4-Copy1

September 30, 2018

1 Instructions

The following Cells need to be executed.

They are used to download and generate a dataset that has an aggregated count of bike trips per hundredth of an hour through the 24 hours in a day.

I put all this here instead of providing you the dataset directly, so that you could learn something along the way :)

The assignment is in the last cell.

```
In [1]: !pip3 install seaborn

Requirement already satisfied: seaborn in /usr/local/lib/python3.7/site-packages (0.9.0)
Requirement already satisfied: matplotlib>=1.4.3 in /usr/local/lib/python3.7/site-packages (from
Requirement already satisfied: pandas>=0.15.2 in /usr/local/lib/python3.7/site-packages (from se
Requirement already satisfied: numpy>=1.9.3 in /usr/local/lib/python3.7/site-packages (from seab
Requirement already satisfied: scipy>=0.14.0 in /usr/local/lib/python3.7/site-packages (from sea
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/site-packages (from matp
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/site-packages (f
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/site-packages (from
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.7/site-packages (from panda
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages (from cycler>=0.10-
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/site-packages (from kiwiso
```

1.1 This cell automatically downloads Capital Bikeshare data

```
In [2]: import pandas as pd
        bikes = pd.read_csv('../data/2016-Q1-Trips-History-Data.csv')
        bikes.head()

Out[2]:    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
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
```

In [3]:
```
# Placed the same script in a different cell to see what result this
# code yields
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.head()

# It looks like the same information but in a different format
```

Out[3]:
```
   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
0 2016-04-01 00:04:00
1 2016-04-01 00:08:00
2 2016-04-01 00:08:00
```

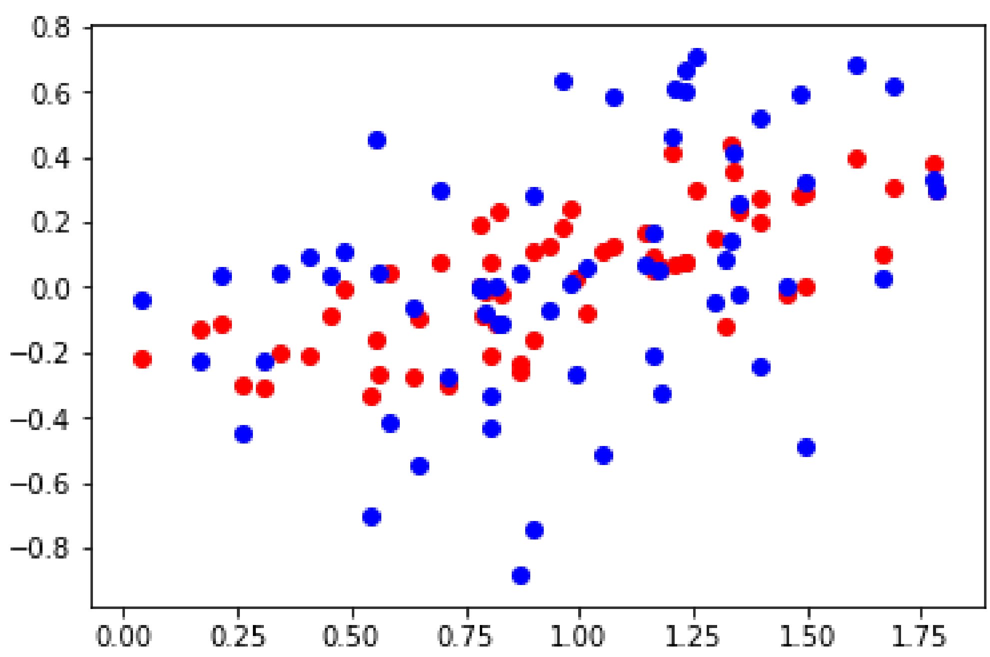
```
3 2016-04-01 00:09:00
4 2016-03-31 23:59:00
```

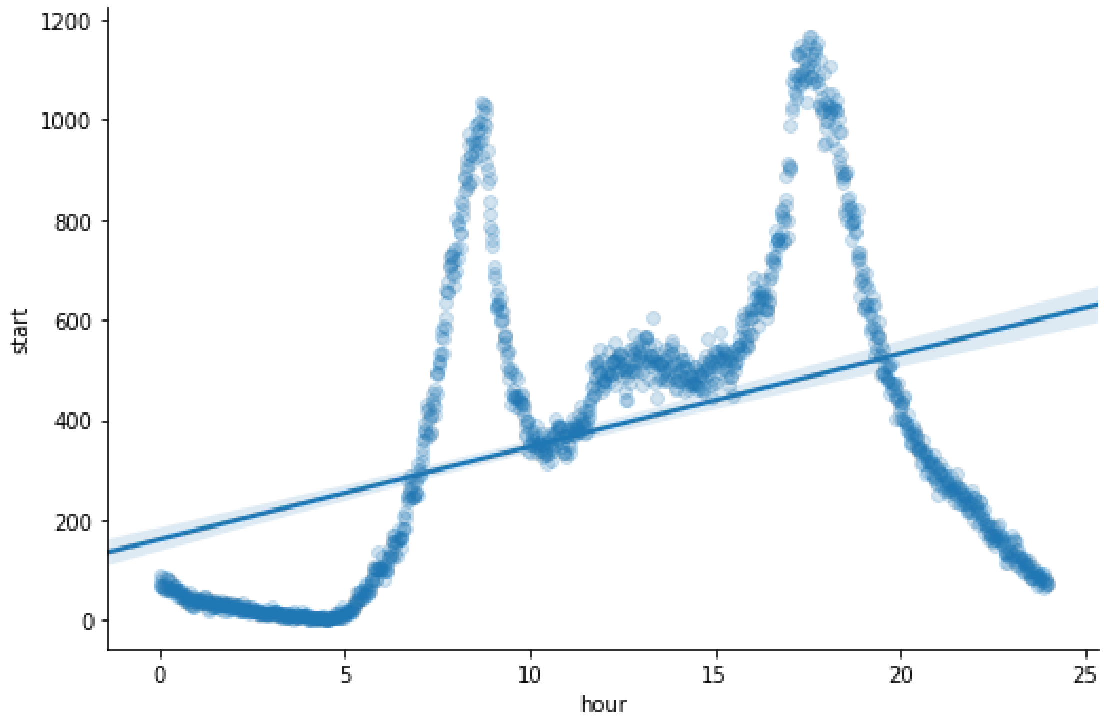
1.1.1 Create a new column that represents the hour of the day

```
In [4]: # So, the hour of day is really concerning the starting timepoint of bikeshare uses
        bikes['hour_of_day'] = (bikes.start.dt.hour + (bikes.start.dt.minute/60).round(2))
```

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

```
In [5]: import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        # Aggregating the column dimension, then indexing it
        hours = bikes.groupby('hour_of_day').agg('count')
        hours['hour'] = hours.index

        # plotting the aggregate number
        hours.start.plot()
        sns.lmplot(x='hour', y='start', data=hours, aspect=1.5, scatter_kws={'alpha':0.2})

Out[5]: <seaborn.axisgrid.FacetGrid at 0x107d64cc0>
```



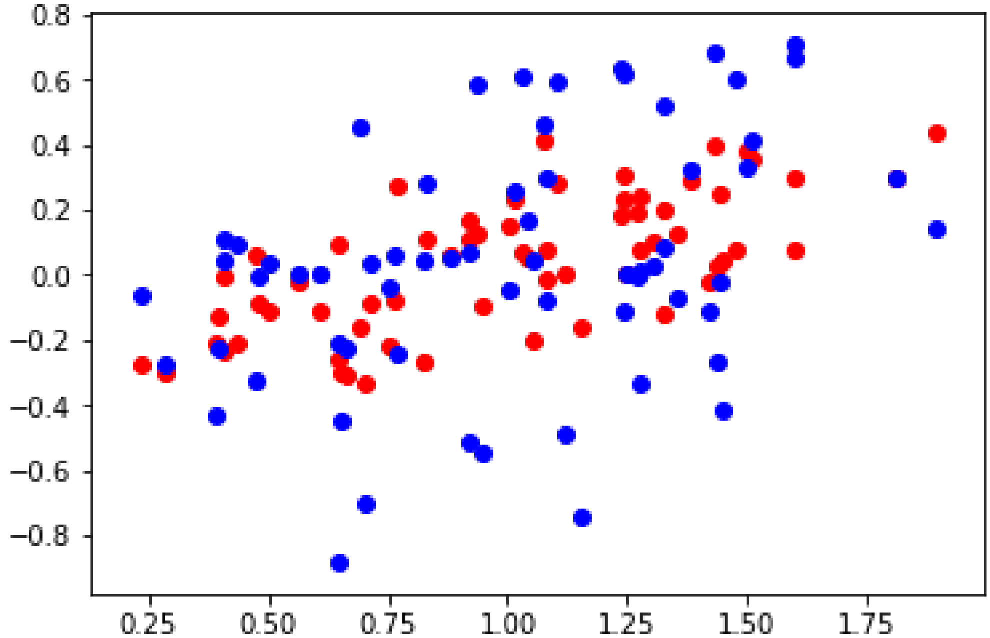


2 Assignment 4

Using the `hours` dataframe and the `hour_of_day` column, perform the following cells.
Explain the results in a **paragraph + charts** of to describe which model you'd recommend

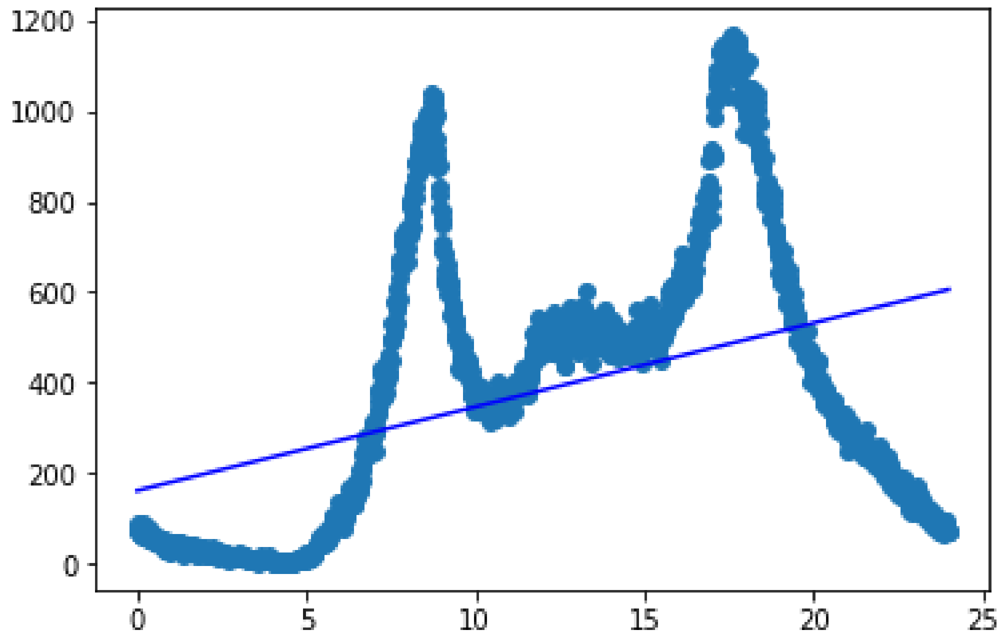
2.1 1. Create 3 models fit to `hour_of_day` with varying polynomial degrees

```
In [6]: # I'm following along your Loss Function Comparison File
        from sklearn import linear_model # adding additional module not already listed from abov

        # I prefer not using x and y as names
        x_times = hours[['hour']]
        y_starting = hours.start

        plt.scatter(x_times, y_starting)

Out[6]: <matplotlib.collections.PathCollection at 0x108644e48>
```



In [7]:
```
# For my reference - Setting up regular linear regresssion
from sklearn.linear_model import LinearRegression

linear = LinearRegression()
linear.fit(x_times, y_starting)

# Plotting
plt.scatter(x_times, y_starting)
plt.plot(x_times, x_times*linear.coef_ + linear.intercept_, c = "b" )
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x108ebbc18>]
```




```
In [8]: # Model 1, polynomial linear regression x25 degree
        from sklearn.preprocessing import PolynomialFeatures

        poly_25 = PolynomialFeatures(degree=25)
        x_25 = poly_25.fit_transform(x_times) # got rid of .reshape
        x_25

Out[8]: array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
                 0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
                [1.00000000e+00, 2.00000000e-02, 4.00000000e-04, ...,
                 8.38860800e-40, 1.67772160e-41, 3.35544320e-43],
                [1.00000000e+00, 3.00000000e-02, 9.00000000e-04, ...,
                 9.41431788e-36, 2.82429536e-37, 8.47288609e-39],
                ...,
                [1.00000000e+00, 2.39500000e+01, 5.73602500e+02, ...,
                 5.29696262e+31, 1.26862255e+33, 3.03835100e+34],
                [1.00000000e+00, 2.39700000e+01, 5.74560900e+02, ...,
                 5.39963971e+31, 1.29429364e+33, 3.10242185e+34],
                [1.00000000e+00, 2.39800000e+01, 5.75040400e+02, ...,
                 5.45168949e+31, 1.30731514e+33, 3.13494170e+34]])

In [9]: linear = linear_model.LinearRegression()
        linear.fit(x_25, y_starting)
        linear.coef_, linear.intercept_

        # So far just copy and pasting
```

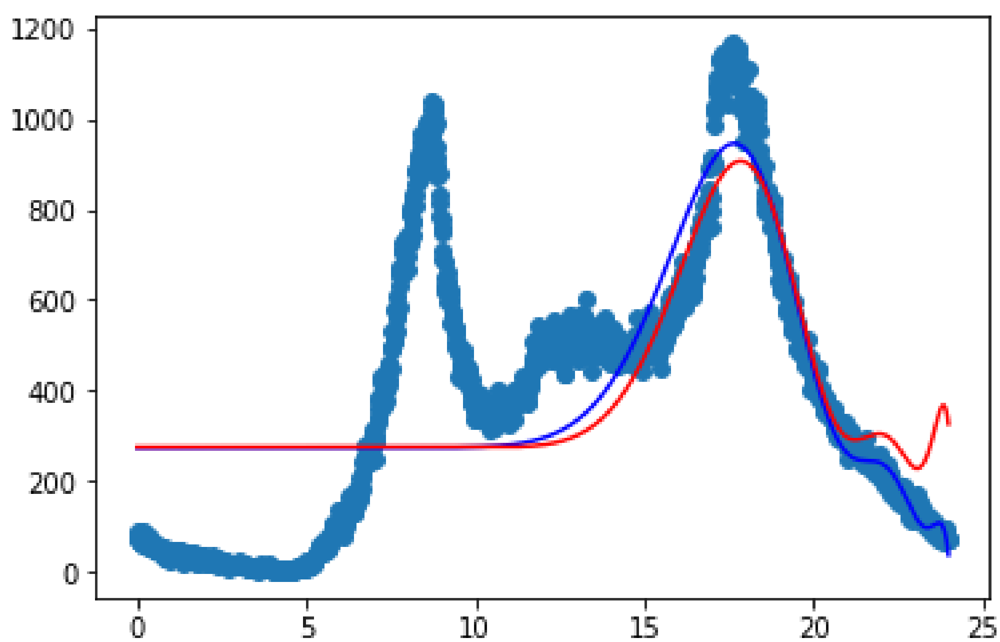
```
Out[9]: (array([-4.70053389e-21,  4.25577597e-19,  2.86340819e-22, -2.87151676e-25,
                 1.52942891e-27,  2.27110474e-30,  2.04295496e-30,  2.37799436e-29,
                 2.78276140e-28,  3.21951564e-27,  3.65473938e-26,  4.03725517e-25,
                 4.29819285e-24,  4.35744774e-23,  4.14022223e-22,  3.60547568e-21,
                 2.78224971e-20,  1.79851813e-19,  8.73208023e-19,  2.38526543e-18,
                -5.17154209e-19,  4.14076916e-20, -1.45700494e-21,  1.47044652e-23,
                 3.45890175e-25, -7.07597579e-27]), 273.72942946950593)
```

```
In [10]: ridge = linear_model.Ridge()
         ridge.fit(x_25, y_starting)
         ridge.coef_, ridge.intercept_
```

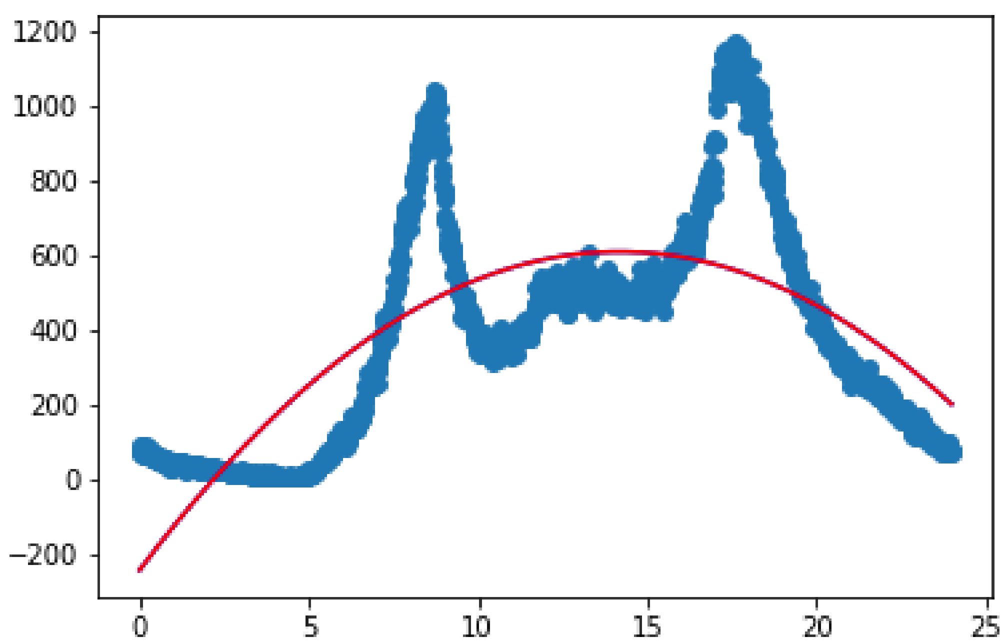
```
Out[10]: (array([-9.82678123e-18,  2.38239504e-16,  1.41333181e-16,  2.66195958e-16,
                  6.35468615e-17, -4.56806931e-17,  5.84980919e-18,  2.96815590e-18,
                 -4.81236845e-17,  5.85990452e-17,  8.27015192e-17,  8.41027834e-17,
                  1.07612985e-16,  1.25774171e-16, -5.43810832e-17, -1.01639415e-16,
                 -1.38116955e-18,  3.12792320e-17, -1.23109784e-16,  3.78594551e-17,
                 -4.94899947e-18,  3.47754823e-19, -1.40282240e-20,  3.19538425e-22,
                 -3.67705795e-24,  1.50466242e-26]), 275.45907916227975)
```

```
In [11]: # Model 1 : 25
         import numpy as np # I didn't need to until now

         plt.scatter(x_times, y_starting)
         plt.plot(x_times, np.dot(x_25, linear.coef_) + linear.intercept_, c='b')
         plt.plot(x_times, np.dot(x_25, ridge.coef_) + ridge.intercept_, c='r')
```

```
Out[11]: [<matplotlib.lines.Line2D at 0x108f1ee48>]
```



```
In [12]: # Okay.... I wasn't expecting that, but THAT IS COOL!
         # I will be using this for work most likely

In [13]: # Model 2, polynomial linear regression x2 degree
         # More concise this time.
         poly_2 = PolynomialFeatures(degree=2)
         x_2 = poly_2.fit_transform(x_times) # got rid of .reshape
         linear.fit(x_2, y_starting)
         ridge.fit(x_2, y_starting)

         plt.scatter(x_times, y_starting)
         plt.plot(x_times, np.dot(x_2, linear.coef_) + linear.intercept_, c='b')
         plt.plot(x_times, np.dot(x_2, ridge.coef_) + ridge.intercept_, c='r')

Out[13]: [<matplotlib.lines.Line2D at 0x1090635c0>]
```

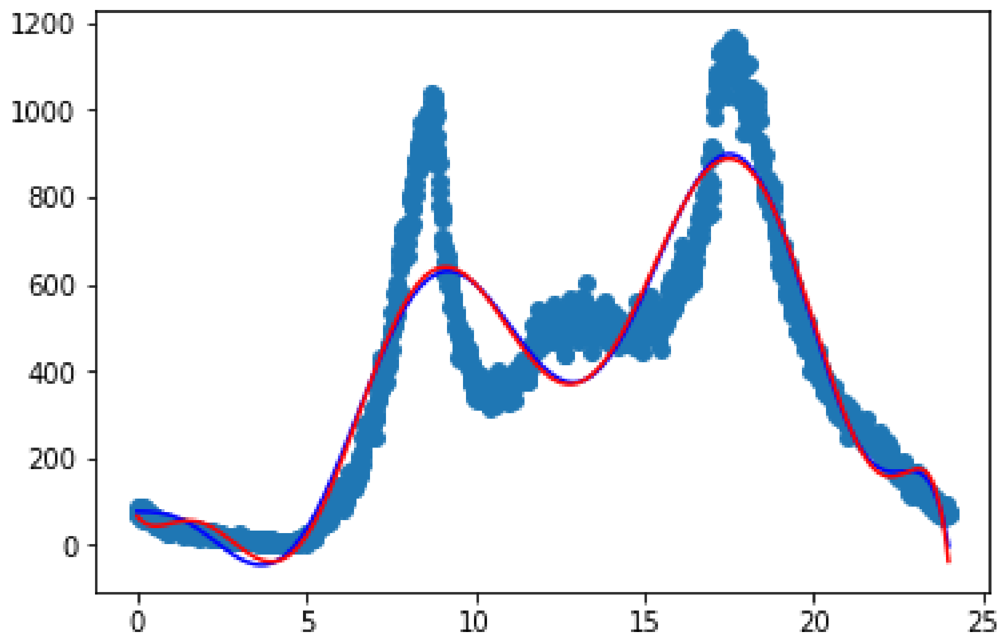



```
In [14]: # Model 3, polynomial linear regression x12 degree
         poly_12 = PolynomialFeatures(degree=12)
         x_12 = poly_12.fit_transform(x_times) # got rid of .reshape
         linear.fit(x_12, y_starting)
         ridge.fit(x_12, y_starting)

         plt.scatter(x_times, y_starting)
```

```
plt.plot(x_times, np.dot(x_12, linear.coef_) + linear.intercept_, c='b')
plt.plot(x_times, np.dot(x_12, ridge.coef_) + ridge.intercept_, c='r')
```

```
/Users/dell/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:112: LinAlgWarni
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number1.474511e-35
  overwrite_a=True).T
```

Out[14]: [<matplotlib.lines.Line2D at 0x1090eaba8>]



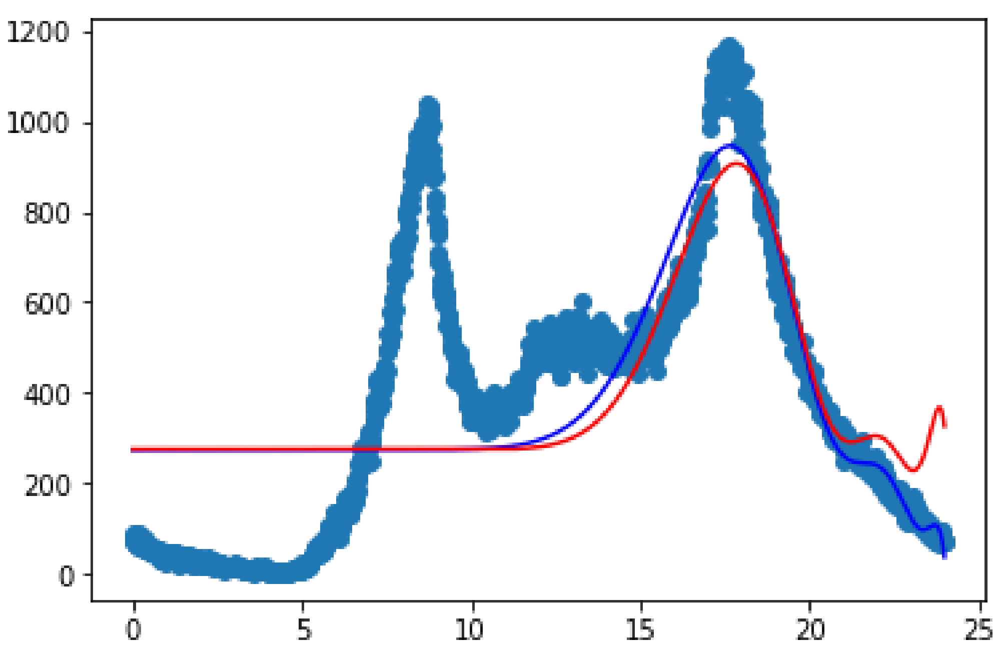
In [15]:
```
# I want to see what will happen if I added more data points like you
# had in the Loss Function Comparison file. But, adding more fake data
# points here would not have the same meaning. While we've already known
# how the scatterplot will look in the Loss Function Comparison file,
# adding more points after 24 hours doesn't make sense. That'll be
# like breaking the law of physics or going against social conventions of
# measuring time.

# In Model 1, it appears that the linear method appears to follow
# along the datapoints at the x=24 mark while the ridge method "refuses"
# to fall down the slope. This is a good thing. I will explain in a sec.
# In the second and third model, both linear and ridge lines are traveling
# more-or-less the same trajectory.

# In determining which model would be the best recommendation, I have
```

```
# been thinking about the context of the graphs. For example, although I
# really like how the third model glides along the spiked datapoints,
# the predictive direction appears to takeabout a dramatic end. And,
# the question I should be asking at this point is - what does this mean
# in the context of the graph drawn?

# After 24 hours, the clock will circle back to 0. We already know the
# value of y at x = 0. If we wanted to predict something, then
# we would have to revisit x = 0 rather than asking the
# value of y after 24. I don't think Medel 2 really gives us
# any useful information other than reflecting an increase and decrease
# of trend. Finally, we are left with Model 1, where the ridge method,
# as mentioned above, "refuses" to fall down the spike. But, earlier
# times' trajections appear to be producing high residuals.
```

2.2 2. Choose one of the polynomial models and create 3 new models fit to `hour_of_day` with different Ridge Regression α (alpha) Ridge Coefficient values

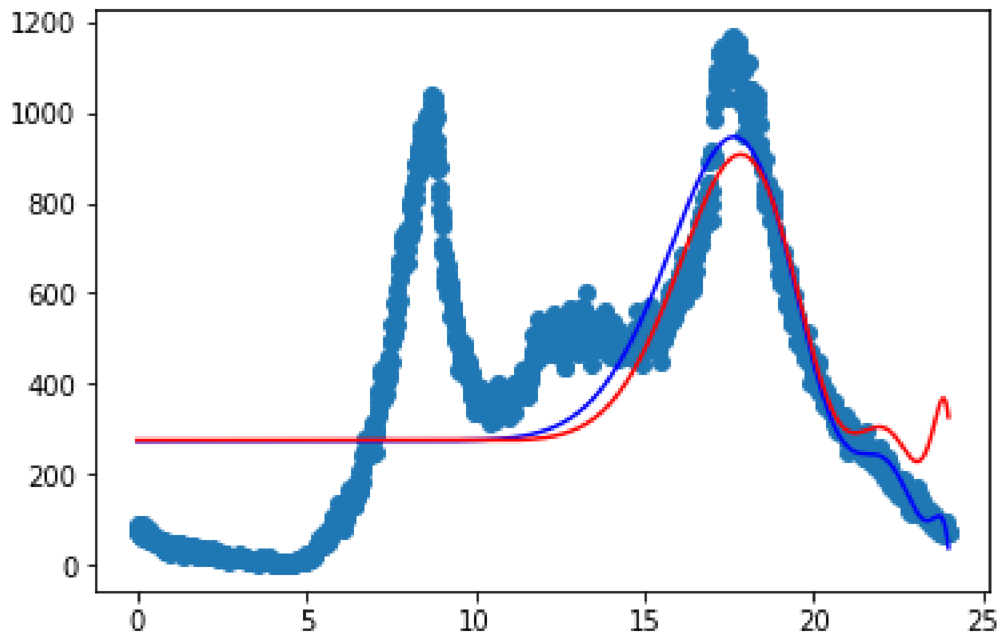
In [16]:
```
# Model 1, polynomial linear regression x25 degree + alpha = .1
linear.fit(x_25, y_starting)
ridge1 = linear_model.Ridge(alpha = .1)
ridge1.fit(x_25, y_starting)

plt.scatter(x_times, y_starting)
plt.plot(x_times, np.dot(x_25, linear.coef_) + linear.intercept_, c='b')
plt.plot(x_times, np.dot(x_25, ridge1.coef_) + ridge1.intercept_, c='r')
```

Out[16]: [<matplotlib.lines.Line2D at 0x109b61c88>]



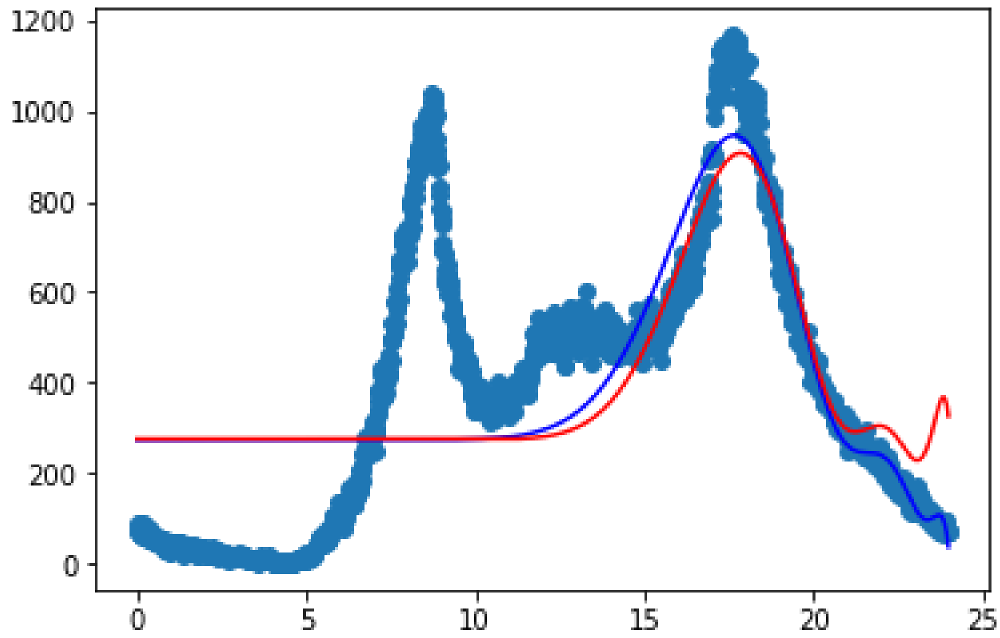
In [17]:
```
# Model 2, polynomial linear regression x25 degree + alpha = .5
linear.fit(x_25, y_starting)
ridge2 = linear_model.Ridge(alpha = .5)
ridge2.fit(x_25, y_starting)

plt.scatter(x_times, y_starting)
plt.plot(x_times, np.dot(x_25, linear.coef_) + linear.intercept_, c='b')
plt.plot(x_times, np.dot(x_25, ridge2.coef_) + ridge2.intercept_, c='r')
```

Out[17]: [<matplotlib.lines.Line2D at 0x109cf2dd8>]



In [18]:
```
# Model 3, polynomial linear regression x25 degree + alpha = 20
linear.fit(x_25, y_starting)
ridge3 = linear_model.Ridge(alpha = 20)
ridge3.fit(x_25, y_starting)

plt.scatter(x_times, y_starting)
plt.plot(x_times, np.dot(x_25, linear.coef_) + linear.intercept_, c='b')
plt.plot(x_times, np.dot(x_25, ridge3.coef_) + ridge3.intercept_, c='r')
```

Out[18]: [<matplotlib.lines.Line2D at 0x109dbdef0>]



In [19]:
```
# Changing the alpha values do not appear to induce any change
# to graph illustrations. So, I wanted to find out if the values were
# infact changing when alphas changed

# Checking to see if this compare method would work to test alphas
from IPython.display import display
same_A = 1
same_B = 1
display(same_A is same_B)

diff_A = 1
diff_B = 2
display(diff_A is diff_B)
```

```
True
```

```
False
```

In [20]:
```
# Above method worked, so I'm going to use it to test alphas

# Control
ridge_pointOne = linear_model.Ridge(alpha=.1)
ridge_A = ridge_pointOne.fit(x_25, y_starting)
ridge_B = ridge_pointOne.fit(x_25, y_starting)
```

```
display(ridge_A is ridge_B)

# Different alphas
ridge1 = linear_model.Ridge(alpha=.1)
ridge2 = linear_model.Ridge(alpha=.5)
ridge3 = linear_model.Ridge(alpha=20)

ridge_1 = ridge1.fit(x_25, y_starting)
ridge_2 = ridge2.fit(x_25, y_starting)
ridge_3 = ridge3.fit(x_25, y_starting)

display(ridge_1 is ridge_2)
display(ridge_1 is ridge_3)
display(ridge_2 is ridge_3)
```

```
True
```

```
False
```

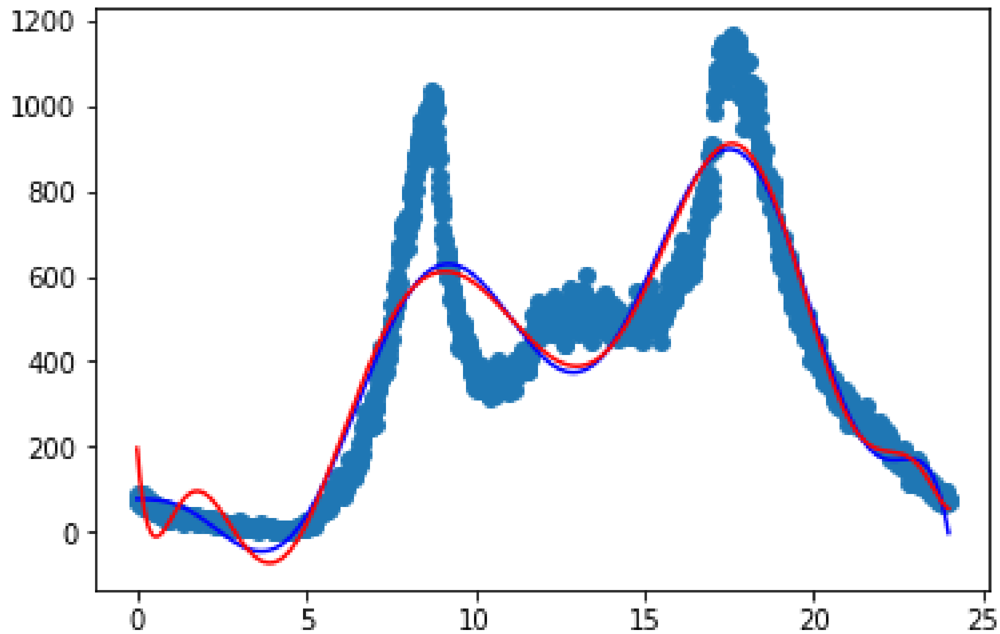
```
False
```

```
False
```

In [21]:
```
# I'm wondering if I've done something wrong. So, I'm priting graphs for another degree

linear.fit(x_12, y_starting)
ridge1 = linear_model.Ridge(alpha = .1)
ridge1.fit(x_12, y_starting)

plt.scatter(x_times, y_starting)
plt.plot(x_times, np.dot(x_12, linear.coef_) + linear.intercept_, c='b')
plt.plot(x_times, np.dot(x_12, ridge1.coef_) + ridge1.intercept_, c='r')
```

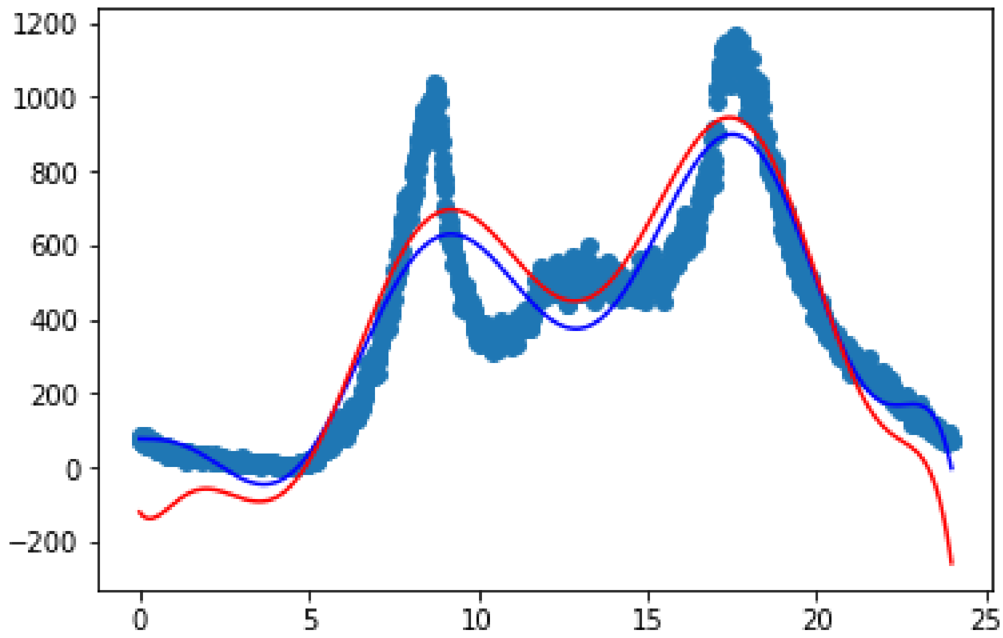
```
/Users/dell/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:112: LinAlgWarni
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number1.320701e-36
  overwrite_a=True).T
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x109e9c358>]
```



In [22]:
```
linear.fit(x_12, y_starting)
ridge1 = linear_model.Ridge(alpha = .5)
ridge1.fit(x_12, y_starting)

plt.scatter(x_times, y_starting)
plt.plot(x_times, np.dot(x_12, linear.coef_) + linear.intercept_, c='b')
plt.plot(x_times, np.dot(x_12, ridge1.coef_) + ridge1.intercept_, c='r')
```

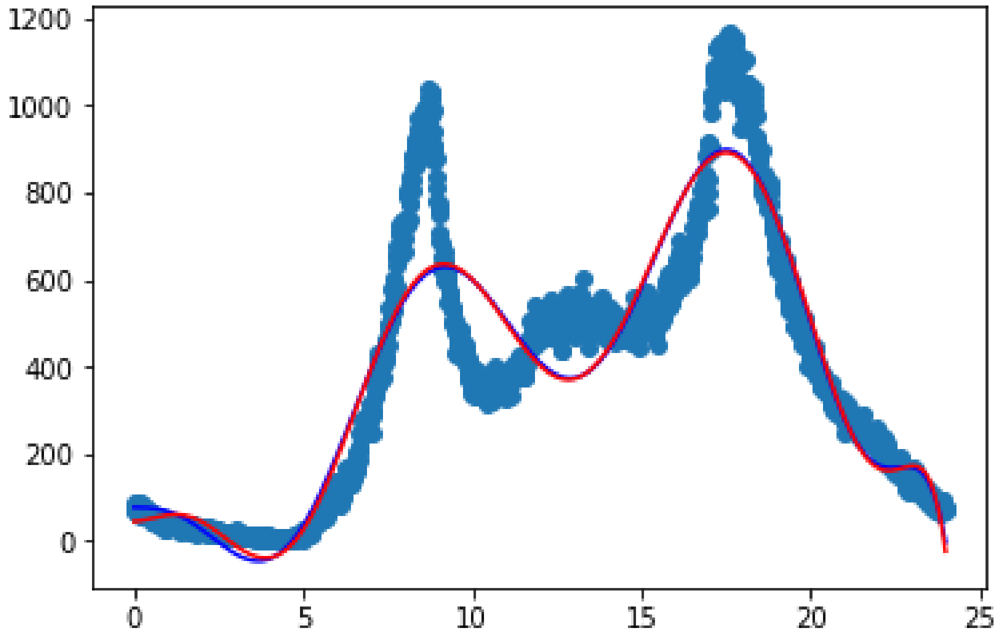
Out[22]:
```
[<matplotlib.lines.Line2D at 0x109e33358>]
```




```
In [23]: linear.fit(x_12, y_starting)
         ridge1 = linear_model.Ridge(alpha = 20)
         ridge1.fit(x_12, y_starting)

         plt.scatter(x_times, y_starting)
         plt.plot(x_times, np.dot(x_12, linear.coef_) + linear.intercept_, c='b')
         plt.plot(x_times, np.dot(x_12, ridge1.coef_) + ridge1.intercept_, c='r')
```

```
/Users/dell/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:112: LinAlgWarni
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number2.512771e-34
  overwrite_a=True).T
```

```
Out[23]: [<matplotlib.lines.Line2D at 0x10d02a860>]
```




```
In [24]: # So, it's not my method that's wrong. x25 does produce very similar
         # looking graphs regardless the alpha value
```