데이터과학 실습 6 주차

컴퓨터공학과
201201886 나동희

1. 과제를 해결한 방법

1) 파이썬 시각화

: 1 명의 데이터를 골라 10 일 간의 데이터를 모두 파싱한 후 심장 박동 수를 발걸음을 기준으로 평균치를 내어 분석해보고 또 다른 요소인 수면 상태까지 포함해서 평균치를 내어 심장 박동수를 분석해본다. 또한 선형 회귀 분석법을 이용하여 심장 박동수 예측 모델을 만들어 본다.

2) R 시각화

: 위의 파이썬을 이용해서 분석한 결과를 csv 파일로 저장한 후 불러와서 심장 박동 수와 Bean 그래프를 이용해서 시각화한다.

3) D3.js 시각화

: 총 걸음 수, 최대 걸음 수, 최소 걸음수, 평균 심박수, 최대 심박수, 최소 심박수를 버블차트로 만들어 본다.

2. 과제를 해결하기 위해 알아야 하는것

- 기본적인 파이썬의 이해
- 기본적인 머신러닝의 이해(?)
- 기본적인 R 문법의 이해
- D3.js 와 자바스크립트에 대한 이해

3. 결과 화면 캡쳐와 설명

1) 파이썬 시각화

: 우선 분석을 시작하기 위해서 10 일치의 데이터를 모두 합쳐야 했다. 하지만 수면 로그가 30 초 단위로 되어 있기 때문에 다른 데이터와 병합을 위해서 0 초 단위로 바꾸어 준다.
그리고 선형회귀법의 Training set 와 Cross Validation set 을 만들어 주기 위해서 2016 년 4 월 1 일 자료부터 2016 년 4 월 7 일까지의 자료를 Training set, 이후 자료를 Cross Validation set 으로 사용한다. Cross Validation set 은 Training set 으로 만든 모델의 검증용 데이터로 사용한다.

```
    heart_json_df = heart_json_df.rename(columns = {u'value': u'heart_beat'})
    sleep_json_df = sleep_json_df.rename(columns = {u'dateTime':u'time'})
    sleep_json_df = sleep_json_df.fillna(4)
    for t in range(len(sleep_json_df[u'time'])):
        sleep_json_df[u'time'][t] = sleep_json_df[u'time'][t].split(':')[0] + ':' + sleep_json_df[u'time'][t].split(':')[1] + u':00'

    merge_data = step_json_df.merge(heart_json_df, left_on = u'time', right_on = u'time', how = 'outer')
    multiple_merge_data = merge_data.merge(sleep_json_df, left_on = u'time', right_on = u'time', how = 'outer')
    for t in range(len(multiple_merge_data[u'status'])):
        if pd.isnull(multiple_merge_data[u'status'][t]):
            multiple_merge_data[u'status'][t] = 4
    return multiple_merge_data
```

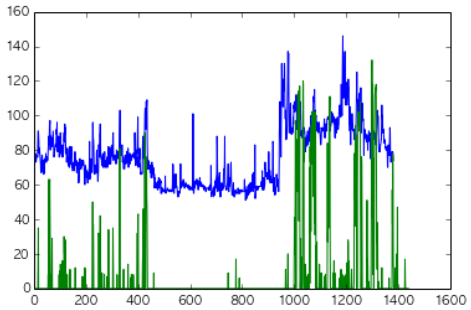
2016년 4월 1일 로그부터 2016년 4월 9일까지의 로그를 모으고 합친다

이 때 수면시간도 다른 시간과 같은 시간대로 (30초 단위 -> 0초 단위) 조절한다.

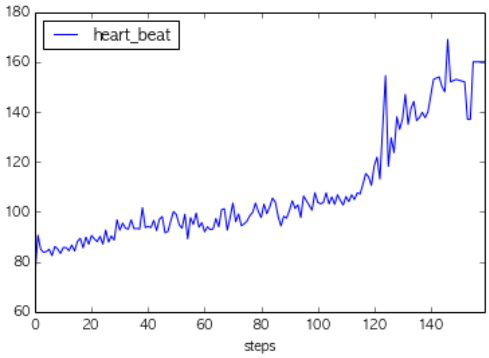
```
In [13]: data_list = [20160401, 20160402, 20160403, 20160404, 20160405, 20160406, 20160407]
data_frame_list = list()
print "create training data set"
for i in data_list:
    print i
    data_frame_list.append(getData(i))

training_data = pd.concat(data_frame_list)
data_list = [20160408, 20160409]
data_frame_list = list()
print "create test data set"
for i in data_list:
    print i
    data_frame_list.append(getData(i))

test_data = pd.concat(data_frame_list)
training_data.to_csv('data.csv')
```

```
create training data set
20160401
```

우선 걸음수와 심장박동수의 추이를 봐본다.



매우 비슷한 분포를 보이므로 발걸음 수를 기준으로 한 평균 심장 박동수를 그래프로 표현해본다



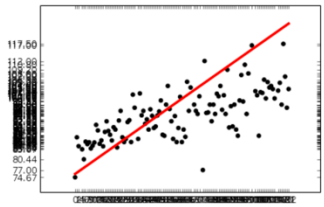
.

실제로 상관관계가 있음을 알 수 있다.
그래서 선형회귀 모델 라이브러리를 이용해서 구현해본다.

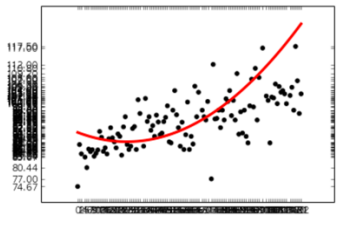
어느정도 선형성이 있음을 확인하였으므로 선형 회귀를 통해 심장박동 수를 예측해본다

In [6]:
```
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
train_x, test_x = np.array(X), np.array(testX)
train_y, test_y = np.array(Y), np.array(testY)
train_x = train_x.reshape(-1, 1)
test_x =test_x.reshape(-1, 1)
for i in range(5):
    polynomial_features = PolynomialFeatures(degree=i+1,include_bias=False)
    regr = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", regr)])
    pipeline.fit(train_x, train_y)

    plt.scatter(test_x, test_y,  color='black')
    plt.plot(test_x, pipeline.predict(test_x), color='red', linewidth=3)
    plt.xticks(test_x)
    plt.yticks(test_y)
    plt.show()
    print('Coefficients:', regr.coef_)
    print("RMSE: %.2f" % np.sqrt(np.mean((pipeline.predict(test_x) - test_y) ** 2)))
    print('Variance score: %.2f' % pipeline.score(test_x, test_y))
```

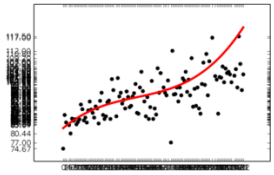
Polynomial Regression 모델을 구현해서 degree 를 늘려가 보면서 적절한 모델을 구한다.(1<= degree <= 5)



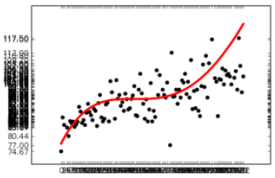

```
('Coefficients:', array([ 0.39542979]))
RMSE: 11.22
Variance score: -1.00
```



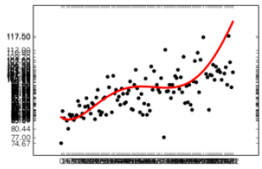

```
('Coefficients:', array([-0.21256604,  0.00397262]))
RMSE: 8.98
Variance score: -0.28
```




```
('Coefficients:', array([  4.71752614e-01,  -7.09165935e-03,   4.76228295e-05]))
RMSE: 7.37
Variance score: 0.14
```



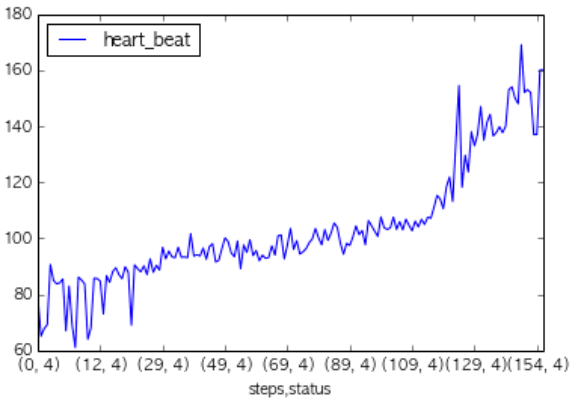

```
('Coefficients:', array([  1.11174990e+00,  -2.56466390e-02,   2.32667394e-04,
        -5.92111387e-07]))
RMSE: 7.60
Variance score: 0.08
```




```
('Coefficients:', array([ -3.71046943e-01,   4.13624844e-02,  -9.11855781e-04,
         7.62848920e-06,  -2.09567280e-08]))
RMSE: 7.35
Variance score: 0.14
```

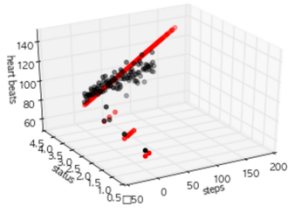
RMSE(Root Mean Square Error)가 너무 크면 Under fit 너무 작으면 일반적으로 Over fit 이라고 하는데 보통 RMSE 값이 작은 경우에서 고르면 되는데 이 경우에는 High Variance 일 수록 Ove fit 이므로 적당히 RMSE 도 작고 Variance 값도 작은 degree 가 3 인 회귀 곡선이 심장박동 수 예측 모델과 가장 적합하다고 생각이 든다. (RMSE: 7.37 Variance Score = 0.14)
(검정색이 실제 값이고 빨간 선이 예측 값이다.)

이번에는 수면 상태도 포함을 시켜서 분석을 해본다. 수면 상태가 입력이 되어 있지 않는 경우는 비수면상태로 4 로 설정을 해서 숫자가 작을 수록 더 깊은 수면상태로 설정을 한다.

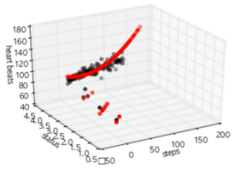


마찬가지로 어느정도 상관관계를 가지고 있음을 알 수 있다.
마찬가지로 선형회귀 모델을 구현해본다.

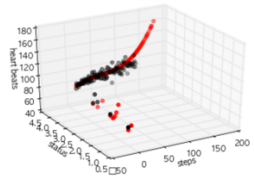
```
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import interactive
train_x, test_x = np.array(X), np.array(testX)
train_y, test_y = np.array(Y), np.array(testY)


for i in range(5):
    polynomial_features = PolynomialFeatures(degree=i+1,include_bias=False)
    regr = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", regr)])
    pipeline.fit(train_x, train_y)

    threed_graph = plt.figure().gca(projection='3d')
    threed_graph.scatter(train_x.T[0].tolist(), train_x.T[1].tolist(),  pipeline.predict(train_x).tolist(),  color='red')
    threed_graph.scatter(test_x.T[0].tolist(),  test_x.T[1].tolist(), test_y.tolist(),  color='black')
    threed_graph.set_xlabel('steps')
    threed_graph.set_ylabel('status')
    threed_graph.set_zlabel('heart beats')
    threed_graph.view_init(30, 240)
    plt.show()
    print('Coefficients:', regr.coef_)
    print("RMSE: %.2f" % np.sqrt(np.mean((pipeline.predict(test_x) - test_y) ** 2)))
    print('Variance score: %.2f' % pipeline.score(test_x, test_y))
```



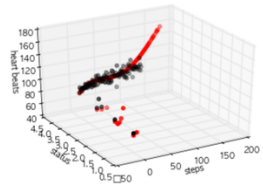
```
('Coefficients:', array([ 0.39675406,  4.36602422]))
RMSE: 11.09
Variance score: -0.57
```



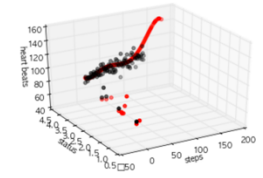
```
('Coefficients:', array([  1.99620080e+00,  -2.43317517e+01,   4.00610266e-03,
        -5.53660203e-01,   6.64291520e+00]))
RMSE: 8.91
Variance score: -0.02
```



```
('Coefficients:', array([  2.45291190e+00,   3.64328773e+01,   3.60541915e-01,
        -3.79421457e+00,  -1.61382850e+01,   4.65477499e-05,
        -9.18348614e-02,   8.23243081e-01,   2.41345426e+00]))
RMSE: 7.49
Variance score: 0.28
```



```
('Coefficients:', array([  1.38398794e-01,   3.10765115e+00,   8.88147600e-01,
        -4.06083148e+00,   3.71238769e+00,   1.25652402e-02,
        -6.06699734e-01,   2.27925249e+00,  -2.50334453e+00,
        -5.68151988e-07,  -3.08527432e-03,   9.46274437e-02,
        -3.01566260e-01,   4.15544610e-01]))
RMSE: 7.61
Variance score: 0.26
```



```
('Coefficients:', array([  2.87286127e-01,   6.26404089e-01,   8.89437950e-02,
         1.16272121e-01,   1.21761082e+00,   1.87381955e-01,
        -1.21259827e+00,   2.74307046e-01,   1.03518314e+00,
        -1.31436501e-02,   8.08504015e-02,  -1.92849987e-01,
         1.04001270e+00,  -9.44307235e-01,  -2.15137129e-08,
         3.28788026e-03,  -3.19833975e-02,   1.23300421e-01,
        -2.81880610e-01,   1.72343545e-01]))
RMSE: 7.21
Variance score: 0.34
```

마찬가지로 RMSE(Root Mean Square Error)가 크면 Under fit 너무 작으면 일반적으로 Over fit 이라고 하는데 보통 RMSE 값이 작은 경우에서 고르면 되는데 이 경우에는 High Variance 일 수록 Over fit 이므로 적당히 RMSE 도 작고 Variance 값도 작은 degree 가 3 인 회귀 곡선이 심장박동 수 예측 모델과 가장 적합하다고 판단이 된다.(RMSE: 7.49 Variance Score = 0.28)

(검정색이 실제 값이고 빨간 선이 예측 값이다.)

이러한 예측 모델을 통해서 만약에 Fitbit 이 가방에 있어 심박수가 측정이 되지 않을 경우 심박수를 측정 할 수있고, 또한 Status 가 4 인 경우에도 steps 가 0 인 경우 학생이 졸고 있는지 여부도 SVM 과 같은 모델로 예측을 해볼 수도 있을 것 같다.

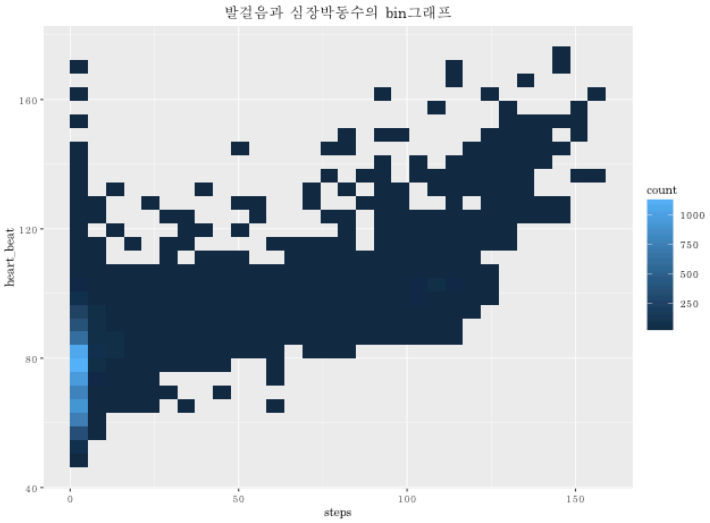
그리고 이 예측모델을 통해 알 수 있는 사실은 어쩌면 당연한 이야기 이지만 사람이 움직이면 심박수가 빨라지고 수면상태에 빠질 수록 에너지 소비를 줄이기 위해 심장 심박수가 줄어든다고 볼 수 있다.

2) R 시각화

```
library(ggplot2)
setwd('/Users/corona10/cnu_datascience/week6/R')
rf <- colorRampPalette(rev(brewer.pal(11,'Spectral')))
r <- rf(32)
# 파이썬에서 작업한 csv파일을 import한다.
data_set <- read.csv('data.csv', sep = ',')
p <- ggplot(data_set, aes(steps, heart_beat))
t <- theme(text=element_text(family = "AppleMyungjo"))
title <- ggtitle("발걸음과 심장박동수의 bin그래프")
h3 <- p + t + stat_bin2d() + title
h3
```

(Top Level) R Script

파이썬에서 작업한 csv 파일을 가져와서 bin 그래프를 그렸다.



위의 bin 차트를 통해서 어느 심박수가 가장 많이 발생되는지 쉽게 알 수 있다.

3) D3.js 시각화

사용자의 10 일간 로그를 분석하여 총 걸음 수, 최대 걸음 수, 최소 걸음수, 평균 심박수, 최대 심박수, 최소 심박수를 버블차트로 만들어 본다.


```
$(document).ready(function () {
  var bubbleChart = new d3.svg.BubbleChart({
    supportResponsive: true,
    //container: => use @default
    size: 600,
    //viewBoxSize: => use @default
    innerRadius: 600 / 3.5,
    //outerRadius: => use @default
    radiusMin: 50,
    //radiusMax: use @default
    //intersectDelta: use @default
    //intersectInc: use @default
    //circleColor: use @default
    data: {
      items: [
        {text: "최대 심박수", count: "175"},
        {text: "최소 심박수", count: "49"},
        {text: "평균 심박수", count: "80"},
        {text: "최대 걸음 수", count: "159"},
        {text: "총 걸음 수", count: "125928"},
        {text: "최소 걸음수", count: "0"},
      ],
      eval: function (item) {return item.count/10;},
      classed: function (item) {return item.text.split(" ").join("");}
    },
    plugins: [
      {
        name: "lines",
        options: {
          format: [
            {// Line #0
              textField: "count",
              classed: {count: true},
              style: {
                "font-size": "28px",
```

위의 bubble 차트를 통해서 Fitbit 의 통계데이터를 쉽게 볼 수 있다.