asnmt

February 20, 2022

1 Instructions

The assignment is at the bottom!

1.1 This cell automatically downloads Capital Bikeshare data

1.1.1 And here we read in the data

[224]:
```
#
```

[1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 20, 10
import pandas as pd
import numpy as np
bikes = pd.read_csv('../data/bikeshare.csv.gz')
bikes.head()
bikes['start'] = pd.to_datetime(bikes['Start date'], infer_datetime_format=True)
bikes['end'] = pd.to_datetime(bikes['End date'], infer_datetime_format=True)
bikes["dur"] = (bikes['Duration (ms)']/1000).astype(int)
bikes.head()
```

[1]:
```
   Duration (ms)       Start date         End date  Start station number  \
0         301295  3/31/2016 23:59    4/1/2016 0:04                 31280
1         557887  3/31/2016 23:59    4/1/2016 0:08                 31275
2         555944  3/31/2016 23:59    4/1/2016 0:08                 31101
3         766916  3/31/2016 23:57    4/1/2016 0:09                 31226
4         139656  3/31/2016 23:57  3/31/2016 23:59                 31011

                   Start station  End station number  \
0                  11th & S St NW               31506
1  New Hampshire Ave & 24th St NW               31114
2                  14th & V St NW               31221
3      34th St & Wisconsin Ave NW               31214
4               23rd & Crystal Dr               31009

                 End station Bike number Member Type               start  \
```

```
0  1st & Rhode Island Ave NW      W00022  Registered 2016-03-31 23:59:00
1   18th St & Wyoming Ave NW      W01294  Registered 2016-03-31 23:59:00
2             18th & M St NW      W01416  Registered 2016-03-31 23:59:00
3      17th & Corcoran St NW      W01090  Registered 2016-03-31 23:57:00
4          27th & Crystal Dr      W21934  Registered 2016-03-31 23:57:00

                  end  dur
0 2016-04-01 00:04:00  301
1 2016-04-01 00:08:00  557
2 2016-04-01 00:08:00  555
3 2016-04-01 00:09:00  766
4 2016-03-31 23:59:00  139
```

[2]:
```
bikes.dur.mean()
```

[2]:
```
992.8716543657755
```

[3]:
```
bikes.dur.std()
```

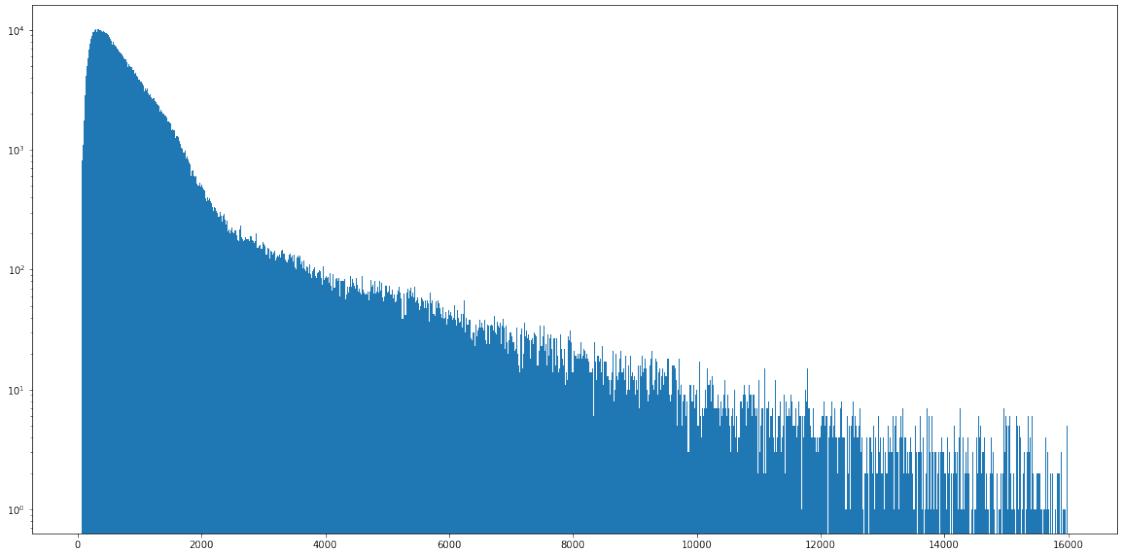
[3]:
```
2073.9809135296514
```

[4]:
```
bikes[bikes.dur>16000].shape
```

[4]:
```
(973, 12)
```

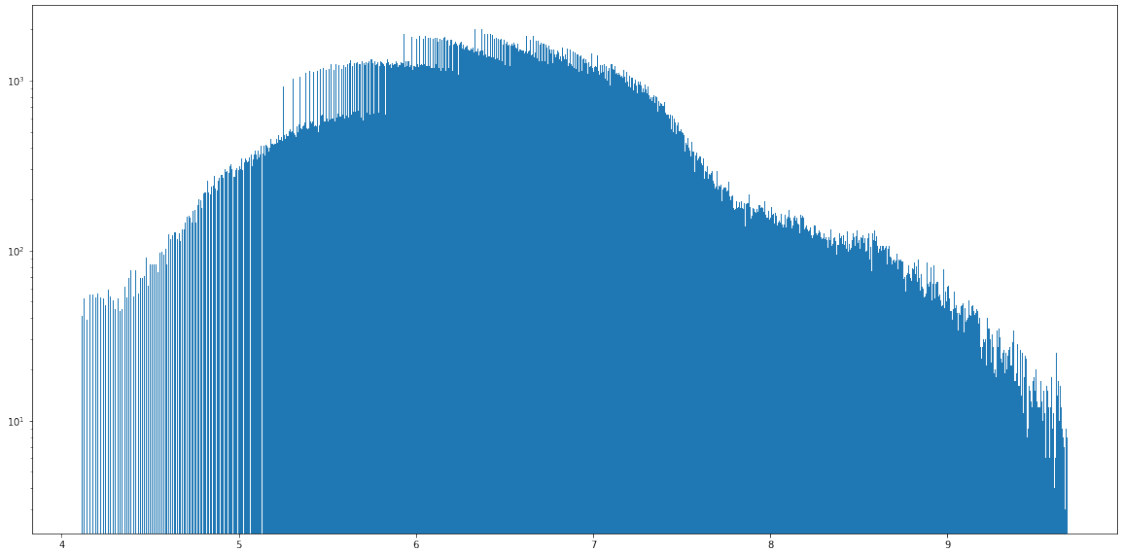
[7]:
```
plt.rcParams['figure.figsize'] = 20, 10
```

[5]:
```
_=plt.hist(bikes[bikes.dur<16000].dur, log=True, bins=1000)
```



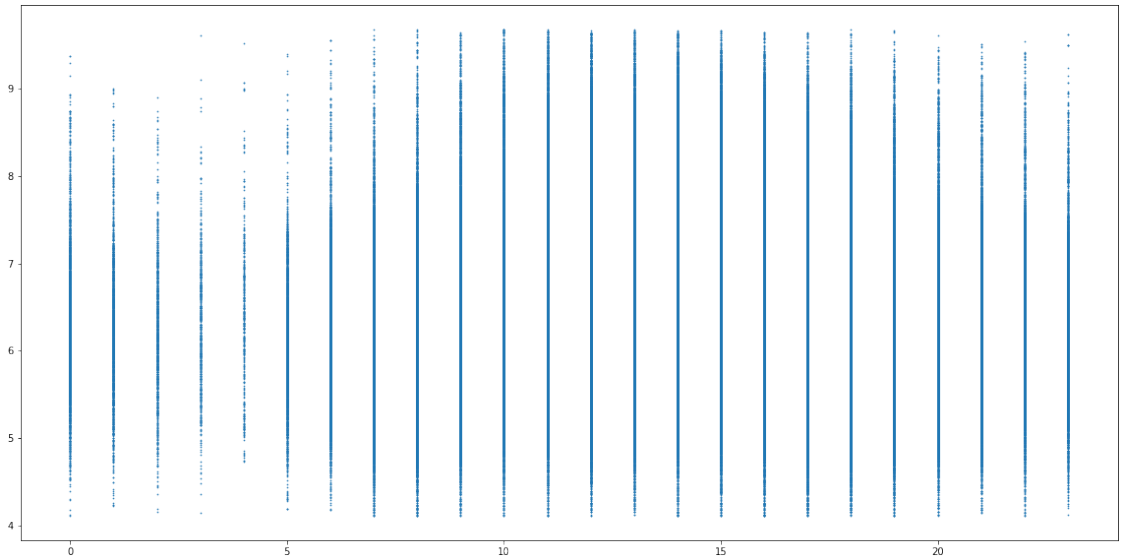
[6]:
```
short = bikes[bikes.dur<16000]
```

[7]:
```
_=plt.hist(np.log1p(short.dur), log=True, bins=1000)
```



[8]:
```
plt.scatter(short.start.dt.hour, np.log1p(short.dur), s=.4)
```

[8]:
```
<matplotlib.collections.PathCollection at 0x7fe598520610>
```



[9]:
```
np.log1p(0), np.log(0)
```

```
/var/folders/xs/2nmb23_93dzdp7pp9j4yl1yr0000gn/T/ipykernel_57323/1076539907.py:1
: RuntimeWarning: divide by zero encountered in log
  np.log1p(0), np.log(0)
```

[9]:
```
(0.0, -inf)
```

[10]:
```
bikes['log_dur'] = np.round(np.log1p(bikes.dur), 1)
```

[11]:
```
monday = bikes[bikes.start.dt.dayofweek==1]
```

[12]:
```
dur_hour = monday.groupby(['log_dur', monday.start.dt.hour]).count()
```

[13]:
```
dur_hour
```

[13]:

		Duration (ms)	Start date	End date	Start station number	\
log_dur	start					
4.1	7	1	1	1	1	
	9	2	2	2	2	
	11	1	1	1	1	
	14	2	2	2	2	
	16	2	2	2	2	
…	…	…	…	…	…	
11.2	21	2	2	2	2	
11.3	14	1	1	1	1	
	17	1	1	1	1	
	19	1	1	1	1	
11.4	18	1	1	1	1	

		Start station	End station number	End station	Bike number	\
log_dur	start					
4.1	7	1	1	1	1	
	9	2	2	2	2	
	11	1	1	1	1	
	14	2	2	2	2	
	16	2	2	2	2	
…	…	…	…	…	…	
11.2	21	2	2	2	2	
11.3	14	1	1	1	1	
	17	1	1	1	1	
	19	1	1	1	1	
11.4	18	1	1	1	1	

		Member Type	start	end	dur
log_dur	start				
4.1	7	1	1	1	1
	9	2	2	2	2
	11	1	1	1	1

```
        14              2      2    2    2
        16              2      2    2    2
…                   …     …   …   …
11.2    21              2      2    2    2
11.3    14              1      1    1    1
        17              1      1    1    1
        19              1      1    1    1
11.4    18              1      1    1    1

[1184 rows x 12 columns]
```

[14]:
```
duration_hour = dur_hour.start.unstack().T.fillna(0)
duration_hour
```

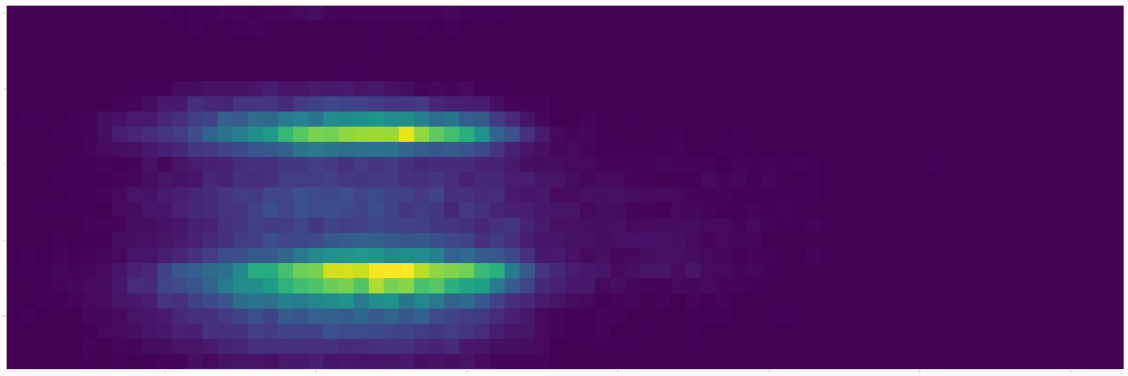
[14]:
```
log_dur  4.1   4.2   4.3   4.4   4.5   4.6   4.7   4.8   4.9   5.0   … \
start                                                                  …
0         0.0   0.0   0.0   0.0   0.0   1.0   1.0   1.0   2.0   3.0  …
1         0.0   0.0   1.0   1.0   0.0   0.0   0.0   0.0   3.0   1.0  …
2         0.0   0.0   0.0   0.0   0.0   0.0   0.0   2.0   0.0   0.0  …
3         0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0  …
4         0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0   1.0  …
5         0.0   0.0   1.0   0.0   0.0   1.0   4.0   1.0   7.0   6.0  …
6         0.0   0.0   0.0   2.0   1.0   2.0   4.0   9.0  11.0  21.0  …
7         1.0   5.0   4.0   1.0   5.0  12.0  25.0  31.0  46.0  46.0  …
8         0.0   3.0   2.0   6.0   7.0  11.0  22.0  52.0  68.0  79.0  …
9         2.0   3.0   2.0   4.0   3.0  11.0  18.0  22.0  28.0  42.0  …
10        0.0   0.0   1.0   3.0   5.0   7.0   8.0   5.0  10.0  31.0  …
11        1.0   0.0   2.0   5.0   4.0   7.0   7.0  10.0  13.0  22.0  …
12        0.0   0.0   4.0   2.0   7.0   6.0  12.0  16.0  36.0  30.0  …
13        0.0   2.0   6.0   3.0   5.0   6.0   4.0  15.0  20.0  36.0  …
14        2.0   0.0   1.0   1.0   3.0   8.0   9.0  11.0  26.0  24.0  …
15        0.0   3.0   0.0   5.0   1.0   7.0   6.0  22.0  26.0  31.0  …
16        2.0   6.0   1.0  11.0   6.0  10.0  14.0  17.0  36.0  35.0  …
17        3.0   7.0   7.0  13.0  12.0  14.0  20.0  36.0  57.0  71.0  …
18        0.0   4.0   7.0   9.0  13.0  20.0  21.0  40.0  79.0  75.0  …
19        3.0   0.0   7.0   7.0   9.0  16.0  19.0  34.0  43.0  52.0  …
20        0.0   7.0   2.0   4.0   2.0  13.0  14.0  19.0  34.0  38.0  …
21        1.0   2.0   1.0   2.0   3.0   6.0  16.0  19.0  26.0  35.0  …
22        1.0   0.0   2.0   2.0   1.0   8.0   1.0  13.0  10.0  20.0  …
23        0.0   0.0   1.0   0.0   2.0   5.0   4.0   8.0   3.0   5.0  …

log_dur  10.5  10.6  10.7  10.8  10.9  11.0  11.1  11.2  11.3  11.4
start
0         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3         0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
```

4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	4.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	3.0	1.0	1.0	1.0	0.0	1.0	0.0
18	0.0	0.0	2.0	4.0	1.0	0.0	1.0	1.0	0.0	1.0
19	0.0	1.0	2.0	3.0	0.0	1.0	0.0	0.0	1.0	0.0
20	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
21	1.0	2.0	0.0	1.0	0.0	0.0	1.0	2.0	0.0	0.0
22	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

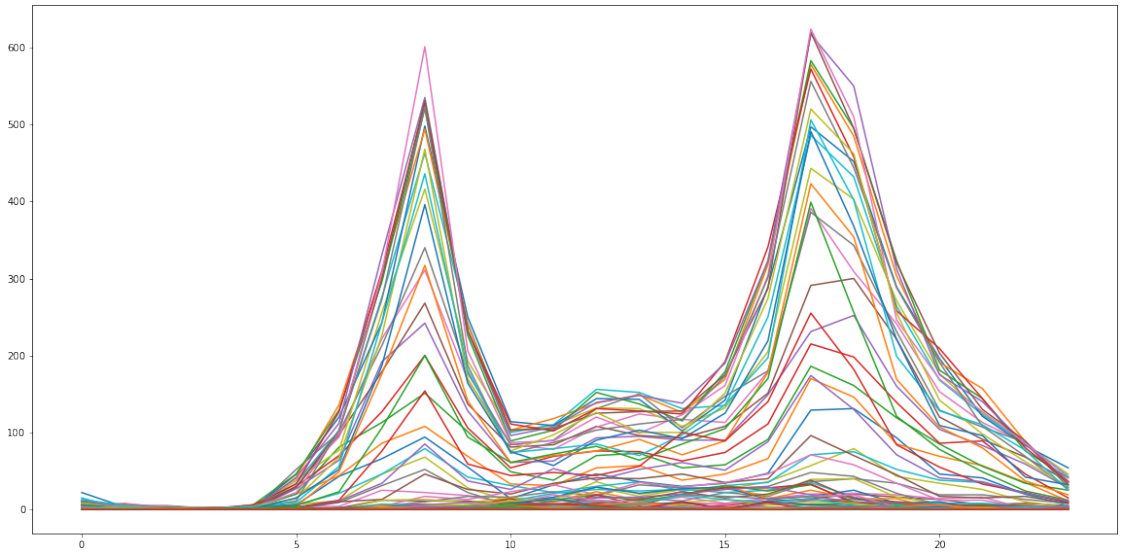
[24 rows x 74 columns]

[15]:
```
plt.figure(figsize=(100,100))
plt.imshow(duration_hour)
```

[15]: <matplotlib.image.AxesImage at 0x7fe59857aaf0>



[16]:
```
_=plt.plot(duration_hour)
```



[20]:
```
bikes['Member Type'].value_counts()
```

[20]:
```
Registered    467432
Casual         84967
Name: Member Type, dtype: int64
```

1.1.2 Create a new column that represents the hour+minute of the day as a fraction (i.e. 1:30pm = 13.5)

[21]:
```
np.round(.65, 1)
```

[21]:
```
0.6
```

[22]:
```
37//6, (37//6)/10, 37/60
```

[22]:
```
(6, 0.6, 0.6166666666666667)
```

[17]:
```
bikes['hour_of_day'] = (bikes.start.dt.hour + (bikes.start.dt.minute//6)/10)
```

[26]:
```
bikes['roundhour_of_day'] = (bikes.start.dt.hour ) # keep the hour handy as well

bikes.head(20)
```

[26]:

	Duration (ms)	Start date	End date	Start station number	\
0	301295	3/31/2016 23:59	4/1/2016 0:04	31280	
1	557887	3/31/2016 23:59	4/1/2016 0:08	31275	
2	555944	3/31/2016 23:59	4/1/2016 0:08	31101	
3	766916	3/31/2016 23:57	4/1/2016 0:09	31226	

```
4          139656  3/31/2016 23:57  3/31/2016 23:59                31011
5          967713  3/31/2016 23:57    4/1/2016 0:13                31266
6          534836  3/31/2016 23:57    4/1/2016 0:06                31222
7          243864  3/31/2016 23:56    4/1/2016 0:00                31228
8          372524  3/31/2016 23:55    4/1/2016 0:01                31113
9          215194  3/31/2016 23:55  3/31/2016 23:59                31263
10         498903  3/31/2016 23:55    4/1/2016 0:03                31243
11         389082  3/31/2016 23:54    4/1/2016 0:01                31079
12        1680745  3/31/2016 23:54    4/1/2016 0:22                31258
13        1687026  3/31/2016 23:54    4/1/2016 0:23                31258
14         544541  3/31/2016 23:53    4/1/2016 0:02                31245
15        1001144  3/31/2016 23:51    4/1/2016 0:08                31106
16        1262663  3/31/2016 23:51    4/1/2016 0:12                31111
17         451821  3/31/2016 23:51  3/31/2016 23:59                31613
18         305172  3/31/2016 23:50  3/31/2016 23:55                31269
19        5230964  3/31/2016 23:50    4/1/2016 1:17                31248

                                          Start station  End station number  \
0                                        11th & S St NW                31506
1                         New Hampshire Ave & 24th St NW                31114
2                                        14th & V St NW                31221
3                            34th St & Wisconsin Ave NW                31214
4                                    23rd & Crystal Dr                 31009
5                                       11th & M St NW                 31600
6                             New York Ave & 15th St NW                 31278
7                                        8th & H St NW                 31600
8                           Columbia Rd & Belmont St NW                 31234
9                                       10th & K St NW                 31265
10                        Maryland & Independence Ave SW                31209
11                              Lee Hwy & N Cleveland St                31093
12                                      Lincoln Memorial                31269
13                                      Lincoln Memorial                31269
14                        7th & R St NW / Shaw Library                  31505
15                            Calvert & Biltmore St NW                  31226
16                                      10th & U St NW                  31226
17  Eastern Market Metro / Pennsylvania Ave & 7th …                    31617
18                           3rd St & Pennsylvania Ave SE               31639
19               Smithsonian / Jefferson Dr & 12th St SW                31248

                                End station Bike number Member Type  \
0                    1st & Rhode Island Ave NW      W00022  Registered
1                     18th St & Wyoming Ave NW      W01294  Registered
2                               18th & M St NW      W01416  Registered
3                          17th & Corcoran St NW      W01090  Registered
4                              27th & Crystal Dr      W21934  Registered
5                                  5th & K St NW      W20562      Casual
6                               18th & R St NW      W20222  Registered
```

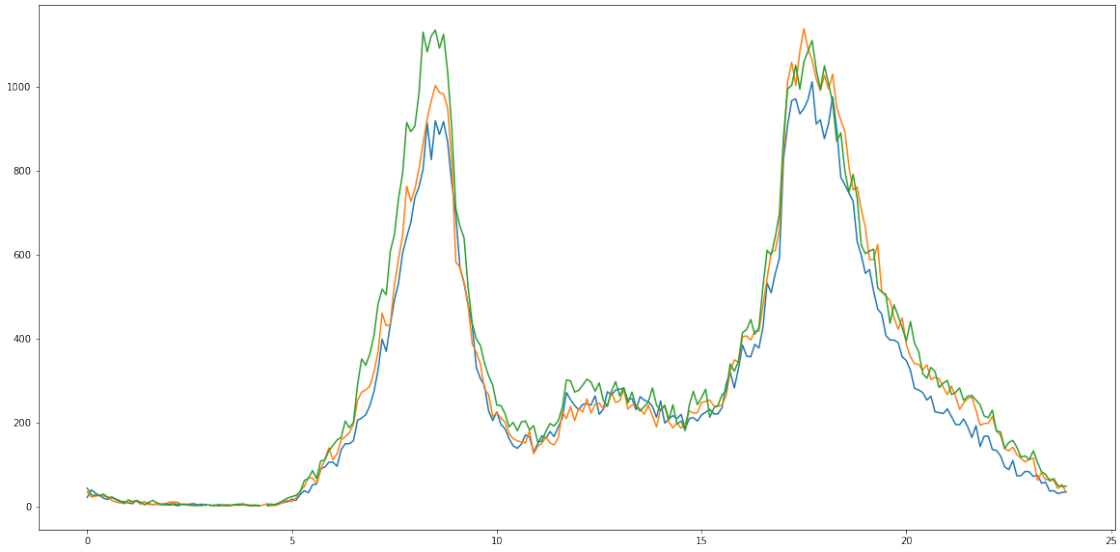
```
7                            5th & K St NW      W20291  Registered
8              20th & O St NW / Dupont South      W20590  Registered
9              5th St & Massachusetts Ave NW      W21876  Registered
10                           1st & N St  SE      W20973  Registered
11                 21st St N & N Pierce St       W01197  Registered
12              3rd St & Pennsylvania Ave SE      W01191      Casual
13              3rd St & Pennsylvania Ave SE      W20449      Casual
14                    Eckington Pl & Q St NE      W20888  Registered
15                34th St & Wisconsin Ave NW      W22196      Casual
16                34th St & Wisconsin Ave NW      W21553      Casual
17            Bladensburg Rd & Benning Rd NE      W20614  Registered
18                           2nd & G St NE        W22068  Registered
19  Smithsonian / Jefferson Dr & 12th St SW       W01458      Casual

                 start                 end   dur  log_dur  hour_of_day  \
0  2016-03-31 23:59:00 2016-04-01 00:04:00   301      5.7         23.9
1  2016-03-31 23:59:00 2016-04-01 00:08:00   557      6.3         23.9
2  2016-03-31 23:59:00 2016-04-01 00:08:00   555      6.3         23.9
3  2016-03-31 23:57:00 2016-04-01 00:09:00   766      6.6         23.9
4  2016-03-31 23:57:00 2016-03-31 23:59:00   139      4.9         23.9
5  2016-03-31 23:57:00 2016-04-01 00:13:00   967      6.9         23.9
6  2016-03-31 23:57:00 2016-04-01 00:06:00   534      6.3         23.9
7  2016-03-31 23:56:00 2016-04-01 00:00:00   243      5.5         23.9
8  2016-03-31 23:55:00 2016-04-01 00:01:00   372      5.9         23.9
9  2016-03-31 23:55:00 2016-03-31 23:59:00   215      5.4         23.9
10 2016-03-31 23:55:00 2016-04-01 00:03:00   498      6.2         23.9
11 2016-03-31 23:54:00 2016-04-01 00:01:00   389      6.0         23.9
12 2016-03-31 23:54:00 2016-04-01 00:22:00  1680      7.4         23.9
13 2016-03-31 23:54:00 2016-04-01 00:23:00  1687      7.4         23.9
14 2016-03-31 23:53:00 2016-04-01 00:02:00   544      6.3         23.8
15 2016-03-31 23:51:00 2016-04-01 00:08:00  1001      6.9         23.8
16 2016-03-31 23:51:00 2016-04-01 00:12:00  1262      7.1         23.8
17 2016-03-31 23:51:00 2016-03-31 23:59:00   451      6.1         23.8
18 2016-03-31 23:50:00 2016-03-31 23:55:00   305      5.7         23.8
19 2016-03-31 23:50:00 2016-04-01 01:17:00  5230      8.6         23.8

    roundhour_of_day
0                 23
1                 23
2                 23
3                 23
4                 23
5                 23
6                 23
7                 23
8                 23
9                 23
```

10	23
11	23
12	23
13	23
14	23
15	23
16	23
17	23
18	23
19	23

1.1.3 Aggregate to get a count per hour/minute of the day across all trips

[37]:
```
reg_bikes = bikes[bikes['Member Type']=='Registered']
hours = reg_bikes.groupby([reg_bikes.hour_of_day, reg_bikes.start.dt.
 ↪dayofweek]).agg('count')
#hours['hour'] = hours.index

day_hour_count = hours.dur.unstack()
plt.figure(figsize=(20,10))
plt.plot(day_hour_count.index, day_hour_count[0])
plt.plot(day_hour_count.index, day_hour_count[1])
plt.plot(day_hour_count.index, day_hour_count[2])
# plt.plot(y.index, day_hour_count[3])
# plt.plot(y.index, day_hour_count[4])
# plt.plot(y.index, day_hour_count[5])
# plt.plot(y.index, day_hour_count[6])
```

[37]:
```
[<matplotlib.lines.Line2D at 0x7fe597c90160>]
```

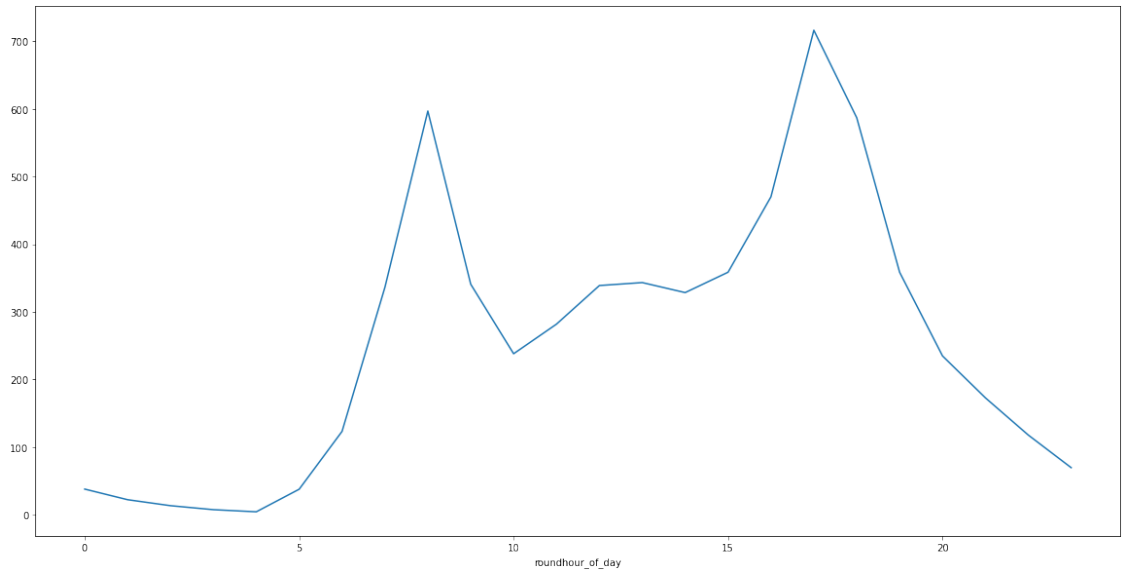


[27]:
```
day_hour_count
```

[27]:
```
start           0     1     2      3      4      5      6
hour_of_day
0.0          21.0  34.0  43.0   47.0   51.0   89.0  106.0
0.1          39.0  22.0  27.0   37.0   56.0   87.0  100.0
0.2          31.0  24.0  26.0   42.0   50.0   98.0   77.0
0.3          26.0  27.0  25.0   29.0   52.0   99.0   87.0
0.4          19.0  24.0  29.0   29.0   50.0   98.0   69.0
…             …     …     …      …      …      …
23.5         36.0  65.0  60.0   94.0   80.0   93.0   28.0
23.6         37.0  61.0  66.0  100.0   81.0   95.0   28.0
23.7         30.0  42.0  49.0   80.0  101.0  105.0   27.0
23.8         33.0  52.0  47.0   79.0   91.0   93.0   24.0
23.9         34.0  33.0  48.0   65.0  105.0  111.0   23.0

[240 rows x 7 columns]
```

[38]:
```
hoursn = bikes.groupby('roundhour_of_day').agg('count')
hoursn['hour'] = hoursn.index
(hoursn.start/90).plot() # 90 days in a quarter
```

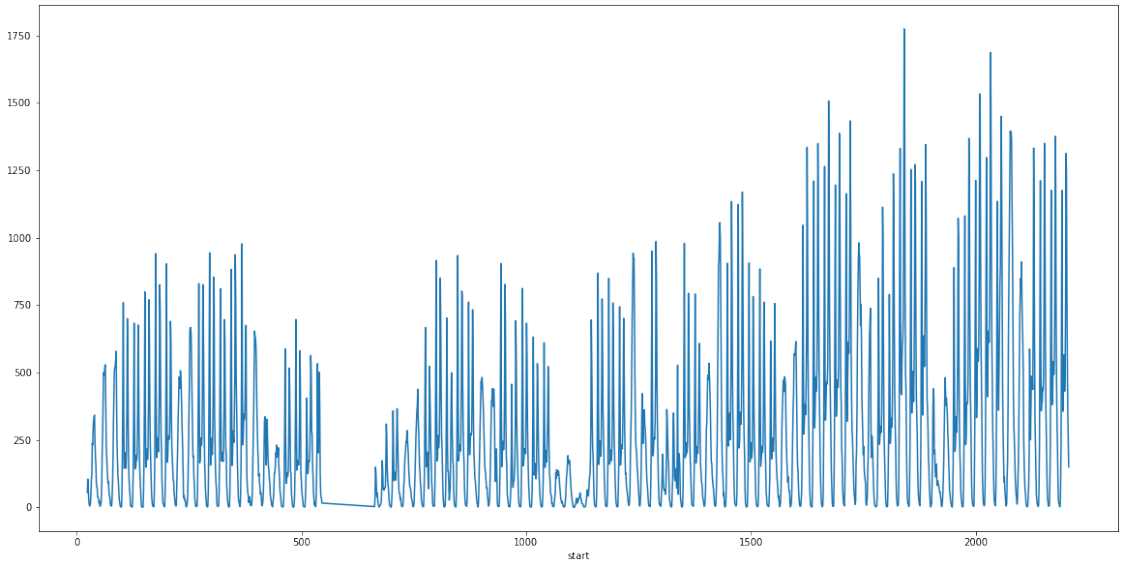
[38]:
```
<AxesSubplot:xlabel='roundhour_of_day'>
```



[29]:
```
hour_count = bikes.groupby(bikes.start.dt.dayofyear*24 + bikes.start.dt.hour).
↪count()
```

[30]:
```
plt.figure(figsize=(20,10))
hour_count.start.plot()
```

[30]:
```
<AxesSubplot:xlabel='start'>
```



[31]:
```
day_count = bikes.groupby(bikes.start.dt.dayofyear).count()
```

[32]:
```
day_hour = bikes.groupby([bikes.start.dt.dayofyear, bikes.start.dt.hour]).
↪count()
```

[33]:
```
day_hour.start.unstack()
```

[33]:

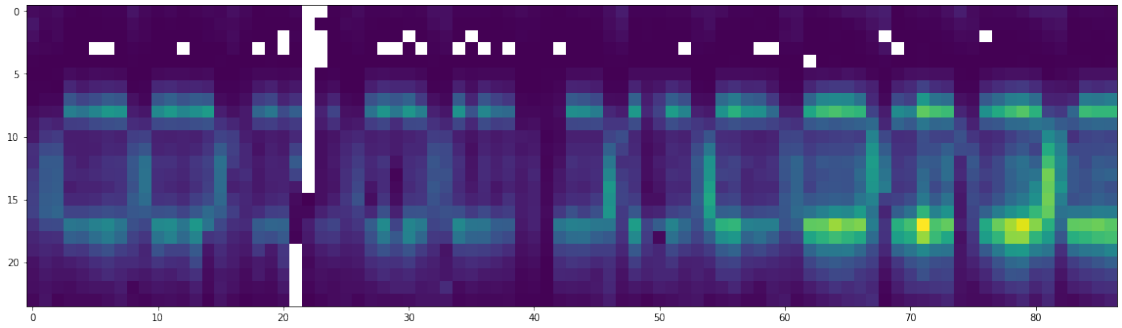
start	0	1	2	3	4	5	6	7	8	9	…	\
start											…	
1	56.0	105.0	74.0	32.0	13.0	5.0	10.0	14.0	54.0	101.0	…	
2	37.0	31.0	17.0	23.0	4.0	7.0	10.0	34.0	80.0	203.0	…	
3	59.0	42.0	39.0	15.0	6.0	9.0	5.0	33.0	87.0	168.0	…	
4	20.0	6.0	2.0	1.0	3.0	58.0	192.0	468.0	759.0	321.0	…	
5	5.0	5.0	3.0	1.0	2.0	42.0	131.0	363.0	683.0	329.0	…	
…	…	…	…	…	…	…	…	…	…	…	…	
87	113.0	82.0	50.0	34.0	12.0	24.0	94.0	166.0	297.0	509.0	…	
88	15.0	7.0	2.0	3.0	8.0	42.0	81.0	197.0	587.0	464.0	…	
89	31.0	11.0	9.0	3.0	8.0	79.0	240.0	727.0	1211.0	564.0	…	
90	31.0	18.0	4.0	6.0	7.0	79.0	215.0	703.0	1176.0	593.0	…	
91	28.0	16.0	10.0	2.0	8.0	80.0	240.0	750.0	1175.0	589.0	…	

start	14	15	16	17	18	19	20	21	22	23
start										

```
1      324.0  338.0  342.0   247.0   185.0  160.0   90.0   75.0   70.0   39.0
2      495.0  525.0  529.0   392.0   232.0  188.0  150.0  114.0   91.0   96.0
3      524.0  546.0  579.0   398.0   237.0  172.0  115.0   96.0   64.0   28.0
4      145.0  206.0  365.0   700.0   547.0  293.0  146.0   96.0   62.0   44.0
5      175.0  208.0  365.0   676.0   519.0  279.0  178.0  122.0   86.0   45.0
…        …      …      …       …       …      …      …      …      …      …
87     910.0  761.0  667.0   611.0   475.0  243.0  158.0  101.0   62.0   51.0
88     481.0  437.0  696.0  1332.0  1113.0  620.0  324.0  226.0  148.0   45.0
89     433.0  473.0  700.0  1350.0  1159.0  700.0  400.0  279.0  178.0   82.0
90     493.0  545.0  749.0  1376.0  1215.0  722.0  468.0  312.0  231.0  108.0
91     431.0  504.0  746.0  1312.0  1241.0  806.0  536.0  345.0  240.0  150.0

[87 rows x 24 columns]
```

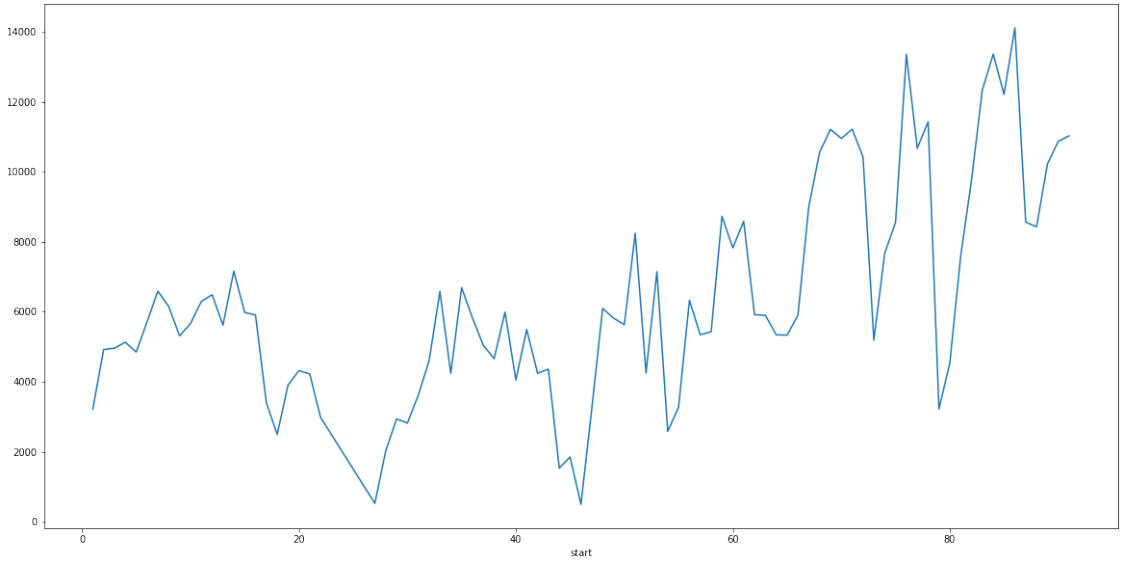
[34]:
```
plt.figure(figsize=(20,10))
plt.imshow(day_hour.start.unstack().T)
```

[34]:
```
<matplotlib.image.AxesImage at 0x13e55b430>
```



[35]:
```
day_count.start.plot()
```

[35]:
```
<AxesSubplot:xlabel='start'>
```



[36]:
```
bikes.start.dt.dayofyear
```

[36]:
```
0         91
1         91
2         91
3         91
4         91
          ..
552394     1
552395     1
552396     1
552397     1
552398     1
Name: start, Length: 552399, dtype: int64
```

[ ]:
```
bikes[bikes.start=="2016-01-10"].shape
```

2 Assignment 4

Explain the results in a **paragraph + charts** of to describe which model you'd recommend. This means show the data and the model's line on the same chart. The paragraph is a simple justification and comparison of the several models you tried.

3 1. Using the `day_hour_count` dataframe create two dataframes `monday` and `saturday` that represent the data for those days. (hint: Monday is day=0)

[116]:
```
#Monday
monday = day_hour_count[[0]].copy()
monday["hour"] = monday.index

#Saturaday
saturday = day_hour_count[[5]].copy()
saturday["hour"] = saturday.index
```

[101]:
```

```

[128]:
```

```

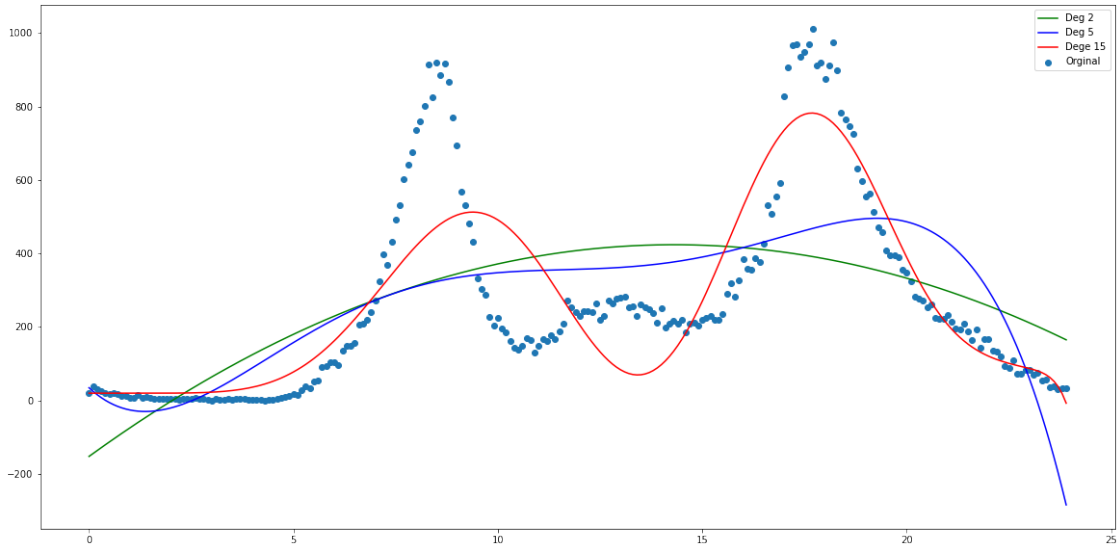
3.1 2a. Create 3 models fit to `monday.hour_of_day` with varying polynomial degrees (choose from `n=1,2,3,5,10,15`). (Repeat for `saturday` below)

3.2 Plot all the results for each polynomial.

[212]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

X = day_hour_count.index

y = monday[0].fillna(0)

ln_model = LinearRegression()

# Monday Degree 2
X = np.array(day_hour_count.index)
y = monday[0].fillna(0)

poly_2 = PolynomialFeatures(degree=2)
X_2 = poly_2.fit_transform(X.reshape(-1, 1))
ln_model.fit(X_2, y)
(linear_model.coef_, linear_model.intercept_)
linear_coef_2, linear_intercept_2 = ln_model.coef_, ln_model.intercept_

#Monday Degree 5
poly_5 = PolynomialFeatures(degree=5)
X_5 = poly_5.fit_transform(X.reshape(-1, 1))
ln_model.fit(X_5, y)
(linear_model.coef_, linear_model.intercept_)
linear_coef_5, linear_intercept_5 = ln_model.coef_, ln_model.intercept_
```

```
#Monday Degree 155
poly_15 = PolynomialFeatures(degree=15)
X_15 = poly_15.fit_transform(X.reshape(-1, 1))
ln_model.fit(X_15, y)
linear_coef_15, linear_intercept_15 = ln_model.coef_, ln_model.intercept_

plt.scatter(X,y, label = 'Orginal')
plt.plot(X, np.dot(X_2, linear_coef_2) + linear_intercept_2, c='g', label =␣
↪'Deg 2')
plt.plot(X, np.dot(X_5, linear_coef_5) + linear_intercept_5, c='b', label =␣
↪'Deg 5')
plt.plot(X, np.dot(X_15, linear_coef_15) + linear_intercept_15, c='r', label =␣
↪'Dege 15')

plt.legend()
```

[212]: <matplotlib.legend.Legend at 0x7fe57ec44e80>



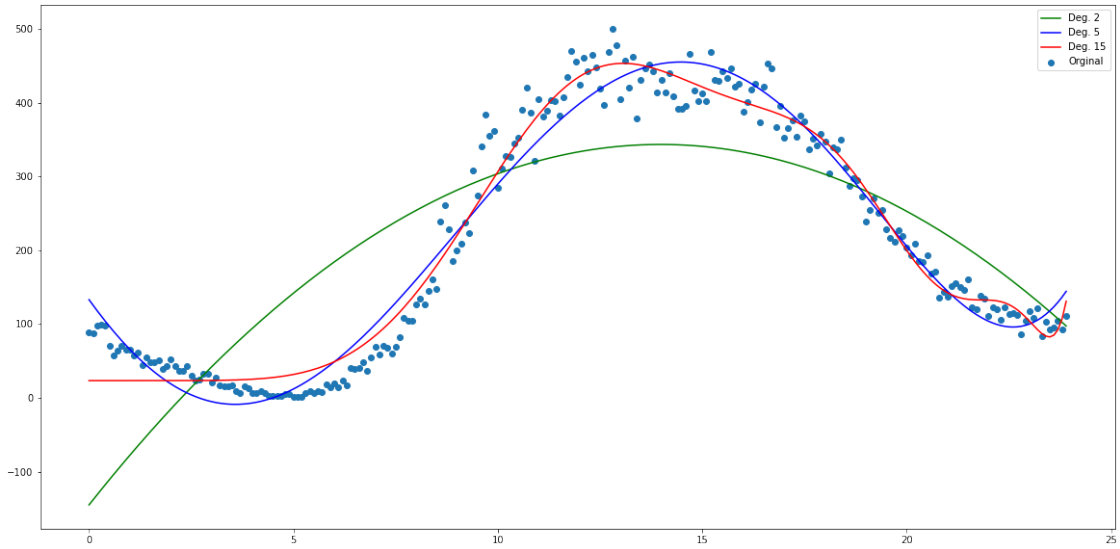
Saturday Ploynomial Degree 15 gives the fitting. It's the most recommended model althought might cause overfitting.

3.3 2b. Repeat 2a for `saturday.hour_of_day`

[208]:
```
# Saturday Degree 2
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

```
linear_model = LinearRegression()

X = np.array(day_hour_count.index)
y = saturday[5].fillna(0)

poly_2 = PolynomialFeatures(degree=2)
X_2 = poly_2.fit_transform(X.reshape(-1, 1))
linear_model.fit(X_2, y)
linear_coef_2, linear_intercept_2 = linear_model.coef_, linear_model.intercept_

#Saturday Degree 5
poly_5 = PolynomialFeatures(degree=5)
X_5 = poly_5.fit_transform(X.reshape(-1, 1))
linear_model.fit(X_5, y)
linear_coef_5, linear_intercept_5 = linear_model.coef_, linear_model.intercept_


#Saturday Degree 15
poly_15 = PolynomialFeatures(degree=15)
X_15 = poly_15.fit_transform(X.reshape(-1, 1))
linear_model.fit(X_15, y)
linear_coef_15, linear_intercept_15 = linear_model.coef_, linear_model.
 ↪intercept_


plt.scatter(X,y, label = 'Orginal')
plt.plot(X, np.dot(X_2, linear_coef_2) + linear_intercept_2, c='g', label =␣
 ↪'Deg. 2')
plt.plot(X, np.dot(X_5, linear_coef_5) + linear_intercept_5, c='b', label =␣
 ↪'Deg. 5')
plt.plot(X, np.dot(X_15, linear_coef_15) + linear_intercept_15, c='r', label =␣
 ↪'Deg. 15')
#plt.legend(handles = ['Degree 2', 'Degree 5', 'Degree 15'])
plt.legend()
```

[208]: <matplotlib.legend.Legend at 0x7fe57c623580>

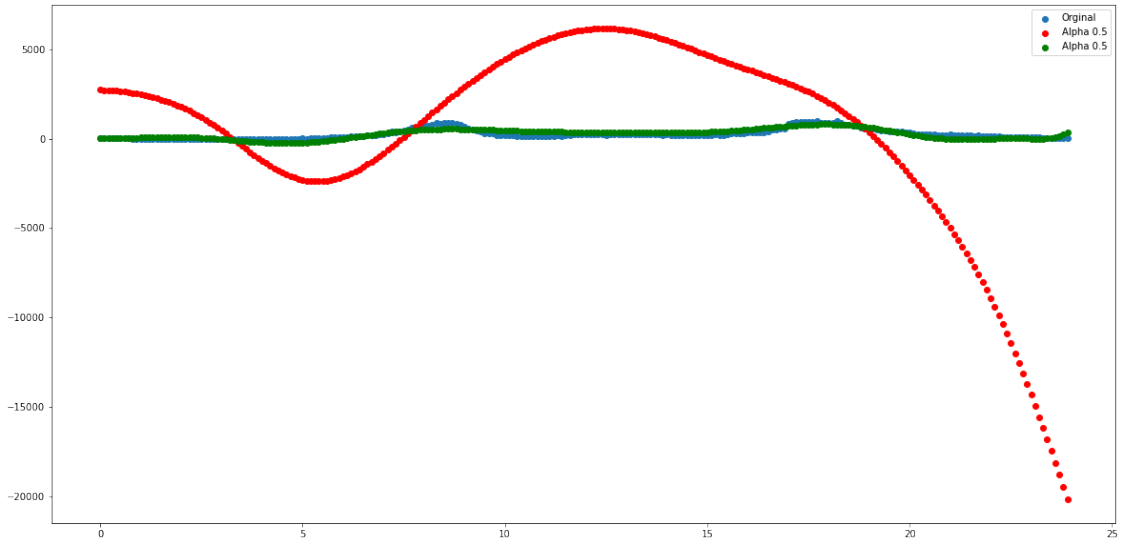


Saturday Ploynomial Degree 5 and Degree 15 give relatively better fitting. As we can see from the chart above, Degree 5 is the best model. however, both dgree 5 and 15, might cause overfitting.

3.4 3. create 3 new models fit to `hour_of_day` with different Ridge Regression α (alpha) Ridge Coefficient values using the `monday` and `saturday` datasets.

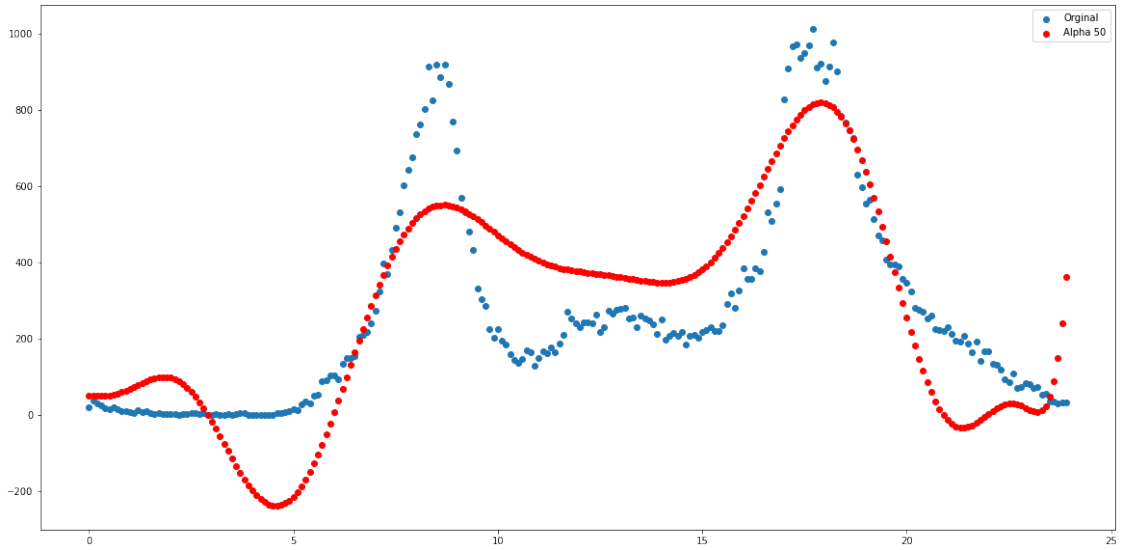
[216]:
```
#Monday
#We'll use Poly 15 (since it produces the best result above) to test it with␣
 ↪different alpha values

from sklearn import linear_model

#Alpha = 0.5
ridge_05 = linear_model.Ridge(alpha = 0.5)
ridge_05 = ridge_05.fit(X_15, y)

plt.scatter(X, y, label = 'Orginal')
plt.scatter(X, ridge_05.predict(X_15), c='r', label = 'Alpha 0.5')


plt.legend()
```

[216]:
```
<matplotlib.legend.Legend at 0x7fe57f7e87c0>
```



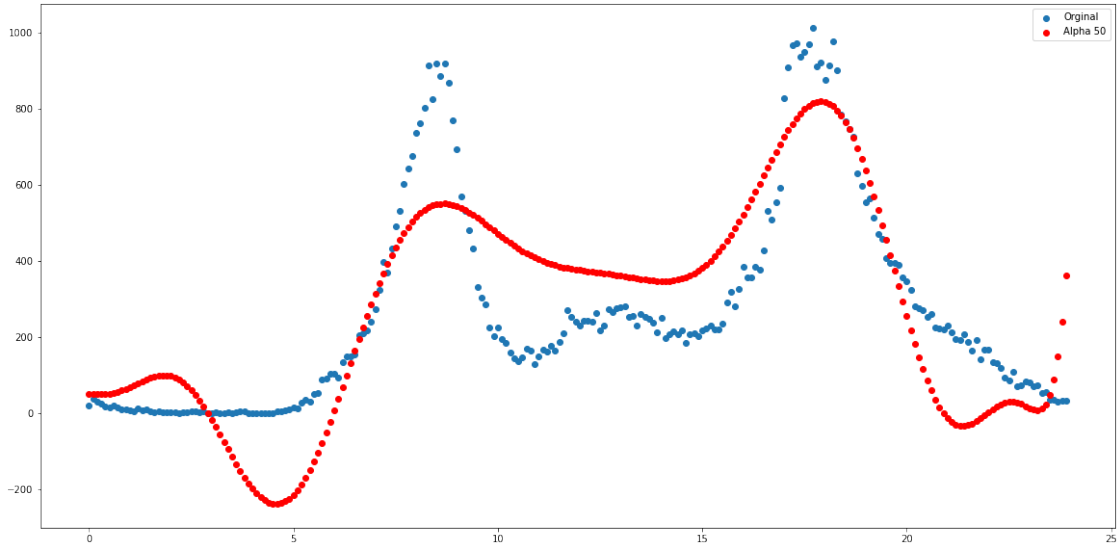
[218]:
```
#Alpha = 50
ridge_50 = linear_model.Ridge(alpha = 50)
ridge_50 = ridge_50.fit(X_15, y)
plt.scatter(X, y, label = 'Orginal')
plt.scatter(X, ridge_50.predict(X_15), c='r', label = 'Alpha 50')
plt.legend()
```

[218]: <matplotlib.legend.Legend at 0x7fe58011ffd0>



[219]:
```
#Alpha = 1000
ridge_1000 = linear_model.Ridge(alpha = 50)
ridge_1000 = ridge_1000.fit(X_15, y)
plt.scatter(X, y, label = 'Orginal')
plt.scatter(X, ridge_1000.predict(X_15), c='r', label = 'Alpha 50')
plt.legend()
```

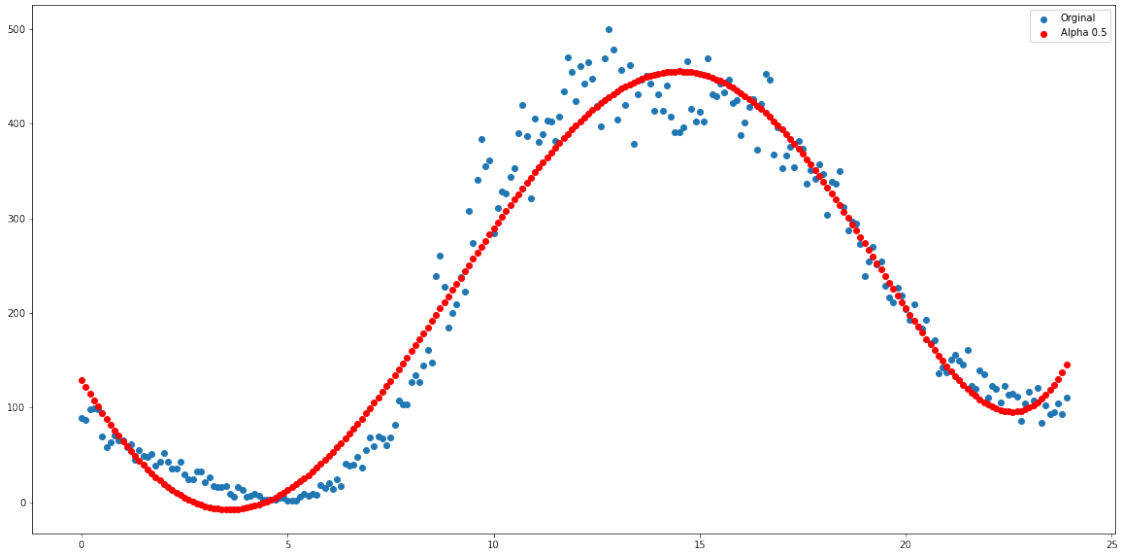
[219]:
```
<matplotlib.legend.Legend at 0x7fe580443e20>
```



Monday Ridge Using Ploynomial degree 15, low alpha values tend to overshot. Higher alpha values (alpha > ~50) give much better fitting as we can see in the charts above.

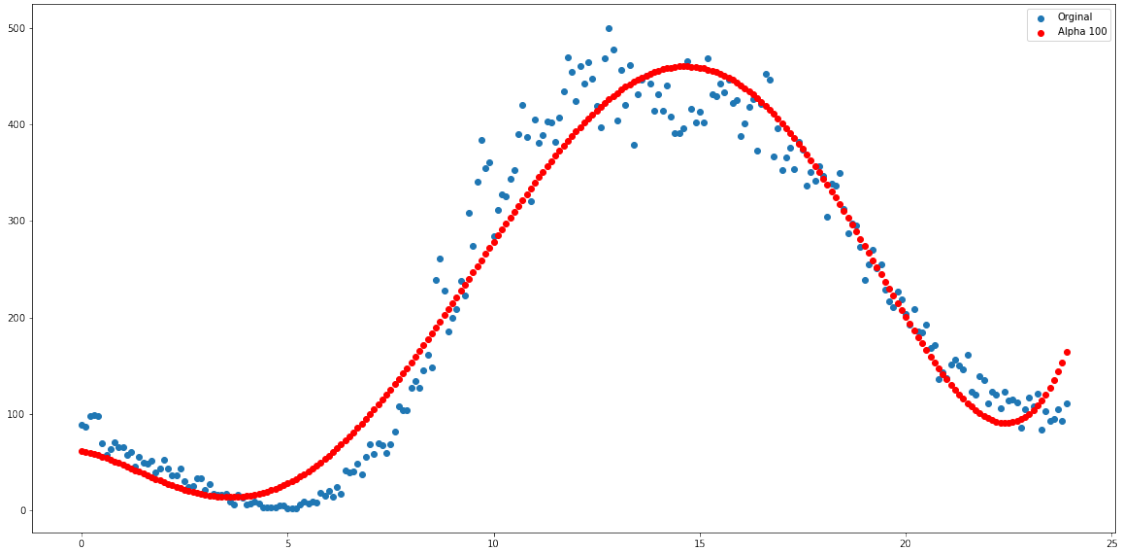
[220]:
```
#Saturday
#We'll use Poly 5 (since it produces the best result above) to test it with␣
 ↪different alpha values

#Alpha = 0.5
X = np.array(day_hour_count.index)
y = saturday[5].fillna(0)
poly_5 = PolynomialFeatures(degree=5)
X_5 = poly_5.fit_transform(X.reshape(-1, 1))


ridge = linear_model.Ridge(alpha = 0.5)
ridge_5 = ridge.fit(X_5, y)
plt.scatter(X, y, label = 'Orginal')
plt.scatter(X, ridge_5.predict(X_5), c='r', label = 'Alpha 0.5')

plt.legend()
```

[220]: <matplotlib.legend.Legend at 0x7fe580758c10>



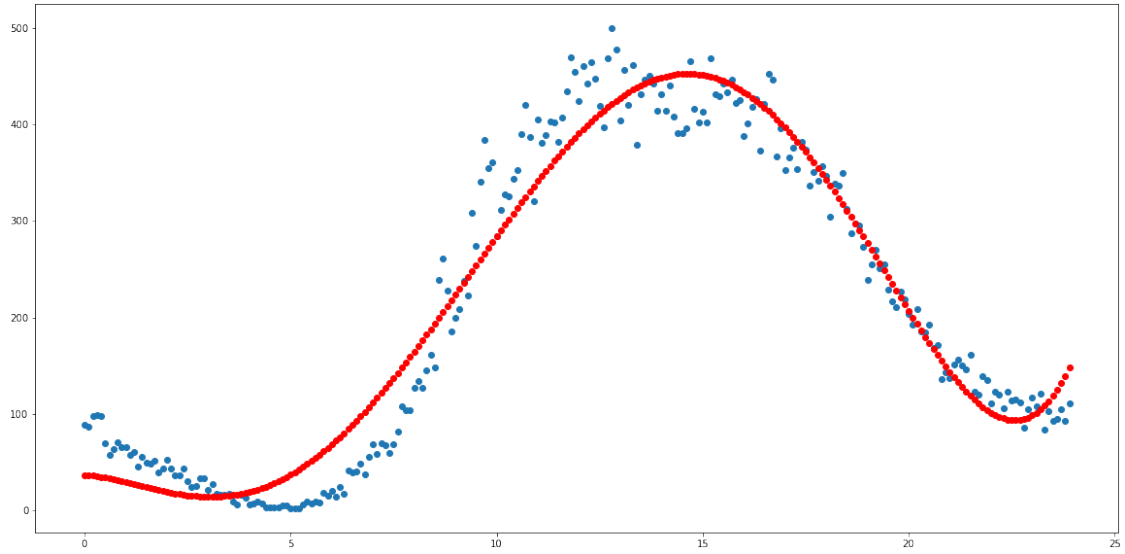
[222]:
```
#Alpha = 100
X = np.array(day_hour_count.index)
y = saturday[5].fillna(0)
poly_5 = PolynomialFeatures(degree=5)
X_5 = poly_5.fit_transform(X.reshape(-1, 1))


ridge = linear_model.Ridge(alpha = 100)
ridge_5 = ridge.fit(X_5, y)
plt.scatter(X, y, label = 'Orginal')
plt.scatter(X, ridge_5.predict(X_5), c='r', label = 'Alpha 100')

plt.legend()
```

[222]: <matplotlib.legend.Legend at 0x7fe580addd60>



[223]:
```
#Alpha = 100
X = np.array(day_hour_count.index)
y = saturday[5].fillna(0)
poly_5 = PolynomialFeatures(degree=5)
X_5 = poly_5.fit_transform(X.reshape(-1, 1))


ridge = linear_model.Ridge(alpha = 1000)
ridge_5 = ridge.fit(X_5, y)
plt.scatter(X, y, label = 'Orginal')
plt.scatter(X, ridge_5.predict(X_5), c='r', label = 'Alpha 1000')
```

[223]:
```
<matplotlib.collections.PathCollection at 0x7fe580deeaf0>
```



Monday Ridge Using Ploynomial degree 5, changing alpha values hasn't changed our model's prediction significantly.

[ ]: