Nastasia Pollas

DATA 71200

For this class, I chose a Kaggle dataset, called Canada Pizza Price Prediction. I know typically with classes such as these, students would opt for data that reflects their capstone. Truth be told, I have no idea what my capstone will be on, despite it rapidly approaching, and I did not want to attempt such a new course with complex data. I searched in Kaggle for machine learning or supervised learning and found this set. The dataset contained 129 rows and 9 columns, and the aim was to predict the pizza company based on the other variables, such as diameter in inches, price in CAD, toppings, variant and size. The diameter and price were the two numeric variables, and the rest were categorical.

Fortunately, the author created a relatively clean dataset; there was no missing data or anything that required a conversion, aside from correcting the spelling of two or three things. However, after trying to create dummy variables, I noticed during supervised learning that the feature extractions of extra cheese, extra sauce, and extra mushrooms did not separate cleanly. Those three variables had the same responses: yes/no. So as dummies, I was unable to distinguish them. After consulting with the professor, I recoded the yes/no so that they were easily distinguishable as dummies (yes_ec/no_ec, yes_es/no_es, & yes_em/no_em).

## Project 1

The goal of project one was to load the data and visualize it. I plotted the two numeric fields, price_CAD & diameter_in as histograms with varying bin sizes. Both variables were slightly right skewed. Using a scatterplot to examine further, I saw that there was an overall positive correlation. As price increased, so did the diameter. The correlation matrix provided further evidence that there was a strong positive correlation between price and diameter (n=0.814). The transformations of the data using squared and cubed shows charts with an even greater right skew.

However, there were mistakes made in the first project that possibly carried over into subsequent work. First, I had trouble initially running the training/testing split. After it finally worked, I was able to run the transformations. The problem was that I apparently ran them on the full dataset instead of the training set. However, trying to rectify that has proven to be very difficult and filled with errors. Based on the professor's feedback, the high correlation between the diameter_in and price_cad meant that I should have dropped one of those features before further analysis. Also, I should have converted the categorical labels of the target variable into numbers. But if I did that, I knew I would have been confused on how to proceed with the target variable. It seems that from the lectures and data camp that the target variable is supposed to be a single variable instead of several variables. I did rerun it and created dummies for the company variable, but then I became confused on the next steps. Perhaps I had to store the multiple variables as a list to pass through further codes.

**Project 2**

For the supervised learning project, after loading it, creating dummy variables, and splitting it into training and testing sets, I first created a bar chart to visualize the target variable. The chart showed that companies C and E were tied as the most frequent in the training set, with approximately 22; A came in third place with about 18, B was next with approximately 16.

D came in last with 15. The two learning algorithms I used were k-Nearest Neighbors and Random Forest. K-Nearest Neighbors is the simplest algorithm that labels a data point based on the class of its nearest data point. If you adjust the k (the number of neighbors) to greater than 1, then it votes on the most common label amongst the neighbors.

The rank scores showed that best performing k's were 8 and 9, and the worst performing was 23. K=9 had a stronger prediction than k=8, with a training set score accuracy of 0.55 and testing set score of 0.48 This indicated that for k=9, the model could correctly predicted the class for 55% of samples in training set and 48% of the samples in the testing set. For k=23, the model was 39% accurate for predicting training samples and 30% accurate for predicting test samples.

Next, I adjusted the k parameter for best performing to k=4 and got accuracy scores of 61% for the training set and 42% for the testing set. I adjusted the k parameter for worst performing to k=1 and got a score of 83% for the training set and 33% for the testing set. Since this dataset had few features and they were not sparse, k-Nearest Neighbors would be a reliable algorithm to use, especially considering how relatively easy it is to understand.

A random forest operates on the notion that while one decision tree might predict well and overfit the data, we can reduce overall overfitting by averaging the amount of a group of trees that overfit differently. With the default random forest, that was set to n_estimators=5, the score was 0.812 on the training set and 0.455 on the testing set, meaning that the model could accurately predict the class for 81% of the samples in the training set, and 45% of the sample in the testing set. I adjusted the parameters to n_estimators=15, 8, 27, and 3. The results showed that overall, as the number of estimators increased the accuracy of the training set increased but that of the testing set decreased. The accuracy scores were the same for n_estimators= 15 and 27—84.4% on the training set and 45.5% on the testing set. For n_estimators = 8, the accuracy on training set was 82.3% but only 39.4% for the testing set. For n_estimator = 3, the accuracy on training set was 78.1% and 51.5% for testing.

I found it interesting that in the feature importance chart extra mushrooms were deemed most important (value=0.10) for the decision a tree made. In second place was diameter_in. This was surprising because extra mushrooms did not seem to be something "distinctive" enough to hold such weight; plenty of companies would offer extra mushrooms—all you do is charge the customer extra—but the sizes can vary according to company, with different companies having the ability to create "unique" size pizzas.

Reflecting on supervised learning, I can see that it is a bit difficult to understand what makes something a great model. Is it a high score on the training set or high score on testing set? I assume ideally it would be high scores on both, but the k-nearest neighbors and Random Forests show trade-offs in model efficiency when adjusting the parameters. For example, k=1 is ranked 20th on the rank test score, but it has one of the highest training set scores at 0.83, albeit with a very mediocre testing set score of 0.33. But the best neighbors have scores of 0.55 and 0.48. Is a model considered good if it can only predict an estimated 50% of samples from a dataset?

**Project 3**

For unsupervised learning, I am not certain if the entire Principal Component Analysis (PCA) feature selection worked because the feature scaling code returned an error. Moreover, I had a challenging time interpreting the many PCA charts based on the class code. The principal components portion was especially confusing.

PCA is an algorithm that rotates the data around the axes, creating statistically uncorrelated features and without losing much information. This technique's goal is dimension reduction, which helps make predictions. PCA assumes that high variance features are more useful, and low variance ones are noise to be discarded. For unscaled data, the accuracies were 0.42 and 0.30 on the training set and the testing set, respectively. It is important to scale the data before using StandardScaler from sklearn. preprocessing. For scaled data, the accuracy sharply declined for both datasets—0.29 on training set, and 0.24 testing set. Compared to supervised learning, the scores for the unscaled data performed similarly to the worst performing k-nearest neighbors (n=23). The scaled datasets performed even worse than either the k-nearest neighbors or the random forest.

**Reflection**

I came into this class with not a shred of knowledge about machine learning algorithms and left with a less than rudimentary understanding. When I took an intro to Python course last year, the professor said that the basics (lists, functions) were easy but advancing one's knowledge in Python is very difficult; she was NOT kidding. My head was spinning trying to decipher the class codes. This was also the first time that I used Google Colab, as the intro to Python utilized Jupyter Notebook in Anaconda. So, it felt like I was learning about or how to use several different processes and concepts in a very short time span. However, the predictive text and Gemini AI were a huge help. Personally, I have been slow in accepting AI help for coding, but when the class code was not working, Gemini AI helped me pushed through.

The book and data camp exercises helped clarify the higher-level concepts, but I still struggled mightily in this course. Normally, class was noticeably quiet, so sometimes I felt like the only person on the outside looking in, completely outside of my element. What really hindered me was the lack of visualizations along with my general confusion for the visuals that were available. It is one thing to say that this model has an accuracy test score, but what does that look like exactly?

If I had more time, I would have strictly adhered to the data-camp course schedule instead of doing them all at the end; that was a huge mistake and disservice to my comprehension. This was perhaps the most difficult course I have taken in the CUNY program, and it is most definitely a course that would have benefited greatly from a full semester or at least had I already taken the course preceding this (taking it this fall). Hopefully, the course next semester will help explain these concepts more slowly. Ultimately, I never felt like I found what made a good model; it all seemed like a delicate balance played between the training and testing datasets. This class was quite the battle, but I am always grateful for any opportunity to learn new ways to build my data analysis skills.