Khalid Ali

August 5, 2017

SF Biking Capstone

Definition

Project Overview

Bike share programs are increasingly becoming available in large cities worldwide. They provide tourists and residents a cheap form of transportation to get around the city. One problem that arises for the city is the maintenance of these bikes. When bikes are left outside in the weather, they begin to deteriorate and need maintenance sooner than if they were inside. The bike program I will be looking at is the one in the San Francisco Bay Area. The dataset I found contains ~600k entries for bike rides for all the stations during the time of August 2013 and August 2015. Using this data, I created a model that will predict the number of people that will rent bikes at a certain location so that the company can move the excess bikes into a sheltered location, likely a warehouse.

Problem Statement

The goal of this project is to create a regression model that will predict the number of bikes needed at a station for a given hour. To accomplish this task, I will need to:

- Load and preprocess the SF Bay Area Bike Share data
- Split the dataset into each respective station
- Train a regression model to determine the number of bikes used for a particular hour
- Test model accuracy
- Find the best time to move bikes and replace them

The final model will be able to accurately find times of low activity so the city can move bikes into a sheltered location.

Metrics

I will be using the Median Absolute Error to determine the absolute fit of my model. The error is calculated by first finding how far each value is from the median. Then, it returns the median value of the previous distances. I chose this metric because it gives me a median value of the error between the predicted users and the actual users. I opted not to use mean error because of the large range of values that may cause the score to be artificially high or low. This helps me assess the tolerance level of my predictions, and therefore more accurately find the correct number of bikes that will be used.

Analysis

Data Exploration

The SF Bay Area Bike Share dataset is a collection of bike trip data from August 2013 to August 2015. The data is split into four different datasets of which I only use two. These are weather.csv and trip.csv. The weather dataset contains 21 features pertaining to the weather of the day in a

Sample One Row

	max_temperature_f	mean_temperature_f	min_temperature_f	max_dew_point_f	mean_dew_point_f
0	81	72	63	62	61

min_dew_point_f	max_humidity	mean_humidity	min_humidity	max_sea_level_pressure_inches
59	87	69	51	30.06

mean_sea_level_pressure_inche	min_sea_level_pressure_inches	max_visibility_miles	mean_visibility_miles
30.02	29.96	10	10

min_visibility_miles	max_wind_Speed_mph	mean_wind_speed_mph	max_gust_speed_mph
10	16	7	24

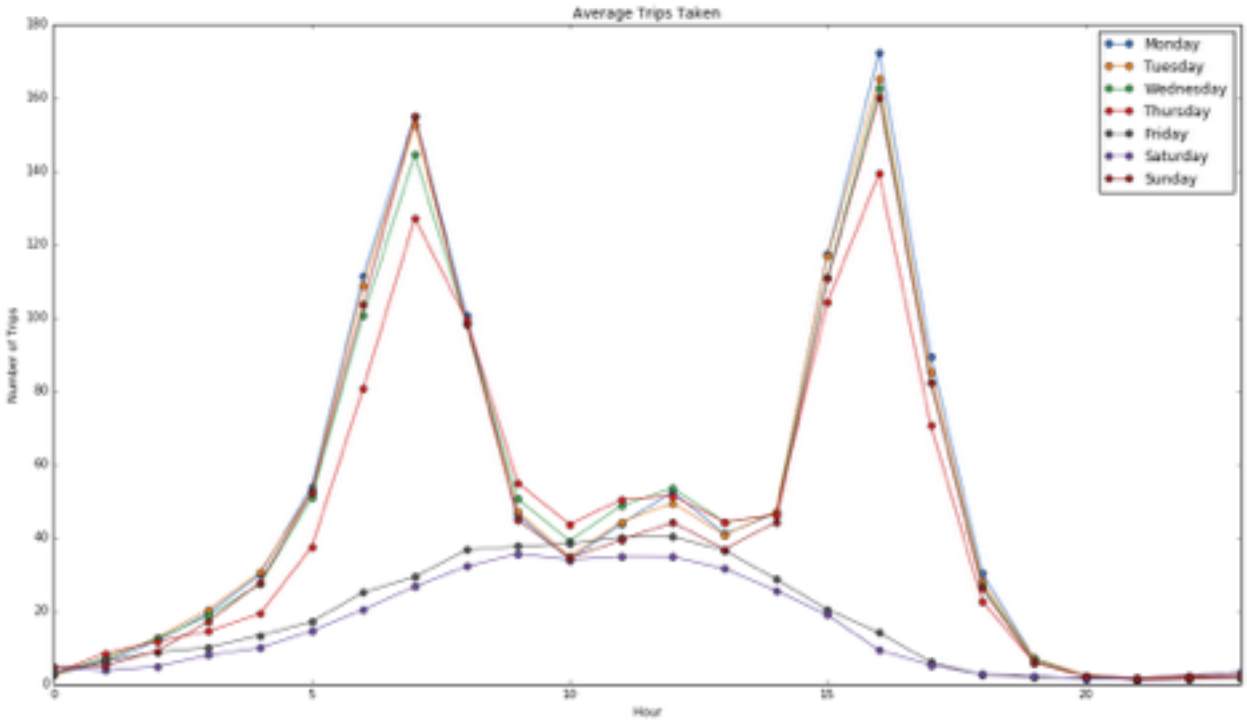
precipitation_inches	cloud_cover	wind_dir_degrees	Fog	Fog-Rain	Normal	Rain	count
0	4	320	0	0	1	0	50

year	month	hour	weekday
0	8	10	2

particular zip code. There are only five zip codes and they are close together, so the weather is very similar. The trip dataset contains about 666,000 entries of bike data. Each entry consists of a date, down to the second, of when the bike was rented and when the bike was returned as well as a station id of both the start and end stations. The final dataset will contain 31 features.

Exploratory Visualization

This plot shows the average number of trips taken per hour. This is helpful in determining trends in the data as well as seeing the importance of certain features. Here, we notice there are two distinct patterns, one with two spikes that corresponds to weekdays and a gradual slope that corresponds with weekends. The unusual pattern of weekdays may indicate the trips people take to work, once in the morning and once again in the afternoon.



Algorithms and Techniques

The regression model I will be training is the Random Forest Regressor, an ensemble method. It has been known to be highly accurate with datasets with large amounts of data. In addition, it can also give an estimate on the features that are the most important, which is a nice bonus. The algorithm is known to overfit to noisy data, but there are few, if any, outliers in the data.
The following parameters can be tuned to optimize the classifier:

- Training parameters:
    - N_estimators (The number of trees in the forest)
    - Criterion (Whether to use mean squared error or mean absolute error)
    - Max_features (Number of features to consider)
    - Max_depth (The max depth of the tree)

During training, the modified dataset is loaded into RAM before it is processed by the CPU. The Random Forest Regressor creates a forest of random decision trees during training. Each decision tree takes a subsample of the training data and randomly chooses predictor variables at each node to create the best split of the data. After training, I cross validate the data to try and emulate real-life data. During testing, the regressor will run the test data through all the trees and take a weighted average of the results from each individual tree before presenting the final product. At the end, I take the median of the scores to get a final score for my data.

Benchmark

In a perfect world, the score of Root Median Square Error would be zero to indicate that there were no excess / lack of bikes expected. Since the number of bike docks per station is about eighteen, the model should accurately predict with an accuracy of ninety percent of the bikes, which is about two bikes.

Methodology

Data Preprocessing

The two datasets required a number of changes before I achieved a usable training set. First, I stripped the dataset of any values that were categorical. Next, I aggregated the trip data by the date so that I had a number of trips per hour on a given day. Then, I chose the weather data from one of the five zip codes with the least amount of unknown values. Since the zip codes were close together, the weather inaccuracy is insignificant. The max_gust_speed_mph had some nulls, so I used the mean of the max_wind_speed_mph because they had a correlation of approximately 94%. Another feature that I changed was the precipitation_inches. It had some values of 'T' mixed in which conflicted with most of the numerical data in the column. Hence, I assumed T meant true and replaced its value with the current median of the feature. I merged the weather data with the modified trip data to come up with a final dataset. I took 75% for training and 25% for testing.

Implementation

To do the project, I first loaded the data into memory. Next, I did the preprocessing steps as mentioned in the above section. Then, I created an instance of the Random Forest Regressor with some basic training parameters. I first attempted to use a GridSearchCV to find the best parameters for training. Next, I trained it with GridSearchCV to find the optimum parameters. I tested it using cross-validation with ten folds as one test. I also predicted the values from the test set with the actual values and scored it.

Refinement

Using the base estimator gave a test score with an error of 4.5 bikes per day, which is very accurate. To optimize the estimator, I used the GridSearchCV to optimize three parameters.

- max_features:
    - By increasing max_features, you generally increase the performance of the model. In doing so, you lose speed in training the model. In addition, you also lose diversity of an individual tree due to the higher number of options that are considered.
- n_estimators:
    - This value is the total number of trees to build in the forest before taking the prediction. A higher value gives you better performance, but it takes longer to train.
- min_sample_leaf:
    - A large value of min_sample_leaf helps the data give more correct readings when there is a lot of noise in the training data. Though, too high a value can cause also cause performance to suffer.

After tweaking the training parameters and using GridSearchCV, I got an error of 4.16 bikes per day from the optimized model.

Results

Model Evaluation and Validation

The following parameters were chosen because they were the best balance of performance and speed.

- max_features was set to ‘auto’, or all the features
- n_estimators was set to 200 because the small amount of performance gain after this number does not justify how long it takes to train
- min_sample_leaf set to a low value of 5 gives the best value surprisingly.

To check the robustness, I took one hundred random values from the original dataset and scored it against the model. I got an error of 3.15 bikes per day.

Justification

The final results show that the error is below the benchmark by two bikes. This is only eighty percent accuracy in terms of station by station, but in the bigger picture, this model does approximate the total number bikes that will be out in a certain hour with a tolerance of four more/less bikes. There will be times when special events held in San Francisco may cause an influx of bike riders, so the model is not fully robust, but enough to solve the problem. The results found from the model can be trusted since there does seem to be an underlying pattern to the number of bikes taken every hour. Overall, the final model does not perform to the benchmark I set (off by two bikes), but it is reasonably close and it aligns with the solution's expectations.

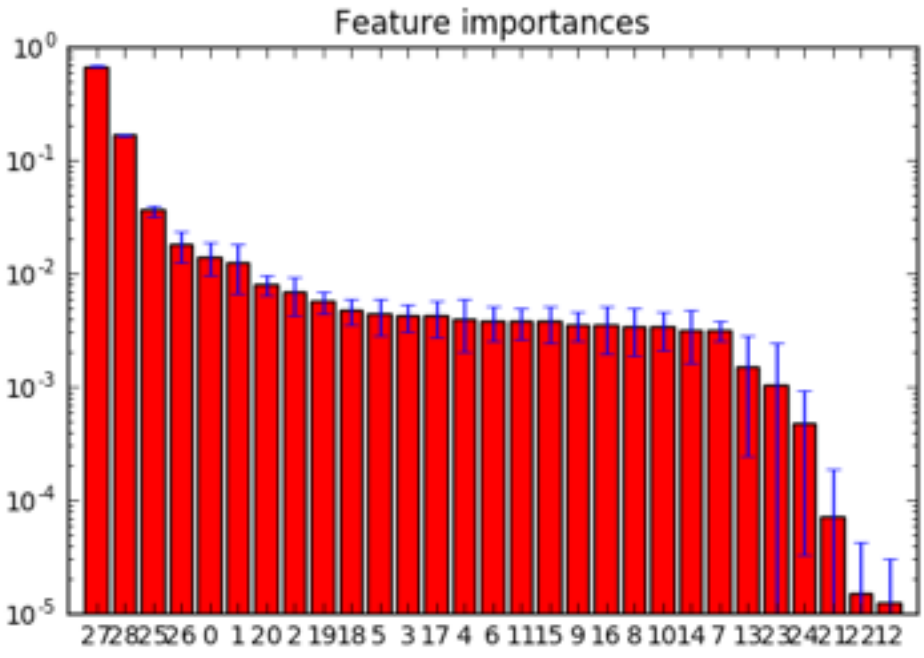
Conclusion

Free-Form Visualization

One of the significant features of the model is how it ranks the importance of the given features. When I look at the graph of feature importances, I see that there is a heavy reliance on the hour parameter, which may be causing the wrong predictions.

Feature ranking:

1. feature 27 (0.676732) hour
2. feature 28 (0.166133) weekday
3. feature 25 (0.036103) year
4. feature 26 (0.017804) month
5. feature 0 (0.014069) max_temperature_f



6. feature 1 (0.012280) mean_temperature_f
7. feature 20 (0.007955) wind_dir_degrees
8. feature 2 (0.006829) min_temperature_f
9. feature 19 (0.005728) cloud_cover
10. feature 18 (0.004763) precipitation_inches

Reflection

The steps I took to complete this project are as follows:

1. I found an initial problem and a relevant dataset was found
2. The data was downloaded and preprocessed
3. A benchmark was created for the regression model
4. The estimator was trained multiple times till the best parameters were found
5. I tested the estimator against a subset of the original data and got a final score

The most interesting part of the project was actually doing the testing portion of the project because I got to see a final score. I also thought that finding the dataset was interesting since I was able to look at the variety of information available to use. The most difficult part was probably step two. Preprocessing the data was a nightmare because of all the transformations that I had to do, but on the flip side, I learned a lot about pandas data frames and their inner workings. The final model and solution does fit my expectation since it turned out to be pretty accurate. I think that the ensemble method Random Forest Regressor worked very well for this problem and should be used to solve other problems of a similar type.

Improvement

Well, for one, I would like to test other ensemble methods and regression models to see if they work better than the one I chose. In addition, this data becomes more and more irrelevant as time

goes on since it does not account for population growth. One way around this is to get population data of the San Francisco area, but generally this is hard to find online since it pertains to a very specific area. Another problem is that the data will ‘grow old’ over time. To remedy this, my solution would need to reach out to the SF bike share API and pull the latest data as well as a weather API to get the weather during this time. This way, the model will always be up to date and can maintain its accuracy.