bike_sharing_report

February 27, 2020

In this report, we go through the analysis and modeling of the San Francisco bike share data. The goal is to provide a model that predicts the net change of bikes in each station for the next hour.

1 Data Exploration

The first part is to import the data and explore them. That means understand the data, clean then, find missing values, and understand which one could be important for our case.

[43]:
```
import pandas as pd
from pandas.tseries.holiday import USFederalHolidayCalendar
from pandas.tseries.offsets import CustomBusinessDay
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

[45]:
```
# I ignore the warning so that the report is easier to read
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
trips = pd.read_csv("../data/trip_data.csv",parse_dates=[1,3],dayfirst=True)
weather = pd.read_csv("../data/weather_data.csv",parse_dates=[0],dayfirst=True)
stations = pd.read_csv("../data/station_data.csv")
```

[4]:
```
trips.head(5)
```

[4]:
```
   Trip ID          Start Date  Start Station            End Date  \
0   913460 2015-08-31 23:26:00             50 2015-08-31 23:39:00
1   913459 2015-08-31 23:11:00             31 2015-08-31 23:28:00
2   913455 2015-08-31 23:13:00             47 2015-08-31 23:18:00
3   913454 2015-08-31 23:10:00             10 2015-08-31 23:17:00
4   913453 2015-08-31 23:09:00             51 2015-08-31 23:22:00

   End Station Subscriber Type
0           70      Subscriber
1           27      Subscriber
2           64      Subscriber
3            8      Subscriber
```

```
4           60        Customer
```

[5]:
```
trips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 354152 entries, 0 to 354151
Data columns (total 6 columns):
Trip ID            354152 non-null int64
Start Date         354152 non-null datetime64[ns]
Start Station      354152 non-null int64
End Date           354152 non-null datetime64[ns]
End Station        354152 non-null int64
Subscriber Type    354152 non-null object
dtypes: datetime64[ns](2), int64(3), object(1)
memory usage: 16.2+ MB
```

Explore if missing values are contained in the trips dataframe

[6]:
```
trips.isnull().sum()
```

[6]:
```
Trip ID            0
Start Date         0
Start Station      0
End Date           0
End Station        0
Subscriber Type    0
dtype: int64
```

No missing values are contained. Moreover thinking about our end goal the idea would be to collect from the trips dataframe how many trips started and ended in each station for each hour.

[7]:
```
weather.head(5)
```

[7]:
```
        Date  Max TemperatureF  Mean TemperatureF  Min TemperatureF  \
0 2014-09-01              83.0               70.0              57.0
1 2014-09-02              72.0               66.0              60.0
2 2014-09-03              76.0               69.0              61.0
3 2014-09-04              74.0               68.0              61.0
4 2014-09-05              72.0               66.0              60.0

   Max Dew PointF  MeanDew PointF  Min DewpointF  Max Humidity  Mean Humidity  \
0            58.0            56.0           52.0          86.0           64.0
1            58.0            57.0           55.0          84.0           73.0
2            57.0            56.0           55.0          84.0           69.0
3            57.0            57.0           56.0          84.0           71.0
4            57.0            56.0           54.0          84.0           71.0

   Min Humidity  …  Mean VisibilityMiles  Min VisibilityMiles  \
```

```
0          42.0  ...                10.0                    8.0
1          61.0  ...                10.0                    7.0
2          53.0  ...                10.0                   10.0
3          57.0  ...                10.0                    8.0
4          57.0  ...                 9.0                    7.0

   Max Wind SpeedMPH  Mean Wind SpeedMPH  Max Gust SpeedMPH  PrecipitationIn  \
0               16.0                 7.0               20.0              0.0
1               21.0                 8.0                NaN              0.0
2               21.0                 8.0               24.0              0.0
3               22.0                 8.0               25.0              0.0
4               18.0                 8.0               32.0              0.0

   CloudCover  Events  WindDirDegrees    Zip
0         0.0     NaN           290.0  94107
1         5.0     NaN           290.0  94107
2         4.0     NaN           276.0  94107
3         5.0     NaN           301.0  94107
4         4.0     NaN           309.0  94107

[5 rows x 24 columns]
```

[8]:
```
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1825 entries, 0 to 1824
Data columns (total 24 columns):
Date                         1825 non-null datetime64[ns]
Max TemperatureF             1821 non-null float64
Mean TemperatureF            1821 non-null float64
Min TemperatureF             1821 non-null float64
Max Dew PointF               1775 non-null float64
MeanDew PointF               1775 non-null float64
Min DewpointF                1775 non-null float64
Max Humidity                 1775 non-null float64
Mean Humidity                1775 non-null float64
Min Humidity                 1775 non-null float64
Max Sea Level PressureIn     1824 non-null float64
Mean Sea Level PressureIn    1824 non-null float64
Min Sea Level PressureIn     1824 non-null float64
Max VisibilityMiles          1820 non-null float64
Mean VisibilityMiles         1820 non-null float64
Min VisibilityMiles          1820 non-null float64
Max Wind SpeedMPH            1824 non-null float64
Mean Wind SpeedMPH           1824 non-null float64
Max Gust SpeedMPH            1284 non-null float64
PrecipitationIn              1824 non-null float64
```

```
CloudCover                    1824 non-null float64
Events                        287 non-null object
WindDirDegrees                1824 non-null float64
Zip                           1825 non-null int64
dtypes: datetime64[ns](1), float64(21), int64(1), object(1)
memory usage: 342.3+ KB
```

We can see that the weather dataframe contains a lot of weather information. One important remark is that the weather info are daily and not hourly,as we want our predictive model to be.

[9]:
```python
weather.Zip.unique()
```

[9]:
```
array([94107, 94063, 94301, 94041, 95113])
```

We have 365 uniques dates in our date range and the weather data contain 1825 entries. That means that we have weather data for each one of the zip locations every day.

[10]:
```python
weather.isnull().sum()
```

[10]:
```
Date                         0
Max TemperatureF             4
Mean TemperatureF            4
Min TemperatureF             4
Max Dew PointF              50
MeanDew PointF              50
Min DewpointF               50
Max Humidity                50
Mean Humidity               50
Min Humidity                50
Max Sea Level PressureIn     1
Mean Sea Level PressureIn    1
Min Sea Level PressureIn     1
Max VisibilityMiles          5
Mean VisibilityMiles         5
Min VisibilityMiles          5
Max Wind SpeedMPH            1
Mean Wind SpeedMPH           1
Max Gust SpeedMPH          541
PrecipitationIn              1
CloudCover                   1
Events                    1538
WindDirDegrees               1
Zip                          0
dtype: int64
```

[11]:
```python
weather.Events.unique()
```

[11]:
```
array([nan, 'Rain', 'Fog', 'Fog-Rain', 'Rain-Thunderstorm'], dtype=object)
```

We can see that Events column contains multiple nan values, but it can be safely assumed that no event happened in that day, thus it was a day with a normal weather.

[12]:
```
# The empty events data are changed to Normal and then we convert the␣
↪categorical values to indicator values
weather.loc[weather.Events.isnull(), 'Events'] = "Normal"
#events = pd.get_dummies(weather["Events"])
```

[13]:
```
weather
```

[13]:
```
           Date  Max TemperatureF  Mean TemperatureF  Min TemperatureF  \
0    2014-09-01              83.0               70.0              57.0
1    2014-09-02              72.0               66.0              60.0
2    2014-09-03              76.0               69.0              61.0
3    2014-09-04              74.0               68.0              61.0
4    2014-09-05              72.0               66.0              60.0
…           …                 …                  …                 …
1820 2015-08-27              92.0               78.0              63.0
1821 2015-08-28              95.0               80.0              64.0
1822 2015-08-29              80.0               72.0              64.0
1823 2015-08-30              78.0               70.0              62.0
1824 2015-08-31              85.0               72.0              59.0

      Max Dew PointF  MeanDew PointF  Min DewpointF  Max Humidity  \
0               58.0            56.0           52.0          86.0
1               58.0            57.0           55.0          84.0
2               57.0            56.0           55.0          84.0
3               57.0            57.0           56.0          84.0
4               57.0            56.0           54.0          84.0
…                …               …              …            …
1820            57.0            51.0           40.0          78.0
1821            64.0            56.0           52.0          93.0
1822            65.0            62.0           54.0          93.0
1823            60.0            57.0           53.0          84.0
1824            59.0            55.0           51.0          84.0

      Mean Humidity  Min Humidity  …  Mean VisibilityMiles  \
0              64.0          42.0  …                  10.0
1              73.0          61.0  …                  10.0
2              69.0          53.0  …                  10.0
3              71.0          57.0  …                  10.0
4              71.0          57.0  …                   9.0
…               …            …  …                   …
1820           48.0          18.0  …                  10.0
1821           60.0          26.0  …                  10.0
1822           70.0          47.0  …                  10.0
1823           64.0          43.0  …                  10.0
```

```
1824           58.0          32.0  …                  10.0

      Min VisibilityMiles  Max Wind SpeedMPH  Mean Wind SpeedMPH  \
0                     8.0               16.0                 7.0
1                     7.0               21.0                 8.0
2                    10.0               21.0                 8.0
3                     8.0               22.0                 8.0
4                     7.0               18.0                 8.0
…                    …                 …                  …
1820                 10.0               23.0                 6.0
1821                 10.0               25.0                 7.0
1822                 10.0               21.0                 9.0
1823                 10.0               22.0                10.0
1824                 10.0               20.0                 6.0

      Max Gust SpeedMPH  PrecipitationIn  CloudCover  Events  WindDirDegrees  \
0                  20.0              0.0         0.0  Normal           290.0
1                   NaN              0.0         5.0  Normal           290.0
2                  24.0              0.0         4.0  Normal           276.0
3                  25.0              0.0         5.0  Normal           301.0
4                  32.0              0.0         4.0  Normal           309.0
…                 …               …         …      …           …
1820               29.0              0.0         3.0  Normal           313.0
1821               30.0              0.0         3.0  Normal           307.0
1822               26.0              0.0         4.0  Normal           312.0
1823               29.0              0.0         3.0  Normal           291.0
1824               24.0              0.0         1.0  Normal           308.0

        Zip
0     94107
1     94107
2     94107
3     94107
4     94107
…     …
1820  95113
1821  95113
1822  95113
1823  95113
1824  95113

[1825 rows x 24 columns]
```

Since the missing values are float we can fill them with the mean values of each respective column

[14]:
```
weather.fillna(weather.mean(), inplace=True)
```

[15]:
```
## We can check now the missing values again
weather.isnull().sum()
```

[15]:
```
Date                         0
Max TemperatureF             0
Mean TemperatureF            0
Min TemperatureF             0
Max Dew PointF               0
MeanDew PointF               0
Min DewpointF                0
Max Humidity                 0
Mean Humidity                0
Min Humidity                 0
Max Sea Level PressureIn     0
Mean Sea Level PressureIn    0
Min Sea Level PressureIn     0
Max VisibilityMiles          0
Mean VisibilityMiles         0
Min VisibilityMiles          0
Max Wind SpeedMPH            0
Mean Wind SpeedMPH           0
Max Gust SpeedMPH            0
PrecipitationIn              0
CloudCover                   0
Events                       0
WindDirDegrees               0
Zip                          0
dtype: int64
```

Now we move the the stations exploration.

[16]:
```
stations.head(5)
```

[16]:
```
   Id                                 Name        Lat        Long  Dock Count  \
0   2  San Jose Diridon Caltrain Station  37.329732 -121.901782          27
1   3              San Jose Civic Center  37.330698 -121.888979          15
2   4             Santa Clara at Almaden  37.333988 -121.894902          11
3   5                   Adobe on Almaden  37.331415 -121.893200          19
4   6                   San Pedro Square  37.336721 -121.894074          15

       City
0  San Jose
1  San Jose
2  San Jose
3  San Jose
4  San Jose
```

[17]:
```
stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76 entries, 0 to 75
Data columns (total 6 columns):
Id            76 non-null int64
Name          76 non-null object
Lat           76 non-null float64
Long          76 non-null float64
Dock Count    76 non-null int64
City          76 non-null object
dtypes: float64(2), int64(2), object(2)
memory usage: 3.7+ KB
```

[18]:
```
stations.isnull().sum()
```

[18]:
```
Id            0
Name          0
Lat           0
Long          0
Dock Count    0
City          0
dtype: int64
```

Some of the stations are moved and changed, however there is the information missing about the installation time of each station, thus we consider each station independant.

We can proceed now with starting to combining the dataframes and generating the train data.

The idea is to create a dataframe for the date range 01/09/2014 until 31/08/2015 with one row for each hour of the day. We should do the same for each station ID.

[46]:
```
dates = pd.date_range(start="9/1/2014",end="9/1/2015",freq="H")
weekdays = dates.weekday
only_date = dates.date

#Find business days and holidays
calendar = USFederalHolidayCalendar()
holidays = calendar.holidays(start=dates.date.min(), end=dates.date.max())

us_bd = CustomBusinessDay(calendar=USFederalHolidayCalendar())
business_days = pd.DatetimeIndex(start=dates.date.min(), end=dates.date.max(),␣
 ↪freq=us_bd)

business_days = pd.to_datetime(business_days, format='%d/%m/%Y').date
holidays = pd.to_datetime(holidays, format='%d/%m/%Y').date
```

```
date_df = pd.DataFrame(dates,columns=["Time"])
date_df["Weekday"] = weekdays
date_df["Date"] = only_date

# Add the business and holidays

date_df['Business_day'] = date_df['Date'].isin(business_days)
date_df['Holiday'] = date_df['Date'].isin(holidays)

date_df.Business_day = date_df.Business_day.map(lambda x: 1 if x == True else 0)
date_df.Holiday = date_df.Holiday.map(lambda x: 1 if x == True else 0)

# Change format
dates = pd.to_datetime(dates).strftime('%d/%m/%Y %H')
only_date = pd.to_datetime(only_date)#.strftime('%d/%m/%Y')
# Replace with correct format
date_df["Date"] = only_date
date_df["Time"] = dates
```

[20]:

```
date_df
```

[20]:

```
                Time  Weekday       Date  Business_day  Holiday
0      01/09/2014 00        0 2014-09-01             0        1
1      01/09/2014 01        0 2014-09-01             0        1
2      01/09/2014 02        0 2014-09-01             0        1
3      01/09/2014 03        0 2014-09-01             0        1
4      01/09/2014 04        0 2014-09-01             0        1
…                …       …        …           …       …
8756   31/08/2015 20        0 2015-08-31             1        0
8757   31/08/2015 21        0 2015-08-31             1        0
8758   31/08/2015 22        0 2015-08-31             1        0
8759   31/08/2015 23        0 2015-08-31             1        0
8760   01/09/2015 00        1 2015-09-01             1        0

[8761 rows x 5 columns]
```

We create a date dataframe containing each day and hour from 01/09/2014 00 till 01/09/2015 00. The weekdays column is also added, showing 0–>Monday and 6–> Sunday. Also Business day is added and Holiday columns as features.

Next part would be to match the location the stations with the zip location of the weather data. The matching is the following: {94107: 'San Francisco', 94063: 'Redwood City', 94301: 'Palo Alto', 94041: 'Mountain View', 95113: 'San Jose'}

[21]:

```
stations["Zip"] = np.nan
stations.loc[stations['City'] == 'San Jose','Zip'] = 95113
stations.loc[stations['City'] == 'Redwood City','Zip'] = 94063
stations.loc[stations['City'] == 'San Francisco','Zip'] = 94107
```

```
stations.loc[stations['City'] == 'Palo Alto','Zip'] = 94301
stations.loc[stations['City'] == 'Mountain View','Zip'] = 94041
```

[22]:
```
stations
```

[22]:
```
    Id                             Name        Lat        Long  Dock Count  \
0    2  San Jose Diridon Caltrain Station  37.329732 -121.901782          27
1    3              San Jose Civic Center  37.330698 -121.888979          15
2    4             Santa Clara at Almaden  37.333988 -121.894902          11
3    5                   Adobe on Almaden  37.331415 -121.893200          19
4    6                   San Pedro Square  37.336721 -121.894074          15
..  ..                                ...        ...         ...         ...
71  77                  Market at Sansome  37.789625 -122.400811          27
72  80      Santa Clara County Civic Center  37.352601 -121.905733          15
73  82            Broadway St at Battery St  37.798541 -122.400862          15
74  83                         Mezes Park  37.491269 -122.236234          15
75  84                        Ryland Park  37.342725 -121.895617          15

             City      Zip
0        San Jose  95113.0
1        San Jose  95113.0
2        San Jose  95113.0
3        San Jose  95113.0
4        San Jose  95113.0
..            ...      ...
71  San Francisco  94107.0
72       San Jose  95113.0
73  San Francisco  94107.0
74   Redwood City  94063.0
75       San Jose  95113.0

[76 rows x 7 columns]
```

[23]:
```
# Here all the dates for each one of the stations is concatenated to create the␣
↪complete 1year24hour representation for
# each one of the stations

dates_df = pd.concat([date_df]*len(stations.index), ignore_index=True)
## List of indices and list of stations
loi = np.arange(0,674597,8761)

station_ids = []
dock_count = []
zip_loc = []
for _,row in stations.iterrows():

    station_ids.append(row["Id"])
```

```
    dock_count.append(row["Dock Count"])
    zip_loc.append(row["Zip"])

for i in range(len(loi)-1):
    dates_df.loc[loi[i]:loi[i+1]-1, "Station_ID"] = station_ids[i]
    dates_df.loc[loi[i]:loi[i+1]-1, "Dock_count"] = dock_count[i]
    dates_df.loc[loi[i]:loi[i+1]-1, "Zip"] = zip_loc[i]
```

[24]:
```
dates_df
```

[24]:
```
                 Time  Weekday        Date  Business_day  Holiday  Station_ID  \
0       01/09/2014 00        0  2014-09-01             0        1         2.0
1       01/09/2014 01        0  2014-09-01             0        1         2.0
2       01/09/2014 02        0  2014-09-01             0        1         2.0
3       01/09/2014 03        0  2014-09-01             0        1         2.0
4       01/09/2014 04        0  2014-09-01             0        1         2.0
…                 …        …           …             …        …           …
665831  31/08/2015 20        0  2015-08-31             1        0        84.0
665832  31/08/2015 21        0  2015-08-31             1        0        84.0
665833  31/08/2015 22        0  2015-08-31             1        0        84.0
665834  31/08/2015 23        0  2015-08-31             1        0        84.0
665835  01/09/2015 00        1  2015-09-01             1        0        84.0

        Dock_count      Zip
0             27.0  95113.0
1             27.0  95113.0
2             27.0  95113.0
3             27.0  95113.0
4             27.0  95113.0
…              …        …
665831        15.0  95113.0
665832        15.0  95113.0
665833        15.0  95113.0
665834        15.0  95113.0
665835        15.0  95113.0

[665836 rows x 8 columns]
```

Next part which is important for our case is to count how many trips started and ended for each station every hour, e.g give me the trips started at station 2 at 01/09/2014 from 1:00 till 1:59. Then it would be possible to find the net change at this specific station for the following hour by subtracting (trips ended - trips started)

[25]:
```
# I take the starting datetime and split it to date and time
trips_dates = pd.to_datetime(trips["Start Date"])
#start_dates = trips_dates.apply(lambda x: x.strftime('%d/%m/%Y %H'))
trips["New_dates"] = pd.to_datetime(trips_dates).dt.date
```

```
trips["Hours"] = pd.to_datetime(trips_dates).dt.hour

# I groupby the trips together with the dates and hours and count how many␣
↪trips started at each station

start = trips["Start Station"].groupby([trips['New_dates'],trips['Hours']]).
↪value_counts().to_frame()
start['Date'] = start.index.get_level_values('New_dates')
start['Hours'] = start.index.get_level_values('Hours')
start['Station_ID'] = start.index.get_level_values('Start Station')
start.columns = ['Start_CountTrips', 'Date','Hours','Station_ID']
start.reset_index(drop=True,inplace=True)
start["Date"] = pd.to_datetime(start.Date)
```

[26]:
```
start
```

[26]:
```
        Start_CountTrips       Date  Hours  Station_ID
0                      3 2014-09-01      0          66
1                      1 2014-09-01      3          50
2                      1 2014-09-01      4          39
3                      1 2014-09-01      4          66
4                      1 2014-09-01      5          68
…                    …          …    …           …
142510                 1 2015-08-31     23          10
142511                 1 2015-08-31     23          31
142512                 1 2015-08-31     23          47
142513                 1 2015-08-31     23          50
142514                 1 2015-08-31     23          68

[142515 rows x 4 columns]
```

[27]:
```
start.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142515 entries, 0 to 142514
Data columns (total 4 columns):
Start_CountTrips    142515 non-null int64
Date                142515 non-null datetime64[ns]
Hours               142515 non-null int64
Station_ID          142515 non-null int64
dtypes: datetime64[ns](1), int64(3)
memory usage: 4.3 MB
```

[28]:
```
dates_df["Hours"] = pd.to_datetime(dates_df.Time).dt.hour
dates_df["Date"] = pd.to_datetime(dates_df.Date)
dates_df
```

[28]:
```
                   Time  Weekday       Date  Business_day  Holiday  Station_ID  \
0       01/09/2014 00        0 2014-09-01             0        1         2.0
1       01/09/2014 01        0 2014-09-01             0        1         2.0
2       01/09/2014 02        0 2014-09-01             0        1         2.0
3       01/09/2014 03        0 2014-09-01             0        1         2.0
4       01/09/2014 04        0 2014-09-01             0        1         2.0
…                 …      …          …           …       …          …
665831  31/08/2015 20        0 2015-08-31             1        0        84.0
665832  31/08/2015 21        0 2015-08-31             1        0        84.0
665833  31/08/2015 22        0 2015-08-31             1        0        84.0
665834  31/08/2015 23        0 2015-08-31             1        0        84.0
665835  01/09/2015 00        1 2015-09-01             1        0        84.0

        Dock_count      Zip  Hours
0             27.0  95113.0      0
1             27.0  95113.0      1
2             27.0  95113.0      2
3             27.0  95113.0      3
4             27.0  95113.0      4
…              …       …     …
665831        15.0  95113.0     20
665832        15.0  95113.0     21
665833        15.0  95113.0     22
665834        15.0  95113.0     23
665835        15.0  95113.0      0

[665836 rows x 9 columns]
```

At this part I will merge the two dataframes, i.e dates_df with the start df I created. The merge will happen using the columns Date,Time and StationID and is outer merge

[29]:
```
merged_start = pd.
 ↪merge(dates_df,start,on=["Date","Hours","Station_ID"],how="outer")
merged_start['Start_CountTrips'].fillna(0,inplace=True)
```

[30]:
```
merged_start
```

[30]:
```
                   Time  Weekday       Date  Business_day  Holiday  Station_ID  \
0       01/09/2014 00        0 2014-09-01             0        1         2.0
1       01/09/2014 01        0 2014-09-01             0        1         2.0
2       01/09/2014 02        0 2014-09-01             0        1         2.0
3       01/09/2014 03        0 2014-09-01             0        1         2.0
4       01/09/2014 04        0 2014-09-01             0        1         2.0
…                 …      …          …           …       …          …
665831  31/08/2015 20        0 2015-08-31             1        0        84.0
665832  31/08/2015 21        0 2015-08-31             1        0        84.0
665833  31/08/2015 22        0 2015-08-31             1        0        84.0
```

```
665834  31/08/2015 23        0 2015-08-31            1        0        84.0
665835  01/09/2015 00        1 2015-09-01            1        0        84.0

        Dock_count      Zip  Hours  Start_CountTrips
0             27.0  95113.0      0               0.0
1             27.0  95113.0      1               0.0
2             27.0  95113.0      2               0.0
3             27.0  95113.0      3               0.0
4             27.0  95113.0      4               0.0
…              …       …     …            …
665831        15.0  95113.0     20               0.0
665832        15.0  95113.0     21               0.0
665833        15.0  95113.0     22               0.0
665834        15.0  95113.0     23               0.0
665835        15.0  95113.0      0               0.0

[665836 rows x 10 columns]
```

The hourly trip counts are merged with the total dates dataframe. The same procedure follows for the stop trips.

[31]:
```
trips_cp = trips.copy()

# I take the starting datetime and split it to date and time
stop_trips_dates = pd.to_datetime(trips["End Date"])
trips_cp["New_dates"] = pd.to_datetime(stop_trips_dates).dt.date
trips_cp["Hours"] = pd.to_datetime(stop_trips_dates).dt.hour

# I groupby the trips together with the dates and hours and count how many␣
 ↪trips started at each station

stop = trips_cp["End Station"].
 ↪groupby([trips_cp['New_dates'],trips_cp['Hours']]).value_counts().to_frame()
stop['Date'] = stop.index.get_level_values('New_dates')
stop['Hours'] = stop.index.get_level_values('Hours')
stop['Station_ID'] = stop.index.get_level_values('End Station')
stop.columns = ['Stop_CountTrips', 'Date','Hours','Station_ID']
stop.reset_index(drop=True,inplace=True)
stop["Date"] = pd.to_datetime(stop.Date)
```

[32]:
```
stop
```

[32]:
```
        Stop_CountTrips       Date  Hours  Station_ID
0                     3 2014-09-01      0          57
1                     1 2014-09-01      4          65
2                     1 2014-09-01      5          70
3                     1 2014-09-01      5          72
```

```
4                     1 2014-09-01      6          74
…                …        …    …         …
138660                2 2015-08-31     23          60
138661                2 2015-08-31     23          70
138662                1 2015-08-31     23           8
138663                1 2015-08-31     23          27
138664                1 2015-08-31     23          64

[138665 rows x 4 columns]
```

[33]:

```python
# I can now merge the start trips at stations with the stop trips at stations

start_stop_merged = pd.
 ↪merge(merged_start,stop,on=["Date","Hours","Station_ID"],how="outer")
start_stop_merged['Stop_CountTrips'].fillna(0,inplace=True)
start_stop_merged
```

[33]:

```
                    Time  Weekday        Date  Business_day  Holiday  Station_ID  \
0       01/09/2014 00        0 2014-09-01             0        1         2.0
1       01/09/2014 01        0 2014-09-01             0        1         2.0
2       01/09/2014 02        0 2014-09-01             0        1         2.0
3       01/09/2014 03        0 2014-09-01             0        1         2.0
4       01/09/2014 04        0 2014-09-01             0        1         2.0
…                  …      …         …            …      …          …
665831  31/08/2015 20        0 2015-08-31             1        0        84.0
665832  31/08/2015 21        0 2015-08-31             1        0        84.0
665833  31/08/2015 22        0 2015-08-31             1        0        84.0
665834  31/08/2015 23        0 2015-08-31             1        0        84.0
665835  01/09/2015 00        1 2015-09-01             1        0        84.0

        Dock_count      Zip  Hours  Start_CountTrips  Stop_CountTrips
0             27.0  95113.0      0               0.0              0.0
1             27.0  95113.0      1               0.0              0.0
2             27.0  95113.0      2               0.0              0.0
3             27.0  95113.0      3               0.0              0.0
4             27.0  95113.0      4               0.0              0.0
…              …       …     …               …                …
665831        15.0  95113.0     20               0.0              0.0
665832        15.0  95113.0     21               0.0              0.0
665833        15.0  95113.0     22               0.0              0.0
665834        15.0  95113.0     23               0.0              0.0
665835        15.0  95113.0      0               0.0              0.0

[665836 rows x 11 columns]
```

Our target variable should be in each column the difference between Stop_CountTrips and Start_CountTrips.

[34]:
```
start_stop_merged["Net_Rate"] =␣
↪start_stop_merged["Stop_CountTrips"]-start_stop_merged["Start_CountTrips"]
```

Another addition to the training data would be to add the weather conditions for each station each day(since we are missing hourly data). This is done by merging the weather data with the merged dataframe that was created.

[35]:
```
merged_weather = pd.
↪merge(start_stop_merged,weather,on=["Date","Zip"],how="outer")
```

[36]:
```
# The data with date 01/09/2015 are removed since there is no weather and trip␣
↪data at this date.

# Drop the data samples with dates 1/9/2015 and then date column
merged_weather.drop(merged_weather[merged_weather['Date'] == '2015-09-01'].
↪index, inplace=True)
#merged_weather.drop(columns=['Date'],inplace=True)
```

[37]:
```
dataset = merged_weather
dataset
```

[37]:
```
                    Time  Weekday        Date  Business_day  Holiday  Station_ID  \
0       01/09/2014 00        0 2014-09-01             0        1         2.0
1       01/09/2014 01        0 2014-09-01             0        1         2.0
2       01/09/2014 02        0 2014-09-01             0        1         2.0
3       01/09/2014 03        0 2014-09-01             0        1         2.0
4       01/09/2014 04        0 2014-09-01             0        1         2.0
…                  …       …          …             …        …           …
665792  31/08/2015 19        0 2015-08-31             1        0        82.0
665793  31/08/2015 20        0 2015-08-31             1        0        82.0
665794  31/08/2015 21        0 2015-08-31             1        0        82.0
665795  31/08/2015 22        0 2015-08-31             1        0        82.0
665796  31/08/2015 23        0 2015-08-31             1        0        82.0

        Dock_count      Zip  Hours  Start_CountTrips  …  \
0             27.0  95113.0      0               0.0  …
1             27.0  95113.0      1               0.0  …
2             27.0  95113.0      2               0.0  …
3             27.0  95113.0      3               0.0  …
4             27.0  95113.0      4               0.0  …
…                …        …      …                 …  …
665792        15.0  94107.0     19               1.0  …
665793        15.0  94107.0     20               0.0  …
665794        15.0  94107.0     21               0.0  …
665795        15.0  94107.0     22               0.0  …
665796        15.0  94107.0     23               0.0  …
```

```
        Max VisibilityMiles  Mean VisibilityMiles  Min VisibilityMiles  \
0                      10.0                  10.0                 10.0   
1                      10.0                  10.0                 10.0   
2                      10.0                  10.0                 10.0   
3                      10.0                  10.0                 10.0   
4                      10.0                  10.0                 10.0   
…                       …                     …                    …   
665792                 10.0                  10.0                  9.0   
665793                 10.0                  10.0                  9.0   
665794                 10.0                  10.0                  9.0   
665795                 10.0                  10.0                  9.0   
665796                 10.0                  10.0                  9.0   

        Max Wind SpeedMPH  Mean Wind SpeedMPH  Max Gust SpeedMPH  \
0                    17.0                 5.0               22.0   
1                    17.0                 5.0               22.0   
2                    17.0                 5.0               22.0   
3                    17.0                 5.0               22.0   
4                    17.0                 5.0               22.0   
…                     …                   …                 …   
665792               18.0                 9.0               21.0   
665793               18.0                 9.0               21.0   
665794               18.0                 9.0               21.0   
665795               18.0                 9.0               21.0   
665796               18.0                 9.0               21.0   

        PrecipitationIn  CloudCover  Events  WindDirDegrees  
0                   0.0         0.0  Normal           296.0  
1                   0.0         0.0  Normal           296.0  
2                   0.0         0.0  Normal           296.0  
3                   0.0         0.0  Normal           296.0  
4                   0.0         0.0  Normal           296.0  
…                   …         …     …               …  
665792              0.0         1.0  Normal           246.0  
665793              0.0         1.0  Normal           246.0  
665794              0.0         1.0  Normal           246.0  
665795              0.0         1.0  Normal           246.0  
665796              0.0         1.0  Normal           246.0  

[665760 rows x 34 columns]
```

[38]:
```
dataset.describe()
```

[38]:
```
             Weekday   Business_day        Holiday     Station_ID  \
count  665760.000000  665760.000000  665760.000000  665760.000000   
mean        2.991781       0.687671       0.027397      46.513158   
std         2.003406       0.463443       0.163238      25.969887   
```

min	0.000000	0.000000	0.000000	2.000000
25%	1.000000	0.000000	0.000000	25.750000
50%	3.000000	1.000000	0.000000	47.500000
75%	5.000000	1.000000	0.000000	68.250000
max	6.000000	1.000000	1.000000	90.000000

	Dock_count	Zip	Hours	Start_CountTrips	\
count	665760.000000	665760.000000	665760.000000	665760.000000	
mean	17.815789	94320.263158	11.500000	0.531951	
std	3.989168	413.141403	6.922192	1.629249	
min	11.000000	94041.000000	0.000000	0.000000	
25%	15.000000	94107.000000	5.750000	0.000000	
50%	15.000000	94107.000000	11.500000	0.000000	
75%	19.000000	94301.000000	17.250000	0.000000	
max	27.000000	95113.000000	23.000000	52.000000	

	Stop_CountTrips	Net_Rate	…	Min Sea Level PressureIn	\
count	665760.000000	665760.000000	…	665760.000000	
mean	0.531951	0.000000	…	29.960170	
std	1.730788	1.592348	…	0.131206	
min	0.000000	-37.000000	…	29.340000	
25%	0.000000	0.000000	…	29.860000	
50%	0.000000	0.000000	…	29.950000	
75%	0.000000	0.000000	…	30.050000	
max	56.000000	50.000000	…	30.360000	

	Max VisibilityMiles	Mean VisibilityMiles	Min VisibilityMiles	\
count	665760.000000	665760.000000	665760.000000	
mean	10.144408	9.509226	7.747467	
std	1.227468	1.255915	3.024935	
min	5.000000	4.000000	0.000000	
25%	10.000000	9.000000	6.000000	
50%	10.000000	10.000000	10.000000	
75%	10.000000	10.000000	10.000000	
max	20.000000	20.000000	20.000000	

	Max Wind SpeedMPH	Mean Wind SpeedMPH	Max Gust SpeedMPH	\
count	665760.000000	665760.000000	665760.000000	
mean	16.935138	6.710553	22.112810	
std	6.880511	3.350704	5.982081	
min	4.000000	0.000000	7.000000	
25%	13.000000	4.000000	20.000000	
50%	17.000000	6.000000	21.690810	
75%	20.000000	9.000000	24.000000	
max	128.000000	23.000000	62.000000	

	PrecipitationIn	CloudCover	WindDirDegrees

```
count    665760.000000  665760.000000   665760.000000
mean          0.033161       3.576408      256.431756
std           0.201655       2.305355       82.000923
min           0.000000       0.000000        0.000000
25%           0.000000       2.000000      244.000000
50%           0.000000       4.000000      281.000000
75%           0.000000       5.000000      307.000000
max           3.360000       8.000000      360.000000

[8 rows x 31 columns]
```

[39]:
```
correlation = dataset.corr()
correlation
```

[39]:
```
                                Weekday  Business_day       Holiday  \
Weekday                    1.000000e+00 -7.227637e-01 -1.333523e-01
Business_day              -7.227637e-01  1.000000e+00 -2.490407e-01
Holiday                   -1.333523e-01 -2.490407e-01  1.000000e+00
Station_ID                 7.405186e-16  5.916985e-16  3.719302e-15
Dock_count                 2.837516e-15  8.975182e-15 -2.136653e-14
Zip                       -2.449051e-15 -6.617513e-15  2.957843e-14
Hours                     -1.921132e-17  4.782342e-21  4.722565e-21
Start_CountTrips          -1.089239e-01  1.347285e-01 -2.525548e-02
Stop_CountTrips           -1.025311e-01  1.267496e-01 -2.377915e-02
Net_Rate                   2.825057e-06 -8.141571e-05 -5.778609e-06
Max TemperatureF           1.630696e-02  4.222482e-03 -2.168771e-02
Mean TemperatureF          2.374296e-02  1.178199e-03 -4.474512e-02
Min TemperatureF           2.593032e-02 -1.597870e-03 -6.355438e-02
Max Dew PointF             5.357322e-02 -1.378715e-02 -7.949395e-02
MeanDew PointF             4.245935e-02  1.073197e-03 -9.426447e-02
Min DewpointF              2.956955e-02  1.315609e-02 -9.836056e-02
Max Humidity               7.736448e-02 -4.124710e-02 -6.431787e-02
Mean Humidity              5.010435e-02 -2.320267e-02 -7.087387e-02
Min Humidity               2.550867e-02 -6.986889e-03 -6.256606e-02
Max Sea Level PressureIn  -3.207496e-02 -7.411865e-03 -2.813410e-03
Mean Sea Level PressureIn -2.901153e-02 -1.323763e-02  3.436900e-03
Min Sea Level PressureIn  -2.946843e-02 -1.641854e-02  1.062251e-02
Max VisibilityMiles        1.494231e-02 -1.325865e-02 -1.146948e-02
Mean VisibilityMiles      -4.543548e-04 -1.822459e-02 -4.749646e-03
Min VisibilityMiles        9.304483e-03 -3.254694e-02  6.784081e-03
Max Wind SpeedMPH         -2.783175e-02  3.001860e-02 -2.663027e-02
Mean Wind SpeedMPH        -4.373204e-02  3.729389e-02 -1.423743e-02
Max Gust SpeedMPH         -3.274058e-02  1.816629e-02 -1.617639e-03
PrecipitationIn           -4.950631e-03  4.663640e-02 -2.734886e-02
CloudCover                 2.707276e-02 -2.333388e-02 -5.623731e-02
WindDirDegrees            -1.807789e-02  2.225218e-03  2.016563e-02
```

	Station_ID	Dock_count	Zip	\
Weekday	7.405186e-16	2.837516e-15	-2.449051e-15	
Business_day	5.916985e-16	8.975182e-15	-6.617513e-15	
Holiday	3.719302e-15	-2.136653e-14	2.957843e-14	
Station_ID	1.000000e+00	2.710607e-01	-5.696462e-01	
Dock_count	2.710607e-01	1.000000e+00	-1.814409e-01	
Zip	-5.696462e-01	-1.814409e-01	1.000000e+00	
Hours	-8.320204e-18	1.159494e-21	0.000000e+00	
Start_CountTrips	1.664825e-01	1.475961e-01	-1.285710e-01	
Stop_CountTrips	1.562364e-01	1.434431e-01	-1.210091e-01	
Net_Rate	-5.209000e-04	4.897616e-03	2.084573e-05	
Max TemperatureF	-1.359855e-01	-6.209219e-02	1.482285e-01	
Mean TemperatureF	-8.863504e-02	-4.708105e-02	7.889046e-02	
Min TemperatureF	-1.455483e-02	-2.401572e-02	-2.634033e-02	
Max Dew PointF	-1.927221e-02	-3.709633e-02	-4.119868e-02	
MeanDew PointF	2.685460e-02	-5.891730e-03	-6.493383e-02	
Min DewpointF	4.974050e-02	8.909358e-03	-8.095258e-02	
Max Humidity	5.536891e-02	2.041245e-02	-6.253836e-02	
Mean Humidity	1.521108e-01	4.525578e-02	-2.163782e-01	
Min Humidity	1.834735e-01	8.207485e-02	-2.068124e-01	
Max Sea Level PressureIn	1.574651e-03	9.145774e-05	-2.077219e-02	
Mean Sea Level PressureIn	4.479843e-03	-1.012751e-03	-2.595390e-02	
Min Sea Level PressureIn	4.781608e-03	-2.880571e-03	-2.689629e-02	
Max VisibilityMiles	-5.650236e-02	-9.410032e-02	-2.803920e-03	
Mean VisibilityMiles	-5.395323e-02	-4.640723e-02	2.701259e-02	
Min VisibilityMiles	-7.957851e-02	-4.162253e-02	6.611461e-02	
Max Wind SpeedMPH	7.564176e-02	5.524657e-02	-2.472153e-02	
Mean Wind SpeedMPH	1.911419e-01	1.142939e-01	-9.843400e-02	
Max Gust SpeedMPH	1.087102e-01	4.791440e-02	-1.009448e-01	
PrecipitationIn	1.244041e-02	1.913469e-02	4.085402e-03	
CloudCover	1.523160e-01	9.639319e-02	-4.054533e-02	
WindDirDegrees	-2.252190e-02	-2.292182e-02	2.572599e-03	

	Hours	Start_CountTrips	Stop_CountTrips	\
Weekday	-1.921132e-17	-0.108924	-0.102531	
Business_day	4.782342e-21	0.134729	0.126750	
Holiday	4.722565e-21	-0.025255	-0.023779	
Station_ID	-8.320204e-18	0.166483	0.156236	
Dock_count	1.159494e-21	0.147596	0.143443	
Zip	0.000000e+00	-0.128571	-0.121009	
Hours	1.000000e+00	0.070981	0.075777	
Start_CountTrips	7.098116e-02	1.000000	0.552240	
Stop_CountTrips	7.577669e-02	0.552240	1.000000	
Net_Rate	9.738696e-03	-0.422922	0.521903	
Max TemperatureF	1.751385e-19	-0.015143	-0.014239	
Mean TemperatureF	-7.245185e-19	-0.006638	-0.006240	
Min TemperatureF	-1.424518e-18	0.002706	0.002545	

Max Dew PointF	-5.390589e-20	-0.004057	-0.003854
MeanDew PointF	-1.312710e-18	0.019361	0.018196
Min DewpointF	1.541398e-18	0.029189	0.027450
Max Humidity	-4.301013e-20	0.003993	0.003720
Mean Humidity	1.629630e-18	0.016738	0.015698
Min Humidity	-3.454982e-21	0.046242	0.043480
Max Sea Level PressureIn	-1.145213e-20	-0.012683	-0.011900
Mean Sea Level PressureIn	1.181641e-20	-0.009186	-0.008584
Min Sea Level PressureIn	-2.350206e-20	-0.005993	-0.005569
Max VisibilityMiles	1.130477e-20	-0.037517	-0.035202
Mean VisibilityMiles	-9.698317e-20	-0.011066	-0.010417
Min VisibilityMiles	-3.930924e-18	-0.015057	-0.014188
Max Wind SpeedMPH	3.821511e-18	0.037483	0.035245
Mean Wind SpeedMPH	-1.061505e-17	0.087053	0.081988
Max Gust SpeedMPH	6.684674e-18	0.023691	0.022349
PrecipitationIn	-2.934658e-19	-0.017643	-0.016694
CloudCover	-6.687929e-22	0.049997	0.047071
WindDirDegrees	1.293222e-18	0.009333	0.008751

	Net_Rate	…	Min Sea Level PressureIn \
Weekday	2.825057e-06	…	-2.946843e-02
Business_day	-8.141571e-05	…	-1.641854e-02
Holiday	-5.778609e-06	…	1.062251e-02
Station_ID	-5.209000e-04	…	4.781608e-03
Dock_count	4.897616e-03	…	-2.880571e-03
Zip	2.084573e-05	…	-2.689629e-02
Hours	9.738696e-03	…	-2.350206e-20
Start_CountTrips	-4.229215e-01	…	-5.993354e-03
Stop_CountTrips	5.219029e-01	…	-5.569187e-03
Net_Rate	1.000000e+00	…	7.886749e-05
Max TemperatureF	1.625632e-05	…	-3.896178e-01
Mean TemperatureF	9.726345e-06	…	-5.167107e-01
Min TemperatureF	-1.874894e-06	…	-5.483771e-01
Max Dew PointF	-3.824974e-05	…	-4.262522e-01
MeanDew PointF	-3.120750e-05	…	-4.553188e-01
Min DewpointF	-2.909095e-05	…	-4.346906e-01
Max Humidity	-4.146381e-05	…	1.705796e-02
Mean Humidity	-6.297958e-05	…	-1.957816e-02
Min Humidity	-5.313440e-05	…	-4.070000e-02
Max Sea Level PressureIn	4.161867e-05	…	9.334396e-01
Mean Sea Level PressureIn	6.838992e-05	…	9.780578e-01
Min Sea Level PressureIn	7.886749e-05	…	1.000000e+00
Max VisibilityMiles	1.237259e-04	…	-4.775944e-02
Mean VisibilityMiles	1.706418e-18	…	-7.241481e-02
Min VisibilityMiles	-1.559190e-05	…	2.086946e-02
Max Wind SpeedMPH	-4.222552e-05	…	-3.003662e-01
Mean Wind SpeedMPH	4.532468e-05	…	-3.688797e-01

Max Gust SpeedMPH	5.231129e-05	…	-3.315007e-01
PrecipitationIn	-9.349868e-05	…	-2.447719e-01
CloudCover	7.774289e-06	…	-1.589703e-01
WindDirDegrees	-3.807625e-05	…	-2.961600e-02

	Max VisibilityMiles	Mean VisibilityMiles \
Weekday	1.494231e-02	-4.543548e-04
Business_day	-1.325865e-02	-1.822459e-02
Holiday	-1.146948e-02	-4.749646e-03
Station_ID	-5.650236e-02	-5.395323e-02
Dock_count	-9.410032e-02	-4.640723e-02
Zip	-2.803920e-03	2.701259e-02
Hours	1.130477e-20	-9.698317e-20
Start_CountTrips	-3.751730e-02	-1.106571e-02
Stop_CountTrips	-3.520247e-02	-1.041653e-02
Net_Rate	1.237259e-04	1.706418e-18
Max TemperatureF	7.592158e-02	2.644111e-01
Mean TemperatureF	8.726146e-02	2.371651e-01
Min TemperatureF	8.909831e-02	1.558717e-01
Max Dew PointF	1.089254e-01	-1.303840e-02
MeanDew PointF	9.787465e-02	-3.068177e-02
Min DewpointF	7.201240e-02	-1.041659e-02
Max Humidity	-2.631401e-02	-3.980222e-01
Mean Humidity	-5.298275e-02	-4.264280e-01
Min Humidity	-1.465501e-02	-3.545712e-01
Max Sea Level PressureIn	-7.216426e-02	-1.276114e-01
Mean Sea Level PressureIn	-6.125860e-02	-9.878491e-02
Min Sea Level PressureIn	-4.775944e-02	-7.241481e-02
Max VisibilityMiles	1.000000e+00	3.898081e-01
Mean VisibilityMiles	3.898081e-01	1.000000e+00
Min VisibilityMiles	6.631160e-02	7.685520e-01
Max Wind SpeedMPH	7.335757e-02	9.237266e-02
Mean Wind SpeedMPH	7.900461e-02	1.428197e-01
Max Gust SpeedMPH	1.017682e-02	3.364707e-02
PrecipitationIn	-1.935009e-02	-2.866093e-01
CloudCover	-4.642078e-02	-2.377469e-01
WindDirDegrees	4.187986e-02	1.179137e-01

	Min VisibilityMiles	Max Wind SpeedMPH \
Weekday	9.304483e-03	-2.783175e-02
Business_day	-3.254694e-02	3.001860e-02
Holiday	6.784081e-03	-2.663027e-02
Station_ID	-7.957851e-02	7.564176e-02
Dock_count	-4.162253e-02	5.524657e-02
Zip	6.611461e-02	-2.472153e-02
Hours	-3.930924e-18	3.821511e-18
Start_CountTrips	-1.505725e-02	3.748253e-02

Stop_CountTrips	-1.418825e-02	3.524472e-02
Net_Rate	-1.559190e-05	-4.222552e-05
Max TemperatureF	2.907515e-01	7.015479e-02
Mean TemperatureF	2.228393e-01	1.804320e-01
Min TemperatureF	9.437339e-02	2.631750e-01
Max Dew PointF	-1.123120e-01	1.017349e-01
MeanDew PointF	-1.305633e-01	1.285866e-01
Min DewpointF	-1.006111e-01	1.302262e-01
Max Humidity	-5.027478e-01	-1.163478e-01
Mean Humidity	-5.345755e-01	-6.709736e-02
Min Humidity	-4.568994e-01	2.606795e-02
Max Sea Level PressureIn	-5.228635e-02	-3.013564e-01
Mean Sea Level PressureIn	-1.172801e-02	-3.059214e-01
Min Sea Level PressureIn	2.086946e-02	-3.003662e-01
Max VisibilityMiles	6.631160e-02	7.335757e-02
Mean VisibilityMiles	7.685520e-01	9.237266e-02
Min VisibilityMiles	1.000000e+00	6.539802e-02
Max Wind SpeedMPH	6.539802e-02	1.000000e+00
Mean Wind SpeedMPH	9.155204e-02	6.403803e-01
Max Gust SpeedMPH	8.239495e-03	5.498513e-01
PrecipitationIn	-3.071337e-01	1.872397e-01
CloudCover	-3.223139e-01	1.554907e-01
WindDirDegrees	1.857964e-01	-9.934955e-03

	Mean Wind SpeedMPH	Max Gust SpeedMPH \
Weekday	-4.373204e-02	-3.274058e-02
Business_day	3.729389e-02	1.816629e-02
Holiday	-1.423743e-02	-1.617639e-03
Station_ID	1.911419e-01	1.087102e-01
Dock_count	1.142939e-01	4.791440e-02
Zip	-9.843400e-02	-1.009448e-01
Hours	-1.061505e-17	6.684674e-18
Start_CountTrips	8.705327e-02	2.369109e-02
Stop_CountTrips	8.198787e-02	2.234935e-02
Net_Rate	4.532468e-05	5.231129e-05
Max TemperatureF	-1.301215e-03	4.400153e-02
Mean TemperatureF	2.183464e-01	1.697155e-01
Min TemperatureF	4.156970e-01	2.732160e-01
Max Dew PointF	1.089263e-01	1.109892e-01
MeanDew PointF	2.106600e-01	1.087857e-01
Min DewpointF	2.547827e-01	9.413161e-02
Max Humidity	-2.079145e-01	-9.421811e-02
Mean Humidity	-3.355616e-02	-2.743240e-02
Min Humidity	1.635984e-01	1.951427e-02
Max Sea Level PressureIn	-4.034512e-01	-3.039390e-01
Mean Sea Level PressureIn	-3.948909e-01	-3.230705e-01
Min Sea Level PressureIn	-3.688797e-01	-3.315007e-01

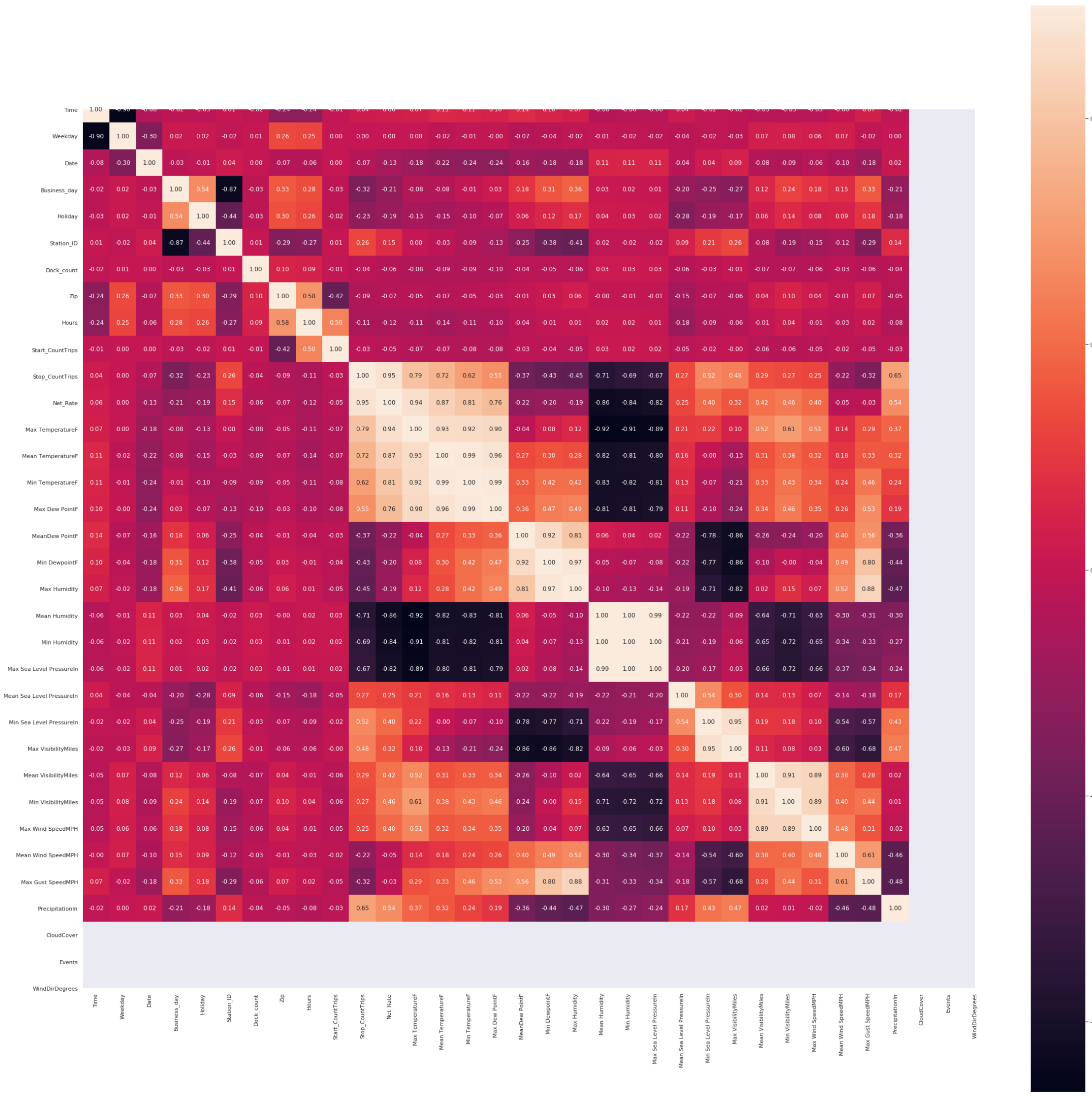
```
Max VisibilityMiles              7.900461e-02       1.017682e-02
Mean VisibilityMiles             1.428197e-01       3.364707e-02
Min VisibilityMiles              9.155204e-02       8.239495e-03
Max Wind SpeedMPH                6.403803e-01       5.498513e-01
Mean Wind SpeedMPH               1.000000e+00       5.925353e-01
Max Gust SpeedMPH                5.925353e-01       1.000000e+00
PrecipitationIn                  1.865885e-01       2.676104e-01
CloudCover                       3.385662e-01       1.391855e-01
WindDirDegrees                  -9.428782e-03      -2.773903e-02

                           PrecipitationIn    CloudCover  WindDirDegrees
Weekday                      -4.950631e-03  2.707276e-02   -1.807789e-02
Business_day                  4.663640e-02 -2.333388e-02    2.225218e-03
Holiday                      -2.734886e-02 -5.623731e-02    2.016563e-02
Station_ID                    1.244041e-02  1.523160e-01   -2.252190e-02
Dock_count                    1.913469e-02  9.639319e-02   -2.292182e-02
Zip                           4.085402e-03 -4.054533e-02    2.572599e-03
Hours                        -2.934658e-19 -6.687929e-22    1.293222e-18
Start_CountTrips             -1.764292e-02  4.999682e-02    9.333367e-03
Stop_CountTrips              -1.669389e-02  4.707085e-02    8.750783e-03
Net_Rate                     -9.349868e-05  7.774289e-06   -3.807625e-05
Max TemperatureF             -1.475797e-01 -3.388355e-01    3.229784e-01
Mean TemperatureF            -8.135362e-02 -6.038073e-02    2.846965e-01
Min TemperatureF              1.162343e-02  2.748162e-01    1.801837e-01
Max Dew PointF                7.345187e-02  1.551442e-01    1.706159e-01
MeanDew PointF                6.935004e-02  3.034806e-01    1.489675e-01
Min DewpointF                 5.693042e-02  3.719641e-01    1.121332e-01
Max Humidity                  1.587742e-01  1.927637e-01   -9.099028e-02
Mean Humidity                 2.077293e-01  5.131858e-01   -1.725922e-01
Min Humidity                  2.110215e-01  6.294940e-01   -1.939128e-01
Max Sea Level PressureIn     -1.556567e-01 -1.566836e-01   -1.105545e-01
Mean Sea Level PressureIn    -2.117144e-01 -1.645076e-01   -6.691296e-02
Min Sea Level PressureIn     -2.447719e-01 -1.589703e-01   -2.961600e-02
Max VisibilityMiles          -1.935009e-02 -4.642078e-02    4.187986e-02
Mean VisibilityMiles         -2.866093e-01 -2.377469e-01    1.179137e-01
Min VisibilityMiles          -3.071337e-01 -3.223139e-01    1.857964e-01
Max Wind SpeedMPH             1.872397e-01  1.554907e-01   -9.934955e-03
Mean Wind SpeedMPH            1.865885e-01  3.385662e-01   -9.428782e-03
Max Gust SpeedMPH             2.676104e-01  1.391855e-01   -2.773903e-02
PrecipitationIn               1.000000e+00  2.483970e-01   -2.148042e-01
CloudCover                    2.483970e-01  1.000000e+00   -2.301525e-01
WindDirDegrees               -2.148042e-01 -2.301525e-01    1.000000e+00

[31 rows x 31 columns]
```

Here it can be seen how our features are corellated. For example we can see that business day and weekdays are highly corellated. Also zip with stationID and the temperatures with the dew points.

```
[54]: correlation_map = np.corrcoef(correlation)
sns.set(font_scale=1.0)
heatmap = sns.heatmap(correlation_map, cbar=True, annot=True, square=True,␣
 ↪fmt='.2f', yticklabels=dataset.columns, xticklabels=dataset.columns)
plt.rcParams['figure.figsize'] = [40, 40]
plt.show()
```

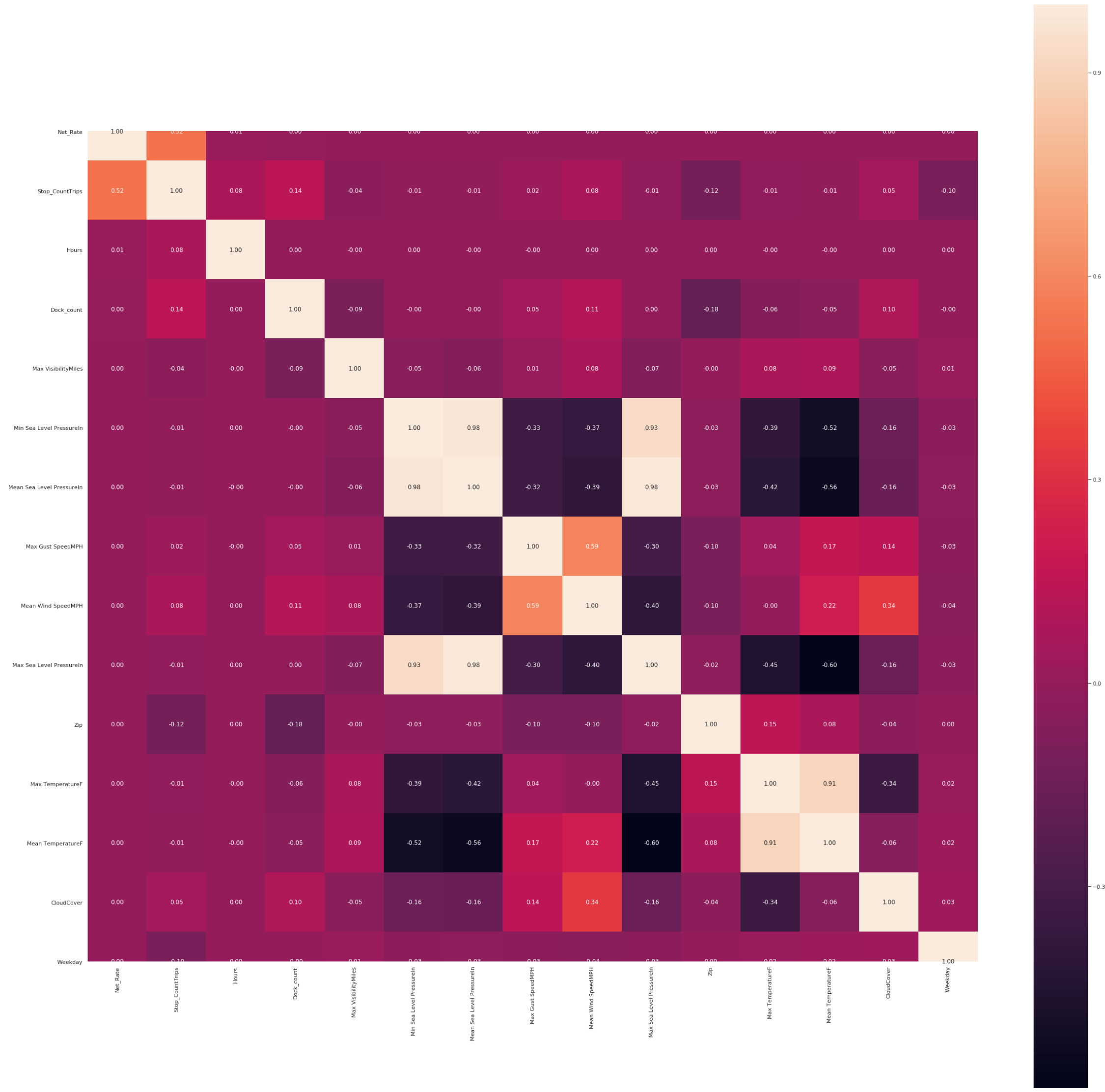


In this part, it is shown which features are mostly corellated with our target variable Net_Rate

```
[55]: correlation = dataset.corr(method='pearson')
columns = correlation.nlargest(15, 'Net_Rate').index
columns
```

```
[55]: Index(['Net_Rate', 'Stop_CountTrips', 'Hours', 'Dock_count',
       'Max VisibilityMiles', 'Min Sea Level PressureIn',
       'Mean Sea Level PressureIn', 'Max Gust SpeedMPH', 'Mean Wind SpeedMPH',
       'Max Sea Level PressureIn', 'Zip', 'Max TemperatureF',
       'Mean TemperatureF', 'CloudCover', 'Weekday'],
      dtype='object')
```

```
[56]: correlation_map = np.corrcoef(dataset[columns].values.T)
sns.set(font_scale=1.0)
heatmap = sns.heatmap(correlation_map, cbar=True, annot=True, square=True,␣
 ↪fmt='.2f', yticklabels=columns.values, xticklabels=columns.values)
plt.rcParams['figure.figsize'] = [25, 25]
plt.show()
```



At this point, a first table with train data is obtained, that could provide the necessary predictive model. However, the table will be further analysed and also specific columns will be chosen in the modeling phase.

2 Modelling Approach

The problem is a regression problem. Thus, a gradient boosted regressor was chosen to learn and model the above data. The XGBoost model is tuned, using the Grid search approach of sklearn and also cross validation is performed to better estimate the model's performance in an unseen dataset.

Regarding our features, first the Time column is dropped, since each specific date occurs only once in our dataset. Moreover, start and stop trip columns should be removed, because they contain all the info about the target variable(the model would easily learn to subtract the 2 columns and achieve accurate results). Since the data are time series, it should be obvious to split the data to past for train and future for prediction. However, there is no future information linked to our estimations, since each prediction is based on the current status of each station.

[57]:
```
import xgboost as xgb
from xgboost import plot_importance, plot_tree
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split, GridSearchCV,␣
↪cross_val_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import preprocessing
```

[58]:
```
# The specific day is not relevant since it happens only once. Keep only month␣
↪from the date and hour.
dataset["Month"] = pd.to_datetime(dataset["Date"]).dt.strftime('%m')
dataset.drop(columns=['Time'],inplace=True)
##Remove the start stop count
dataset.drop(columns=['Start_CountTrips','Stop_CountTrips'],inplace=True)
```

[59]:
```
# Change the order of the columns in the dataframe
dataset = dataset[['Weekday','Business_day','Holiday','Station_ID',␣
↪'Dock_count','Zip','Month', 'Hours','Max TemperatureF',
       'Mean TemperatureF', 'Min TemperatureF',
       'Max Dew PointF', 'MeanDew PointF', 'Min DewpointF', 'Max Humidity',
       'Mean Humidity', 'Min Humidity', 'Max Sea Level PressureIn',
       'Mean Sea Level PressureIn', 'Min Sea Level PressureIn',
       'Max Wind SpeedMPH', 'Mean Wind SpeedMPH', 'Max Gust SpeedMPH',
       'PrecipitationIn', 'CloudCover','Events' , 'WindDirDegrees', 'Net_Rate']]
```

[60]:
```
# Some of the features are necessary to be encoded. This is done using the␣
↪Label encoder
```

```
lbl = preprocessing.LabelEncoder()
dataset['Month'] = lbl.fit_transform(dataset['Month'])
dataset['Hours'] = lbl.fit_transform(dataset['Hours'])
dataset['Events']=dataset['Events'].apply(str)
dataset['Events'] = lbl.fit_transform(dataset['Events'])
dataset['Zip'] = lbl.fit_transform(dataset['Zip'])

#Change type of values depending on their nature
dataset['Station_ID']=dataset['Station_ID'].apply(int)
dataset['Dock_count']=dataset['Dock_count'].apply(int)
dataset['Net_Rate']=dataset['Net_Rate'].apply(int)
```

[61]:
```
# Separate features and target variable

X, y = dataset.iloc[:,:-1], dataset.iloc[:,-1]

#print(y)
data_dmatrix = xgb.DMatrix(data=X, label=y)

# Random split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=123)
```

[121]:
```
# Use GridSearch method to get the best parameters of the model
# Moreover, multiple experiments were done to get an insight of the model␣
 ↪parameters
# It will take some time to complete

xg_reg = xgb.XGBRegressor(objective ='reg:squarederror')

parameters= {
       "colsample_bytree":[0.6,1.0],
       "subsample":[0.6,1.0],
       "min_child_weight":[1,4],
       "learning_rate": [0.1,0.3],
       "max_depth" :[7,12],
       "reg_alpha" :[1,30],
       "n_estimators" :[50]
}

xg_reg_gs = GridSearchCV(xg_reg, parameters, n_jobs=2, cv=2, refit=True)

xg_reg_gs.fit(X_train,y_train, eval_metric='rmse',verbose=True)
```

[121]:
```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    colsample_bylevel=1, colsample_bynode=1,
```

```
                                    colsample_bytree=1, gamma=0,
                                    importance_type='gain', learning_rate=0.1,
                                    max_delta_step=0, max_depth=3,
                                    min_child_weight=1, missing=None,
                                    n_estimators=100, n_jobs=1, nthread=None,
                                    objective='reg:squarederror',
                                    random_st…ambda=1,
                                    scale_pos_weight=1, seed=None, silent=None,
                                    subsample=1, verbosity=1),
             iid='warn', n_jobs=2,
             param_grid={'colsample_bytree': [0.6, 1.0],
                         'learning_rate': [0.1, 0.3], 'max_depth': [7, 12],
                         'min_child_weight': [1, 4], 'n_estimators': [50],
                         'reg_alpha': [1, 30], 'subsample': [0.6, 1.0]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[123]:
```
results = xg_reg_gs.cv_results_
score = xg_reg_gs.best_score_
print('score:', score)
print(xg_reg_gs.best_params_)
```

```
score: 0.5490230880964797
{'colsample_bytree': 1.0, 'learning_rate': 0.3, 'max_depth': 12,
'min_child_weight': 4, 'n_estimators': 50, 'reg_alpha': 30, 'subsample': 1.0}
```

I start with a single training to get an idea about the parameters and use the best combination that found using GridSearch. The GridSearch provides us with a good combination of parameters. Then, additional experimentation was made.

[178]:
```
eval_set = [(X_train, y_train), (X_test, y_test)]

# The XGB regressor object with the parameters
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', colsample_bytree = 1.
↪0, learning_rate = 0.30, subsample=1.0,
                max_depth = 12, reg_alpha=30, min_child_weight=4 ,n_estimators␣
↪= 100) #15
```

[179]:
```
# Train the regressor

xg_reg.fit(X_train,y_train, eval_metric='rmse',eval_set=eval_set,verbose=True)
```

```
[0]     validation_0-rmse:1.48216       validation_1-rmse:1.45994
[1]     validation_0-rmse:1.36307       validation_1-rmse:1.35077
[2]     validation_0-rmse:1.27211       validation_1-rmse:1.267
[3]     validation_0-rmse:1.2201        validation_1-rmse:1.22056
[4]     validation_0-rmse:1.17999       validation_1-rmse:1.18468
[5]     validation_0-rmse:1.15612       validation_1-rmse:1.16574
```

```
[6]	validation_0-rmse:1.13206	validation_1-rmse:1.14693
[7]	validation_0-rmse:1.11799	validation_1-rmse:1.13505
[8]	validation_0-rmse:1.10291	validation_1-rmse:1.12465
[9]	validation_0-rmse:1.089	validation_1-rmse:1.11682
[10]	validation_0-rmse:1.08287	validation_1-rmse:1.1137
[11]	validation_0-rmse:1.07345	validation_1-rmse:1.10631
[12]	validation_0-rmse:1.06795	validation_1-rmse:1.10228
[13]	validation_0-rmse:1.06242	validation_1-rmse:1.09908
[14]	validation_0-rmse:1.05673	validation_1-rmse:1.09484
[15]	validation_0-rmse:1.04787	validation_1-rmse:1.0888
[16]	validation_0-rmse:1.04209	validation_1-rmse:1.08594
[17]	validation_0-rmse:1.03855	validation_1-rmse:1.08374
[18]	validation_0-rmse:1.03443	validation_1-rmse:1.08133
[19]	validation_0-rmse:1.03202	validation_1-rmse:1.07964
[20]	validation_0-rmse:1.02673	validation_1-rmse:1.07571
[21]	validation_0-rmse:1.02479	validation_1-rmse:1.07511
[22]	validation_0-rmse:1.02363	validation_1-rmse:1.07448
[23]	validation_0-rmse:1.01379	validation_1-rmse:1.06714
[24]	validation_0-rmse:1.00925	validation_1-rmse:1.06527
[25]	validation_0-rmse:1.00435	validation_1-rmse:1.06274
[26]	validation_0-rmse:1.00098	validation_1-rmse:1.06039
[27]	validation_0-rmse:0.998711	validation_1-rmse:1.05957
[28]	validation_0-rmse:0.996615	validation_1-rmse:1.0584
[29]	validation_0-rmse:0.995317	validation_1-rmse:1.05818
[30]	validation_0-rmse:0.994434	validation_1-rmse:1.05801
[31]	validation_0-rmse:0.992789	validation_1-rmse:1.05711
[32]	validation_0-rmse:0.99172	validation_1-rmse:1.05702
[33]	validation_0-rmse:0.989151	validation_1-rmse:1.05639
[34]	validation_0-rmse:0.988115	validation_1-rmse:1.05606
[35]	validation_0-rmse:0.986486	validation_1-rmse:1.05595
[36]	validation_0-rmse:0.984001	validation_1-rmse:1.05521
[37]	validation_0-rmse:0.981574	validation_1-rmse:1.05438
[38]	validation_0-rmse:0.97998	validation_1-rmse:1.05361
[39]	validation_0-rmse:0.977651	validation_1-rmse:1.05256
[40]	validation_0-rmse:0.973702	validation_1-rmse:1.05
[41]	validation_0-rmse:0.971481	validation_1-rmse:1.04926
[42]	validation_0-rmse:0.96964	validation_1-rmse:1.04864
[43]	validation_0-rmse:0.966056	validation_1-rmse:1.04705
[44]	validation_0-rmse:0.965178	validation_1-rmse:1.04679
[45]	validation_0-rmse:0.964238	validation_1-rmse:1.04642
[46]	validation_0-rmse:0.962668	validation_1-rmse:1.04604
[47]	validation_0-rmse:0.96163	validation_1-rmse:1.04579
[48]	validation_0-rmse:0.96021	validation_1-rmse:1.0459
[49]	validation_0-rmse:0.95959	validation_1-rmse:1.04589
[50]	validation_0-rmse:0.958522	validation_1-rmse:1.04527
[51]	validation_0-rmse:0.957601	validation_1-rmse:1.04518
[52]	validation_0-rmse:0.956767	validation_1-rmse:1.04492
[53]	validation_0-rmse:0.954527	validation_1-rmse:1.04419
```

```
[54]	validation_0-rmse:0.95301	validation_1-rmse:1.04424
[55]	validation_0-rmse:0.951773	validation_1-rmse:1.0439
[56]	validation_0-rmse:0.949919	validation_1-rmse:1.04348
[57]	validation_0-rmse:0.948327	validation_1-rmse:1.0433
[58]	validation_0-rmse:0.947088	validation_1-rmse:1.04343
[59]	validation_0-rmse:0.946343	validation_1-rmse:1.04311
[60]	validation_0-rmse:0.945301	validation_1-rmse:1.04323
[61]	validation_0-rmse:0.943985	validation_1-rmse:1.043
[62]	validation_0-rmse:0.942927	validation_1-rmse:1.04295
[63]	validation_0-rmse:0.942081	validation_1-rmse:1.04265
[64]	validation_0-rmse:0.940719	validation_1-rmse:1.0425
[65]	validation_0-rmse:0.939991	validation_1-rmse:1.0424
[66]	validation_0-rmse:0.939184	validation_1-rmse:1.04251
[67]	validation_0-rmse:0.938165	validation_1-rmse:1.04211
[68]	validation_0-rmse:0.936825	validation_1-rmse:1.0417
[69]	validation_0-rmse:0.936061	validation_1-rmse:1.04159
[70]	validation_0-rmse:0.935202	validation_1-rmse:1.04145
[71]	validation_0-rmse:0.933876	validation_1-rmse:1.04093
[72]	validation_0-rmse:0.933229	validation_1-rmse:1.04086
[73]	validation_0-rmse:0.932067	validation_1-rmse:1.04102
[74]	validation_0-rmse:0.930731	validation_1-rmse:1.04137
[75]	validation_0-rmse:0.929698	validation_1-rmse:1.04134
[76]	validation_0-rmse:0.928696	validation_1-rmse:1.0412
[77]	validation_0-rmse:0.927357	validation_1-rmse:1.04129
[78]	validation_0-rmse:0.926699	validation_1-rmse:1.04119
[79]	validation_0-rmse:0.925799	validation_1-rmse:1.04158
[80]	validation_0-rmse:0.92516	validation_1-rmse:1.04148
[81]	validation_0-rmse:0.924286	validation_1-rmse:1.0415
[82]	validation_0-rmse:0.923445	validation_1-rmse:1.0415
[83]	validation_0-rmse:0.92258	validation_1-rmse:1.04085
[84]	validation_0-rmse:0.921617	validation_1-rmse:1.04066
[85]	validation_0-rmse:0.921058	validation_1-rmse:1.04042
[86]	validation_0-rmse:0.919823	validation_1-rmse:1.04001
[87]	validation_0-rmse:0.91902	validation_1-rmse:1.03998
[88]	validation_0-rmse:0.918666	validation_1-rmse:1.04006
[89]	validation_0-rmse:0.918006	validation_1-rmse:1.0401
[90]	validation_0-rmse:0.917116	validation_1-rmse:1.04019
[91]	validation_0-rmse:0.916715	validation_1-rmse:1.04015
[92]	validation_0-rmse:0.91619	validation_1-rmse:1.03994
[93]	validation_0-rmse:0.915564	validation_1-rmse:1.03993
[94]	validation_0-rmse:0.914831	validation_1-rmse:1.03985
[95]	validation_0-rmse:0.9142	validation_1-rmse:1.03989
[96]	validation_0-rmse:0.913157	validation_1-rmse:1.03952
[97]	validation_0-rmse:0.912588	validation_1-rmse:1.03938
[98]	validation_0-rmse:0.911893	validation_1-rmse:1.03948
[99]	validation_0-rmse:0.911012	validation_1-rmse:1.03947
```

[179]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1.0, gamma=0,
             importance_type='gain', learning_rate=0.3, max_delta_step=0,
             max_depth=12, min_child_weight=4, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=30, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1.0, verbosity=1)
```

[128]:
```
# 3-fold Cross Validation is being performed
params = {"objective":"reg:squarederror", "colsample_bytree": 1.0,␣
↪"learning_rate":0.30, "subsample":1.0,
          "min_child_weight":4,"max_depth":12, "reg_alpha":30}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=200, early_stopping_rounds=10,␣
↪metrics="rmse", as_pandas=True, seed=123)

print(cv_results)

print((cv_results["test-rmse-mean"]).tail(1))
```

```
     train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0           1.475910        0.003836        1.479549       0.015773
1           1.351627        0.006594        1.361126       0.014861
2           1.265852        0.007994        1.280736       0.010898
3           1.214202        0.004217        1.233338       0.008858
4           1.173336        0.006123        1.196828       0.011783
..               …               …               …              …
103         0.905968        0.001255        1.043520       0.002262
104         0.905327        0.001217        1.043483       0.002357
105         0.904442        0.001266        1.043343       0.002362
106         0.903755        0.001175        1.043318       0.002392
107         0.903035        0.001259        1.043225       0.002287

[108 rows x 4 columns]
107    1.043225
Name: test-rmse-mean, dtype: float64
```

[133]:
```
# Predictions at test set
preds = xg_reg.predict(X_test)
# Predictions at train
train_preds = xg_reg.predict(X_train)

# Root mean square error at test set
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE_test: %f" % (rmse))
```

```
# Root mean square error at train set
rmse_train = np.sqrt(mean_squared_error(y_train, train_preds))
print("RMSE_train: %f" % (rmse_train))

gain = xg_reg._Booster.get_score(importance_type='gain')
print(gain)

# cover = xg_reg._Booster.get_score(importance_type='cover')
# print(cover)
```

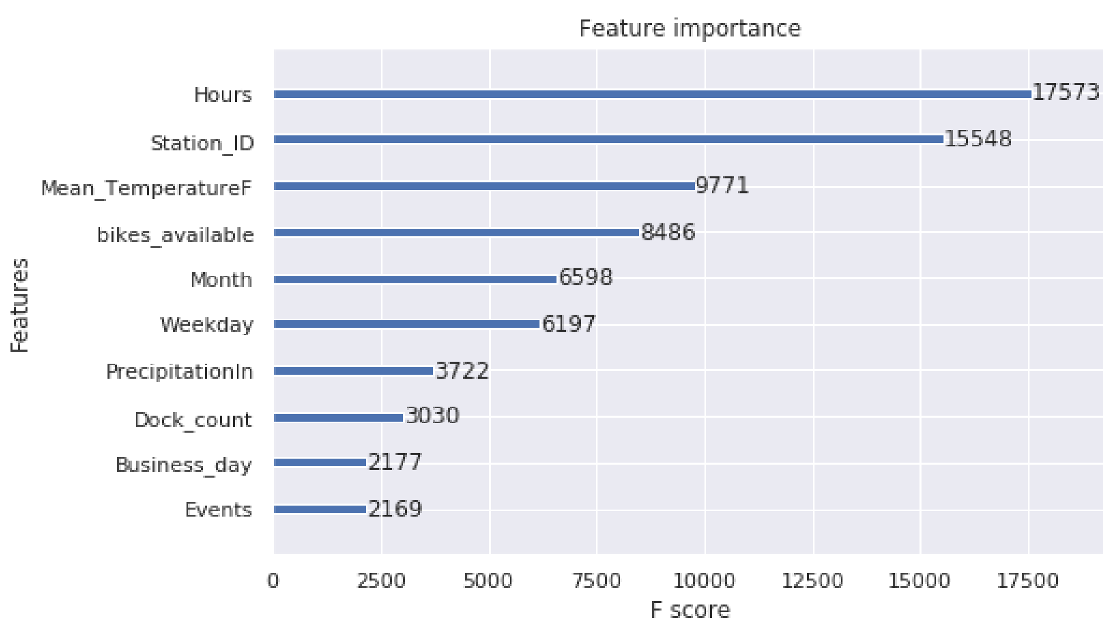
```
RMSE_test: 1.039538
RMSE_train: 0.911369
{'bikes_available': 13.420537390559128, 'Hours': 24.349498986767784,
'Dock_count': 35.527951883091454, 'Station_ID': 39.879495341058394,
'Business_day': 74.19203978539136, 'Events': 2.369128516518445,
'PrecipitationIn': 2.8087168859307163, 'Month': 14.447384585626978, 'Weekday':
11.512204469236078, 'Mean_TemperatureF': 4.314044789481135}
```

Above we can also see the gain each features add to the model's estimation.

[134]:
```
# Here we plot the important features of the Regressor
plot_importance(xg_reg._Booster)

plt.rcParams['figure.figsize'] = [8, 5]

plt.show()
```



Feature Engineering By running some experiments which are skipped above, the following conclusions, about which features improve the performance, are made: * with Dock_count * without Zip * with PrecipitationIN * without WindSpeed * without Holidays * with Business_days * With Temperature

We can see that it is better to remove highly corellated columns from our features. Also, a high correlated feature with the target variable, like WindDir does not mean that could provide meaningful information in predicting the net rate, and may lead to overfitting. Thus, the models will be shown to perform better after removing them.

No improvement in the validation RMSE was shown while removing the features based on the above list.

[137]:
```
dataset2 = dataset[['Weekday','Business_day','Station_ID','Dock_count','Month',␣
↪'Hours',
       'Mean TemperatureF', 'PrecipitationIn','Events', 'Net_Rate']]
```

[138]:
```
dataset2
```

[138]:
```
        Weekday  Business_day  Station_ID  Dock_count  Month  Hours  \
0             0             0           2          27      8      0
1             0             0           2          27      8      1
2             0             0           2          27      8      2
3             0             0           2          27      8      3
4             0             0           2          27      8      4
…           …             …           …          …      …      …
665792        0             1          82          15      7     19
665793        0             1          82          15      7     20
665794        0             1          82          15      7     21
665795        0             1          82          15      7     22
665796        0             1          82          15      7     23

        Mean TemperatureF  PrecipitationIn  Events  Net_Rate
0                    72.0              0.0       2         0
1                    72.0              0.0       2         0
2                    72.0              0.0       2         0
3                    72.0              0.0       2         0
4                    72.0              0.0       2         0
…                     …             …       …         …
665792               69.0              0.0       2        -1
665793               69.0              0.0       2         1
665794               69.0              0.0       2         2
665795               69.0              0.0       2         0
665796               69.0              0.0       2         0

[665760 rows x 10 columns]
```

[139]:
```
# Separate features and target variable

X, y = dataset2.iloc[:,:-1], dataset2.iloc[:,-1]

#print(y)
data_dmatrix = xgb.DMatrix(data=X, label=y)

# Random split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,␣
↪random_state=123)
```

[146]:
```
eval_set = [(X_train, y_train), (X_test, y_test)]

# The XGB regressor object with the parameters
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror',colsample_bytree=1,␣
↪colsample_bylevel=1,
                          colsample_bynode=1, learning_rate = 0.3, subsample=1,␣
↪max_depth = 12, min_child_weight=4,
                          reg_alpha=30, n_estimators = 100) #15
```

[147]:
```
xg_reg.fit(X_train,y_train, eval_metric='rmse',eval_set=eval_set,verbose=True)
```

```
[0]     validation_0-rmse:1.43273       validation_1-rmse:1.41124
[1]     validation_0-rmse:1.29361       validation_1-rmse:1.28126
[2]     validation_0-rmse:1.21332       validation_1-rmse:1.21038
[3]     validation_0-rmse:1.16355       validation_1-rmse:1.16685
[4]     validation_0-rmse:1.129 validation_1-rmse:1.13796
[5]     validation_0-rmse:1.10849       validation_1-rmse:1.12361
[6]     validation_0-rmse:1.09188       validation_1-rmse:1.11126
[7]     validation_0-rmse:1.08121       validation_1-rmse:1.10413
[8]     validation_0-rmse:1.07156       validation_1-rmse:1.0956
[9]     validation_0-rmse:1.06458       validation_1-rmse:1.08969
[10]    validation_0-rmse:1.06118       validation_1-rmse:1.08817
[11]    validation_0-rmse:1.0588        validation_1-rmse:1.08642
[12]    validation_0-rmse:1.05444       validation_1-rmse:1.08241
[13]    validation_0-rmse:1.04882       validation_1-rmse:1.08021
[14]    validation_0-rmse:1.04652       validation_1-rmse:1.07861
[15]    validation_0-rmse:1.04369       validation_1-rmse:1.0774
[16]    validation_0-rmse:1.0429        validation_1-rmse:1.07702
[17]    validation_0-rmse:1.04123       validation_1-rmse:1.0763
[18]    validation_0-rmse:1.04034       validation_1-rmse:1.07618
[19]    validation_0-rmse:1.0397        validation_1-rmse:1.07587
[20]    validation_0-rmse:1.03745       validation_1-rmse:1.07563
[21]    validation_0-rmse:1.03602       validation_1-rmse:1.07491
[22]    validation_0-rmse:1.03379       validation_1-rmse:1.07395
[23]    validation_0-rmse:1.03186       validation_1-rmse:1.07363
[24]    validation_0-rmse:1.03055       validation_1-rmse:1.07314
```

```
[25]	validation_0-rmse:1.02879	validation_1-rmse:1.07222
[26]	validation_0-rmse:1.02704	validation_1-rmse:1.0709
[27]	validation_0-rmse:1.02572	validation_1-rmse:1.07046
[28]	validation_0-rmse:1.02315	validation_1-rmse:1.06982
[29]	validation_0-rmse:1.0214	validation_1-rmse:1.06933
[30]	validation_0-rmse:1.01972	validation_1-rmse:1.06867
[31]	validation_0-rmse:1.01866	validation_1-rmse:1.0686
[32]	validation_0-rmse:1.0178	validation_1-rmse:1.06869
[33]	validation_0-rmse:1.01567	validation_1-rmse:1.06802
[34]	validation_0-rmse:1.01445	validation_1-rmse:1.06785
[35]	validation_0-rmse:1.01352	validation_1-rmse:1.06701
[36]	validation_0-rmse:1.012	validation_1-rmse:1.06658
[37]	validation_0-rmse:1.01144	validation_1-rmse:1.06642
[38]	validation_0-rmse:1.01057	validation_1-rmse:1.06628
[39]	validation_0-rmse:1.00985	validation_1-rmse:1.06619
[40]	validation_0-rmse:1.00947	validation_1-rmse:1.06595
[41]	validation_0-rmse:1.00887	validation_1-rmse:1.06599
[42]	validation_0-rmse:1.00756	validation_1-rmse:1.0655
[43]	validation_0-rmse:1.00685	validation_1-rmse:1.06492
[44]	validation_0-rmse:1.00594	validation_1-rmse:1.0649
[45]	validation_0-rmse:1.00501	validation_1-rmse:1.06512
[46]	validation_0-rmse:1.00416	validation_1-rmse:1.06497
[47]	validation_0-rmse:1.00354	validation_1-rmse:1.06509
[48]	validation_0-rmse:1.00267	validation_1-rmse:1.06502
[49]	validation_0-rmse:1.00181	validation_1-rmse:1.06524
[50]	validation_0-rmse:1.00075	validation_1-rmse:1.06557
[51]	validation_0-rmse:0.999998	validation_1-rmse:1.06587
[52]	validation_0-rmse:0.998909	validation_1-rmse:1.06613
[53]	validation_0-rmse:0.997907	validation_1-rmse:1.06608
[54]	validation_0-rmse:0.996799	validation_1-rmse:1.06629
[55]	validation_0-rmse:0.995884	validation_1-rmse:1.0661
[56]	validation_0-rmse:0.994647	validation_1-rmse:1.06591
[57]	validation_0-rmse:0.994066	validation_1-rmse:1.06597
[58]	validation_0-rmse:0.993564	validation_1-rmse:1.06598
[59]	validation_0-rmse:0.993049	validation_1-rmse:1.06618
[60]	validation_0-rmse:0.992447	validation_1-rmse:1.06609
[61]	validation_0-rmse:0.991763	validation_1-rmse:1.06637
[62]	validation_0-rmse:0.991399	validation_1-rmse:1.06651
[63]	validation_0-rmse:0.991028	validation_1-rmse:1.06674
[64]	validation_0-rmse:0.990516	validation_1-rmse:1.06675
[65]	validation_0-rmse:0.989997	validation_1-rmse:1.0667
[66]	validation_0-rmse:0.989344	validation_1-rmse:1.06688
[67]	validation_0-rmse:0.98893	validation_1-rmse:1.06691
[68]	validation_0-rmse:0.988504	validation_1-rmse:1.06705
[69]	validation_0-rmse:0.988041	validation_1-rmse:1.06717
[70]	validation_0-rmse:0.987463	validation_1-rmse:1.06722
[71]	validation_0-rmse:0.986956	validation_1-rmse:1.06728
[72]	validation_0-rmse:0.986521	validation_1-rmse:1.06728
```

```
[73]	validation_0-rmse:0.986016	validation_1-rmse:1.0675
[74]	validation_0-rmse:0.985395	validation_1-rmse:1.0675
[75]	validation_0-rmse:0.984783	validation_1-rmse:1.0674
[76]	validation_0-rmse:0.984383	validation_1-rmse:1.06742
[77]	validation_0-rmse:0.983721	validation_1-rmse:1.06765
[78]	validation_0-rmse:0.98326	validation_1-rmse:1.06746
[79]	validation_0-rmse:0.982683	validation_1-rmse:1.06764
[80]	validation_0-rmse:0.981976	validation_1-rmse:1.06772
[81]	validation_0-rmse:0.98149	validation_1-rmse:1.06776
[82]	validation_0-rmse:0.981335	validation_1-rmse:1.06786
[83]	validation_0-rmse:0.981051	validation_1-rmse:1.06795
[84]	validation_0-rmse:0.980611	validation_1-rmse:1.06791
[85]	validation_0-rmse:0.98	validation_1-rmse:1.06804
[86]	validation_0-rmse:0.979483	validation_1-rmse:1.06812
[87]	validation_0-rmse:0.978911	validation_1-rmse:1.06793
[88]	validation_0-rmse:0.978349	validation_1-rmse:1.06813
[89]	validation_0-rmse:0.977977	validation_1-rmse:1.06814
[90]	validation_0-rmse:0.977624	validation_1-rmse:1.06852
[91]	validation_0-rmse:0.977234	validation_1-rmse:1.06866
[92]	validation_0-rmse:0.976976	validation_1-rmse:1.06875
[93]	validation_0-rmse:0.976659	validation_1-rmse:1.06891
[94]	validation_0-rmse:0.976459	validation_1-rmse:1.06893
[95]	validation_0-rmse:0.976132	validation_1-rmse:1.06903
[96]	validation_0-rmse:0.975919	validation_1-rmse:1.06915
[97]	validation_0-rmse:0.975563	validation_1-rmse:1.06911
[98]	validation_0-rmse:0.975226	validation_1-rmse:1.06919
[99]	validation_0-rmse:0.974966	validation_1-rmse:1.06924
```

[147]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.3, max_delta_step=0,
             max_depth=12, min_child_weight=4, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=30, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

3-fold Cross Validation is also performed using the above model and features.

[149]:
```
params = {"objective":"reg:squarederror","colsample_bytree":1,␣
 ↪"colsample_bylevel":1,
                          "colsample_bynode":1, "learning_rate":0.3,␣
 ↪"subsample":1, "max_depth":12, #10
                          "reg_alpha":35, "n_estimators":100}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=200, early_stopping_rounds=10,␣
 ↪metrics="rmse", as_pandas=True, seed=123)
```

```python
print(cv_results)

print((cv_results["test-rmse-mean"]).tail(1))
```

```
    train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0          1.438285        0.002765        1.442784       0.015968
1          1.303374        0.005520        1.312569       0.012718
2          1.216102        0.001674        1.231996       0.004024
3          1.165109        0.001787        1.185945       0.002657
4          1.131991        0.003641        1.156493       0.003203
5          1.108919        0.002817        1.136379       0.000703
6          1.091899        0.001365        1.122956       0.001298
7          1.079283        0.000759        1.113080       0.002849
8          1.069938        0.001268        1.106310       0.003550
9          1.065356        0.001431        1.103305       0.003224
10         1.060217        0.000242        1.099307       0.003601
11         1.056682        0.001451        1.097250       0.003360
12         1.050912        0.002446        1.093144       0.002196
13         1.048210        0.001630        1.091978       0.001755
14         1.046007        0.002588        1.090743       0.001158
15         1.042810        0.002310        1.088959       0.000920
16         1.041432        0.002418        1.088214       0.000708
17         1.038034        0.001897        1.086832       0.000928
18         1.035411        0.001616        1.085437       0.000812
19         1.033467        0.001614        1.084976       0.000838
20         1.032412        0.001243        1.084430       0.001202
21         1.031033        0.001995        1.083896       0.001315
22         1.029602        0.001394        1.083375       0.001293
23         1.027759        0.001959        1.082467       0.001004
24         1.026333        0.001641        1.081993       0.000932
25         1.025310        0.001677        1.081888       0.001090
26         1.024551        0.001564        1.081864       0.001250
27         1.023624        0.001597        1.081739       0.001411
28         1.022490        0.001403        1.081509       0.001231
29         1.020878        0.001110        1.081354       0.001230
30         1.020159        0.001281        1.081224       0.001178
31         1.019274        0.001476        1.081121       0.001190
32         1.017987        0.001485        1.080987       0.001308
33         1.016580        0.001148        1.080803       0.001168
34         1.015150        0.001136        1.080316       0.001094
35         1.014134        0.000997        1.080049       0.000992
36         1.013355        0.000963        1.079926       0.001096
37         1.012516        0.000772        1.079889       0.001285
38         1.011702        0.000824        1.079951       0.001468
39         1.011263        0.000828        1.079918       0.001485
40         1.010529        0.000891        1.079847       0.001462
```

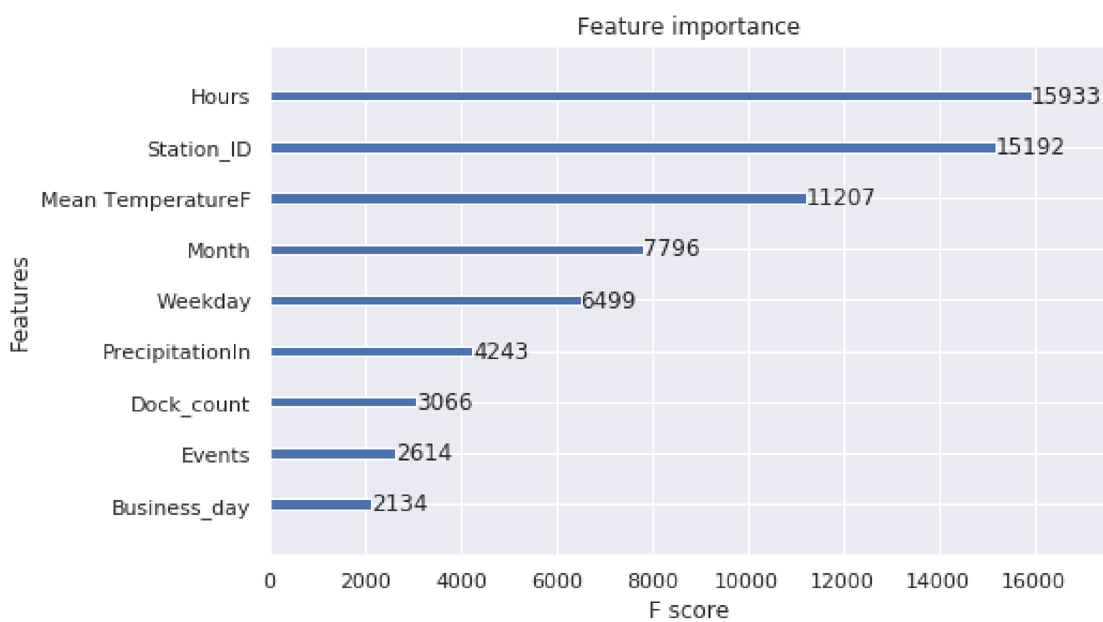
```
41          1.009488         0.000954         1.079678        0.001316
42          1.008527         0.001038         1.079489        0.001400
43          1.007711         0.001349         1.079328        0.001633
44          1.006963         0.001345         1.079285        0.001616
45          1.006208         0.001203         1.079252        0.001642
46          1.005376         0.001198         1.079182        0.001607
47          1.004647         0.001326         1.079205        0.001634
48          1.003770         0.001412         1.079113        0.001693
49          1.002928         0.001344         1.078897        0.001725
50          1.002229         0.001349         1.078728        0.001792
50    1.078728
Name: test-rmse-mean, dtype: float64
```

[150]:

```
# Here we plot the important features of the Regressor
plot_importance(xg_reg._Booster)

plt.rcParams['figure.figsize'] = [8, 5]

plt.show()
```



It can be seen that with the removal of the data, the performance of the model didn't improve, and it even got worse. Moreover, it was much easier to **overfit**.

Improvement of the dataset using external data I decided to extent the original dataset by adding a new feature. In Kaggle another cvs file containing status data was found. Here I will try to extract information and see if by using that our model can be improved. I follow the feature analysis made above, so keep only the features that seem relevant, and the add this new feature. The data source can be found as status.csv here: https://www.kaggle.com/benhamner/sf-bay-area-bike-share

The status data are already cleaned to include only information relevant for our date range. This step is not shown here since the file was really large. This step is included in the source code. After I clean the data, we can see that the file contains the availability of bikes and docks at each station every one minute. The plan for this dataframe is to extract and add in my dataset two columns containing the available bikes and free dock stations at the beginning of every hour. Thus I have to groupby the info contained in the statusDf on hour,time and stationID. Then I will keep only the last value and use it as info for the next hour, meaning that this value is the number of available bikes at the beginning of the next hour.

[151]:
```
avail_bikes = pd.read_csv("../code/rel_data.
 ↪csv",parse_dates=[3],dayfirst=True,index_col=0)
```

[152]:
```
avail_bikes
```

[152]:
```
        station_id  bikes_available  docks_available                 time
0                2               15               12  2014-09-01 00:59:03
1                3                9                6  2014-09-01 00:59:03
2                4                5                6  2014-09-01 00:59:03
3                5                8               11  2014-09-01 00:59:03
4                6                8                7  2014-09-01 00:59:03
…              …              …                …                    …
611630          77               13               14  2015-08-31 23:59:02
611631          80                7                8  2015-08-31 23:59:02
611632          82                5               10  2015-08-31 23:59:02
611633          83                5               10  2015-08-31 23:59:02
611634          84                8                7  2015-08-31 23:59:02

[611635 rows x 4 columns]
```

[153]:
```
avail_bikes["Date"] = pd.to_datetime(avail_bikes["time"]).dt.date
avail_bikes["Hours"] = pd.to_datetime(avail_bikes["time"]).dt.hour
#Rename station_id col to fit the other col name
avail_bikes.columns=␣
 ↪["Station_ID","bikes_available","docks_available","time","Date","Hours"]
avail_bikes["Date"] = pd.to_datetime(avail_bikes["Date"])
```

[154]:
```
avail_bikes['docks_available']=avail_bikes['docks_available'].apply(int)
avail_bikes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 611635 entries, 0 to 611634
Data columns (total 6 columns):
Station_ID          611635 non-null int64
bikes_available     611635 non-null int64
docks_available     611635 non-null int64
```

```
time                611635 non-null object
Date                611635 non-null datetime64[ns]
Hours               611635 non-null int64
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 32.7+ MB
```

[155]:
```
# Here I merge the above table with the feature table merging the two␣
↪dataframes on date, hour and stationID
dataset_avail = pd.
↪merge(merged_weather,avail_bikes,on=["Date","Hours","Station_ID"],how="outer")
```

[156]:
```
dataset_avail.drop(columns=['Date','time'],inplace=True)
```

[157]:
```
dataset_avail
```

[157]:
```
        Weekday  Business_day  Holiday  Station_ID  Dock_count      Zip  \
0             0             0        1         2.0        27.0  95113.0
1             0             0        1         2.0        27.0  95113.0
2             0             0        1         2.0        27.0  95113.0
3             0             0        1         2.0        27.0  95113.0
4             0             0        1         2.0        27.0  95113.0
…           …           …        …         …           …        …
665755        0             1        0        82.0        15.0  94107.0
665756        0             1        0        82.0        15.0  94107.0
665757        0             1        0        82.0        15.0  94107.0
665758        0             1        0        82.0        15.0  94107.0
665759        0             1        0        82.0        15.0  94107.0

        Hours  Net_Rate  Max TemperatureF  Mean TemperatureF  …  \
0           0       0.0              86.0               72.0  …
1           1       0.0              86.0               72.0  …
2           2       0.0              86.0               72.0  …
3           3       0.0              86.0               72.0  …
4           4       0.0              86.0               72.0  …
…         …       …                …                 …  …
665755     19      -1.0              78.0               69.0  …
665756     20       1.0              78.0               69.0  …
665757     21       2.0              78.0               69.0  …
665758     22       0.0              78.0               69.0  …
665759     23       0.0              78.0               69.0  …

        Max Wind SpeedMPH  Mean Wind SpeedMPH  Max Gust SpeedMPH  \
0                    17.0                 5.0               22.0
1                    17.0                 5.0               22.0
2                    17.0                 5.0               22.0
3                    17.0                 5.0               22.0
4                    17.0                 5.0               22.0
```

```
…                       …                 …                 …
665755                 18.0               9.0              21.0
665756                 18.0               9.0              21.0
665757                 18.0               9.0              21.0
665758                 18.0               9.0              21.0
665759                 18.0               9.0              21.0

        PrecipitationIn  CloudCover  Events  WindDirDegrees  Month  \
0                   0.0         0.0  Normal           296.0     09
1                   0.0         0.0  Normal           296.0     09
2                   0.0         0.0  Normal           296.0     09
3                   0.0         0.0  Normal           296.0     09
4                   0.0         0.0  Normal           296.0     09
…                     …         …     …                …     …
665755              0.0         1.0  Normal           246.0     08
665756              0.0         1.0  Normal           246.0     08
665757              0.0         1.0  Normal           246.0     08
665758              0.0         1.0  Normal           246.0     08
665759              0.0         1.0  Normal           246.0     08

        bikes_available  docks_available
0                  15.0             12.0
1                  15.0             12.0
2                  14.0             13.0
3                  15.0             12.0
4                  15.0             12.0
…                     …              …
665755              0.0             15.0
665756              4.0             11.0
665757              6.0              9.0
665758              6.0              9.0
665759              5.0             10.0

[665760 rows x 33 columns]
```

[158]:

```
# We can see that the size of the status data obtained has less rows(611635␣
↪instead of 665760) than the one we have. That is
# we have some missing values as we see below
dataset_avail.isnull().sum()
```

[158]:

```
Weekday                          0
Business_day                     0
Holiday                          0
Station_ID                       0
Dock_count                       0
Zip                              0
Hours                            0
```

```
Net_Rate                       0
Max TemperatureF               0
Mean TemperatureF              0
Min TemperatureF               0
Max Dew PointF                 0
MeanDew PointF                 0
Min DewpointF                  0
Max Humidity                   0
Mean Humidity                  0
Min Humidity                   0
Max Sea Level PressureIn       0
Mean Sea Level PressureIn      0
Min Sea Level PressureIn       0
Max VisibilityMiles            0
Mean VisibilityMiles           0
Min VisibilityMiles            0
Max Wind SpeedMPH              0
Mean Wind SpeedMPH             0
Max Gust SpeedMPH              0
PrecipitationIn                0
CloudCover                     0
Events                         0
WindDirDegrees                 0
Month                          0
bikes_available            54125
docks_available            54125
dtype: int64
```

We repeat now the same steps as before in the modeling to train our model.

[170]:
```python
# Change the order of the columns in the dataframe
dataset_avail2 =␣
↪dataset_avail[['Weekday','Business_day','Station_ID','Dock_count','Month',␣
↪'Hours',
       'Mean TemperatureF','PrecipitationIn','Events' ,'bikes_available',␣
↪'Net_Rate']]
```

[171]:
```python
dataset_avail2
```

[171]:
```
        Weekday  Business_day  Station_ID  Dock_count Month  Hours  \
0             0             0         2.0        27.0    09      0
1             0             0         2.0        27.0    09      1
2             0             0         2.0        27.0    09      2
3             0             0         2.0        27.0    09      3
4             0             0         2.0        27.0    09      4
…          …            …          …           …    …    …
665755        0             1        82.0        15.0    08     19
```

```
665756         0            1         82.0         15.0     08      20
665757         0            1         82.0         15.0     08      21
665758         0            1         82.0         15.0     08      22
665759         0            1         82.0         15.0     08      23

        Mean TemperatureF  PrecipitationIn  Events  bikes_available  Net_Rate
0                    72.0              0.0  Normal             15.0       0.0
1                    72.0              0.0  Normal             15.0       0.0
2                    72.0              0.0  Normal             14.0       0.0
3                    72.0              0.0  Normal             15.0       0.0
4                    72.0              0.0  Normal             15.0       0.0
…                     …             …    …                 …       …
665755               69.0              0.0  Normal              0.0      -1.0
665756               69.0              0.0  Normal              4.0       1.0
665757               69.0              0.0  Normal              6.0       2.0
665758               69.0              0.0  Normal              6.0       0.0
665759               69.0              0.0  Normal              5.0       0.0

[665760 rows x 11 columns]
```

[172]:
```
# Some of the features are necessary to be encoded. This is done using the␣
↪Label encoder

lbl = preprocessing.LabelEncoder()
dataset_avail2['Month'] = lbl.fit_transform(dataset_avail2['Month'])
dataset_avail2['Hours'] = lbl.fit_transform(dataset_avail2['Hours'])
dataset_avail2['Events']=dataset_avail2['Events'].apply(str)
dataset_avail2['Events'] = lbl.fit_transform(dataset_avail2['Events'])

#Change type of values depending on their nature
dataset_avail2['Station_ID']=dataset_avail2['Station_ID'].apply(int)
dataset_avail2['Dock_count']=dataset_avail2['Dock_count'].apply(int)
#dataset_avail['docks_available']=dataset_avail['docks_available'].apply(int)
dataset_avail2['Net_Rate']=dataset_avail2['Net_Rate'].apply(int)
```

[173]:
```
# Separate features and target variable

X, y = dataset_avail2.iloc[:,:-1], dataset_avail2.iloc[:,-1]

#print(y)
data_dmatrix = xgb.DMatrix(data=X, label=y)

# Random split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,␣
↪random_state=123)
```

I use exactly the same parameters as I used for training the previous model above. These parameters were shown to give the best results, thus it can safely stated that the added feature of available_bikes was important in improving the model's perfomance, by comparing the 2 models RMSE.

[174]:
```
eval_set = [(X_train, y_train), (X_test, y_test)]

# The XGB regressor object with the parameters
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror',colsample_bytree=1,␣
↪colsample_bylevel=1,
                          colsample_bynode=1, learning_rate = 0.11,␣
↪subsample=1, max_depth = 15,
                          reg_alpha=50, reg_lambda=100, n_estimators = 350) #15
```

[175]:
```
xg_reg.fit(X_train,y_train, eval_metric='rmse',eval_set=eval_set,verbose=True)
```

```
[0]     validation_0-rmse:1.61633       validation_1-rmse:1.58648
[1]     validation_0-rmse:1.56421       validation_1-rmse:1.53724
[2]     validation_0-rmse:1.52226       validation_1-rmse:1.49457
[3]     validation_0-rmse:1.48228       validation_1-rmse:1.45406
[4]     validation_0-rmse:1.44503       validation_1-rmse:1.41945
[5]     validation_0-rmse:1.41268       validation_1-rmse:1.38838
[6]     validation_0-rmse:1.38339       validation_1-rmse:1.35963
[7]     validation_0-rmse:1.35869       validation_1-rmse:1.33569
[8]     validation_0-rmse:1.33387       validation_1-rmse:1.31394
[9]     validation_0-rmse:1.31081       validation_1-rmse:1.29319
[10]    validation_0-rmse:1.29121       validation_1-rmse:1.27478
[11]    validation_0-rmse:1.27249       validation_1-rmse:1.25777
[12]    validation_0-rmse:1.25695       validation_1-rmse:1.2437
[13]    validation_0-rmse:1.24047       validation_1-rmse:1.22885
[14]    validation_0-rmse:1.22405       validation_1-rmse:1.2142
[15]    validation_0-rmse:1.20995       validation_1-rmse:1.20174
[16]    validation_0-rmse:1.19844       validation_1-rmse:1.19185
[17]    validation_0-rmse:1.18648       validation_1-rmse:1.18081
[18]    validation_0-rmse:1.17593       validation_1-rmse:1.17185
[19]    validation_0-rmse:1.16588       validation_1-rmse:1.16334
[20]    validation_0-rmse:1.15651       validation_1-rmse:1.15566
[21]    validation_0-rmse:1.14849       validation_1-rmse:1.14926
[22]    validation_0-rmse:1.14074       validation_1-rmse:1.14272
[23]    validation_0-rmse:1.13367       validation_1-rmse:1.13701
[24]    validation_0-rmse:1.1263        validation_1-rmse:1.13085
[25]    validation_0-rmse:1.1197        validation_1-rmse:1.12531
[26]    validation_0-rmse:1.1148        validation_1-rmse:1.12164
[27]    validation_0-rmse:1.11021       validation_1-rmse:1.11808
[28]    validation_0-rmse:1.10459       validation_1-rmse:1.11357
[29]    validation_0-rmse:1.10108       validation_1-rmse:1.11121
[30]    validation_0-rmse:1.0967        validation_1-rmse:1.10813
[31]    validation_0-rmse:1.09322       validation_1-rmse:1.10595
[32]    validation_0-rmse:1.08972       validation_1-rmse:1.10356
```

```
[33]	validation_0-rmse:1.0855	validation_1-rmse:1.1004
[34]	validation_0-rmse:1.08202	validation_1-rmse:1.09801
[35]	validation_0-rmse:1.07804	validation_1-rmse:1.09523
[36]	validation_0-rmse:1.07517	validation_1-rmse:1.09328
[37]	validation_0-rmse:1.07188	validation_1-rmse:1.0911
[38]	validation_0-rmse:1.06824	validation_1-rmse:1.08844
[39]	validation_0-rmse:1.06456	validation_1-rmse:1.08574
[40]	validation_0-rmse:1.06123	validation_1-rmse:1.08355
[41]	validation_0-rmse:1.05833	validation_1-rmse:1.08162
[42]	validation_0-rmse:1.05561	validation_1-rmse:1.07973
[43]	validation_0-rmse:1.05286	validation_1-rmse:1.07766
[44]	validation_0-rmse:1.05026	validation_1-rmse:1.07569
[45]	validation_0-rmse:1.04783	validation_1-rmse:1.07391
[46]	validation_0-rmse:1.04508	validation_1-rmse:1.07217
[47]	validation_0-rmse:1.04312	validation_1-rmse:1.07101
[48]	validation_0-rmse:1.04125	validation_1-rmse:1.06988
[49]	validation_0-rmse:1.03896	validation_1-rmse:1.06842
[50]	validation_0-rmse:1.03685	validation_1-rmse:1.06709
[51]	validation_0-rmse:1.0353	validation_1-rmse:1.06622
[52]	validation_0-rmse:1.03334	validation_1-rmse:1.06512
[53]	validation_0-rmse:1.03202	validation_1-rmse:1.06443
[54]	validation_0-rmse:1.03004	validation_1-rmse:1.06313
[55]	validation_0-rmse:1.02749	validation_1-rmse:1.06142
[56]	validation_0-rmse:1.02501	validation_1-rmse:1.05965
[57]	validation_0-rmse:1.02337	validation_1-rmse:1.05859
[58]	validation_0-rmse:1.02137	validation_1-rmse:1.05729
[59]	validation_0-rmse:1.01942	validation_1-rmse:1.05604
[60]	validation_0-rmse:1.01785	validation_1-rmse:1.05513
[61]	validation_0-rmse:1.01613	validation_1-rmse:1.05413
[62]	validation_0-rmse:1.01447	validation_1-rmse:1.05321
[63]	validation_0-rmse:1.01286	validation_1-rmse:1.05238
[64]	validation_0-rmse:1.01156	validation_1-rmse:1.05166
[65]	validation_0-rmse:1.01026	validation_1-rmse:1.05079
[66]	validation_0-rmse:1.00917	validation_1-rmse:1.05027
[67]	validation_0-rmse:1.00812	validation_1-rmse:1.0497
[68]	validation_0-rmse:1.00673	validation_1-rmse:1.04889
[69]	validation_0-rmse:1.00585	validation_1-rmse:1.04849
[70]	validation_0-rmse:1.00473	validation_1-rmse:1.04792
[71]	validation_0-rmse:1.00335	validation_1-rmse:1.04721
[72]	validation_0-rmse:1.00189	validation_1-rmse:1.04636
[73]	validation_0-rmse:1.00052	validation_1-rmse:1.04561
[74]	validation_0-rmse:0.999431	validation_1-rmse:1.04494
[75]	validation_0-rmse:0.998586	validation_1-rmse:1.0445
[76]	validation_0-rmse:0.997488	validation_1-rmse:1.04383
[77]	validation_0-rmse:0.996308	validation_1-rmse:1.04314
[78]	validation_0-rmse:0.995451	validation_1-rmse:1.04268
[79]	validation_0-rmse:0.994569	validation_1-rmse:1.04222
[80]	validation_0-rmse:0.993969	validation_1-rmse:1.04194
```

```
[81]	validation_0-rmse:0.993313	validation_1-rmse:1.04159
[82]	validation_0-rmse:0.992484	validation_1-rmse:1.04119
[83]	validation_0-rmse:0.991666	validation_1-rmse:1.04081
[84]	validation_0-rmse:0.990867	validation_1-rmse:1.04044
[85]	validation_0-rmse:0.990005	validation_1-rmse:1.04002
[86]	validation_0-rmse:0.989257	validation_1-rmse:1.03973
[87]	validation_0-rmse:0.98846	validation_1-rmse:1.03951
[88]	validation_0-rmse:0.987695	validation_1-rmse:1.03927
[89]	validation_0-rmse:0.987059	validation_1-rmse:1.039
[90]	validation_0-rmse:0.986444	validation_1-rmse:1.03866
[91]	validation_0-rmse:0.985476	validation_1-rmse:1.03824
[92]	validation_0-rmse:0.984851	validation_1-rmse:1.03803
[93]	validation_0-rmse:0.984277	validation_1-rmse:1.03791
[94]	validation_0-rmse:0.983608	validation_1-rmse:1.03763
[95]	validation_0-rmse:0.983436	validation_1-rmse:1.03759
[96]	validation_0-rmse:0.983183	validation_1-rmse:1.03755
[97]	validation_0-rmse:0.982819	validation_1-rmse:1.03757
[98]	validation_0-rmse:0.982631	validation_1-rmse:1.03753
[99]	validation_0-rmse:0.982045	validation_1-rmse:1.03717
[100]	validation_0-rmse:0.981555	validation_1-rmse:1.03706
[101]	validation_0-rmse:0.981258	validation_1-rmse:1.03705
[102]	validation_0-rmse:0.980939	validation_1-rmse:1.03691
[103]	validation_0-rmse:0.980524	validation_1-rmse:1.03683
[104]	validation_0-rmse:0.980042	validation_1-rmse:1.03666
[105]	validation_0-rmse:0.979596	validation_1-rmse:1.03657
[106]	validation_0-rmse:0.979055	validation_1-rmse:1.03641
[107]	validation_0-rmse:0.978581	validation_1-rmse:1.03628
[108]	validation_0-rmse:0.978203	validation_1-rmse:1.03619
[109]	validation_0-rmse:0.977899	validation_1-rmse:1.0362
[110]	validation_0-rmse:0.977571	validation_1-rmse:1.03614
[111]	validation_0-rmse:0.977148	validation_1-rmse:1.03601
[112]	validation_0-rmse:0.976757	validation_1-rmse:1.03584
[113]	validation_0-rmse:0.976509	validation_1-rmse:1.03581
[114]	validation_0-rmse:0.976001	validation_1-rmse:1.03553
[115]	validation_0-rmse:0.975722	validation_1-rmse:1.03548
[116]	validation_0-rmse:0.975374	validation_1-rmse:1.03545
[117]	validation_0-rmse:0.974943	validation_1-rmse:1.03538
[118]	validation_0-rmse:0.974591	validation_1-rmse:1.03534
[119]	validation_0-rmse:0.97428	validation_1-rmse:1.03525
[120]	validation_0-rmse:0.974017	validation_1-rmse:1.0352
[121]	validation_0-rmse:0.973711	validation_1-rmse:1.03515
[122]	validation_0-rmse:0.973439	validation_1-rmse:1.03507
[123]	validation_0-rmse:0.973008	validation_1-rmse:1.03497
[124]	validation_0-rmse:0.972632	validation_1-rmse:1.03488
[125]	validation_0-rmse:0.972155	validation_1-rmse:1.03479
[126]	validation_0-rmse:0.97178	validation_1-rmse:1.03472
[127]	validation_0-rmse:0.971422	validation_1-rmse:1.03463
[128]	validation_0-rmse:0.971054	validation_1-rmse:1.03448
```

```
[129]	validation_0-rmse:0.970791	validation_1-rmse:1.0345
[130]	validation_0-rmse:0.970387	validation_1-rmse:1.03445
[131]	validation_0-rmse:0.970132	validation_1-rmse:1.03443
[132]	validation_0-rmse:0.969804	validation_1-rmse:1.0344
[133]	validation_0-rmse:0.969558	validation_1-rmse:1.03439
[134]	validation_0-rmse:0.969231	validation_1-rmse:1.03436
[135]	validation_0-rmse:0.968669	validation_1-rmse:1.03404
[136]	validation_0-rmse:0.968415	validation_1-rmse:1.03402
[137]	validation_0-rmse:0.968059	validation_1-rmse:1.03397
[138]	validation_0-rmse:0.967676	validation_1-rmse:1.03391
[139]	validation_0-rmse:0.967491	validation_1-rmse:1.03394
[140]	validation_0-rmse:0.967025	validation_1-rmse:1.03367
[141]	validation_0-rmse:0.966555	validation_1-rmse:1.03345
[142]	validation_0-rmse:0.966161	validation_1-rmse:1.0334
[143]	validation_0-rmse:0.96586	validation_1-rmse:1.03337
[144]	validation_0-rmse:0.96549	validation_1-rmse:1.03332
[145]	validation_0-rmse:0.965138	validation_1-rmse:1.03326
[146]	validation_0-rmse:0.96467	validation_1-rmse:1.0332
[147]	validation_0-rmse:0.964358	validation_1-rmse:1.03317
[148]	validation_0-rmse:0.96407	validation_1-rmse:1.03316
[149]	validation_0-rmse:0.963756	validation_1-rmse:1.03315
[150]	validation_0-rmse:0.963303	validation_1-rmse:1.03299
[151]	validation_0-rmse:0.962919	validation_1-rmse:1.03296
[152]	validation_0-rmse:0.962631	validation_1-rmse:1.03294
[153]	validation_0-rmse:0.962174	validation_1-rmse:1.03283
[154]	validation_0-rmse:0.961657	validation_1-rmse:1.03263
[155]	validation_0-rmse:0.961033	validation_1-rmse:1.03222
[156]	validation_0-rmse:0.960668	validation_1-rmse:1.03211
[157]	validation_0-rmse:0.960357	validation_1-rmse:1.03205
[158]	validation_0-rmse:0.959935	validation_1-rmse:1.03196
[159]	validation_0-rmse:0.959503	validation_1-rmse:1.03185
[160]	validation_0-rmse:0.958922	validation_1-rmse:1.03168
[161]	validation_0-rmse:0.958634	validation_1-rmse:1.03163
[162]	validation_0-rmse:0.958012	validation_1-rmse:1.03128
[163]	validation_0-rmse:0.957709	validation_1-rmse:1.03117
[164]	validation_0-rmse:0.957476	validation_1-rmse:1.03115
[165]	validation_0-rmse:0.957195	validation_1-rmse:1.03108
[166]	validation_0-rmse:0.956813	validation_1-rmse:1.03108
[167]	validation_0-rmse:0.956492	validation_1-rmse:1.03105
[168]	validation_0-rmse:0.956238	validation_1-rmse:1.03106
[169]	validation_0-rmse:0.955872	validation_1-rmse:1.031
[170]	validation_0-rmse:0.955541	validation_1-rmse:1.03099
[171]	validation_0-rmse:0.955226	validation_1-rmse:1.03087
[172]	validation_0-rmse:0.954927	validation_1-rmse:1.03084
[173]	validation_0-rmse:0.954723	validation_1-rmse:1.03082
[174]	validation_0-rmse:0.954363	validation_1-rmse:1.03076
[175]	validation_0-rmse:0.954168	validation_1-rmse:1.03078
[176]	validation_0-rmse:0.954027	validation_1-rmse:1.03074
```

```
[177]	validation_0-rmse:0.953773	validation_1-rmse:1.03069
[178]	validation_0-rmse:0.953593	validation_1-rmse:1.03068
[179]	validation_0-rmse:0.953408	validation_1-rmse:1.03069
[180]	validation_0-rmse:0.953232	validation_1-rmse:1.03069
[181]	validation_0-rmse:0.952866	validation_1-rmse:1.03047
[182]	validation_0-rmse:0.952598	validation_1-rmse:1.03042
[183]	validation_0-rmse:0.952185	validation_1-rmse:1.03016
[184]	validation_0-rmse:0.951891	validation_1-rmse:1.03011
[185]	validation_0-rmse:0.951614	validation_1-rmse:1.03012
[186]	validation_0-rmse:0.951362	validation_1-rmse:1.03009
[187]	validation_0-rmse:0.951167	validation_1-rmse:1.02998
[188]	validation_0-rmse:0.950854	validation_1-rmse:1.02988
[189]	validation_0-rmse:0.950591	validation_1-rmse:1.02985
[190]	validation_0-rmse:0.950377	validation_1-rmse:1.02983
[191]	validation_0-rmse:0.950112	validation_1-rmse:1.02975
[192]	validation_0-rmse:0.949801	validation_1-rmse:1.02975
[193]	validation_0-rmse:0.949512	validation_1-rmse:1.02968
[194]	validation_0-rmse:0.949261	validation_1-rmse:1.02965
[195]	validation_0-rmse:0.948958	validation_1-rmse:1.02949
[196]	validation_0-rmse:0.948676	validation_1-rmse:1.0294
[197]	validation_0-rmse:0.948367	validation_1-rmse:1.02941
[198]	validation_0-rmse:0.948202	validation_1-rmse:1.02941
[199]	validation_0-rmse:0.948057	validation_1-rmse:1.02944
[200]	validation_0-rmse:0.947918	validation_1-rmse:1.02947
[201]	validation_0-rmse:0.947761	validation_1-rmse:1.02948
[202]	validation_0-rmse:0.947525	validation_1-rmse:1.02946
[203]	validation_0-rmse:0.947356	validation_1-rmse:1.02943
[204]	validation_0-rmse:0.94711	validation_1-rmse:1.02938
[205]	validation_0-rmse:0.946974	validation_1-rmse:1.02937
[206]	validation_0-rmse:0.946826	validation_1-rmse:1.02937
[207]	validation_0-rmse:0.946607	validation_1-rmse:1.02935
[208]	validation_0-rmse:0.946337	validation_1-rmse:1.02926
[209]	validation_0-rmse:0.946117	validation_1-rmse:1.02927
[210]	validation_0-rmse:0.945964	validation_1-rmse:1.0293
[211]	validation_0-rmse:0.945688	validation_1-rmse:1.02923
[212]	validation_0-rmse:0.94549	validation_1-rmse:1.02925
[213]	validation_0-rmse:0.945242	validation_1-rmse:1.02917
[214]	validation_0-rmse:0.945032	validation_1-rmse:1.02919
[215]	validation_0-rmse:0.944889	validation_1-rmse:1.0292
[216]	validation_0-rmse:0.944688	validation_1-rmse:1.02917
[217]	validation_0-rmse:0.944485	validation_1-rmse:1.02919
[218]	validation_0-rmse:0.944282	validation_1-rmse:1.02919
[219]	validation_0-rmse:0.944069	validation_1-rmse:1.02918
[220]	validation_0-rmse:0.943841	validation_1-rmse:1.02914
[221]	validation_0-rmse:0.943651	validation_1-rmse:1.02915
[222]	validation_0-rmse:0.943453	validation_1-rmse:1.02915
[223]	validation_0-rmse:0.943325	validation_1-rmse:1.02913
[224]	validation_0-rmse:0.943125	validation_1-rmse:1.02913
```

```
[225]	validation_0-rmse:0.942994	validation_1-rmse:1.02912
[226]	validation_0-rmse:0.942763	validation_1-rmse:1.02917
[227]	validation_0-rmse:0.942619	validation_1-rmse:1.02919
[228]	validation_0-rmse:0.94243	validation_1-rmse:1.02919
[229]	validation_0-rmse:0.94218	validation_1-rmse:1.02916
[230]	validation_0-rmse:0.941939	validation_1-rmse:1.02913
[231]	validation_0-rmse:0.94171	validation_1-rmse:1.02914
[232]	validation_0-rmse:0.941527	validation_1-rmse:1.02916
[233]	validation_0-rmse:0.941368	validation_1-rmse:1.02915
[234]	validation_0-rmse:0.941228	validation_1-rmse:1.02918
[235]	validation_0-rmse:0.941091	validation_1-rmse:1.02919
[236]	validation_0-rmse:0.940946	validation_1-rmse:1.02917
[237]	validation_0-rmse:0.940744	validation_1-rmse:1.02921
[238]	validation_0-rmse:0.94055	validation_1-rmse:1.0292
[239]	validation_0-rmse:0.940388	validation_1-rmse:1.0292
[240]	validation_0-rmse:0.940191	validation_1-rmse:1.02919
[241]	validation_0-rmse:0.939962	validation_1-rmse:1.02923
[242]	validation_0-rmse:0.939726	validation_1-rmse:1.0292
[243]	validation_0-rmse:0.939435	validation_1-rmse:1.02908
[244]	validation_0-rmse:0.939205	validation_1-rmse:1.02902
[245]	validation_0-rmse:0.938946	validation_1-rmse:1.02894
[246]	validation_0-rmse:0.938756	validation_1-rmse:1.02887
[247]	validation_0-rmse:0.938573	validation_1-rmse:1.02887
[248]	validation_0-rmse:0.938353	validation_1-rmse:1.02888
[249]	validation_0-rmse:0.93811	validation_1-rmse:1.02885
[250]	validation_0-rmse:0.937866	validation_1-rmse:1.02877
[251]	validation_0-rmse:0.937666	validation_1-rmse:1.02873
[252]	validation_0-rmse:0.9375	validation_1-rmse:1.02875
[253]	validation_0-rmse:0.937348	validation_1-rmse:1.02874
[254]	validation_0-rmse:0.937131	validation_1-rmse:1.02876
[255]	validation_0-rmse:0.936983	validation_1-rmse:1.02875
[256]	validation_0-rmse:0.936839	validation_1-rmse:1.02877
[257]	validation_0-rmse:0.936635	validation_1-rmse:1.02876
[258]	validation_0-rmse:0.936438	validation_1-rmse:1.02873
[259]	validation_0-rmse:0.936283	validation_1-rmse:1.02869
[260]	validation_0-rmse:0.936154	validation_1-rmse:1.02868
[261]	validation_0-rmse:0.935973	validation_1-rmse:1.02867
[262]	validation_0-rmse:0.935742	validation_1-rmse:1.02863
[263]	validation_0-rmse:0.935578	validation_1-rmse:1.0286
[264]	validation_0-rmse:0.935309	validation_1-rmse:1.02855
[265]	validation_0-rmse:0.935091	validation_1-rmse:1.02847
[266]	validation_0-rmse:0.934863	validation_1-rmse:1.02836
[267]	validation_0-rmse:0.934632	validation_1-rmse:1.02832
[268]	validation_0-rmse:0.934489	validation_1-rmse:1.02832
[269]	validation_0-rmse:0.934326	validation_1-rmse:1.02829
[270]	validation_0-rmse:0.934124	validation_1-rmse:1.02825
[271]	validation_0-rmse:0.933954	validation_1-rmse:1.0282
[272]	validation_0-rmse:0.933803	validation_1-rmse:1.02819
```

```
[273]	validation_0-rmse:0.93368	validation_1-rmse:1.02818
[274]	validation_0-rmse:0.933486	validation_1-rmse:1.02814
[275]	validation_0-rmse:0.933311	validation_1-rmse:1.02812
[276]	validation_0-rmse:0.933069	validation_1-rmse:1.02805
[277]	validation_0-rmse:0.932844	validation_1-rmse:1.02794
[278]	validation_0-rmse:0.932621	validation_1-rmse:1.02786
[279]	validation_0-rmse:0.932456	validation_1-rmse:1.02786
[280]	validation_0-rmse:0.932283	validation_1-rmse:1.02783
[281]	validation_0-rmse:0.932002	validation_1-rmse:1.02771
[282]	validation_0-rmse:0.9318	validation_1-rmse:1.02772
[283]	validation_0-rmse:0.931679	validation_1-rmse:1.02769
[284]	validation_0-rmse:0.931539	validation_1-rmse:1.02766
[285]	validation_0-rmse:0.931251	validation_1-rmse:1.02748
[286]	validation_0-rmse:0.931069	validation_1-rmse:1.02742
[287]	validation_0-rmse:0.930925	validation_1-rmse:1.02745
[288]	validation_0-rmse:0.930738	validation_1-rmse:1.02741
[289]	validation_0-rmse:0.930569	validation_1-rmse:1.02736
[290]	validation_0-rmse:0.930394	validation_1-rmse:1.02738
[291]	validation_0-rmse:0.930228	validation_1-rmse:1.02741
[292]	validation_0-rmse:0.930092	validation_1-rmse:1.02739
[293]	validation_0-rmse:0.929989	validation_1-rmse:1.02742
[294]	validation_0-rmse:0.929821	validation_1-rmse:1.02738
[295]	validation_0-rmse:0.929653	validation_1-rmse:1.02738
[296]	validation_0-rmse:0.929512	validation_1-rmse:1.02737
[297]	validation_0-rmse:0.929275	validation_1-rmse:1.02737
[298]	validation_0-rmse:0.929036	validation_1-rmse:1.02731
[299]	validation_0-rmse:0.928799	validation_1-rmse:1.02722
[300]	validation_0-rmse:0.928566	validation_1-rmse:1.02714
[301]	validation_0-rmse:0.928418	validation_1-rmse:1.02714
[302]	validation_0-rmse:0.92833	validation_1-rmse:1.02716
[303]	validation_0-rmse:0.928211	validation_1-rmse:1.0272
[304]	validation_0-rmse:0.92803	validation_1-rmse:1.02717
[305]	validation_0-rmse:0.927881	validation_1-rmse:1.02719
[306]	validation_0-rmse:0.927725	validation_1-rmse:1.02719
[307]	validation_0-rmse:0.927567	validation_1-rmse:1.0272
[308]	validation_0-rmse:0.927451	validation_1-rmse:1.02721
[309]	validation_0-rmse:0.927326	validation_1-rmse:1.02715
[310]	validation_0-rmse:0.927193	validation_1-rmse:1.02713
[311]	validation_0-rmse:0.927076	validation_1-rmse:1.02712
[312]	validation_0-rmse:0.926956	validation_1-rmse:1.0271
[313]	validation_0-rmse:0.926843	validation_1-rmse:1.02711
[314]	validation_0-rmse:0.926736	validation_1-rmse:1.02713
[315]	validation_0-rmse:0.926548	validation_1-rmse:1.02705
[316]	validation_0-rmse:0.926417	validation_1-rmse:1.02707
[317]	validation_0-rmse:0.926235	validation_1-rmse:1.02706
[318]	validation_0-rmse:0.926128	validation_1-rmse:1.02707
[319]	validation_0-rmse:0.925953	validation_1-rmse:1.02704
[320]	validation_0-rmse:0.925816	validation_1-rmse:1.02706
```

```
[321]	validation_0-rmse:0.925692	validation_1-rmse:1.02706
[322]	validation_0-rmse:0.925483	validation_1-rmse:1.02706
[323]	validation_0-rmse:0.925374	validation_1-rmse:1.02704
[324]	validation_0-rmse:0.925251	validation_1-rmse:1.02705
[325]	validation_0-rmse:0.925122	validation_1-rmse:1.02699
[326]	validation_0-rmse:0.924991	validation_1-rmse:1.02698
[327]	validation_0-rmse:0.924822	validation_1-rmse:1.02697
[328]	validation_0-rmse:0.924672	validation_1-rmse:1.02694
[329]	validation_0-rmse:0.924459	validation_1-rmse:1.02689
[330]	validation_0-rmse:0.924328	validation_1-rmse:1.0269
[331]	validation_0-rmse:0.924226	validation_1-rmse:1.02688
[332]	validation_0-rmse:0.924112	validation_1-rmse:1.02689
[333]	validation_0-rmse:0.923946	validation_1-rmse:1.02686
[334]	validation_0-rmse:0.923815	validation_1-rmse:1.02684
[335]	validation_0-rmse:0.923682	validation_1-rmse:1.02686
[336]	validation_0-rmse:0.923524	validation_1-rmse:1.02684
[337]	validation_0-rmse:0.923405	validation_1-rmse:1.02685
[338]	validation_0-rmse:0.923301	validation_1-rmse:1.02686
[339]	validation_0-rmse:0.923153	validation_1-rmse:1.02682
[340]	validation_0-rmse:0.923017	validation_1-rmse:1.02684
[341]	validation_0-rmse:0.922854	validation_1-rmse:1.02683
[342]	validation_0-rmse:0.922743	validation_1-rmse:1.02682
[343]	validation_0-rmse:0.922592	validation_1-rmse:1.02678
[344]	validation_0-rmse:0.92246	validation_1-rmse:1.02683
[345]	validation_0-rmse:0.922285	validation_1-rmse:1.0268
[346]	validation_0-rmse:0.922192	validation_1-rmse:1.02684
[347]	validation_0-rmse:0.922105	validation_1-rmse:1.02684
[348]	validation_0-rmse:0.921987	validation_1-rmse:1.02682
[349]	validation_0-rmse:0.921842	validation_1-rmse:1.02679
```

[175]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.11, max_delta_step=0,
             max_depth=15, min_child_weight=1, missing=None, n_estimators=350,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=50, reg_lambda=100, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

[103]:
```python
# See some predictions of the model

preds = xg_reg.predict(X_test)
preds_df = pd.DataFrame(preds, y_test)
preds_df.head(20)
#preds_df.to_csv("preds_last.csv")
```

[103]:
```
                 0
Net_Rate
```

```
 5         1.508265
 0         0.006555
 0         0.365989
 0         0.004635
-1        -0.163201
 0         0.014411
 2        -1.097256
 0        -0.022960
-1        -0.800496
 0         0.075449
 0        -0.000343
-1        -0.009855
-5        -2.095707
 0        -0.059652
-2        -1.006866
 0         0.003163
 4         0.682527
-4        -1.518595
 4        10.903127
 0        -1.511916
```

[104]:

```python
# Cross Validation
params = {"objective":"reg:squarederror","colsample_bytree":
↪1,"colsample_bynode":1,
          "learning_rate":0.11, "subsample":1, "max_depth":15,"reg_alpha":
↪50,"reg_lambda":100}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=500, early_stopping_rounds=10,␣
↪metrics="rmse", as_pandas=True, seed=123)

print(cv_results)

print((cv_results["test-rmse-mean"]).tail(1))
```

```
     train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0           1.612889        0.004752        1.612300       0.018289
1           1.560940        0.004566        1.563699       0.018147
2           1.517508        0.003433        1.520262       0.018817
3           1.478990        0.003716        1.481423       0.018408
4           1.441445        0.003707        1.447231       0.018229
..               …               …               …              …
353         0.921483        0.000789        1.033871       0.004336
354         0.921350        0.000818        1.033854       0.004320
355         0.921240        0.000807        1.033850       0.004329
356         0.921136        0.000803        1.033840       0.004313
357         0.920977        0.000804        1.033813       0.004317
```

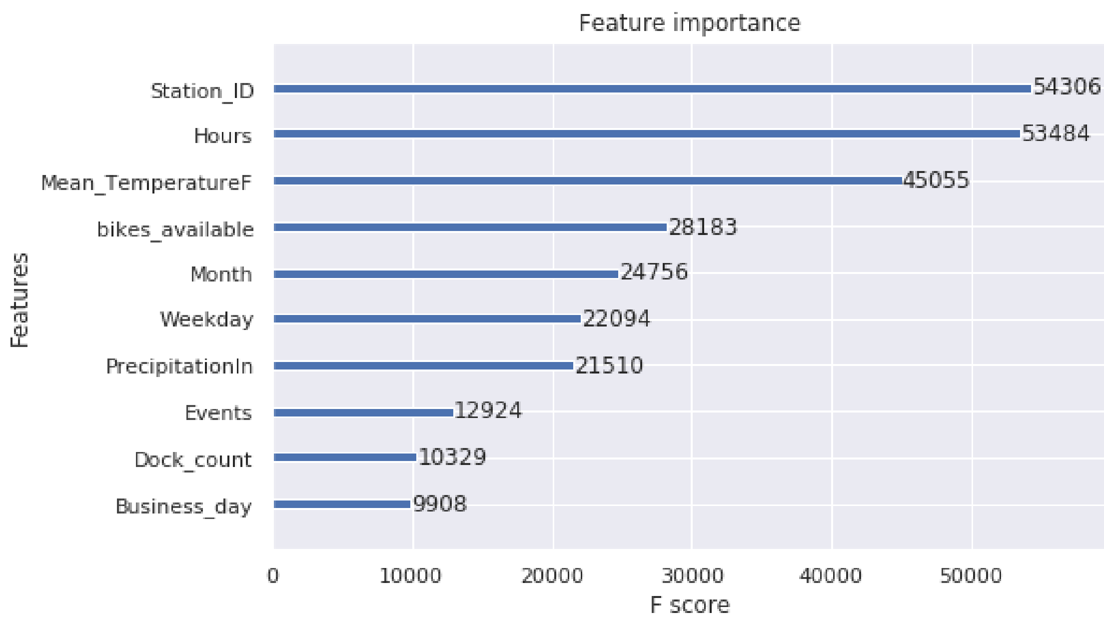
```
[358 rows x 4 columns]
357    1.033813
Name: test-rmse-mean, dtype: float64
```

[105]:

```
# Here we plot the important features of the Regressor
plot_importance(xg_reg._Booster)

plt.rcParams['figure.figsize'] = [8, 5]
plt.show()

# The tree is to large to be visualized
```



Here I try another model for predicting the net rate change. So that we can compare the two models together. I will compare the XGboost regressor with the Gradient Boosting regressor.

[188]:

```
gbr = GradientBoostingRegressor(learning_rate = 0.20,
                                n_estimators = 50,
                                max_depth = 10,
                                min_samples_leaf = 1,
                                criterion="mse",
                                verbose=True,
                                random_state = 2)
```

[185]:

```
# I fill the nan values since gradient boosting cannot work with nan values as␣
↪XGB.
X.fillna(X.mean(), inplace=True)
```

```
X.isnull().sum()
```

[185]:

```
Weekday             0
Business_day        0
Station_ID          0
Dock_count          0
Month               0
Hours               0
Mean TemperatureF   0
PrecipitationIn     0
Events              0
bikes_available     0
dtype: int64
```

[186]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,␣
 ↪random_state=123)

gbr.fit(X_train,y_train)
```

```
      Iter       Train Loss   Remaining Time
         1           2.2083           12.15m
         2           1.9702           10.38m
         3           1.8097            9.55m
         4           1.6608            9.22m
         5           1.5355            8.95m
         6           1.4874            8.65m
         7           1.4529            8.27m
         8           1.4240            8.04m
         9           1.3867            7.80m
        10           1.3179            7.56m
        20           1.0835            5.22m
        30           0.9915            3.31m
        40           0.9204            1.65m
        50           0.8676            0.00s
```

[186]:

```
GradientBoostingRegressor(alpha=0.9, criterion='mse', init=None,
                          learning_rate=0.3, loss='ls', max_depth=10,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=50,
                          n_iter_no_change=None, presort='auto', random_state=2,
                          subsample=1.0, tol=0.0001, validation_fraction=0.1,
                          verbose=True, warm_start=False)
```

[187]:

```
# Predictions at test set
preds_gbr = gbr.predict(X_test)
```

```
# Root mean square error at test set
rmse_gbr = np.sqrt(mean_squared_error(y_test, preds_gbr))
print("RMSE_test: %f" % (rmse_gbr))
```

```
RMSE_test: 1.061003
```

[196]:
```
# Cross validation is being performed

scores = cross_val_score(gbr, X, y, cv=5, n_jobs=2, scoring =␣
 ↪'neg_mean_squared_error')
```

[197]:
```
# See the scores
rmse = np.sqrt(-scores)
print(rmse)
mean_rmse = np.mean(rmse)
mean_rmse
```

```
[0.67129841 0.59370638 1.39719009 1.58182449 1.73171191]
```

[197]:
```
1.1951462571672358
```

3 Performance Analysis

In this section we will discuss the different results, that were achieved at the modelling section. A small explanation will be given on each one of the models and different versions of the train dataset.

DatasetVersion	Model	CrossVal RMSE
WithAllWeatherData	XGBoost	1.043
RelevantFeatures	XGBoost	1.078
WithAvailBikes	XGBoost	1.033
WithAvailBikes	GradientBoosting	1.195

It can be seen, that with a simple feature selection, based on correlation and importance, and the addition of an extra feature we were able to slightly increase the model's performance. It is interesting that all the initial features plus the available bikes did not improve the performance. For that reason, it is not being included in the above table. The extra feature helped more, when most of the weather data were removed. Moreover, the extreme gradient boosting model shows better performance than the gradient boosting model. However, the gradient boosting model parameters were not experimented in great extent, and it is possible that with further tuning, I could be able to increase the performance to similar levels with the XGboost.

4 Conclusions

In this report, I analysed the San Francisco Bike share data, and made a first attempt to generate a model that can predict the hourly change on each station. The data provided were explored and analysed. Correlations and statistics of the features were shown and taken into account in the feature selection process. Finally, the dataset was generated in order to train successfully a model. For modelling, the option of extreme gradient boosting was chosen, since it is shown to produce good results in tabular data. Moreover, it is generally fast and good in avoiding overfitting. Grid search was used to get an insight of the parameters and tune them. Furthermore, the performance was shown and estimated by doing a 3-fold cross validation on different models and different combinations of the features. An external data source was also added, introducing a useful feature for the model to better predict the net change and reduce the RMSE.

Some remarks:

- It was found, that not all the weather information were important to get the model with the best performance. This may be because the weather information are correlated together, but also not all the data are relevant to the net change.
- It is quite easy for the model to overfit. That is why, the small learning rates lead to a better performance.
- XGBoost performed better than Gradient Boosting model, and moreover it was much faster to train.
- High regularization(L1, L2) had to be introduced to avoid the model from overfitting.
- An external data source containing info about the available bikes was a meaningful feature for our model and helped improve the performance.

5 Potential Improvements

In this section, potential improvements are discussed: 1. To begin with, an initial improvement I would suggest, is to try to obtain *hourly weather data*. Since now we have data information only for a specific date. This information could provide us with more information on how the use of bikes changes throughout the day and how this is influenced by the weather. 2. A second idea would be to increase the date range in total, i.e include more years of data. In this case, only the data of one year is used. 3. Another idea would be to change the model approach. An alternative would be to use a time series network. For example, a Recurrent neural network or LSTM, to handle our data as a time series and forecast future changes in the stations. However, the future changes are not always dependant on the past data. 4. Last but not least, I could try to model the stations as a graph neural network. It can be seen that the stations are interconnected, ie. a lot of times bikes that start from one station may end the ride to another. This could be potentially model using a GNN.