Forecasting Bike Rental Demand in SF

data science capstone project 2

Rui Cao

2020-01-30

Introduction

Bike Sharing System is a modern way to rent bikes and its commonly used in metropolitan cities like San Francisco, New York City, and Washington DC. The process of obtaining membership, renting and returning bikes is automated through networks throughout the city. Using these systems, people can rent a bike from one place and return it to another as needed. Currently, there are more than 500 bike-sharing programs worldwide.

Bike share systems offer challenges for both the system owners and users. For example, cities often manually redistribute bikes between stations to ensure availability of bikes and docks. Cities also need to use rider data to inform where future bike stations should be installed. For users, the main questions are: will there be a bike at the nearest station when I need? The uncertainty of bikes at a given station has strong business implications, as unpredictable bike supply can deter ridership and reduce demand.

Objective

In this project, I will use the SF Bay Area Bike Share dataset to evaluates current ridership trends and predicted daily trip counts for the San Francisco bike share system. The four main sections of this project include:

1. Data Wrangling - explore and merge all datasets together, make a new data frame ready to analyze.
2. Data Exploratory Analysis based on a new data frame and see what is the relationship between each factor and the total daily trips.
3. Feature engineering to get the best features which affect the daily trips most.
4. Find the best model to predict daily trips in SF on a selected station.

Data Wrangling

In this project, I will use 3 datasets: trip, station, and weather. they include a variety of bike-share information between Aug 29,2013, to Aug 30, 2015.

- ### Trip

  the trip dataset has 11 columns and 648717 entries trip-level records, includes date, start/end time, start/end station ID, start/end station name, bike ID, rider subscription type, and trip duration.

1. Start by checking the missing values, I found there are 6619 nan values in the zip_code column. because I don't need this feature for future analysis, I will just leave it like this.

```
id                      0
duration                0
start_date              0
start_station_name      0
start_station_id        0
end_date                0
end_station_name        0
end_station_id          0
bike_id                 0
subscription_type       0
zip_code             6619
dtype: int64
```

figure1. missing values

2. Check the duration values and remove the outliers by only choose the trip duration of less than 1 hour.

```
count    669959.000000
mean         18.465831
std         370.923950
min           1.000000
25%           5.733333
50%           8.616667
75%          12.583333
max      287840.000000
Name: duration, dtype: float64
```

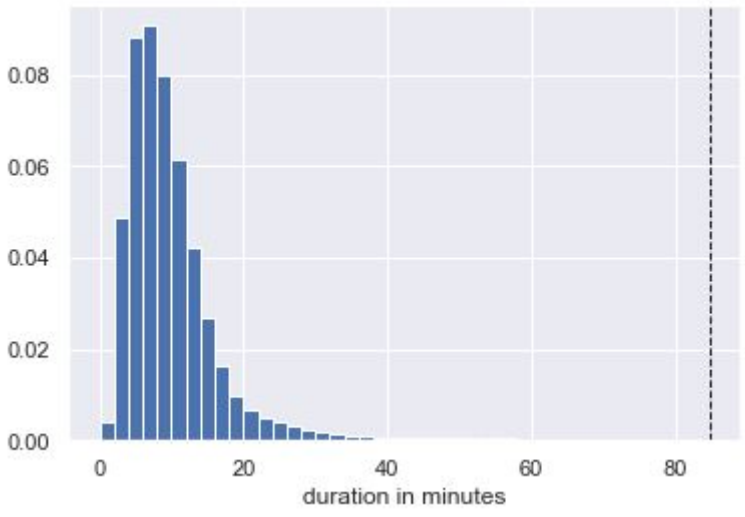


figure 2. duration info

3. Change the date column to DateTime format and use it to add new columns: date, day, hour, month, week. Change the day column pattern from 1-7 to Monday to Friday and month column pattern from 1-12 to January to December. Add holiday and working day columns based on the date and day column.

4. Drop ‘id’ and ‘zip code’ columns and convert all the objective columns to categorical.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 648717 entries, 0 to 669958
Data columns (total 15 columns):
duration              648717 non-null float64
start_date            648717 non-null datetime64[ns]
start_station_name    648717 non-null category
start_station_id      648717 non-null category
end_date              648717 non-null datetime64[ns]
end_station_name      648717 non-null category
end_station_id        648717 non-null category
subscription_type     648717 non-null category
date                  648717 non-null category
day                   648717 non-null category
hour                  648717 non-null category
month                 648717 non-null category
week                  648717 non-null category
holiday               648717 non-null category
workingday            648717 non-null category
dtypes: category(12), datetime64[ns](2), float64(1)
memory usage: 27.9 MB
```

figure3. trip dataset info after wrangling

- ## Station

The station dataset contains information about 70 stations including name, location, dock count, city, and installation date of each station. there are no missing values in this dataset, I just change the columns with ‘object’ data type to ‘category’ and convert installation date to data time format.

	id	name	lat	long	dock_count	city	installation_date
0	2	San Jose Diridon Caltrain Station	37.329732	-121.901782	27	San Jose	8/6/2013
1	3	San Jose Civic Center	37.330698	-121.888979	15	San Jose	8/5/2013
2	4	Santa Clara at Almaden	37.333988	-121.894902	11	San Jose	8/6/2013
3	5	Adobe on Almaden	37.331415	-121.893200	19	San Jose	8/5/2013
4	6	San Pedro Square	37.336721	-121.894074	15	San Jose	8/7/2013

figure 4. station data frame

- ## Weather

The weather dataset is zip-level daily weather patterns for the SF Bay Area: San Francisco, Redwood City, Palo Alto, Mountain View, San Jose. There are almost missing values in every column.

```
date                                 0
max_temperature_f                    4
mean_temperature_f                   4
min_temperature_f                    4
max_dew_point_f                     54
mean_dew_point_f                    54
min_dew_point_f                     54
max_humidity                        54
mean_humidity                       54
min_humidity                        54
max_sea_level_pressure_inches        1
mean_sea_level_pressure_inches       1
min_sea_level_pressure_inches        1
max_visibility_miles                13
mean_visibility_miles               13
min_visibility_miles                13
max_wind_Speed_mph                   1
mean_wind_speed_mph                  1
max_gust_speed_mph                 899
precipitation_inches                 1
cloud_cover                          1
events                            3143
wind_dir_degrees                     1
zip_code                             0
dtype: int64
```

figure 5. missing value in the weather data frame

The events column has the most missing values. It categorizes the weather type in fog, fog-rain, rain, rain- thunderstorms. First, I assume that the NA value represents the normal weather. I fill the nan value with ‘normal’ and then level them up based on wind speed and visibility miles. As the picture shows below, the Normal weather has the highest visibility and acceptable wind speed, I will level it as 0. Ran and Rain_Thunderstorm almost has the same performance. It is so hard to tell which weather will affect the trip counts most: Fog or Rain based on this data. So I would like to temperature level Rain as level 1, Rain-Thunderstorm as level 2, Fog as level 3 and Fog-Rain as level 4 and check the details in EDA. For reference, I use numerical data to level up events only because I want to fill null values for weather dataset easily. In further machine learning, I will use dummy variables instead of this order for model prediction.

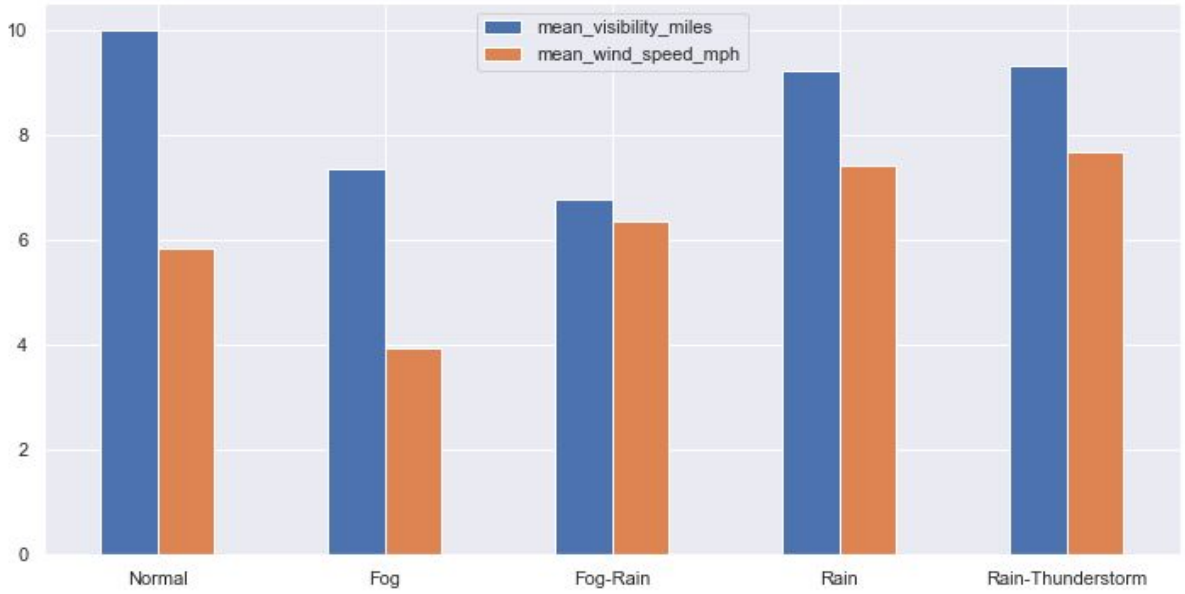


figure 6. weather events on visibility and wind speed

After exploring the dataset make sure there are no unacceptable outliers, I convert objects columns to numeric values and fill nan values with its own average. And finally, map the 5 zip codes with the corresponding city name, change column datatype, adding a new column ‘week’ and drop ’zip_code’ column.

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3665 entries, 2013-08-29 to 2015-08-31
Data columns (total 23 columns):
max_temperature_f                  3665 non-null float64
mean_temperature_f                 3665 non-null float64
min_temperature_f                  3665 non-null float64
max_dew_point_f                    3665 non-null float64
mean_dew_point_f                   3665 non-null float64
min_dew_point_f                    3665 non-null float64
max_humidity                       3665 non-null float64
mean_humidity                      3665 non-null float64
min_humidity                       3665 non-null float64
max_sea_level_pressure_inches      3665 non-null float64
mean_sea_level_pressure_inches     3665 non-null float64
min_sea_level_pressure_inches      3665 non-null float64
max_visibility_miles               3665 non-null float64
mean_visibility_miles              3665 non-null float64
min_visibility_miles               3665 non-null float64
max_wind_Speed_mph                 3665 non-null float64
mean_wind_speed_mph                3665 non-null float64
max_gust_speed_mph                 3665 non-null float64
precipitation_inches               3665 non-null float64
cloud_cover                        3665 non-null float64
events                             3665 non-null int64
wind_dir_degrees                   3665 non-null float64
zip_code                           3665 non-null int64
dtypes: float64(21), int64(2)
memory usage: 687.2 KB
```

figure 7. weather dataset info after wrangling

- ## Merging Dataset

After I clean up all the datasets, next step, I need to merge them together for machine learning purpose. from the trip data frame, group by date,station_name, holiday, weekday to get the trip_count on each day by each station, left merge with station dataframe on station_name column.

	date	station_name	holiday	workingday	trip_count	dock_count	city
0	2013-08-29	2nd at Folsom	False	True	12	19.0	San Francisco
1	2013-08-29	2nd at South Park	False	True	11	15.0	San Francisco
2	2013-08-29	2nd at Townsend	False	True	8	27.0	San Francisco
3	2013-08-29	5th at Howard	False	True	12	15.0	San Francisco
4	2013-08-29	Adobe on Almaden	False	True	3	19.0	San Jose

figure 8. merging station and trip data frame

The new data frame has 1150 missing values in the city and dock_count columns, it is because there are 4 stations in the trip dataset not included in the station dataset. Because there is no significant difference in dock_count between each city, I will fill the city column based on the location of station_name and filling the dock_count column with the overall dock mean value.

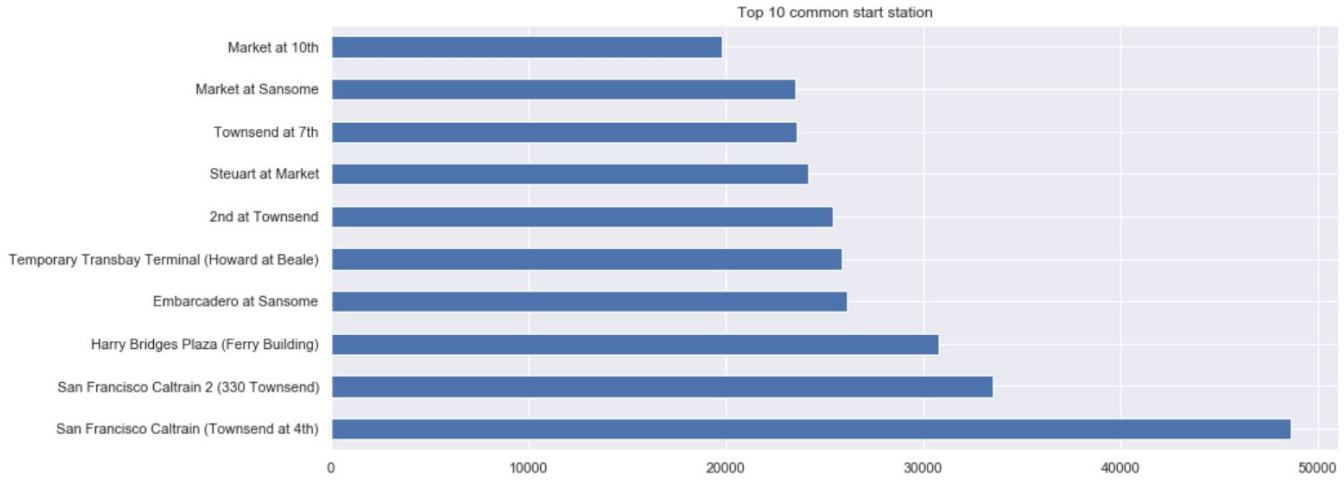
after fixing the missing value, I will keep merging the data frame with the weather on date and city columns. There are no missing values after merging, just need to change columns from object to category data type.

Until now, I've got 3 well-cleaned data frames and a combined data frame for EDA and Machine learning purposes.

Exploratory Data Analysis

- ## Most Commonly Used Stations

There is a total of 70 stations listed in this dataset. My main target in this project is to predict the daily trips for an individual station. So the first thing I need to do is to check out which station is the busiest station in SF. I check the top 10 commonly used stations based on the start-station and end-station and find the station named: San Francisco Caltrain(Townsend at 4th) is the most popular station. I will choose this station for my machine learning.



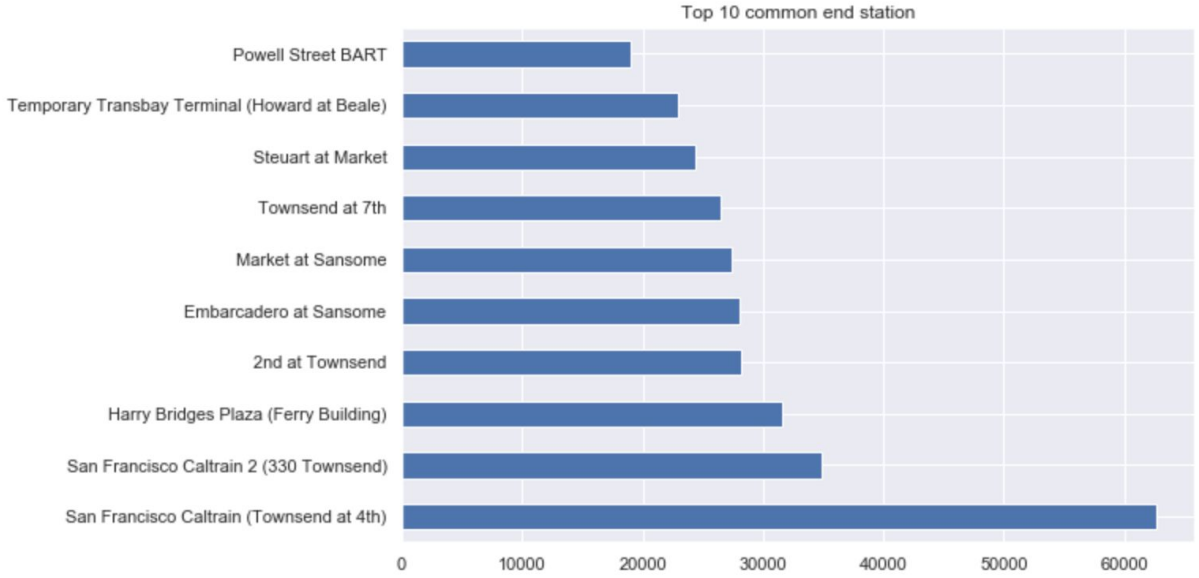


figure 9. most commonly unused station in SF

- ## Total Trip Counts Trend by Subscription Type

After grouping riders by subscription type (Subscriber: monthly or annual member, and Customer: casual pay-as-you-go rider), I explored seasonal, weekly, and daily ridership patterns. All timeframes highlight distinct patterns for Subscribers vs. Customers, supporting the hypothesis that Subscribers are commuters and Customers are largely tourists.Subscriber ridership has a large drop on holidays and weekends,

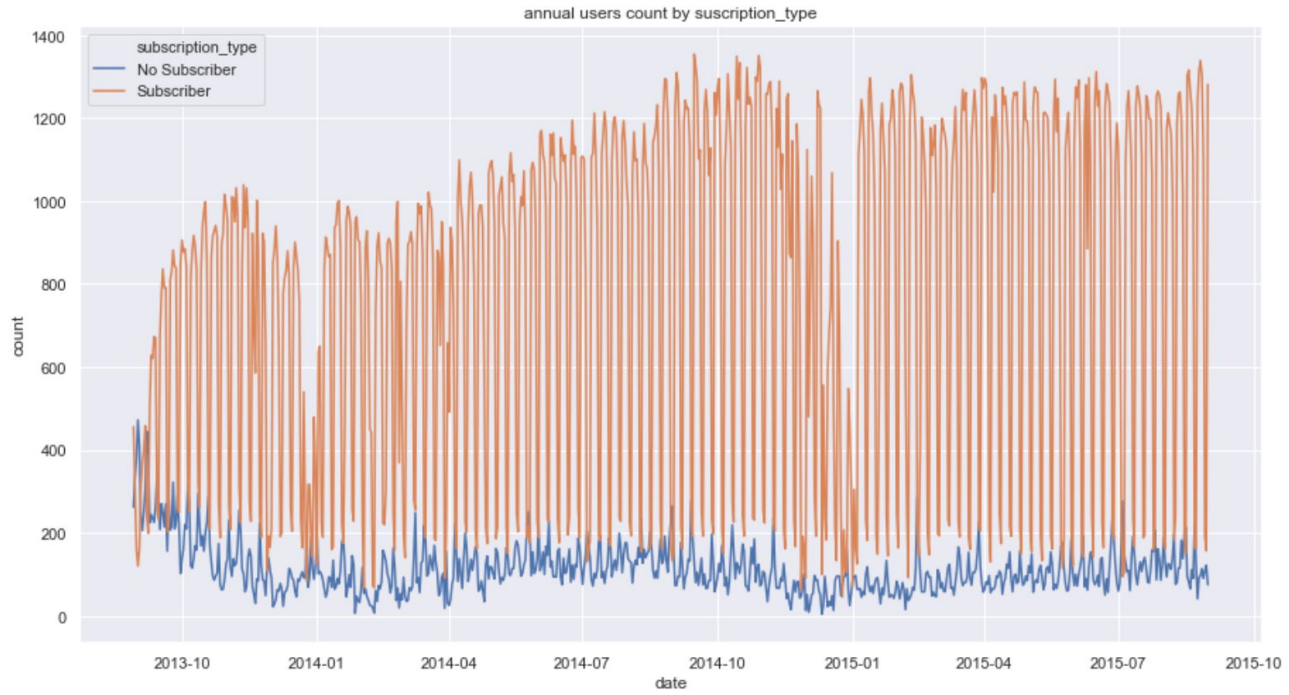


figure 10.overall trips count by subscripthin_type

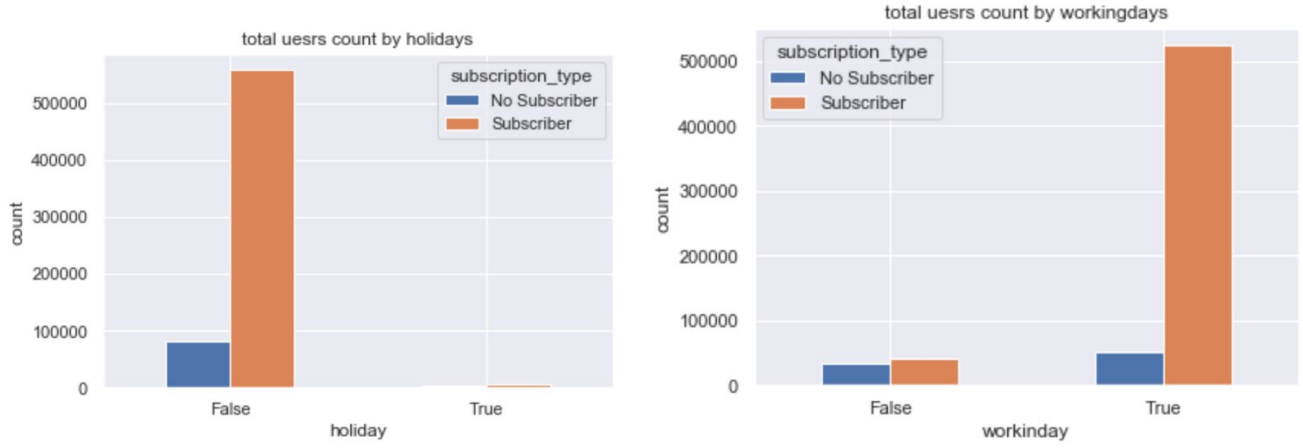


figure 11. trip counts on weekday/holiday by suscription_type

- ## Trip Counts Trend by Month/Day/Hour

Ridership increases slightly in the first half of the year, peaking in summer (week 34/35) in ideal bike weather. rides start to drop off in the Fall.Because the weather in SF is very warm, the trip counts by month has no big drop in wintertime. I believe the slight drop in December is due to holiday reasons.

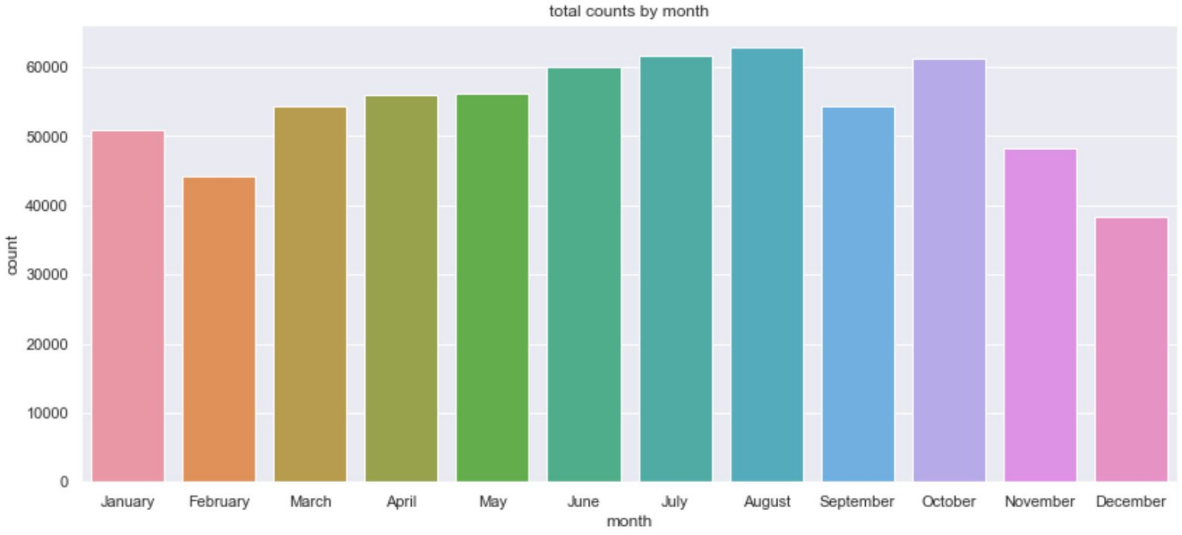


figure 12. trip counts by month

Subscriber show a distinct bimodal daily usage pattern, with peaks during AM & PM commute hours, and a small increase during lunch. Customers gets a later start (as one does on vacation), with steady usage 11am until the early evening for daily usage.The slight drop on Friday's peak hours might due to some personal choice to work at home on that day.

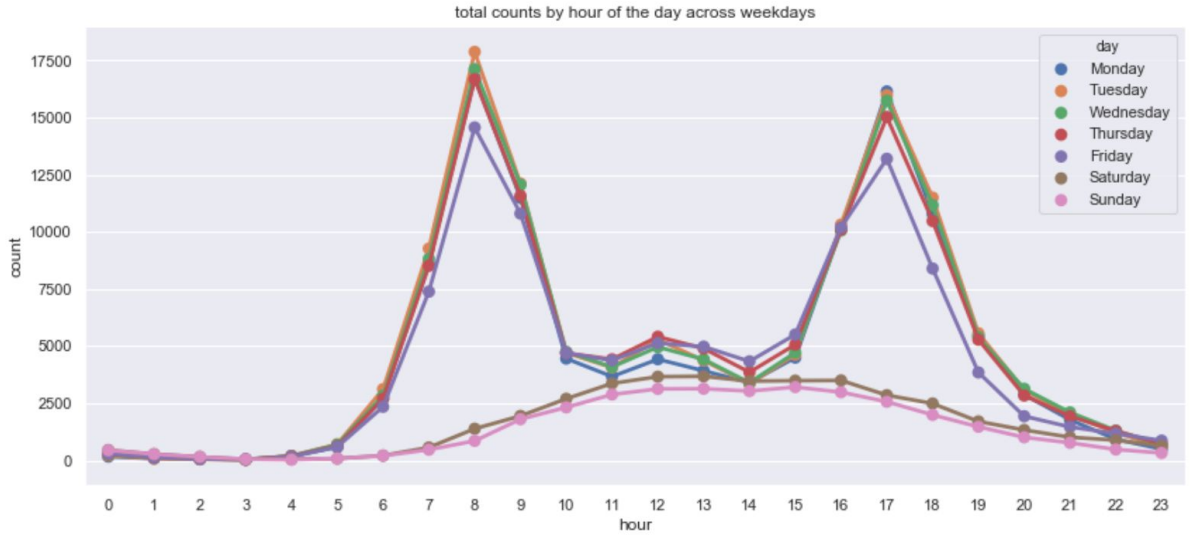


figure 13. trip counts of the day across weekdays

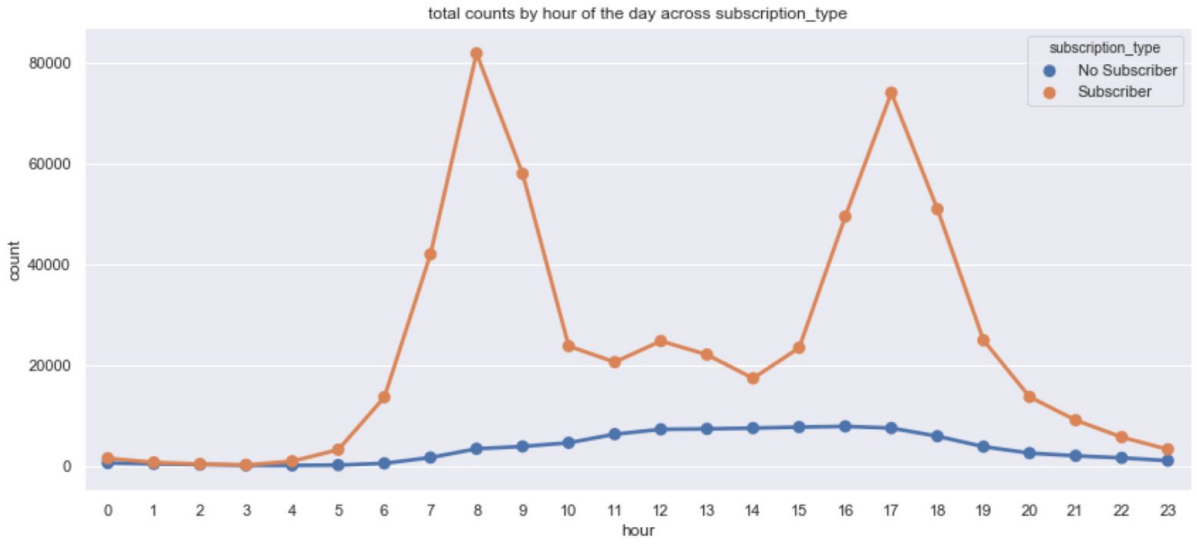


figure 14. trip counts of the day across subscription_type

- Trip counts trend by weather features

There is no clear relationship between each weather features and daily trips. obviously people are willing to use a bike rental service on a normal day rather than fog or raining day. The trip counts have no significant difference between fog and rain but we can see there is more fog weather in wintertime them summer.

figure 15. trip counts trend by week across events

Looking into deeper on only SF city with working day, we can see the Higher the temperature, the more people choose to bike to work until the temperature reach to 65 Fahrenheit, the number start to drop. Same pattern as wind speed, when the wind speed increases more than 10 mph (miles per hour), the trip count starts to drop. And visibility seems not a key feature for trip counts.

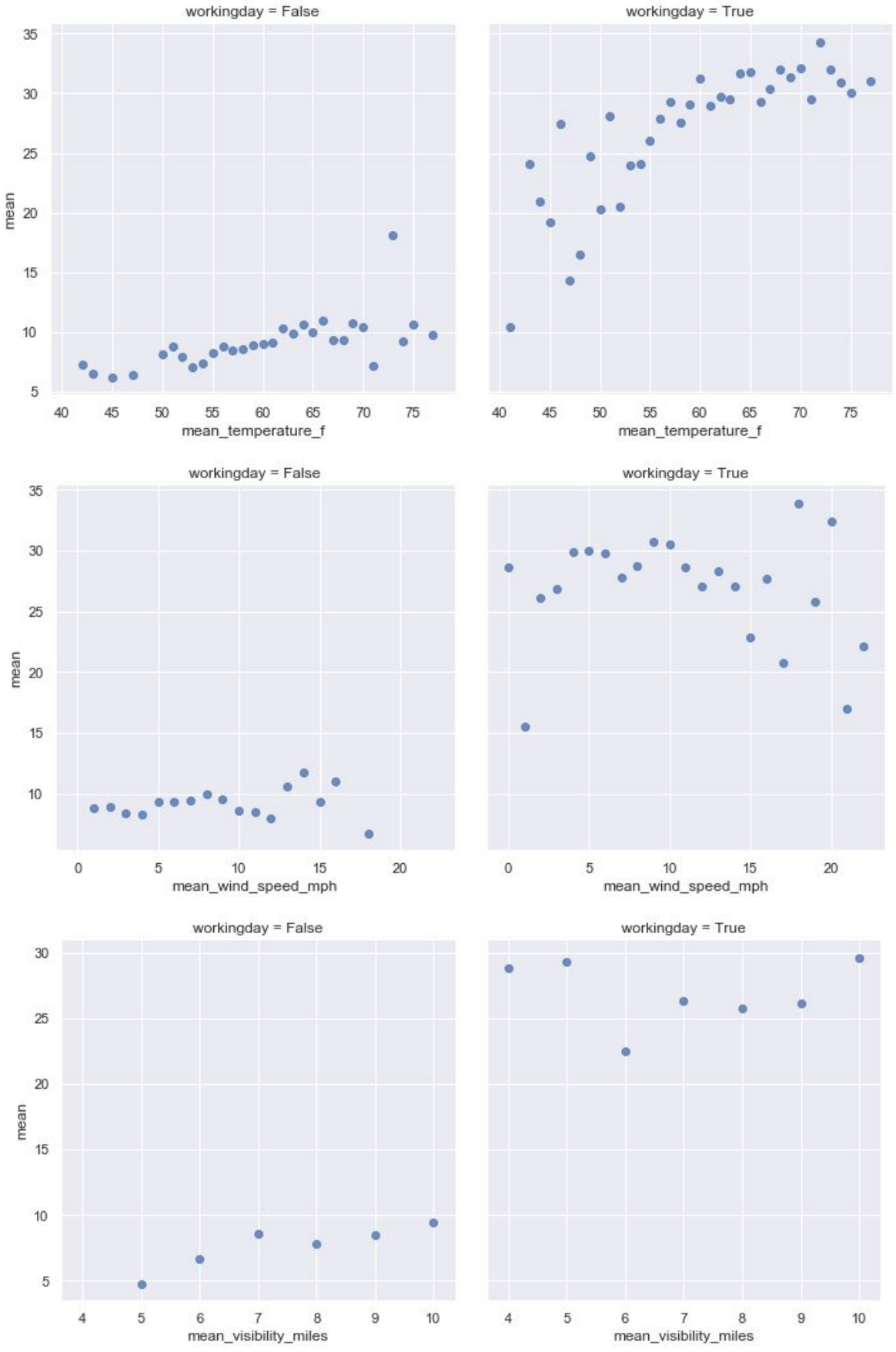


figure 16. trip counts mean vs. temperature,wind_speed, and visibility

Machine Learning

- ## Trip Counts Predictive Model

Now that we've understood the rough trends of data and user behavior, the next step is to build a model for predicting bike trip counts per day per station. Bike trip counts is a continuous quantity values, thus a regression model is appropriate.

In this project, I aim to build a regression model to predict daily trips on a target station based on the weather, dock count, week, holiday and working day. I choose the most popular station: San Francisco Caltrain (Townsend at 4th) as my target station. Then get the dummy variables on the events column, finally, because I already choose the particular station and get all the time-series data from the date column, I will drop the city, station name and date column, set trip counts as my test value,the rest of the data as train values, and set the train, test split ready for modeling.

I create two functions at the beginning: Grid-search function to find the optimal hyperparameters of a model which results in the most 'accurate' predictions. And regressor metrics function which I will check the mean absolute error of each model using the best hyperparameters I have found.

I try the grid search and MAE (mean absolute error) on 5 different regression models: Random Forest, Decision Tree, Linear Regression, Gradient Boosting, and Adaboost. random forest regressor which has the smallest mean absolute error of 11.69.

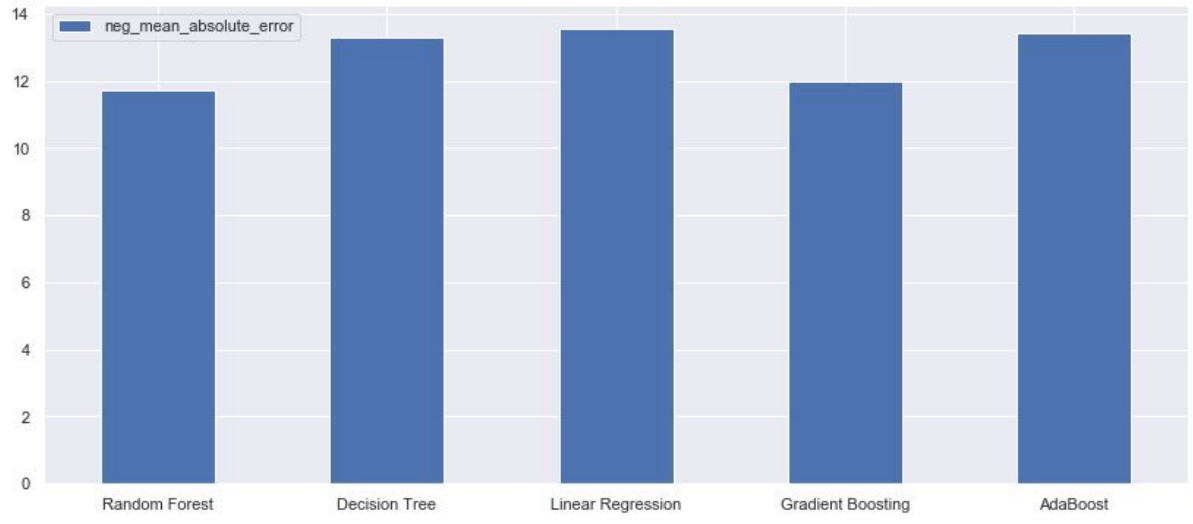


figure 17. mean_absolute_error on different predictive models

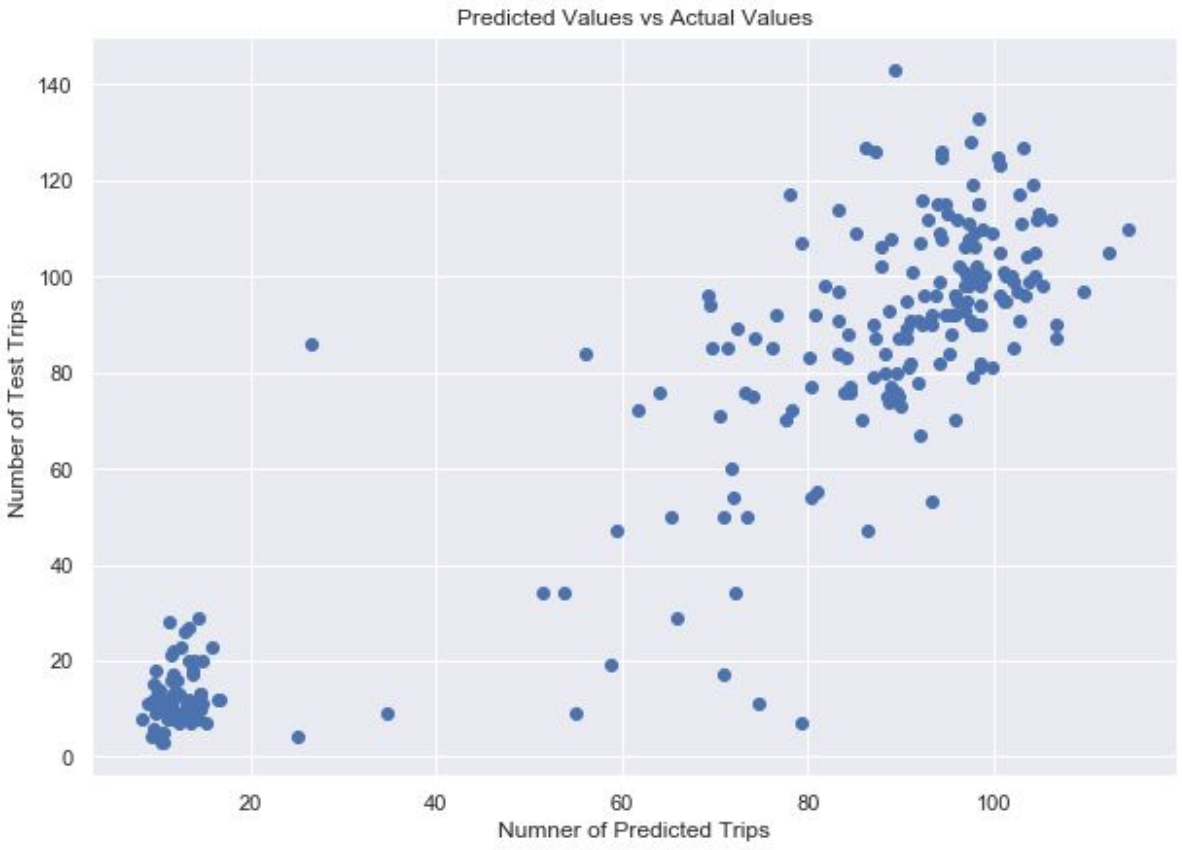


figure 18. predicted value vs. actual value of daily trips

- Feature Engineering

As I check the key features of my prediction model, it does not surprise me that the working day has a significant impact on daily trips based on my Exploratory data analysis. Holidays, week of the year also have some impact on predict daily trips then weather.

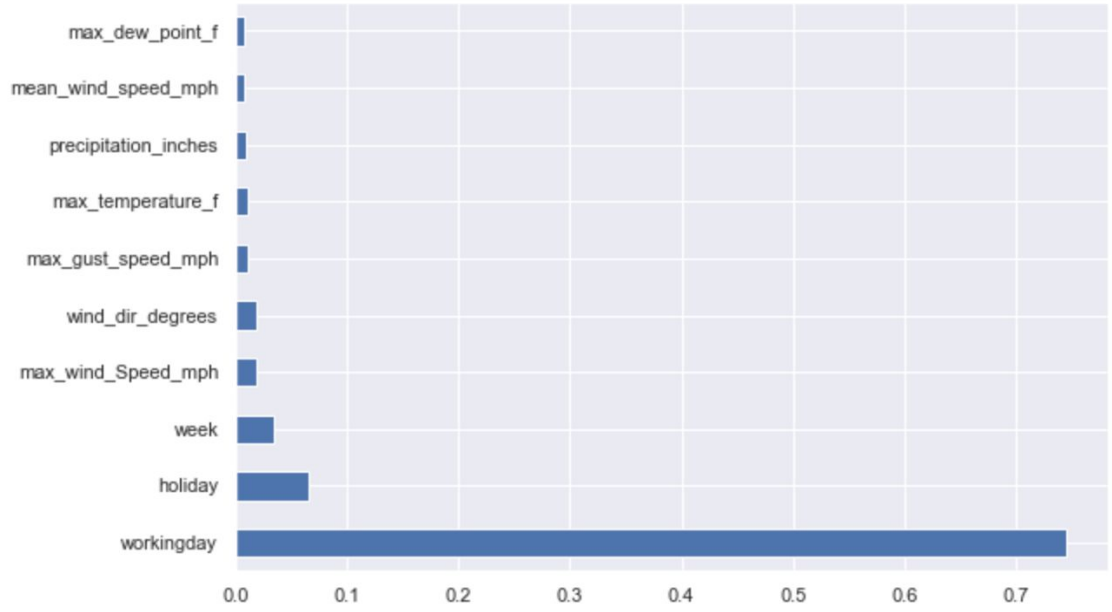


figure 19. Importance Features

Conclusion

I believe I have made a good model to predict daily trips that will take with the bike-sharing service at a particular station. The mean absolute error of my model is about 11 times a day. The company could use my model to predict any station usage they are looking for and get a good rough estimate of the dock account they should keep for each day.

To make the model more accurate and usable, The next step I will focus on the larger number of trips, especially between 60-80 trips per day, try to figure out what features affect the numbers and how to optimize the model to make better predictions on those particular days.