Sarah Kahl
Final Reflection

A 5–8 page paper describing the work you did in projects 1–3 (your dataset and your supervised and unsupervised experiments). The paper should describe both what you did technically and what you learned from the relative performance of the machine learning approaches you applied to your dataset. This assignment should be posted as a PDF in your GitHub repository.

1) Describe your dataset, why you chose it, and what you are trying to predict with it

Coming to this class, I thought I knew what to expect. As a student in the Data Analysis and Visualization master's program, I really wanted to gain some technical skills in data science. I took "Working with Data" with Tim Shortell last spring, and was amazed with what I could do with python. I have a liberal arts background from Oberlin, and the main software package I used for analytics was SPSS. At this program, I want to master more data science skills where I can use a plethora of coding languages – which could help bring me to my goal of finding my career path. As of now, I do feel a little lost finding the exact type of career path to choose. However, I knew that this class was an integral part of figuring out what that next step could be.

I chose a dataset from Kaggle, a site that has many open datasets available about any topic. One potential path I am thinking about is Human Resources. I saw that IBM had a fake dataset they made for others to practice with. It was about employee attrition and retention. Since this could be a project I might do for a boss in the future, I thought this was the most practical dataset to use. I saw that even though there were more rows than what I have previously worked with, I liked how it had a mix of categorical and numerical columns. What I was predicting is whether the person stayed with the company or not (Attrition), which was a clear "y" to me. I figured that I would try to model whether an employee stayed at a company or not given many factors that were used to predict.

2) Detail what you did to clean your data and any changes in the data representation that you applied. Discuss any challenges that arose during this process.

Cleaning data in Python was definitely a new skill I had to acquire for this class. As I mentioned, I'm used to working with much smaller datasets in psychology. Now that I am trying to enter the data science world, I thought learning how to properly clean a dataset in Python is something I can use for the rest of my career. I noticed I had a lot of redundant variables, as well as variables that I was not interested in analyzing (such as whether they invested in stock or not). I decided to drop some of the categorical variables I thought were redundant, and only dropped two non-continuously variables, which both seemed quite similar to "Job Satisfaction". I did this by using the .drop() function in Pandas.

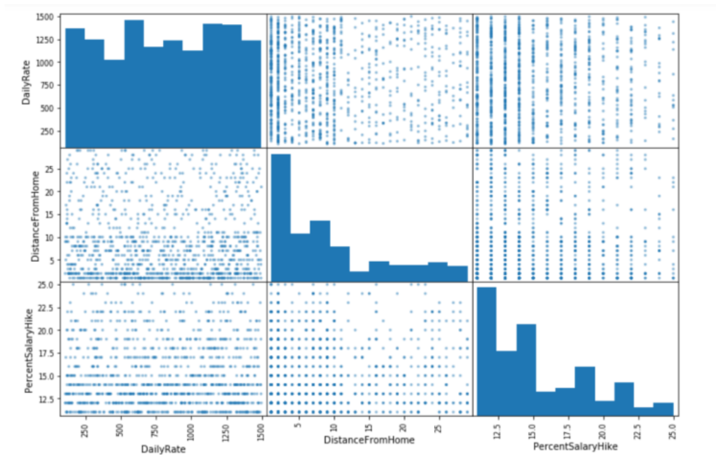
As I kept moving through the data cleaning process, I ended up with a huge problem using dummy variables. I wanted to use this method since it made more sense, had a clear explanation in the book, and I had some practice using dummies in data camp. Statistically, it also made the most sense to me. The get_dummies function can create dummy variables. It adds the new columns as a Data Frame, which includes NaN for the aspect of the variable that is not included with that data point (row value). Gender is a popular example. Instead of having the column be called “Gender”, with a categorical output, we create a new DataFrame which includes the amount of options for that categorical variable had as columns. For Gender in this one dataset, we would take the name of “Gender” and create a new DataFrame with the columns “Gender_Female” and “Gender_Male”. However, I ran into some problems with these dummies. When I tried to run functions, even after using “concat” to combine these two Data Frames, nothing seemed to work. Perhaps it was the fact that I had NaN’s in my dataset after doing this, but I struggled to even get an output for .describe and .head. I thought I did everything possible to make this go away, such as dropping the Gender variable from the DataFrame after they were concatted. Perhaps I would have been more successful if I changed the NaN’s to 0’s in order to perform these functions, however, having integers stand in for your categorical variables can sometimes cause problems if there is a hierarchy. Nonetheless, I decided to go with One Hot Encoder for these variables.

Alongside with Gender, I had “Job Role”, “Marital Status”, and “Department” as the other categorical variables. I used the One Hot Encoder and the Label Encoder so I could get a matrix output once these variables were created. What the Label Encoder does is convert categorical labels to numbers, which is exactly what I needed and not dissimilar from the dummy method (yes, in practice, but not theoretically). I also think this was the better decision because I didn’t need to combine two Data Frames, such as with get dummies. With One Hot Encoder, it adds convert integer categorical values into one-hot vectors, so instead of there being a hierarchical rank to the new label encoded data, they now can exist as numeric categories. My first step was to use the one hot encoder. Then, I used the label encoder to piece them all together. I had a few different columns to account for the options within my categorical variables.

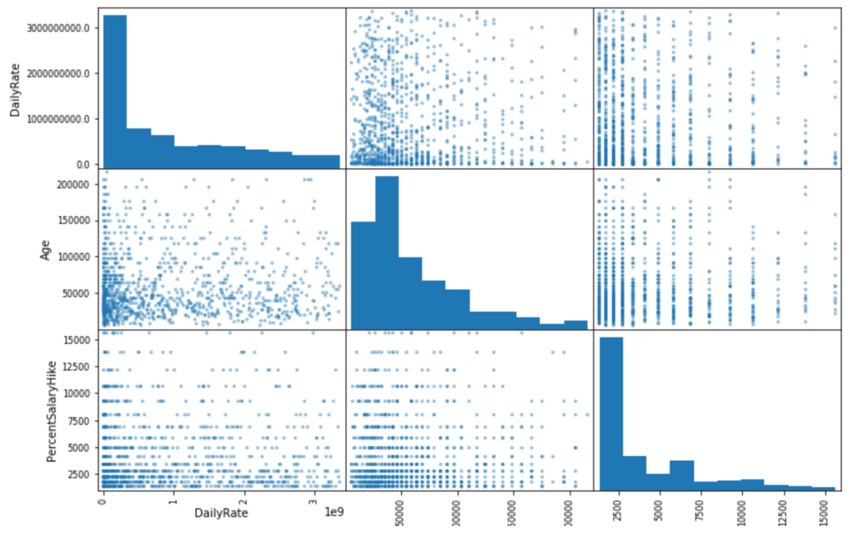
I also used the Ordinal Encoder to clean my target variable, which was attrition. This was also my y variable. Since the options were Yes and No, I thought it would make the most sense to turn them into zeros and ones, with Zero being “No” and One being “Yes”. Ordinal is meant more for hierarchical data, however it worked well in this case. I want to also note that I had to split my data again after cleaning the dataset.

3) Discuss what you learned by visualizing your data

While I had some trouble visualizing my data at first, I did learn some important things about this dataset. I ran multiple scatter plot matrices to see if anything was correlated. At first, I thought I was doing something wrong! I wasn’t getting a graphs that looked like they had any type of pattern. See below where I used Daily Rate (how much they were paid daily), Distance from Home, and Percent Salary Hike since I thought they might be correlated.



As you can see from these graphs, and other similar ones that I completed for project 1, nothing was correlated at all. I kept getting completely straight lines. This tells me that either my data (which is all fake from IBM) is really not related in anyway, or something might be wrong in my data. I wanted to take a closer look and decided to use "Age" as one of the variables in the scatter matrix.



As you can see here, there is something happening with age and daily rate. While they are not extremely correlated, there does seem to be some kind of relation between the two. I also think that the "fake" people that represent the data can't be too far off from reality. More of the older people at IBM probably make more money since they might have more experience or have been at the company for longer. For this project, that meant to watch out for age and daily rate when running my models, and they might both be good variables to predict the target variable of attrition. This was the only real relation I found through my scatter matrices.

4) Describe your experiments with the two supervised learning algorithms you chose. This should include a brief description, in your own words, of what the algorithms do and the parameters that you adjusted. You should also report the relative performance of the

algorithms on predicting your target attribute, reflecting on the reasons for any differences in performance between models and parameter settings.

I first used K Nearest Neighbors. This is a machine learning algorithm that uses the near-by points in clustering to essentially cluster your data together in a way that could make predictions. K Nearest Neighbors only considers exactly one nearest neighbor, which is the closest training data point to the point we want to make a prediction for. First, I made a line graph (on a scale from 1-10) to see what the best neighbor number would be. After running this, it looked like I was best off with 10 neighbors total (that's where the test and train data meet on the graph for a second time). The first time these lines touched together was at 8 neighbors, but then it crossed back over. Due to this crossing-back over, I thought 10 neighbors was going to be the best option. I then ran the algorithm with 4 neighbors, 8 neighbors and 10 neighbors. It looked like the highest accuracy was 86% for 4 neighbors. However, since the elbow graph showed that the testing and training data are different, we can assume that this will either overfit or underfit the data. I continued on with 10 neighbors.

Then, I ran a grid search to see what other parameters needed tuning. I found that 20 neighbors were actually the best decision (there are a lot of data points, so I'm not surprised I needed more neighbors than I thought). I also found that a Manhattan distribution would work best, and well as uniform weights. This did, however, make my accuracy even lower, down to 83.9% accuracy. Once I ran the statistics to see how well the model worked, I found that my F-1 score for the "Yes" option (meaning the person stayed at the company) was actually high, with a score of .91. However, my F-1 score for the "No" was extremely low, coming in a .03 for 10 neighbors. It was a flat zero for the other neighbors. I am going to predict that the model can work well for determining who will stay at the company, but not be able to predict who will leave.

The second attempt I made was using Support Vector Mechanisms, or SVM's. SVM's are a very robust way of clustering data in the supervised methods. As Wikipedia describes it, "SVM maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall." This definition is definitely the one that sticks in my mind the most. It is a clustering algorithm, like KNN's, but uses the vectors to make sure that the space between the categories is as wide as possible. This can make it a better predictor for any new data that is added.

I wanted to run a Grid Search again to determine my best parameters for the SVM. I used the rbf kernel since that was the one that worked best with my data, and the poly SVM did not work. This ended up taking a lot of work, and I decided to switch around the log space (C_range = np.logspace(-2, 10, 2) gamma_range = np.logspace(-9, 3, 2)) in the grid for both the c range and gamma. Having two folds worked better than having 125 fits, I just needed to make sure that the n_split should be smaller to have less folds (since I had way too many to run this), and I ended up splitting it twice for the stratified split. I then found my C and gamma for the SVM. When I ran the SVM, I got an accuracy of 84%, similar to my KNN's. As I will discuss later in this paper, I couldn't really escape that number of 84% throughout this project. Similar to my KNN with 2 or 4 neighbors, I had .91 for the F-1 score for the "Yes's", and zero for the "No's". This led me to believe that I was better off with the KNN algorithm for training my data.

5) Describe your experiments using PCA for feature selection, discussing whether it improved any of your results with your best performing supervised learning algorithm.

Long story short: It did not. I did run my PCA, first in class. I needed to turn my data frame for X_train into an array with NumPy. Once I did this, my PCA ran smoothly. I also made a heat map that showed me two variables with a different color, which I interpreted as two variables that were principal components in different ways. I didn't think much of it at the time, but I did get an error that stated "'numpy.ndarray' object has no attribute 'columns', thinking that had to do with the variable X.columns and wasn't a truly meaningful error in representing the data. I will discuss below how my PCA ended up with one variable accounting for 95% of the variance, which I think might have something to do with that error. After, I used the K nearest neighbors' algorithm, with 10 neighbors since that was the amount where there was a score that wasn't zero for the "No's". I also wanted to see the line graph again to see how many neighbors I needed now that the PCA was run. However, the graph looked identical to the one from before. I then ran the algorithm and got a result of .838, which was a slightly (so marginal it is insignificant) lower number than before.

I will talk about how running the PCA with 95% of variance explained below only result in one variable (accounting for 95% of the variance), however for the purposes of this question, I did not realize that this was the outcome of my PCA. Therefore, I don't know if my data in general was clean enough to have an impact, or if there was another problem with the data that resulted in just one variable accounting for the variance. Perhaps using Z-scores or another type of standardization could help with this. I think I would need to pick two or more components and re-do the PCA to get a better idea if it could improve my K Nearest Neighbors. However, this could be completely useless if something with my data is wrong, or if it is worth it given that the other variable that would be chosen would account for such little variance.

6) Discuss the results of using PCA as a pre-processing step for clustering. This should include a brief description, in your own words, of what the algorithms do. If you used the Wine dataset for this, briefly explain why your original dataset wasn't appropriate.

This turned out to be a little bit of a nightmare! No matter what I was doing, I couldn't get anything to run after I used the PCA as a pre-processing step for clustering with unsupervised methods. I kept tweaking my parameters (such as slicing where the X_train started) but could not get a graph for any of the unsupervised methods. I noticed that the only way it worked was when I sliced both X_trains at zero, meaning it was just comparing that one array to itself, which is completely useless. These unsupervised methods worked out before I used the PCA, so clearly something was up.

I did use the PCA with n_components = .95, meaning I could use the variables that accounted for 95% of the variance, which can make the algorithm stronger by not using excess variables that are not accounting for much. When I looked at my X2_train (post PCA), there was only one array. That means that 95% of the variance is contained by one predictor variable in X. This was a huge finding! I ended up re-running the PCA with 2 components instead, so I would have more than one variable. This made the graphs work! I used an X3 so I could still keep the work I did with my X2 – this way I could keep track of why I had to make the decision using 2 components.

The K means is a clustering algorithm that can help find different clusters in the dataset. I thought this one would work well for my categorical y, which was attrition. Agglomerative clustering seems to work backwards comparatively; each observation starts with its own cluster. Pairs of clusters continue to merge until all data points have been successful grouped into a large cluster – not leaving any behind. These two methods worked the best, given their ARI scores. DBSCAN (density-based spatial clustering of applications) was the one that I think worked the poorest for me. It essentially, based on distance, groups together points that are close to each other. We discussed in class how all of these essentially cluster your data, which is why I thought I would have no problems using them with my dataset. Unfortunately, coding and machine learning are not that simple!

Once I saw the graphs for X3 and the unsupervised algorithms, the clustering looked a lot cleaner than it did compared to not using PCA as a pre-processing step. The ARI scores from the first attempt (without PCA) gave me a score of zero for random assignment and DBSCAN, but had a score of -0.04 for agglomerative clustering and -0.05 for K means. The ARI scores stayed pretty much the same, with K-means ARI scoring a -.05 again, and the agglomerative clustering scoring a -.05, slightly larger score (but very insignificant). There was also no score of the random assignment, as no graph was made. Since these numbers are so small, we can tell that the predictive powers are not that good. My silhouette coefficients were .61 for K means, .65 for agglomerative clustering, no score for the DBSCAN and zero for random assignment. The K means and agglomerative changed after the PCA to .13 for both. I think not using the PCA (because then I'm only using one or two components) gave me better scores (more predictive) for the silhouette coefficients, which got much lower after I ran the PCA on my data.

I also made an attempt with the breast cancer dataset. I did this because the actual assignment was to run the algorithms after you account for 95% of the variance with the PCA, which I literally could not do. I did get much larger scores for my ARI's, meaning that this dataset from sci-kit learn was much better than mine, and actually could have legitimate predictions (ARI's were .51 for K means and .68 for agglomerative clustering, however the DBSCAN was still zero.

7) Summarize what you learned across the three projects, including what you think worked and what you would do differently if you had to do it over.

While this might sound odd, learning how machine learning actually works was a big take-away from this class. When I started this summer course, I didn't even know what machine learning was! It seemed like a magical process where the computer just knew what to do. However, I now know that this is NOT true! Machine learning can be done by training a subset of your data, and then using it on the test set. Your algorithm should definitely be based on the type of data and what you are trying to predict. With my background in statistics, this made much more sense, and I was ready to dive in.

The first project was great for learning different ways to clean your data in python. It is so different than using excel or a statistical program. Learning the python language (with pandas) to clean and visualize data was something extremely useful that I hope to do in the future. The second project definitely made my brain run in overdrive. While I am familiar with statistics, making the code work to actually run these algorithms was a tough job. I got stuck numerous times and definitely got more frustrated than I intended. However, I did feel as though I developed a better sense of how these algorithms actually work, especially with seeing the outcomes for its predictive powers. The third project was tricky because I ran into that problem where 95% of my variance was accounted for by one variable. It took me a while to realize that the problems I was seeing, such as the graphs for the unsupervised algorithms and the slicing of the X2_train, was because I only had one array. Once that lightbulb went off in my head, I was able to do two components instead, which led me to experiment better with the unsupervised algorithms and interpreting their ARI and silhouette scores.

Instead of doing this project simultaneously with data camp and the readings, I think I would have benefitted from doing the project in the second half of the class, or even in a different class altogether. I think this intensive was just a little too intense for me, especially with my full-time job of running a preschool. I think what I would do differently is get a tutor, take this class during a regular semester, or do some Data Camp learning prior. I think I also would have picked a slightly less complex dataset. Additionally, I would like to understand the error messages better in python, as I think I would have had vastly different ideas in how to handle my data.