**Homework III**

Name: Dazhi Li

RUID:197007456

Course: Sp.20 Software Engineering Web Application

Professor: Yinglung Liang

Source code:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

import datetime

import pandas\_datareader.data as web

def history\_data(stock,start,end):

start\_time = datetime.datetime.strptime(start, "%Y,%m,%d")

end = datetime.datetime.strptime(end,"%Y,%m,%d")

company = web.DataReader(stock, "yahoo", start\_time, end)

return company

def SSE\_mesure():

sum=0

for i in range(210):

sum+=(y[i]-y\_train[i])\*\*2

return sum

def R\_square():

sum\_1=SSE\_mesure()

sum\_2=0

for i in range(210):

sum\_2+=(y\_train[i]-y\_train.mean())\*\*2

R\_square\_value=1-(sum\_1/sum\_2)

return R\_square\_value

def Adj\_R\_square():

Adj\_R\_square\_value=1-(((1-R\_square())\*(210-1))/(210-M-1))

return Adj\_R\_square\_value

class Bayesian\_curvefitiing():

def \_\_init\_\_(self, alpha=1., beta=1.):

self.alpha = alpha

self.beta = beta

self.mean\_prev = None

self.S = None

def fit(self, X, t):

S\_inv = self.alpha \* np.eye(np.size(X, 1)) + self.beta \* np.matmul(X.T,X)

mean\_prev = np.linalg.solve(

S\_inv,

self.beta \* np.matmul(X.T,t)

)

self.mean\_prev = mean\_prev

self.S = np.linalg.inv(S\_inv)

def predict(self, X):

y = np.matmul(X,self.mean\_prev)

y\_var = 1 / self.beta + np.sum(np.matmul(X,self.S) \* X, axis=1)

y\_std = np.sqrt(y\_var)

return y, y\_std

def Auto\_adjusted\_M(x\_train,y\_train,x\_test):

scores=[]

for i in range(20):

poly\_test = PolynomialFeatures(i)

X\_train = poly\_test.fit\_transform(x\_train)

X\_test = poly\_test.fit\_transform(x\_test)

model\_test = Bayesian\_curvefitiing(alpha=5e-3, beta=11.1)

model\_test.fit(X\_train, y\_train)

y, y\_std = model\_test.predict(X\_test)

sum\_1 = 0

sum\_2 = 0

for j in range(210):

sum\_1 += (y[j] - y\_train[j]) \*\* 2

sum\_2 += (y\_train[j] - y\_train.mean()) \*\* 2

R\_square\_value = 1 - (sum\_1 / sum\_2)

score=1-(((1-R\_square\_value)\*(210-1))/(210-i-1))

scores.append(score)

return scores

def early\_stop(scores):

for i in range(20):

if scores[i+1]-scores[i]<=0.01 and scores[i+2]-scores[i-1]<=0.1:

return i

stockprice = history\_data('AAPL', "2016,1,1", "2017,1,1")['Close'].to\_list()

x\_train = np.linspace(1, 210, 210)

x\_train = x\_train.reshape(-1, 1)

y\_train = stockprice[:210]

y\_train = np.array(y\_train)

y\_train = y\_train.reshape(210)

x\_test = np.linspace(1, 217, 217)

x\_test = x\_test.reshape(-1, 1)

y\_test = stockprice[210:217]

y\_test = np.array(y\_test)

Ms=Auto\_adjusted\_M(x\_train,y\_train,x\_test) #Polynomial degree

print(Ms)

M=early\_stop(Ms)

print(M)

poly = PolynomialFeatures(M)

X\_train = poly.fit\_transform(x\_train)

X\_test = poly.fit\_transform(x\_test)

model = Bayesian\_curvefitiing(alpha=5e-3, beta=11.1)

model.fit(X\_train, y\_train)

y, y\_std = model.predict(X\_test)

print(SSE\_mesure())

print(R\_square())

print(Adj\_R\_square())

fig = plt.figure()

plt.scatter(x\_train, y\_train, facecolor="none", edgecolor="b", s=20, label="training data")

plt.plot(x\_train, y\_train, c="b", label="trained stock price")

plt.plot(x\_test, y, c="r", label="predicted stock price")

real\_price=np.linspace(211,217,7).reshape(-1,1)

plt.plot(real\_price, y\_test, c="g", label="real stock price")

plt.scatter(real\_price, y\_test, c="g", s=20, label="real stock price")

plt.fill\_between(x\_test.flatten(), y - y\_std, y + y\_std, color="pink", label="std.", alpha=0.5)

plt.title("M="+str(M))

plt.legend(loc=2)

plt.show()

Predict result:

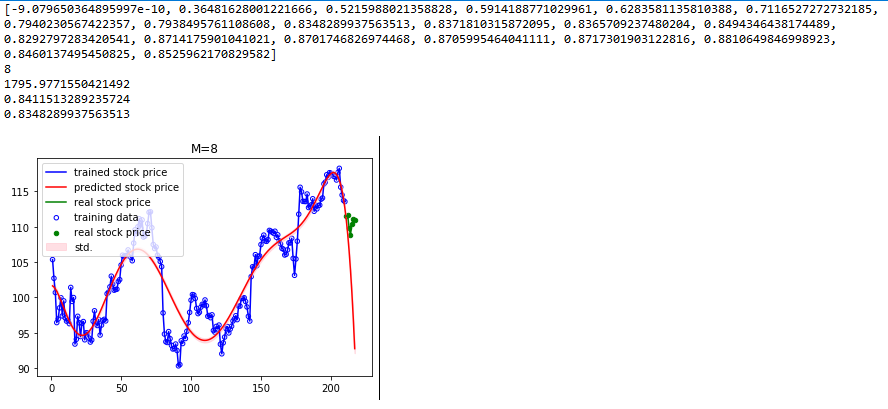


Figure 1

Stock price is downloaded by pandas-datareader. Here I use Apple Inc. as an example. The first 4 row of data means the adjusted R-square when M grows from 0 to 19. The best result without overfitting is when M=8. Red curve means fitted curve, blue and green curve means real stock price, but the blue is training data. Green is the testing data.

I tested 5 day stock price, there are 3 correct out of 5 total. The global accuracy is 0.63, precision is 0.46. Global means I collect data from other stocks in other times and calculate the prediction performance.

Running Requirement:

1. No .csv file needed, just run the code which will automatically download data from Yahoo-Finance!
2. Pandas-datareader package
3. Numpy
4. Matplotlib
5. Sklearn library

Adjust your own stock here:

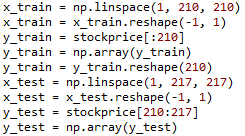


‘AAPL’ refers to stock name

‘2016,1,1’ refers to start time of stock

‘2017,1,1’ refers to end time of stock

Adjust your training set and test set:



Please use python slice to slice a part for training set and another slice for testing set.