are meaningful- the row and the column indexed as "0" represent actual or predicted review scores equal to "1", those indexed as "1" represent actual or predicted review scores equal to "2", and so on. We look at the confusion matrix on a row-by-row basis since the rows represent the actual accuracy scores. For instance, in the row indexed 0, we see that only one review score in the test data is actually equal to 1. As evidenced by the fact that (0,0) = 0, the machine does not guess this correctly. In fact, the machine only predicts that reviews scores will be equal to "1" one time, and it does so incorrectly (it predicts "1" when the actual score equals "2"). As shown by the row indexed as "1", review score is actually equal to 2 25 times, and the machine guesses 10 of these correctly. This seems like an indication that the model might tend to overestimate the review scores of businesses with actual review scores equal to 2. However, moving onto the third and fourth rows (those indexed 3 and 4), we realize that our confusion matrix is actually pretty good. Of 341 actual scores equal to 3 or 4, the model predicts 270 correctly. Each time the actual score is 5, the model predicts that the score is equal to 4. This is certainly not ideal, but it's not that bad. After all, a final indication that the confusion matrix is pretty good is that when the model predicts incorrectly, it almost always misses the correct review score by only one star. There are only two instances in the confusion matrix where this is not the case. For the restaurant random forest model, then, we have a good accuracy score and a good confusion matrix.

*Random Forest: Nonrestaurant Businesses*

I calibrated the random forest model in runme_nonrestaurants.py in the same way that I calibrated the model in runme_restaurants.py. The random forest trained in runme_nonrestaurants.py ultimately gave me an accuracy score of .556.

The confusion matrix for the random forest trained in runme_nonrestaurants.py was as shown in the terminal output here:

```
Confusion matrix from nonrestaurant random forest:
[[  0   3   3   4   2]
 [  0  17  24  29  14]
 [  0   7  86  94  30]
 [  0   4  48 246  63]
 [  0   0  13  88 185]]
Accuracy score from nonrestaurant random forest:
0.55625
..............
r2 score from nonrestaurant linear model:
0.271661183818823
```

We now inspect this terminal output, ignoring the last two lines in the output for now. First, we interpret the accuracy score. It means that roughly 56% of the model's predictions of target_test (the review scores for the test portion of our N=25,000 training sample) were correct. This is decent because over half of the model's predictions about nonrestaurant businesses are correct. Still, it is markedly lower than the accuracy score for restaurants. I discuss this limitation later in the report (in "Addressing the limitations of my models"), concluding that it is not too concerning because the information we are given for non-restaurant businesses is not as informative as that given for restaurants. We now move to the confusion matrix. As argued earlier, this is meaningful because the indexes assigned to the rows and columns are meaningful. We once again look at the confusion matrix on a row-by-row basis. In the row indexed "0", we see that 12 review scores in the test data are equal to 1. However, the machine never predicts that any review score will be equal to 1. In the row indexed "1", we learn that the target_test (actual review scores) is equal to 2 84 times, 17 of which the machine guesses correctly. From the row indexed "2," we learn that of the 217 times that target_test is equal to 3, the machine guesses correctly 86 times. The machine guesses correctly most of the time when target_test is equal to 4 (it guesses correctly 246/361 times) and when target_test is equal to 5 (guessing correctly 185/286 times). This is a redeeming feature of our confusion matrix. Another somewhat redeeming feature of the confusion matrix is that incorrect predictions are concentrated very roughly around the correct review scores (data_test). However, we cannot say of this model that when it predicts incorrectly, it almost always misses the correct review score by only one star. Incorrect predictions sometimes miss the actual review score by two or three stars. Therefore, while our accuracy score and confusion matrix are alright for the nonrestaurant random forest model, they indicate that this model does not perform as well as the restaurant random forest model.

**Comparison to Linear Model**

To show that my random forest models are good, I also constructed and trained linear models for comparison. As shown by the terminal outputs above, the R2 scores are .325 for the restaurant linear model and .271 for the nonrestauarant linear model. R2 is a measure of how much of the variation in the target is predicted by the variables (data) included in the model. Therefore, high R2 means that the linear model is working well for predictions, while a low R2 means it is not. Since only ¼ to ⅓ of the variation in review scores is predictable on the basis of the variables in our data, these R2 values are unsatisfactory. Still, it makes sense that R2 would be so low for these models. While the linear model includes all of the variables that we have in our data, it does nothing to account for interactions between these variables. While we could technically improve R2 by adding many interactions between these variables and

squaring/cubing some variables, this would take arduous trial and error and would probably not give us results as good as those given by our random forest models. We are therefore unsatisfied with the linear models and are confident that our random forest models, which can be said to account very indirectly for interactions between variables in data, work better. This holds true even for the non-restaurant businesses which are predicted with an accuracy score of only .556 by the non-restaurant random forest.

**Addressing the limitations of my models**

There are some limitations to our models- most notably that they only predict whole-star review scores, meaning that they are incapable of predicting that a business has a review score of, say, 2.5 or 3.5. This is warranted because it allows the models to predict well between, for example, review scores 3 and 4 at the cost of losing the ability to predict poorly between, for example, review scores 2.5, 3, and 3.5.. However, this means that our predictions for future businesses will only take whole-star values. Still, this is redeemable. I see two ways in which this is still worth the loss in precision. First, a person who has a generally good experience at a business might rate it 4 if they're in a regular mood, a 3.5 if they're in a bad mood, or a 4.5 if they're in a good mood. Similarly, natural variations in temperament (generosity, optimism, etc) between people predispose some to rate a given business 4, while others are predisposed to rate the same business slightly higher or slightly lower than this. For these reasons, a .5 star difference in review scores need not be clearly meaningful. The second justification is more practical. I assume that future users of the model are probably people who want to know how well a future business would do given information about its category, attributes, opening hours, location, etc. Perhaps they're trying to make decisions about their own businesses or to plan future personal investments of money or labor. Presumably, these users would rather get a rough estimate of review score with relatively high accuracy rather than an exact estimate with very low accuracy. The predicted review score should give users a good indication of how well the business is likely to do based on the information in "data," but it doesn't need to be so exact because it's only one of many considerations that wil inform their eventual decisions.

The second limitation of my model is that predictions are not as good for non-restaurant businesses. This is likely because the information that Yelp.com lists for restaurants is a lot more detailed and specific than that given for non-restaurant businesses. To begin with, the "categories" for restaurants reflect different types of restaurants (for example, fast food, Italian, Chinese, Latin American), while the "categories" for non-restaurant businesses communicate business sectors (for example, medical, shopping, beauty and spas). Intuitively, we learn more about a restaurant from the knowledge that it serves fast food than we learn about a business from the knowledge that its main function is "shopping." Additionally, Yelp data about restaurants include other helpful attributes such as noise level, "good for X meal", and ambience. Meanwhile, Yelp data about non-restaurant businesses do not include comparable attributes. The result is that we have better information for restaurants than we do for non-restaurant businesses. For this reason, it is understandable that the accuracy score for the nonrestaurant random forest model should be significantly lower than that of the restaurant random forest model.

Finally, it is also worth noting that my models were trained using a subset of the training dataset (N = 25,000) due to computational power limitations. It is likely that if they were trained with the entire training dataset, accuracy scores would be higher because the models could benefit from gathering more information from a larger training sample.