<u>COURSE NAME:</u> MACHINE LEARNING FOR TRADING

<u>COURSE NUMBER:</u> MSCS 7646

<u>PROJECT NAME:</u> PROJECT 4 – BUILD A STOCK PRICE FORECASTER

<u>STUDENT NAME:</u> MITHUN KUMBLE

<u>GTID NUMBER:</u> 903060297

<u>**PROJECT REPORT**</u>

PROJECT 2 – BUILD A STOCK PRICE FORECASTER

Objective: For a given number of datasets, we need to create a learner (KNN Learner) that can predict the Stock price data. The data is provided as a sine wave from which the parameters have to be extracted and fed as an input to the learner. Additionally, we need to evaluate the performance of the learner by calculating the RMS (Root Mean Square) error and Correlation Coefficient.

<u>i) METHODOLOGY:</u>

The data is analyzed from the past 100 days and then the price of the next 5th day is calculated. The steps are as follows:

a) *Read input data:*

I read the input data (ML4T-000.cv to ML4T-199.csv) using the QSTK data access object. **The files are assumed to be in 'Yahoo' directory of QSTK (QSTK/QSData/Yahoo).** The timestamps for the learning data are from 01/01/2001 to 12/31/2005.

b) *Training the learner:*

- *Extract training sets:*

The input data is analyzed and its parameters (**xTrain**) are extracted. Since the data is known to be a sine wave, the following parameters are extracted:

a. **Amplitude:** The amplitude is calculated by subtracted the minimum altitude from the maximum altitude of the given 100 day period.
b. **Frequency:** The frequency is calculated by doubling the change in value of distance from highest positive value to lowest negative value i.e 2*(difference of distance from highest positive value to lowest negative value)
c. **Phase:** The phase is calculated based on the change in value of the price(positive to negative, negative to positive).
d. **Price Change:** The change in price between subsequent days is also used as a parameter.

For each parameter, I find the mean of the value for the past 100 days. I then create a tuple of the calculated parameters and the y (price value) of the 5th day from now.

The **yTrain** value is the price value of 5 days from today. However, to improve performance I have used the **difference in price as yTrain value**.

The training data is then sent to a KNN Learner with k = 7.

k=1 Correlation Coefficient= 0.883568937318

k=3 Correlation Coefficient = 0.873494012414
k=7 Correlation Coefficient = 0.895667108255
k=10 Correlation Coefficient = 0.873441895218
k=12 Correlation Coefficient = 0.773546687385
k=15 Correlation Coefficient = 0.802716519573
k=20 Correlation Coefficient = 0.868201940047

After analyzing the performance of the learner for different k values, the k value of 7 was selected.

c) *Querying for results:*

After training the learner, I query for results on the datasets:
The datasets used for querying are ML4T-292.csv and ML4T-324.csv

- *Extracting the training sets:*

The input data is analyzed and its parameters are extracted. **Since the data is known to be a sine wave, the following parameters are extracted:**

a. **Amplitude:** The amplitude is calculated by subtracted the minimum altitude from the maximum altitude of the given 100 day period.
b. **Frequency:** The frequency is calculated by doubling the change in value from positive to negative i.e 2*(difference from positive to negative)
c. **Phase:** The phase is calculated based on the change in value of the price.
d. **Price Change:** The change in price between subsequent days is also used as a parameter.

After extracting the parameters (xTest), I then query the learner for yTest. The retrieved yTest is the price difference which is then converted to its absolute value.

d) Calculate the Correlation Coefficient and Root Mean square error:

I then calculate the correlation coefficient and Root Mean square error of yPredicted and YActual for the datasets.

Correlation coefficient is calculated by:
corrCoeff= numpy.corrcoef(yTest, yActual)0,1]

Root Mean Square error is calculated by:
*rmse = numpy.sqrt(numpy.mean((yResult - yActual)**2));*

ii) INPUT/OUTPUTS:

Stock Price Forecaster:

a) Program file: forecaster.py

Execution: python forecaster.py

Execution 1 :

Dataset: ML4T-292.csv

SAMPLE INPUT:

```
python forecaster.py
```

Input files:

Training: ML4T-000.csv to ML4T-199.csv
Querying: ML4T-292.csv

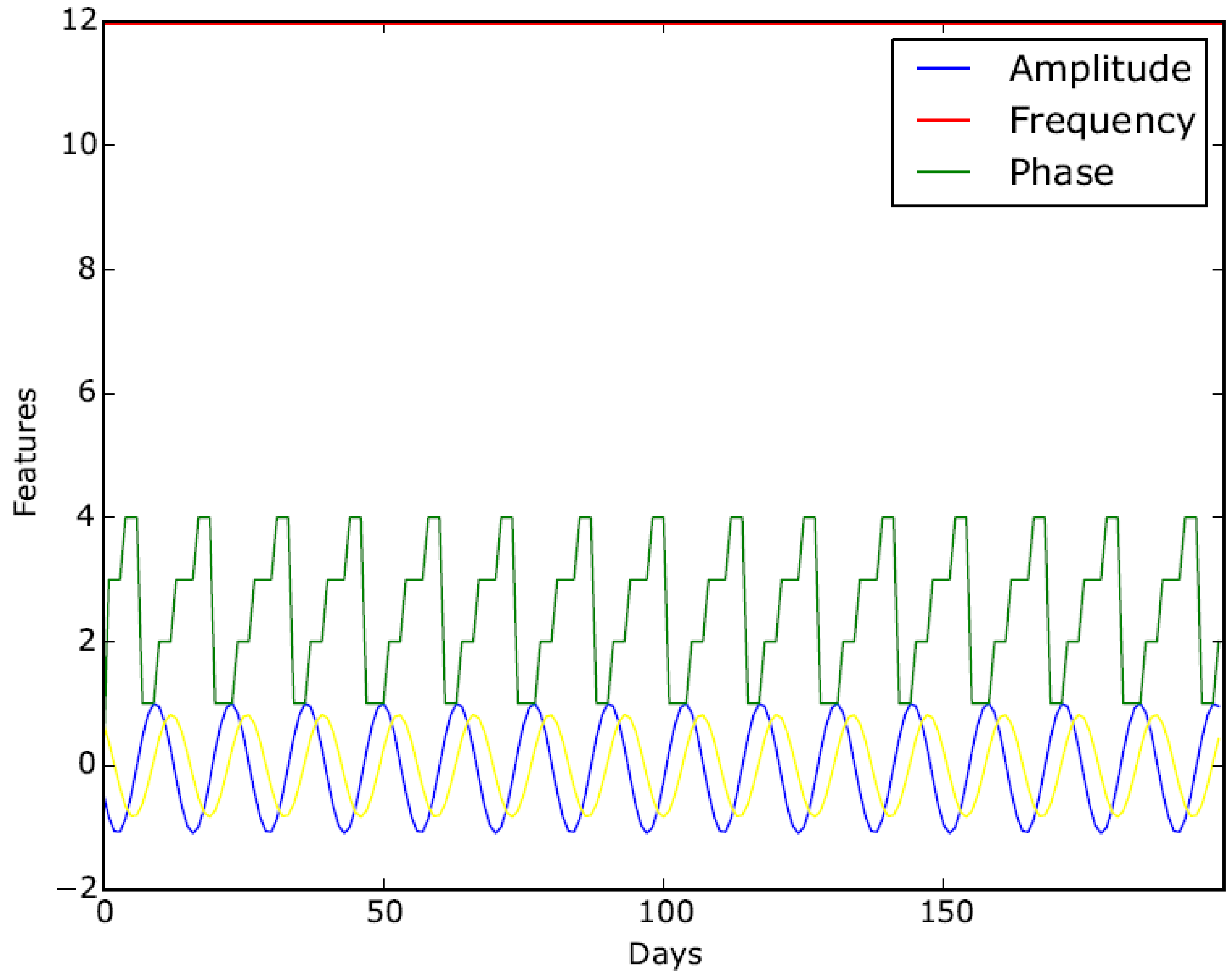
SAMPLE OUTPUT:

1) **Learner: KNNLearner (k=7)**

Calculating Correlation Coefficient for the data set **ML4T-292.csv**...
Correlation Coefficient = 0.895667108255

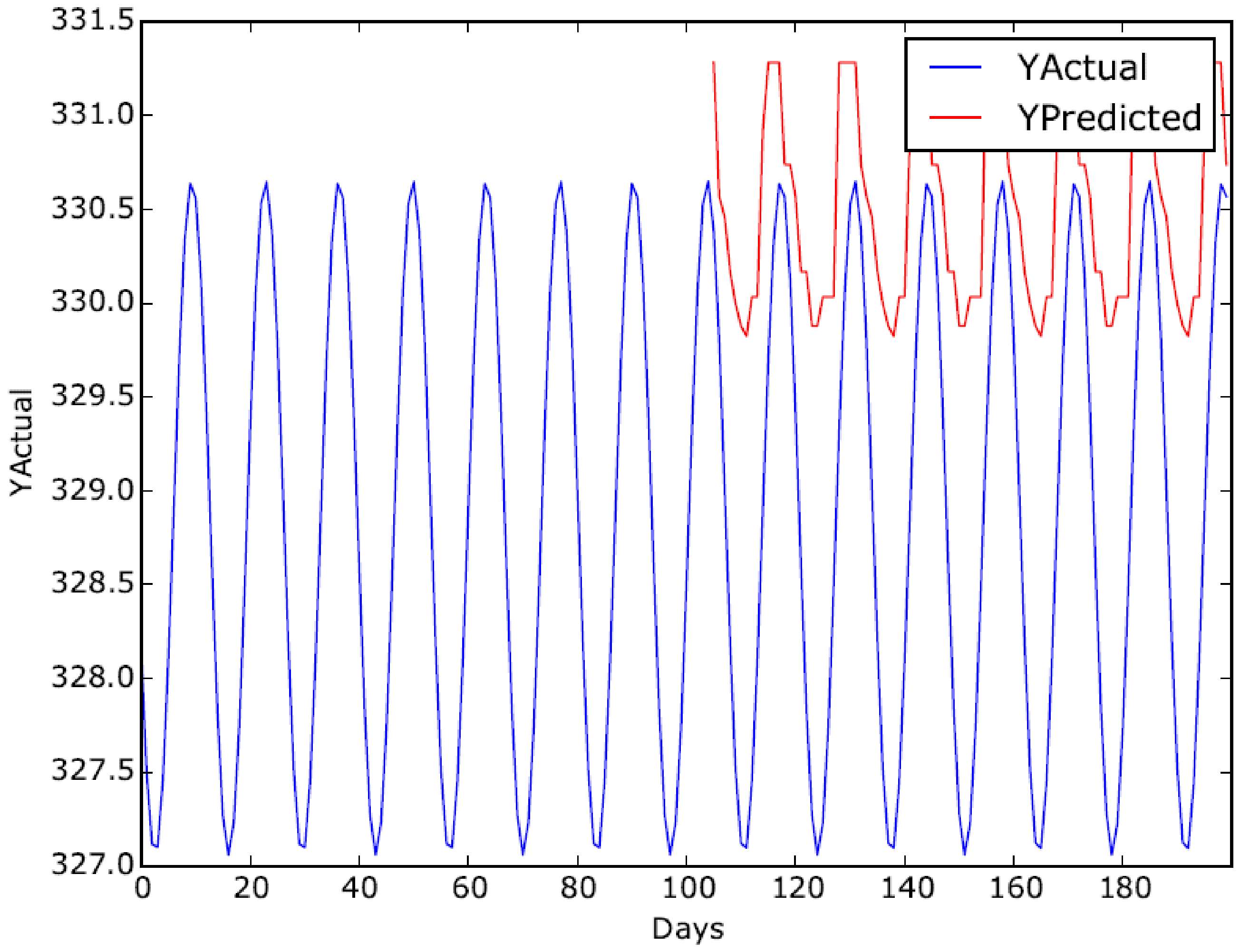
Calculating RMS Error for the data set **ML4T-292.csv**...
RMS Error = 1.8432453371

2) Timeseries plot of Days vs Features(Amplitude, Frequency, Phase, Price difference) for the first 200 days of dataset ML4T-292.csv:



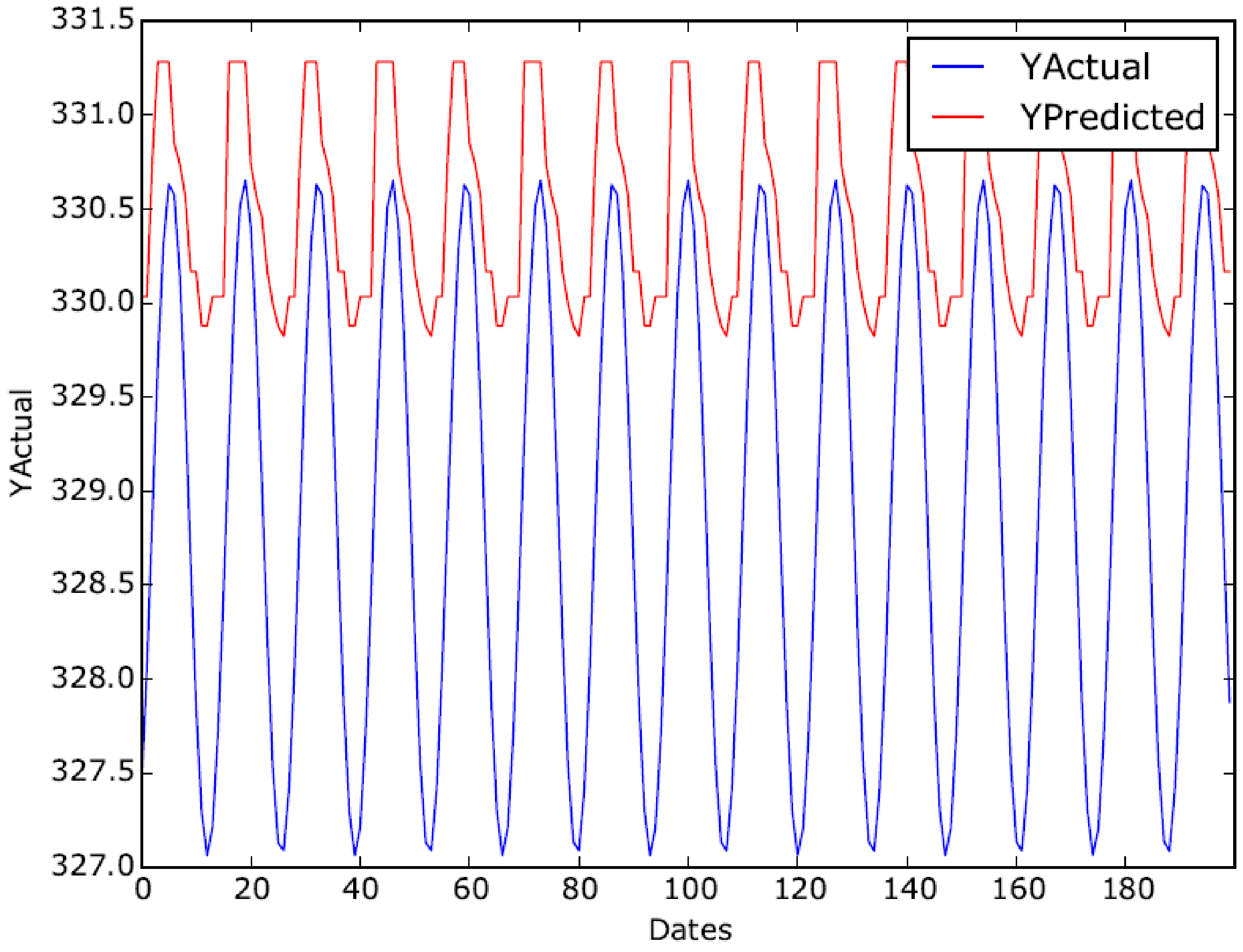
The **frequency(red)**, **amplitude(blue)**, **phase(green)** and **price change(yellow)** **are the features used.**

3) **Timeseries plot of Days vs (YPredict,YActual) for the first 200 days of the dataset ML4T-292.csv:**



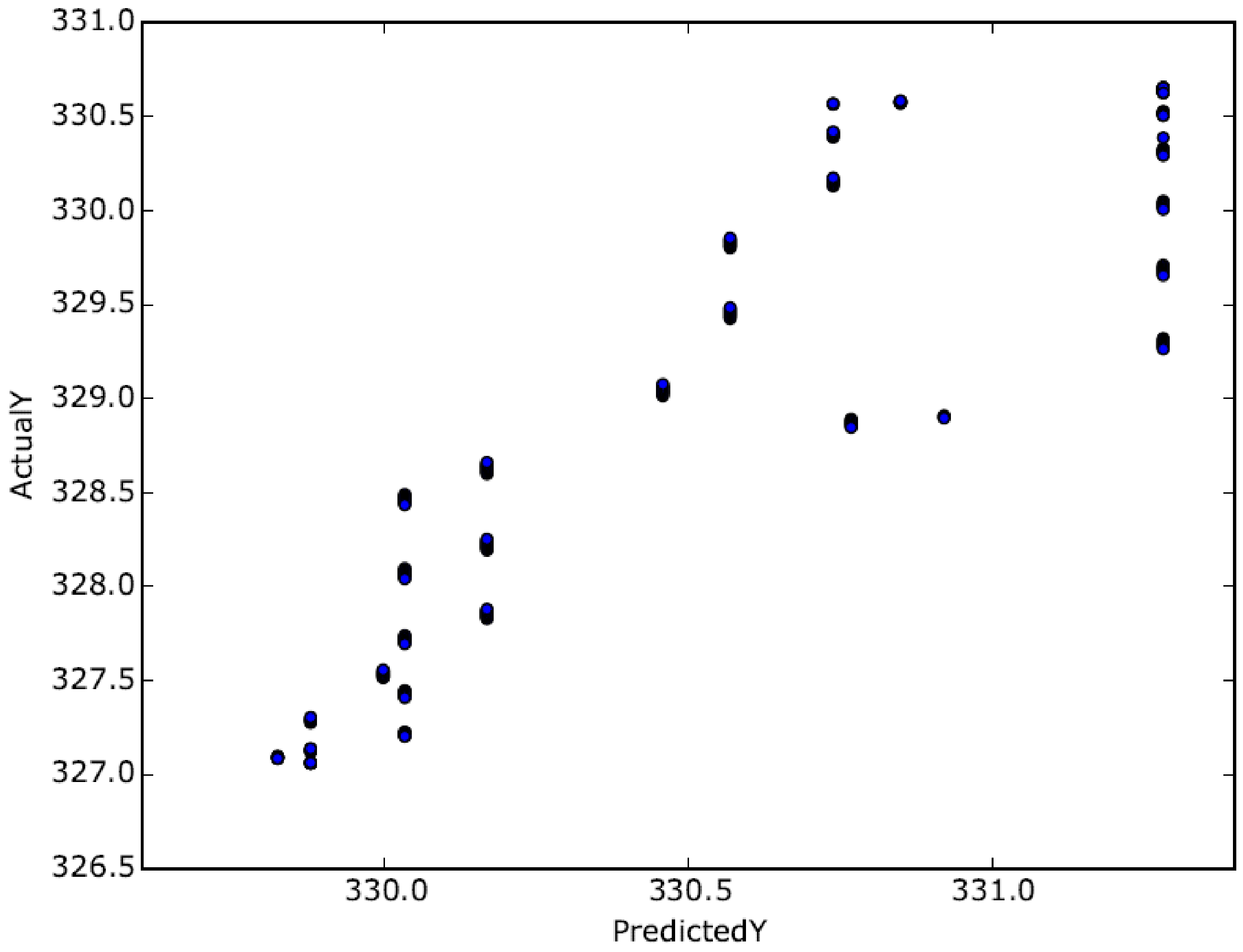
The average difference between actual and predicted values is + OR – 3.

4) **Timeseries plot of Days vs (YPredict,YActual) for the last 200 days of the dataset ML4T-292.csv:**



The average difference between actual and predicted values is + OR – 3.

5) Scatterplot of YPredict vs YActual for the dataset ML4T-292.csv:



__**Execution 2:**__

Dataset: ML4T-324.csv

```
python forecaster.py
```

Input files:

Training: ML4T-000.csv to ML4T-199.csv
Querying: ML4T-324.csv

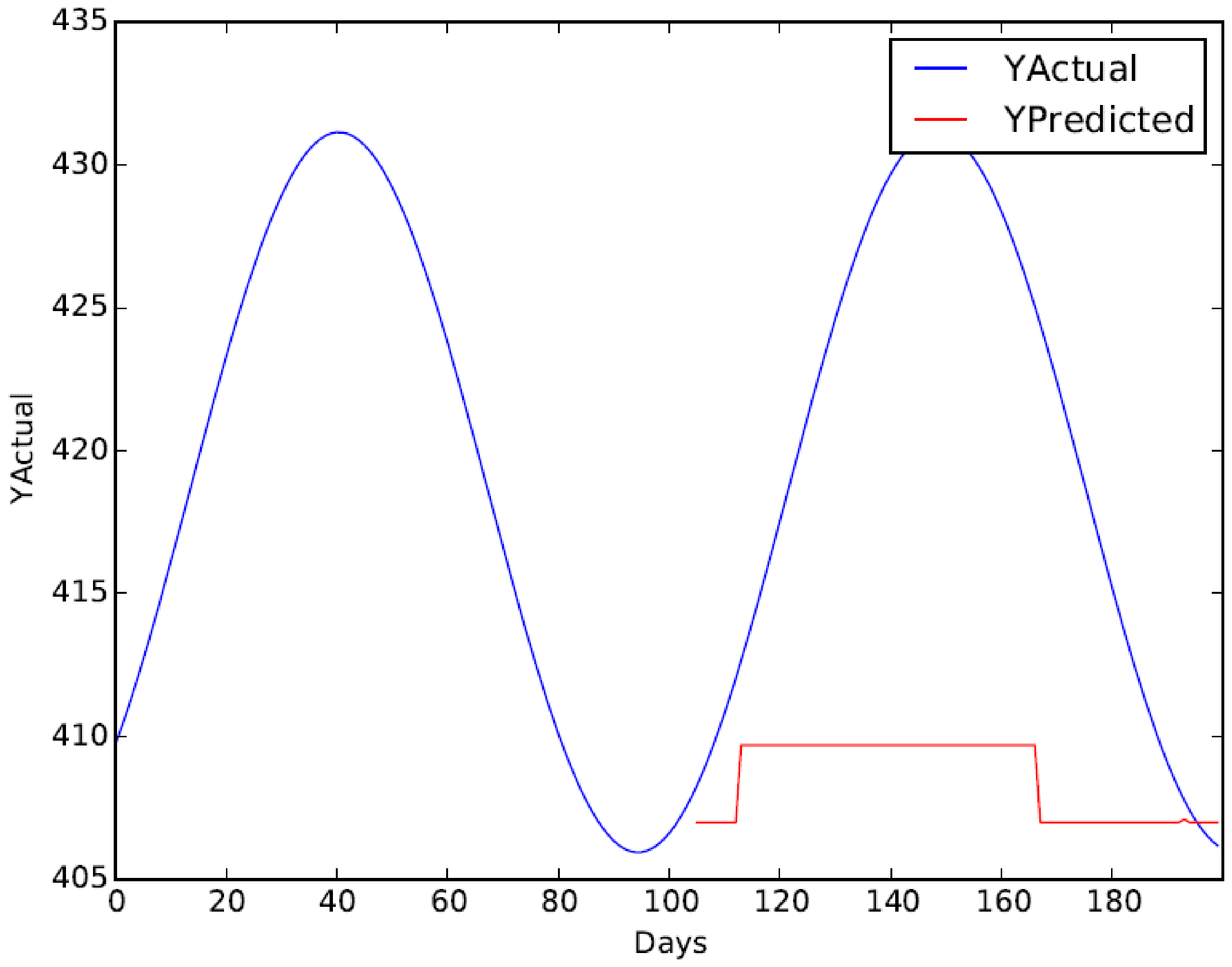
__**SAMPLE OUTPUT:**__

1) **Learner: KNNLearner (k=7)**

Calculating Correlation Coefficient for the data set **ML4T-324.csv**...
Correlation Coefficient = 0.767062789222

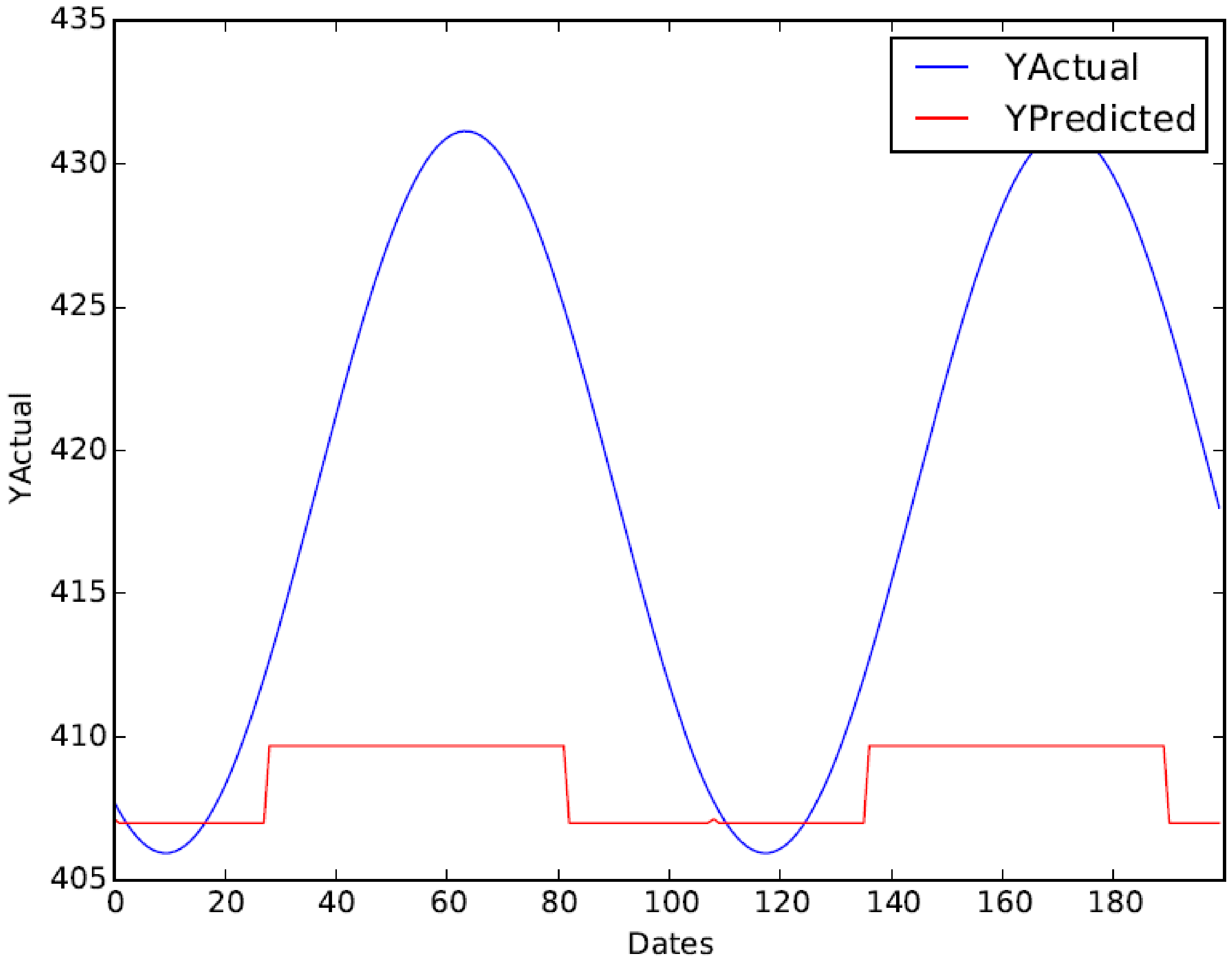
Calculating RMS Error for the data set **ML4T-324.csv**...
RMS Error = 13.4159577232

2) Timeseries plot of Days vs (YPredict,YActual) for the first 200 days of the dataset ML4T-324.csv:



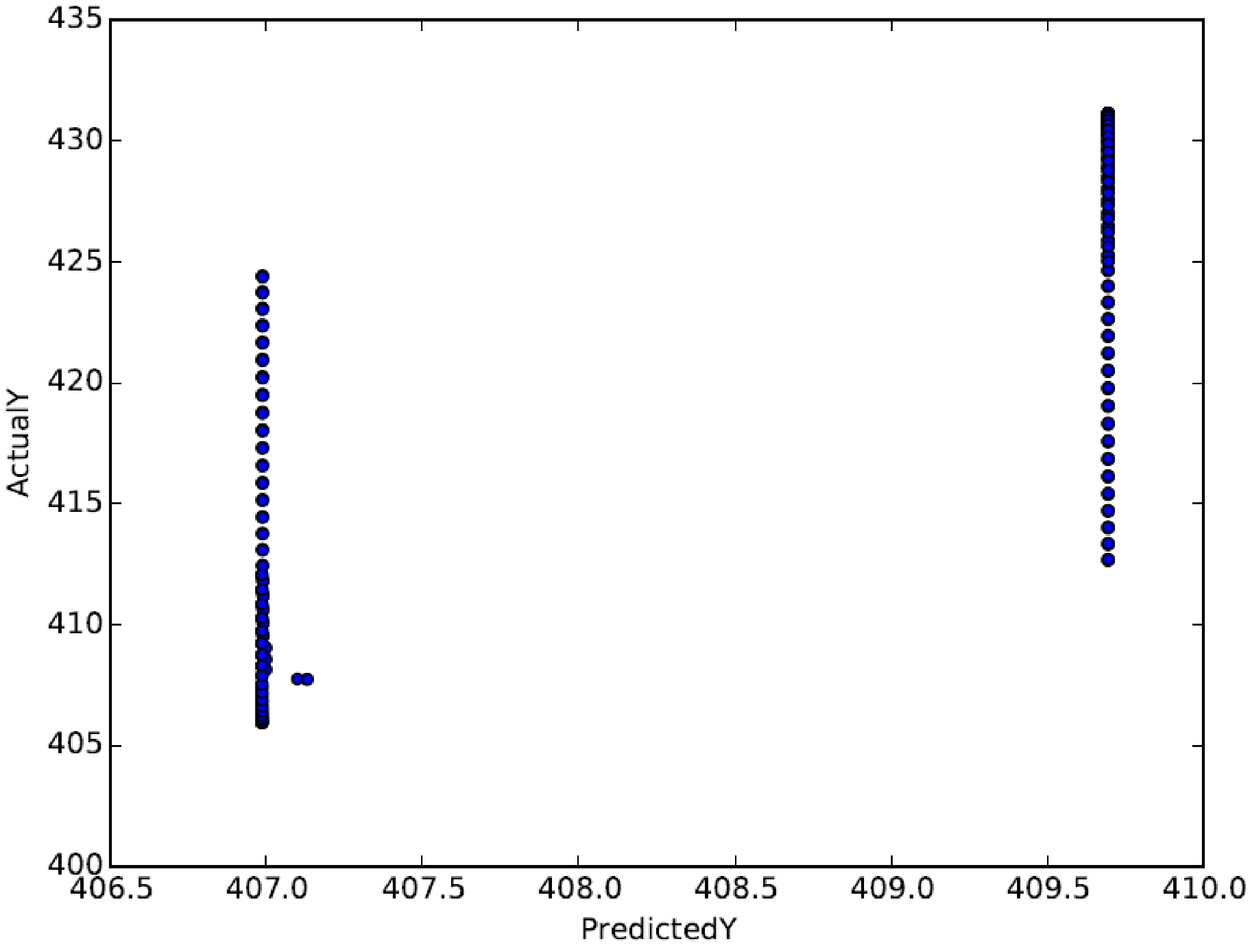
The average difference between actual and predicted values is + OR – 20.

3) Timeseries plot of Days vs (YPredict,YActual) for the last 200 days of the dataset ML4T-324.csv:



The average difference between actual and predicted values is + OR – 20.

4) Scatterplot of YPredict vs YActual for the dataset ML4T-324.csv:



QUESTIONS:

1) **Method and indicators used for learning:**

   I used the KNN Learner (k=7) for training the dataset.

   The value of k was selected based on trial and error:

   - **k=1 Correlation Coefficient= 0.883568937318**
   - **k=3 Correlation Coefficient = 0.873494012414**
   - **k=7 Correlation Coefficient = 0.895667108255**
   - **k=10 Correlation Coefficient = 0.873441895218**
   - **k=12 Correlation Coefficient = 0.773546687385**
   - **k=15 Correlation Coefficient = 0.802716519573**
   - **k=20 Correlation Coefficient = 0.868201940047**

   I find that the correlation coefficient is highest for k=7.

   Method for training:
   Since the input is a sine wave, I consider **Amplitude, Frequency and Phase** as indicators. Additionally, I also consider **Price Change** as an indicator.

   Amplitude: For every 100 days, amplitude is calculated by finding the difference between the maximum and minimum value of price during the period

   Frequency: Frequency is twice the distance between the maximum altitude and the minimum altitude.

   Phase: Phase is calculated by monitoring the change in price values from positive to negative and vice versa.

   Price Change: Price change is calculated by finding the difference between prices of subsequent dates.

   More details about the indicators (parameters) have been explained in the Methodology section.

2) **Why do you think the method worked well?**
   The method worked fairly well (correlation coefficient of 0.896 (for ML4T-292) and 0.767 (for ML4T-324)) because both the test data and the training data had the same shape (i.e a sine wave). So the indicators (phase, amplitude and frequency) learnt by the learner could assist

the learner in querying an unknown value of another sine wave. The method would not perform considerably well if the test data was not a sine wave (Example: the actual stock information).

The current implementation doesnot consider the change in amplitude (for 100 days). Instead, a single max amplitude is selected (for 100 days) resulting in flat amplitudes in the resulting chart.