机器学习第七次作业

Credit数据集

```
# 调用声明
import pandas as pd
import numpy as np
from numpy import *

# 载入数据
Credit = pd.read_csv('C:/Users/mi/Desktop/Credit.csv')

# 变量处理
ID = Credit['ID']
Income = Credit['Income']
Limit = Credit['Limit']
Rating = Credit['Rating']
Cards = Credit['Cards']
Age = Credit['Age']
Education = Credit['Education']
Gender = Credit['Gender'].replace({' Male':1, 'Female':0})
Student = Credit['Student'].replace({'Yes':1, 'No':0})
Married = Credit['Married'].replace({'Yes':1, 'No':0})

Ethnicity = Credit['Ethnicity']
Ethinicities = pd.get_dummies(Ethnicity)

Balance = Credit['Balance']
```

1. 用最小二乘法计算balance对每个自变量的一元回归系数

```
def regression_coefficiency(x, y, intersect = True):
    coeff = 0
    isect = 0
    if not intersect:
        coeff = np.dot(x - np.mean(x), y - np.mean(y))/np.dot(x - np.mean(x),x -
np.mean(x))
        return coeff

    else:
        coeff = np.dot(x,y)/np.dot(x,x)
        isect = np.mean(y) - coeff * np.mean(x)
        return [coeff, isect]

### 1.1 id-balance
coeff_sect = regression_coefficiency(ID, Balance, True)
#[1.9536278685309199, 128.31261235955054]
coeff = regression_coefficiency(ID, Balance, False)
#0.024114900718129488

### 1.2 income-balance
coeff_sect = regression_coefficiency(Income, Balance, True)
```

```
#[9.442942529468013, 93.01566769837643]
coeff = regression_coefficiency(Income, Balance, False)
#6.048363408531562

### 1.3 limit-balance
coeff_sect = regression_coefficiency(Limit, Balance, True)
#[0.12165457521978079, -56.092406410793956]
coeff = regression_coefficiency(Limit, Balance, False)
#0.1716372783714825

### 1.4 rating-balance
coeff_sect = regression_coefficiency(Rating, Balance, True)
#[1.6405427248908104, -62.2792347727443]
coeff = regression_coefficiency(Rating, Balance, False)
#2.56624032734332

### 1.5 cards-balance
coeff_sect = regression_coefficiency(Cards, Balance, True)
#[149.9002118145446, 76.68512355848429]
coeff = regression_coefficiency(Cards, Balance, False)
#28.986948162513205

### 1.6 age-balence
coeff_sect = regression_coefficiency(Age, Balance, True)
#[8.529199211056982, 45.21580291848545]
coeff = regression_coefficiency(Age, Balance, False)
#0.04891139930752879

### 1.7 education-balance
coeff_sect = regression_coefficiency(Education, Balance, True)
#[36.62643919326497, 27.389392850586148]
coeff = regression_coefficiency(Education, Balance, False)
#-1.1859635617141384

### 1.8 gender-balance
male_coeff_sect = regression_coefficiency(Gender, Balance, True)
#[509.8031088082902, 274.03499999999997]
male_coeff = regression_coefficiency(Gender, Balance, False)
#-19.733123075767825

### 1.9 student-balance
student_coeff_sect = regression_coefficiency(Student, Balance, True)
#[876.825, 432.3325]
student_coeff = regression_coefficiency(Student, Balance, False)
#396.4555555555555

### 1.10 married-balance
married_coeff_sect = regression_coefficiency(Married, Balance, True)
#[517.9428571428572, 202.77499999999992]
married_coeff = regression_coefficiency(Married, Balance, False)
#-5.347465437788021

### 1.11 Ethnicity-balance
asian_coeff_sect = regression_coefficiency(Ethinicities['Asian'], Balance, True)
#[512.3137254901961, 389.375]
asian_coeff = regression_coefficiency(Ethinicities['Asian'], Balance, False)
#-10.337281221213326
```

```
caucasian_coeff_sect = regression_coefficiency(Ethinicities['Caucasian'],
Balance, True)
#[518.4974874371859, 262.0625]
caucasian_coeff = regression_coefficiency(Ethinicities['Caucasian'], Balance,
False)
#-3.0199254981374506

African_American_coeff_sect = regression_coefficiency(Ethinicities['African
American'], Balance, True)
#[531.0, 388.5925]
African_American_coeff = regression_coefficiency(Ethinicities['African
American'], Balance, False)
#14.59800664451827
```

2. 用最小二乘法计算balance对limit和rating的回归

```
### 2.1 考虑截距项
X = mat(Credit[['Limit','Rating','Intersect']])
y = mat(Credit['Balance'])

coeff_vector = (X.T * X).I * X.T * y.T
#Limit-Balance: 0.0245144
#Rating-Balance: 2.20167
#Intersect: -377.537

### 2.2 不考虑截距项
X = mat(Credit[['Limit','Rating']])
y = mat(Credit['Balance'])

coeff_vector = (X.T * X).I * X.T * y.T
#Limit-Balance: 0.43236
#Rating-Balance: -4.23338
```

3. 用最小二乘法计算balance对age和rating的回归

```
### 3.1 考虑截距项
X = mat(Credit[['Age','Rating','Intersect']])
y = mat(Credit['Balance'])

coeff_vector = (X.T * X).I * X.T * y.T
#Age-Balance: -2.35078
#Rating-Balance: 2.59328
#Intersect: -269.581

### 3.2 不考虑截距项
X = mat(Credit[['Age','Rating']])
y = mat(Credit['Balance'])

coeff_vector = (X.T * X).I * X.T * y.T
#Age-Balance: -5.43841
#Rating-Balance: 2.36754
```

4. 使用正交系数估算法计算balance对age和rating的回归

```
def Matrix_coeff(X, y):
    if type(X) == matrix and type(y) == matrix:
```

```
        return (X.T * X).I * X.T * y
    else:
        print('TypeError: parameters are not as type
numpy.matrixlib.defmatrix.matrix')

def Schmit_Orthogonalize(X):
    # 默认截距项在最后一列
    nrows = X.shape[0]
    ncols = X.shape[1]
    o_cols = []
    sub = 0
    o_cols.append(X[:,ncols-1])
    for i in range(ncols-1):
        for j in range(i + 1):
            sub += Matrix_coeff(o_cols[j],X[:,i])
            z_neo = X[:,i] - sub
            o_cols.append(z_neo)
    o_cols.append(X[:,ncols-1])
    o_cols.pop(0)
    return mat(np.array(o_cols).T.reshape(nrows, ncols))
```

```
### 4.1 考虑截距项
X = mat(Credit[['Age','Rating','Intersect']])

z_0  = X[:,2]
z_1 = X[:,0] - (z_0.T * z_0).I * z_0.T * X[:,0]
z_2 = X[:,1] - (z_0.T * z_0).I * z_0.T * X[:,1] - (z_1.T * z_1).I * z_1.T *
X[:,1]

X_o = mat(np.array([z_1,z_2,z_0]).T.reshape(400,3))
coeff_vector = (X_o.T * X_o).I * X_o.T * y.T

#Age-Balance: -2.35078
#Rating-Balance: 2.59328
#Intersect: 522.415


### 4.2 不考虑截距项
X = mat(Credit[['Age','Rating']])
z_0  = X[:,1]
z_1 = X[:,0] - (z_0.T * z_0).I * z_0.T * X[:,0]
X_o = mat(np.array([z_1,z_0]).T.reshape(400,2))
coeff_vector = (X_o.T * X_o).I * X_o.T * y.T
#Age-Balance: -5.44345
#Rating-Balance: 2.36649
```

5. 回归系数比较

	Limit	Rating	Age
单变量（截距）	0.12165457521978079	1.6405427248908104	8.529199211056982
单变量	0.1716372783714825	2.56624032734332	0.04891139930752879
多变量1（截距）	0.0245144	2.20167	
多变量1	0.43236	-4.23338	
多变量2（截距）		2.59328	-2.35078
多变量2		2.36754	-5.43841
正交化（截距）		2.59328	-2.35078
		2.36649	-5.44345

Age在单变量回归中系数为正，多变量中系数为负，表明Rating和Age对彼此的回归系数有影响

在正交化之后的回归分析中，Rating的回归系数为正，Age为负，说明消除相互影响的部分之后仍有这种现象，此时Age对Balance的负向影响是真实的

Boston数据集

```
#调用声明
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neighbors import KNeighborsRegressor

from sklearn.model_selection import train_test_split

from matplotlib import pyplot as plt

#载入数据
Boston = pd.read_csv('C:/Users/mi/Desktop/Boston.csv')

lowstat = np.array(Boston['lowstat']).reshape(-1, 1)
medv = np.array(Boston['medv']).reshape(-1, 1)

#划分训练集于测试集
X_train, X_test, y_train, y_test = train_test_split(lowstat, medv, test_size =
0.2, random_state = 1)
```

1. 线性模型

```
reg = LinearRegression()
reg.fit(X_train, y_train)
```

```
plt.figure(1)
plt.scatter(X_test , y_test ,color ='blue')
plt.plot(X_test ,reg.predict(X_test) ,color='orange',linewidth =3)
plt.xlabel('lowstat')
plt.ylabel('medv')
plt.show()

print('Mean squared error: %.3f' %
mean_squared_error(y_test,reg.predict(X_test)))
print('R^2: %.3f' % r2_score(y_test,reg.predict(X_test)))

#Mean squared error: 47.024
#R^2: 0.524
```

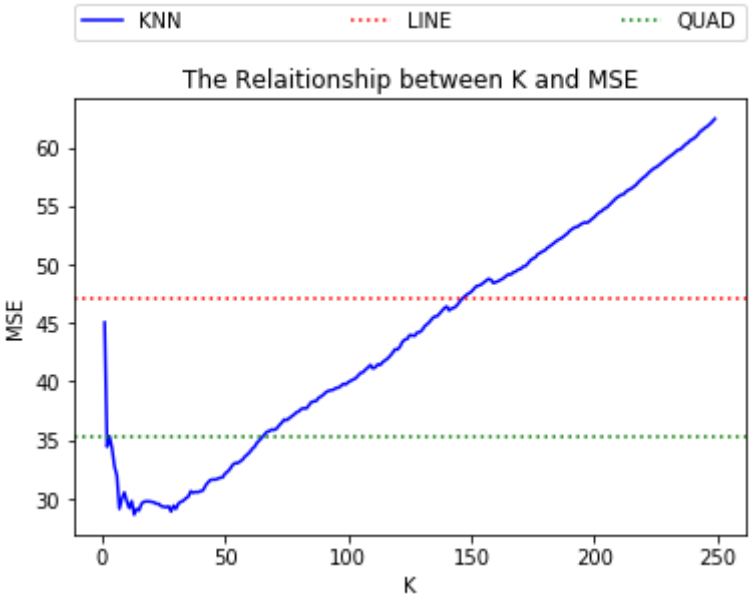
2. 多项式模型

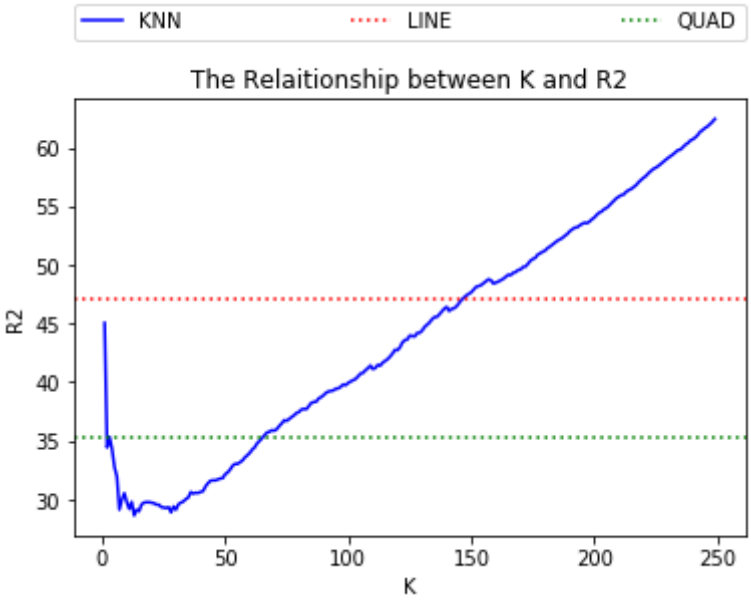
```
poly = np.polyfit(X_train.flatten(), y_train.flatten(), 2)
y_pred = np.polyval(poly, X_test.flatten())

plt.figure(2)
plt.scatter(X_test , y_test ,color ='blue')
plt.plot(X_test, y_pred, color='orange',linewidth =3 )
plt.xlabel('lowstat')
plt.ylabel('medv')
plt.show()

print('Mean squared error: %.3f' % mean_squared_error(y_test,y_pred))
print('R^2: %.3f' % r2_score(y_test,y_pred))
#Mean squared error: 35.364
#R^2: 0.642
```

3. KNN模型

```
neigh = KNeighborsRegressor(n_neighbors = 2)
neigh.fit(X_train, y_train)

plt.figure(3)
plt.scatter(X_test , y_test ,color ='blue')
plt.plot(X_test ,neigh.predict(X_test) ,color='orange',linewidth =3)
plt.xlabel('lowstat')
plt.ylabel('medv')
plt.show()

print('Mean squared error: %.3f' %
mean_squared_error(y_test,neigh.predict(X_test)))
print('R^2: %.3f' % r2_score(y_test,neigh.predict(X_test)))

#Mean squared error: 34.424
#R^2: 0.652
```

4. 综合评价


KNN
LINE
QUAD
The Relaitionship between K and MSE
MSE
K
60
55
50
45
40
35
30
0
50
100
150
200
250


KNN
LINE
QUAD
The Relaitionship between K and R2
R2
K
60
55
50
45
40
35
30
0
50
100
150
200
250