Capstone 1 - In-depth Analysis (Machine Learning)
The code behind this report can be found at:
https://github.com/dgeihsler15/Springboard/blob/master/Capstone1/Machine_Learning_Analysis.ipynb

A lot of the insights gathered so far revolve around how Divvy's usage fluctuates throughout the week and throughout the year. As the usage differs in this way, I'd like to predict station stocking requirements by the day of the week and month of the year. For example, if Divvy wanted to know what the stocking requirements should be for station x on a Monday in August, the model would be able to provide the answer.

For this model I would like to use the ratio of number of trips in to number of trips out for each station on a given day as the target variable for the station's stocking requirements. I believe this will be more valuable because ratio can relate to the number of empty and filled slots each station should have. The total number of rides associated with the station alone on will not give us the same insight.

With the goal of a supervised model in mind, I began this portion of the analysis by organizing the trip data by station, day of the week, and month of the year so I could calculate the trips in to trips out ratio. The number of rides in and number of rides out were found for each station by day of the week and month of the year, but looking closer I found that not every day of the week was accounted for each month for every station. The reason behind this was that not every station had a ride come in or out on every day of the year. This trip information we'd like to have marked as 0 was not available in the current grouping. By unstacking the day of the week, and the month of the year to the top of the dataframe I was then able to then view the ride counts for every day of the week for every month. Those days without trip numbers in them showed up as NaN, and I was able to fill them in with 0s, and restack the day of the week and month of the year variables.

The ratio could then be calculated using complete number of rides in and number of rides out information. A few things need to be addressed regarding this ratio however. When the rides out (the denominator) is 0 for this ratio, it produces a NaN value. To solve this, when rides out is 0, the ratio is set equal to the numerator (rides in). So if 12 rides came in and 0 rides went out the ratio would be 12, not 0 or undefined. This more accurately represents the reality of that particular day. The next instance of concern would be when the number of rides in for the day (the numerator) is 0. As 0 rides in and 1 ride out, and 0 rides in and 12 rides out would need to be accounted for differently we can't have them both with a ratio of 0. In these cases the ratio is set to 1/rides out, this will more accurately reflect the reality as well. The last potential concern would be days with 0 rides in and 0 rides out. This should not read as 0 either, as an even number of rides in and rides out needs to be represented by a 1. In these cases, the ratio was calculated as a 1.

After the ratios were calculated, the following variables were then added to the newly organized dataframe; average trip duration, station capacity, station latitude, station longitude, station's online date, and group number. The group numbers were assigned in previous analysis and are based on traffic to each station. 1 representing a high traffic station, and 4 representing a low traffic station.

Using longitude and latitude alone would not be a big help while building a model because each set of coordinates will be unique. Two stations right next to each other would not be seen as related by the model as the coordinates would differ. To better group the bike stations by location, I split stations into two different groups, north and south. These were split based on the median latitude. I also split the stations into four territories to further narrow down their location. These four territories are marked as 1 (northwest), 2 (northeast), 3 (southwest), and 4 (southeast). The north and south dividing line was again the median latitude, and the east to west dividing line was the median longitude for the stations.

Capstone 1 - In-depth Analysis (Machine Learning)
The code behind this report can be found at:
https://github.com/dgeihsler15/Springboard/blob/master/Capstone1/Machine_Learning_Analysis.ipynb

The last thing I wanted to look at before feature selection was grouping the stations by their capacity. In previous analysis we saw that the mean ratio didn't differ greatly between station capacities, but the variance was higher in the lower capacity stations. Segmenting the stations will allow models to be created for each group that will ideally better fit each groups intricacies. Grouping the stations by capacities will also help in predicting the ratios for new stations. If we grouped stations by traffic level, for example, there wouldn't be any trip information for a brand new station to help determine what model should be used. The stations were segmented into 4 groups by capacity 1 containing the lowest capacity stations and 4 containing the highest. These designations were recorded in dataframe under the cap_group column.

Once this information was gathered together in the new data frame, I used Lasso regularization to provide some insight on my feature selection. Looking at the Lasso coefficients, none of the variables had strong coefficients on their own. Because this was the case, all of the predictor variables (day of the week, month of the year, average duration, station capacity, group, year online, month online, section, and territory) would be used in the models to keep as much predictive power as possible. With a smaller number of variables we don't have to worry about the dimensions being too high in this case.

As we're looking to predict a continuous variable (the trips in to trips out ratio) I started with the linear regression model. Before training this model though, I needed to change the territory, section, group, day of the week, and month of the year variables to categorical data. Although the data are numbers, they represent different categories, and don't relate to each other by their size. Next, dummy columns were added to turn the categorical data into a series of 1's and 0's depending on what category they belonged to. This will allow us to use the categorical information in the regression model.

After splitting the data into training and testing sets, and training the linear regression model, it became clear that model was not performing well. After trying ridge regularization as well, I found a similar result of less than 1% accuracy. Looking back at scatter plots and correlation coefficients for these variables it was not surprising to find that they don't have a strong linear relationship with the calculated ratio.

To help solve the main goal of predicting stocking requirements for the bike stations I grouped the ratios into 5 different classes. These classes were based on the value of the ratio as the ratios near one another are going to have very similar stocking recommendations. For example whether the ratio is .8 or 1.2, you still have approximately the same amount of bikes coming in vs leaving a station on the particular day, and you would ideally have that station stocked with equal number of filled and empty slots. Once these stocking classes were created, this became a classification problem instead of regression.

Before working with different models, I wanted to have a benchmark to test them against. The benchmark used was the average ratio for each station for all of 2017. I wanted to test if these models would be more effective than putting them in a particular stock class based on this average. Looking at all of the stations together, the accuracy of our benchmark in predicting the correct stocking class was 68%. Separating the stations into the four groups by capacity mentioned previously, the accuracy of the benchmark was 61% for cap_group 1, 64% for cap_group 2, 75% for cap_group 3, and 81% for cap_group 4.

With these benchmarks in mind, I looked at 4 models while searching for the most accurate predictions. KNeighbors, Random Forest, SVM, and Multinomial Naive Bayes. Although often used with text data, the bayes model was included as it can be of help in finding a dependent categorical variable that has more than 2 possible categories.

Capstone 1 - In-depth Analysis (Machine Learning)
The code behind this report can be found at:
https://github.com/dgeihsler15/Springboard/blob/master/Capstone1/Machine_Learning_Analysis.ipynb

Each of these models' hyperparameters were tuned using GridSearchCV. If the value on either end of the grid search was found to be the best parameter, another grid search was run to check the range above or below that value. Classification reports were then run on each model as well.

The accuracy results for these models can be found below:

	Not Grouped	Group 1	Group 2	Group 3	Group 4
Benchmark:	68.47	61.02	63.67	74.71	80.91
Models:					
SVM	72.69	71.57	64.98	74.46	79.76
KNN	70.01	70	63.87	74.24	80.48
Random Forest	66.96	65.4	59.7	72.04	78.98
Multinomial NB	65.97	60.39	61.12	72.35	77.25

SVM and KNeighbors performed the best across the board, and both beat the benchmark accuracy score of 68% for the ungrouped model. When the models were segmented based on on our 4 groups, outperformed the benchmark scores for groups 1 and 2, and were in line with the benchmarks for groups 3 and 4.

Both training data scoring and test data scoring were looked at for each of the models to find if they were overfitting the training data. The Random Forest had the biggest differences between it's training and test scores. Even with the optimal parameter the training scores hovered around 98% but the testing scores did not come close. The training and testing scores were much more consistent in the other three models, and we don't have to worry about overfitting in this case with KNeighbors, SVM, and Multinomial NB.