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:

```
[-9.079650364895997e-10, 0.364816280012216666, 0.5215988021358828, 0.5914188771029961, 0.6283581135810388, 0.7116527272732185,
0.7940230567422357, 0.7938495761108608, 0.8348289937563513, 0.8371810315872095, 0.8365709237480204, 0.8494346438174489,
0.8292797283420541, 0.8714175901041021, 0.8701746826974468, 0.8705995464041111, 0.8717301903122816, 0.8810649846998923,
0.8460137495450825, 0.8525962170829582]
8
1795.9771550421492
0.8411513289235724
0.8348289937563513
```

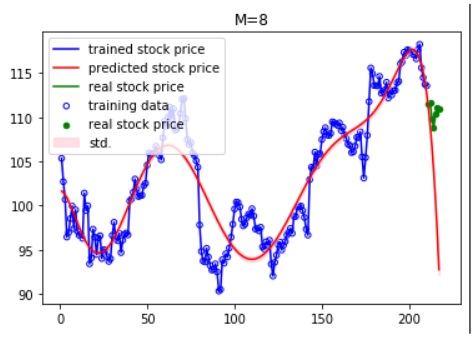

Figure 1

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:

```
stockprice = history_data('AAPL', "2016,1,1", "2017,1,1")['Close'].to_list()
```

'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:

```
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)
```

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