```
import statsmodels.formula.api as smf
import sklearn.metrics as sm
import pandas as pd
import numpy as np
import math
import sys

# The path to the data folder should be given as input
if len(sys.argv) != 2:
    print('bitcoin.py <path to data folder>')
    sys.exit(1)
data_path = sys.argv[1]


# Reading the vectors from the given csv files
train1_90 = pd.read_csv(data_path+'/train1_90.csv')
train1_180 = pd.read_csv(data_path+'/train1_180.csv')
train1_360 = pd.read_csv(data_path+'/train1_360.csv')

train2_90 = pd.read_csv(data_path+'/train2_90.csv')
train2_180 = pd.read_csv(data_path+'/train2_180.csv')
train2_360 = pd.read_csv(data_path+'/train2_360.csv')

test_90 = pd.read_csv(data_path+'/test_90.csv')
test_180 = pd.read_csv(data_path+'/test_180.csv')
test_360 = pd.read_csv(data_path+'/test_360.csv')

#Function to calculate the similarity of two vectors
def similarity(a,b):
    a_mean = np.mean(a)
    b_mean = np.mean(b)
    nr = np.dot(a - a_mean,b - b_mean)
    dr = len(a) * np.std(a) * np.std(b)
    similarity = nr/dr
    return similarity


def computeDelta(wt, X, Xi):
    """
    This function computes equation 6 of the paper, but with the
euclidean distance
    replaced by the similarity function given in Equation 9.
    Parameters
    ----------
    wt : int
        This is the constant c at the top of the right column on page
4.
    X : A row of Panda Dataframe
        Corresponds to (x, y) in Equation 6.
    Xi : Panda Dataframe
```

```
        Corresponds to a dataframe of (xi, yi) in Equation 6.
    Returns
    -------
    float
        The output of equation 6, a prediction of the average price
change.
    """
    similarity_vector = Xi.iloc[:,:-1].apply(lambda x :
similarity(x,X[:-1]),axis = 1)
    similarity_exp = np.exp(wt * similarity_vector)
    price_change = np.dot(Xi.iloc[:,-1],similarity_exp)/
np.sum(similarity_exp)
    return price_change



# Perform the Bayesian Regression to predict the average price change
for each dataset of train2 using train1 as input.
# These will be used to estimate the coefficients (w0, w1, w2, and w3)
in equation 8.
weight = 2  # This constant was not specified in the paper, but we
will use 2.
trainDeltaP90 = np.empty(0)
trainDeltaP180 = np.empty(0)
trainDeltaP360 = np.empty(0)
for i in range(0,len(train1_90.index)) :
  trainDeltaP90 = np.append(trainDeltaP90,
computeDelta(weight,train2_90.iloc[i],train1_90))
for i in range(0,len(train1_180.index)) :
  trainDeltaP180 = np.append(trainDeltaP180,
computeDelta(weight,train2_180.iloc[i],train1_180))
for i in range(0,len(train1_360.index)) :
  trainDeltaP360 = np.append(trainDeltaP360,
computeDelta(weight,train2_360.iloc[i],train1_360))


# Actual deltaP values for the train2 data.
trainDeltaP = np.asarray(train2_360[['Yi']])
trainDeltaP = np.reshape(trainDeltaP, -1)


# Combine all the training data
d = {'deltaP': trainDeltaP,
     'deltaP90': trainDeltaP90,
     'deltaP180': trainDeltaP180,
     'deltaP360': trainDeltaP360 }
trainData = pd.DataFrame(d)


# Feed the data: [deltaP, deltaP90, deltaP180, deltaP360] to train the
```

```
linear model.
# Use the statsmodels ols function.
# Use the variable name model for your fitted model
model = smf.ols(formula = 'deltaP ~ deltaP90 + deltaP180 +
deltaP360',data = trainData).fit()

# Print the weights from the model
print(model.params)


# Perform the Bayesian Regression to predict the average price change
for each dataset of test using train1 as input.
# This should be similar to above where it was computed for train2.
testDeltaP90 = np.empty(0)
testDeltaP180 = np.empty(0)
testDeltaP360 = np.empty(0)
for i in range(0,len(train1_90.index)) :
  testDeltaP90 = np.append(testDeltaP90,
computeDelta(weight,test_90.iloc[i],train1_90))
for i in range(0,len(train1_180.index)) :
  testDeltaP180 = np.append(testDeltaP180,
computeDelta(weight,test_180.iloc[i],train1_180))
for i in range(0,len(train1_360.index)) :
  testDeltaP360 = np.append(testDeltaP360,
computeDelta(weight,test_360.iloc[i],train1_360))


# Actual deltaP values for test data.
testDeltaP = np.asarray(test_360[['Yi']])
testDeltaP = np.reshape(testDeltaP, -1)


# Combine all the test data
d = {'deltaP': testDeltaP,
     'deltaP90': testDeltaP90,
     'deltaP180': testDeltaP180,
     'deltaP360': testDeltaP360}
testData = pd.DataFrame(d)


# Predict price variation on the test data set.
result = model.predict(testData)
compare = { 'Actual': testDeltaP,
            'Predicted': result }
compareDF = pd.DataFrame(compare)


# Compute the MSE and print the result
MSE = 0.0
MSE = sm.mean_squared_error(y_true=testDeltaP,y_pred=result)
```

```
print("The MSE is %f" % (MSE))
```