GENERAL ASSEMBLY

Predicting Changes in Divvy Station Capacity

Jimmy Conway

March 22, 2018

Contents

Problem Statement

Problem Statement
Background

Divvy is a station-based bike sharing system similar to those found in larger cities around the country.

Launched in 2013, the Divvy system has expanded to include 585 stations spread across three cities and approximately 5,800 bikes.

Bike distribution is a major concern for any bike sharing system. There are two scenarios to avoid:

- **Station empty** (users cannot rent bikes)
- **Station full** (users cannot return bikes)

To maintain proper bike distribution, Divvy employs a fleet of trucks to move bikes between stations as needed.

Accurate prediction of near-term changes in station capacity could help Divvy more efficiently utilize its bike distribution resources.

Problem Statement

Target Variable Definition and Use Case Hypothesis

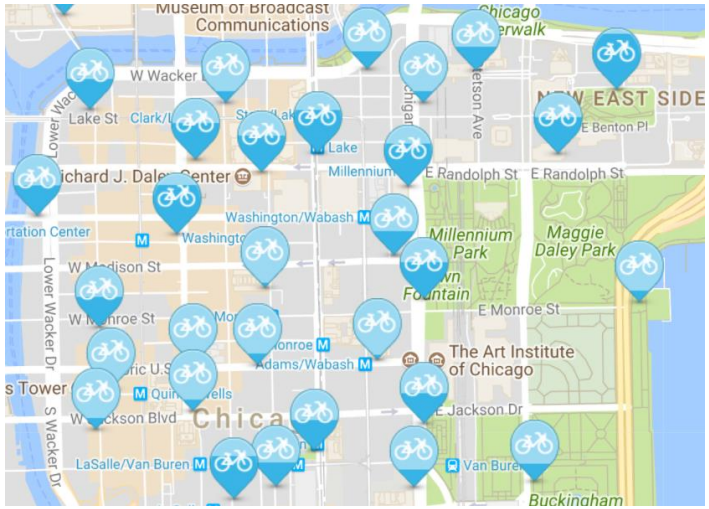
The model's target variable is called **bike differential**, defined as:

- Rides **ending** at station in upcoming time period, **MINUS**
- Rides **beginning** at station in upcoming time period

Divvy could predict station capacity, using this information in combination with data it already has access to:

- Total capacity of each station
- Current-state number of bikes at each station
- Current time and date
- Current weather conditions
- Near-term forecasted weather conditions

Note that by summing bike differential for multiple upcoming periods, Divvy could string together a rolling prediction of capacity at each station for the next hour, day, week, etc.



Divvy stations around "the loop" in downtown Chicago. Icons indicate current station capacity.

Plotting stations projected to have too many or too few bikes in the near future on a map could assist in planning Divvy's daily truck operations.

Data
(~99% of the work)

Data

Sources

There were two data sources utilized:

- **Divvy**: Log of all rides taken, station information
- **National Oceanic and Atmospheric Administration** (NOAA): Daily weather "normals", hourly weather information

Divvy trip data has one row per individual bike trip with the following information:

- Starting/ending station ID and name
- Starting/ending date and time
- Bike ID
- Whether the user was an annual subscriber or one-time-use customer
- If a subscriber, gender and birth year of the user

Divvy station data has one row per station with the following information:

- ID and name
- Location: Latitude, longitude, and city
- Bike capacity
- Date the station went online

NOAA "normals" data is reflects weather averages over the 30-year period from 1981 – 2010. Normals data available includes temperature, precipitation, snowfall, etc. Averages are available by year, season, month, day, and hour.

Only the normal daily temperature data was used for this analysis. Normal daily temperature captures in a numeric variable what otherwise a categorical variable like "month" or "season" might.

Current weather, hour by hour, is captured in the **Local Climatological Data (LCD) Dataset** . This includes every reading from the station for all conceivable weather metrics including temperature, precipitation, dew point, humidity, pressure, etc.

For both NOAA datasets, the weather station used was Midway airport, selected for being the most centrally located of available choices.

Data

Cleanup and Manipulation

The first major task was to "shrink" the data. There were 13.8 million rides taken between 2013 and 2017. How do we craft the observation data we can feed into the model?

- **Set time buckets**. The daily normal temperature will capture the "month" aspect of datetime. Time buckets capture day and time.
- The first step in this task is to visualize how bike activity varies by day and hour.
- As the heat map shows, very consistent weekday/weekend patterns emerge.
- In the end, 8 time buckets were selected. Five for weekdays (early morning, morning rush, afternoon, evening rush, night) and three for weekends (morning, afternoon, night).

HOUR \ WEEKDAY	0	1	2	3	4	5	6	Total
0	0.1%	0.1%	0.1%	0.1%	0.1%	0.2%	0.2%	0.7%
1	0.0%	0.0%	0.0%	0.0%	0.1%	0.1%	0.1%	0.5%
2	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.1%	0.3%
3	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%
4	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%
5	0.1%	0.1%	0.1%	0.1%	0.1%	0.0%	0.0%	0.7%
6	0.4%	0.5%	0.5%	0.5%	0.4%	0.1%	0.1%	2.5%
7	1.0%	1.1%	1.1%	1.0%	1.0%	0.2%	0.2%	5.6%
8	1.3%	1.4%	1.4%	1.3%	1.3%	0.4%	0.3%	7.3%
9	0.6%	0.6%	0.6%	0.6%	0.6%	0.6%	0.6%	4.3%
10	0.5%	0.4%	0.4%	0.4%	0.5%	0.9%	0.8%	4.0%
11	0.6%	0.6%	0.5%	0.6%	0.7%	1.1%	1.1%	5.2%
12	0.7%	0.6%	0.6%	0.6%	0.8%	1.3%	1.2%	6.0%
13	0.7%	0.6%	0.6%	0.6%	0.8%	1.3%	1.2%	6.1%
14	0.7%	0.6%	0.6%	0.6%	0.8%	1.3%	1.3%	6.0%
15	0.9%	0.8%	0.7%	0.8%	1.0%	1.3%	1.3%	6.8%
16	1.4%	1.4%	1.3%	1.3%	1.5%	1.3%	1.2%	9.2%
17	2.0%	2.0%	1.9%	1.9%	1.6%	1.1%	1.1%	11.6%
18	1.3%	1.3%	1.2%	1.3%	1.1%	0.9%	0.9%	8.1%
19	0.8%	0.9%	0.8%	0.8%	0.7%	0.7%	0.7%	5.4%
20	0.5%	0.6%	0.5%	0.6%	0.5%	0.5%	0.5%	3.6%
21	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.3%	2.7%
22	0.2%	0.3%	0.3%	0.3%	0.3%	0.4%	0.2%	2.0%
23	0.1%	0.1%	0.1%	0.2%	0.2%	0.3%	0.1%	1.2%
Total	14.7%	14.6%	13.9%	14.1%	14.6%	14.5%	13.5%	100.0%

The form of the observation data is now set:

One row per station/time bucket/date combination. Sum up all rides beginning and ending at each station over each time bucket on each date.

Data

Cleanup and Manipulation (continued)

The next step is to **create a shell** in which to put the trip and weather data. The shell needs to include one row for each station/time bucket/date combination. **What could go wrong?**

- **Must capture all combinations**, even those for which there are no rides beginning or ending. Capturing those "zeroes" is very important to the analysis. Making tags from the Divvy data and pulling unique values will not do.
- **Date ranges vary by station**, as stations have various inception dates over the course of Divvy's history. Starting all stations back in 2013 would put many improper zeroes in the data.
- First the inception date of each station was inferred using earliest dates a station appears in trip data, checked against inception date in the station data.

Nested "for" loops were used to generate the "shell" of the observations DataFrame, creating only the unique rows we want:

```
temp_list=[]
for station in final_station_data.index.values:
    start_date = final_station_data.loc[station,'first_day_operational']
    for date in pd.date_range(start_date, '2017-12-31'):
        if date.weekday() <= 4:
            for time in range(5):
                tag = 'ID'+str(station)+'Date'+str(date.date())+'Time'+str(time)
                temp_list.append((station, date, time, tag))
        else:
            for time in range(5,8):
                tag = 'ID'+str(station)+'Date'+str(date.date())+'Time'+str(time)
                temp_list.append((station, date, time, tag))

column_list = ['station', 'date', 'time', 'tag']
final_data = pd.DataFrame(temp_list, columns=column_list)
```

Note, in creating the final shell, certain stations were filtered out:

- Not included in most recent station list (i.e. deactivated)
- Inception date in 2017 (not enough data)
- Bike capacity = 0
- Located in Oak Park (discontinuing Divvy operations in 2018)

Ultimately, 565 of 585 current stations were included in the analysis.

Data

Cleanup and Manipulation (continued)

Now that we have our shell, we need to add in the data.

Divvy trip data is relatively straightforward:

- Create a station+date+time tag
- Do value_counts on the tags and turn them into a DataFrame
- Merge the counts into the shell
- Fill NaN's with zero

Daily weather normals are also easy:

- Create a date tag with the year removed in both files
- Merge in the data
- The daily normals, being long-term averages, are very complete; no mess (just add Leap Day)

tag	station	date	time	start_tag_count	end_tag_count	bike_differential	total_activity	normal_temp_range	normal_temp_ave	normal_temp_max
ID2Date2015-05-09Time5	2	2015-05-09	5	0	0	0	0	18.6	58.7	68.0
ID2Date2015-05-09Time6	2	2015-05-09	6	48	43	-5	91	18.6	58.7	68.0
ID2Date2015-05-09Time7	2	2015-05-09	7	1	1	0	2	18.6	58.7	68.0
ID2Date2015-05-10Time5	2	2015-05-10	5	2	2	0	4	18.7	59.0	68.4
ID2Date2015-05-10Time6	2	2015-05-10	6	27	26	-1	53	18.7	59.0	68.4

Hourly weather data is another matter entirely:

- Many different "report types" coming in at irregular times of day
- Incomplete reports
- Suspect values
- Weather station equipment can easily be damaged, out of order, etc.

Much cleaning was required; for a longer-term project, some of the decisions made (e.g. accept all suspect values, replace "trace" precipitation with 0.005 inches) should be revisited.

Finally, our data is ready!

```
In [81]: divvy.dtypes
Out[81]: station                 int64
         date          datetime64[ns]
         time                    int64
         start_tag_count         int64
         end_tag_count           int64
         bike_differential       int64
         total_activity          int64
         normal_temp_range     float64
         normal_temp_ave       float64
         normal_temp_max       float64
         normal_temp_min       float64
         current_temp          float64
         current_dew_point     float64
         current_humidity      float64
         current_wind_speed    float64
         current_pressure      float64
         current_precipitation float64
         null_pred             float64
         dtype: object
```

```
In [80]: divvy.isnull().sum()
Out[80]: station                 0
         date                    0
         time                    0
         start_tag_count         0
         end_tag_count           0
         bike_differential       0
         total_activity          0
         normal_temp_range       0
         normal_temp_ave         0
         normal_temp_max         0
         normal_temp_min         0
         current_temp            0
         current_dew_point       0
         current_humidity        0
         current_wind_speed      0
         current_pressure        0
         current_precipitation   0
         null_pred               0
         dtype: int64
```

Model Selection and Results

Model Selection and Results

Null Model

In order to make the Null Model somewhat competitive, y_pred was based on both **station** and **time bucket**.

For observation with given [station] and [time]:

y_pred = average [bike_differential] across all observations with that [station] and [time]

Null RMSE was calculated for each station and saved in a DataFrame. See descriptive statistics to the right for the Null Model RMSE.

What did we learn?

- As we will come to see, this actually does a very decent job of predicting differential.
- Problems arise with certain stations dealing with very high bike volume. Note that the max RMSE is approximately 7x the 75th percentile.

	null_rmse
count	565.000000
mean	2.492552
std	2.561555
min	0.046822
25%	0.793228
50%	1.975058
75%	3.222389
max	22.053789

Model Selection and Results

Variable Selection

When constructing the observation data, we essentially pulled in **all available weather data**.

It is quite likely that many of these variables are **not predictive of bike activity** or are **closely correlated to each other**.

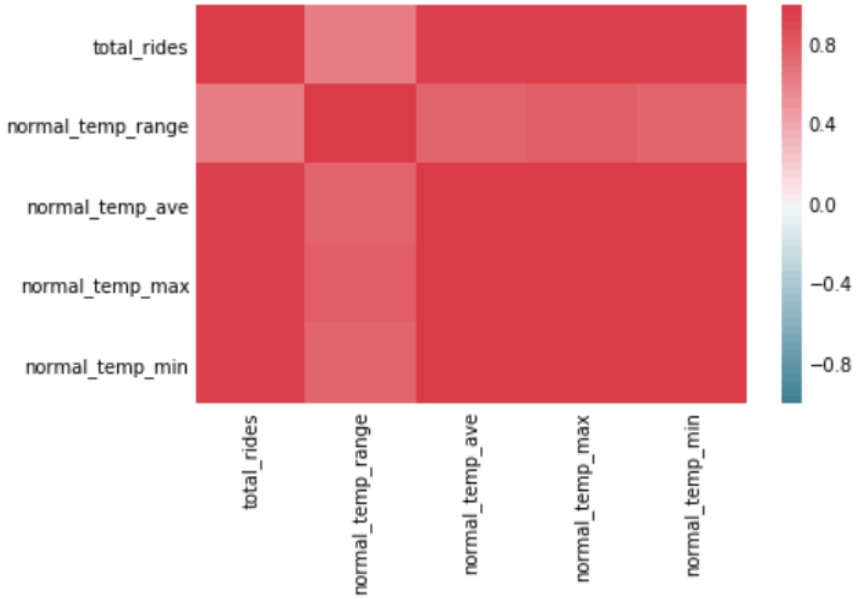
We look at bike activity as a whole, assuming that weather affects ridership similarly across different stations, times of day, etc. Perhaps another thing to revisit later.

The graphics to the right illustrate correlation between total rides and the various weather normals.

What did we learn?

- A sea of red! While normal temperature is certainly correlated with bike activity, we probably only want to choose one of these variables, since they are so closely related to each other.
- Due to having the highest correlation with total_rides, **normal_temp_min** is selected for use in the models to come.

Weather Normals Correlations



	total_rides	normal_temp_range	normal_temp_ave	normal_temp_max	normal_temp_min
total_rides	1.000000	0.643684	0.949477	0.944921	0.953175
normal_temp_range	0.643684	1.000000	0.759075	0.780672	0.734455
normal_temp_ave	0.949477	0.759075	1.000000	0.999425	0.999311
normal_temp_max	0.944921	0.780672	0.999425	1.000000	0.997486
normal_temp_min	0.953175	0.734455	0.999311	0.997486	1.000000

Model Selection and Results

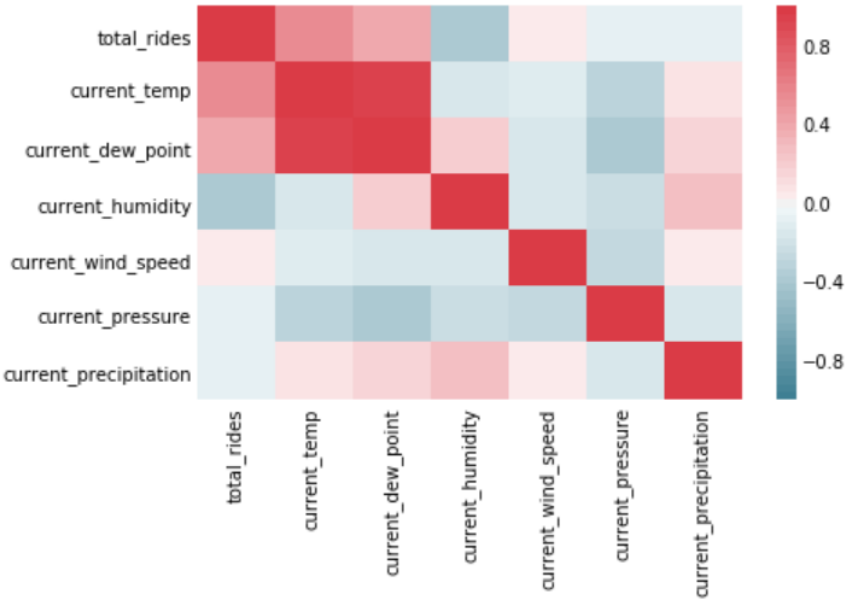
Variable Selection (continued)

Next we do the same process for the hourly weather data. The graphics to the right illustrate correlation between total rides and the various current weather measurements.

What did we learn?

- A much more diverse palette of correlations here. Temperature (+), dew point(+), and humidity (-) have the strongest correlation with bike activity.
- However, dew point is much more strongly correlated with the temperature, so we don't need it.
- **current_temp** and **current_humidity** are selected for use in the models to come.

Hourly Weather Correlations



	total_rides	current_temp	current_dew_point	current_humidity	current_wind_speed	current_pressure	current_precipitation
total_rides	1.000000	0.532708	0.390112	-0.397224	0.055018	-0.073875	-0.063931
current_temp	0.532708	1.000000	0.941542	-0.154007	-0.119430	-0.333144	0.063757
current_dew_point	0.390112	0.941542	1.000000	0.182063	-0.172857	-0.400867	0.142523
current_humidity	-0.397224	-0.154007	0.182063	1.000000	-0.155386	-0.232962	0.257775
current_wind_speed	0.055018	-0.119430	-0.172857	-0.155386	1.000000	-0.259796	0.053768
current_pressure	-0.073875	-0.333144	-0.400867	-0.232962	-0.259796	1.000000	-0.161939
current_precipitation	-0.063931	0.063757	0.142523	0.257775	0.053768	-0.161939	1.000000

Model Selection and Results

Linear Regression Models

There are two linear regression models shown here to illustrate. The first uses polynomials up to a factor of 4. The second is the simplest model possible; the only transformation made to the data was to get dummies for the time buckets.

What did we learn?

- Neither of these models is very good.
- Adding higher-order polynomials could help, but not very much.
- At this point, it's probably not worth the extra complexity; the null model would probably be preferred in practice.

	null_rmse
count	565.000000
mean	2.492552
std	2.561555
min	0.046822
25%	0.793228
50%	1.975058
75%	3.222389
max	22.053789

	lr_RMSE
count	565.000000
mean	2.482930
std	2.512858
min	0.034062
25%	0.798839
50%	1.979332
75%	3.224762
max	20.769727

	lr_RMSE_2
count	565.000000
mean	2.482950
std	2.533575
min	0.042465
25%	0.794816
50%	1.977183
75%	3.230419
max	21.265235

Model Selection and Results

Random Forest Regression Models

Four random forest models were created:

1. Limiter is a min_samples_split of 5% of the data
2. Limiter is a max_depth of 5
3. max_depth of 6
4. max_depth of 8

What did we learn?

- The max_depth-limited models do appear to do a good job of bringing down the top end of the RMSE.
- The appropriate max_depth is probably around 5 or 6.
- These models take forever to run (about an hour). In practice, you may want to fit the model once per day (or less frequently) and then only generate predictions from it, or use a remote server to continuously run the projections and export them to csv automatically.

	null_rmse
count	565.000000
mean	2.492552
std	2.561555
min	0.046822
25%	0.793228
50%	1.975058
75%	3.222389
max	22.053789

	rfreg_RMSE
count	565.000000
mean	2.347739
std	2.287885
min	0.041399
25%	0.793765
50%	1.907099
75%	3.063866
max	20.508510

	rfreg_RMSE_2
count	565.000000
mean	2.326152
std	2.207106
min	0.051840
25%	0.797065
50%	1.908761
75%	3.046848
max	16.351499

	rfreg_RMSE_3
count	565.000000
mean	2.323652
std	2.196599
min	0.052427
25%	0.793029
50%	1.914373
75%	3.055300
max	16.165930

	rfreg_RMSE_4
count	565.000000
mean	2.335409
std	2.201926
min	0.054031
25%	0.801756
50%	1.911849
75%	3.069363
max	16.324823

Model Selection and Results

Summary of Results

For each of the 565 stations, I "ranked" the models (null, 2 linear, 4 random forest) from 1 to 7 by RMSE.

The table below shows the count of ranks for each model. The more stations have a model at lower ranks, the better for the model.

The rank counts, average rank, and average RMSE scores confirm what we saw on the previous slide:

The most effective model for predicting bike differential is a random forest model with a max_depth of 5 or 6.

The most important feature is time bucket, with the weather variables relatively equal in importance.

	rfreg2 Import	rfreg3 Import
Time	0.552	0.501
Normal Temp	0.125	0.141
Current Temp	0.199	0.210
Current Humidity	0.125	0.148

Rank	Null Rank	lr Rank	lr2 Rank	rfreg Rank	rfreg2 Rank	rfreg3 Rank	rfreg4 Rank
1	74	13	60	79	138	162	39
2	46	15	78	96	159	128	43
3	38	51	32	133	124	105	82
4	43	43	25	191	53	32	178
5	202	133	79	36	50	48	17
6	88	83	215	24	32	71	52
7	74	227	76	6	9	19	154
Ave. Rank	**4.44**	**5.52**	**4.65**	**3.19**	**2.73**	**2.94**	**4.53**
Mean RMSE	**2.49**	**2.48**	**2.48**	**2.35**	**2.33**	**2.32**	**2.34**

Conclusion

Conclusion

Considerations for the Future

There are some pre-processing considerations that were overlooked in the interest of time:

- Some stations get **moved over time**. While usually not moved too far, some moves are significant. This could have an impact on ridership that needs to be addressed.
- The hourly weather data is **very messy**. The process for cleaning it could be more robust, use more of the report types, and perhaps check reasonability based on values from other reports around that time.
- Alternatively, another source for current weather data, such as Wunderground, could be considered.
- There are potentially **other external data sources**, having nothing to do with weather, that are significant. For example, how are stations around Wrigley Field impact by whether there is a home Cubs game? How do holidays affect ridership?

There are always more tweaks that could be made to the model selection:

- **Stations are very, very different from one another**, both in how they relate to time buckets, and in general level of activity. Many stations don't see much action, while the station at Navy Pier can run through 2,000 bikes on a weekend afternoon in the summer. **Perhaps using the same model archetype across all stations is incorrect?** For what types of stations does the model fall short now?
- Was it incorrect to eliminate those weather variables at the outset? Precipitation comes to mind. It suffers from the fact that the vast majority of observations have the same precipitation value: zero. This impedes the correlation coefficient, but perhaps it would still be useful in a decision tree?

Thank you for listening!

Questions?