Kaan Yeşildal - 21202704
EEE492-1
2019-05-24

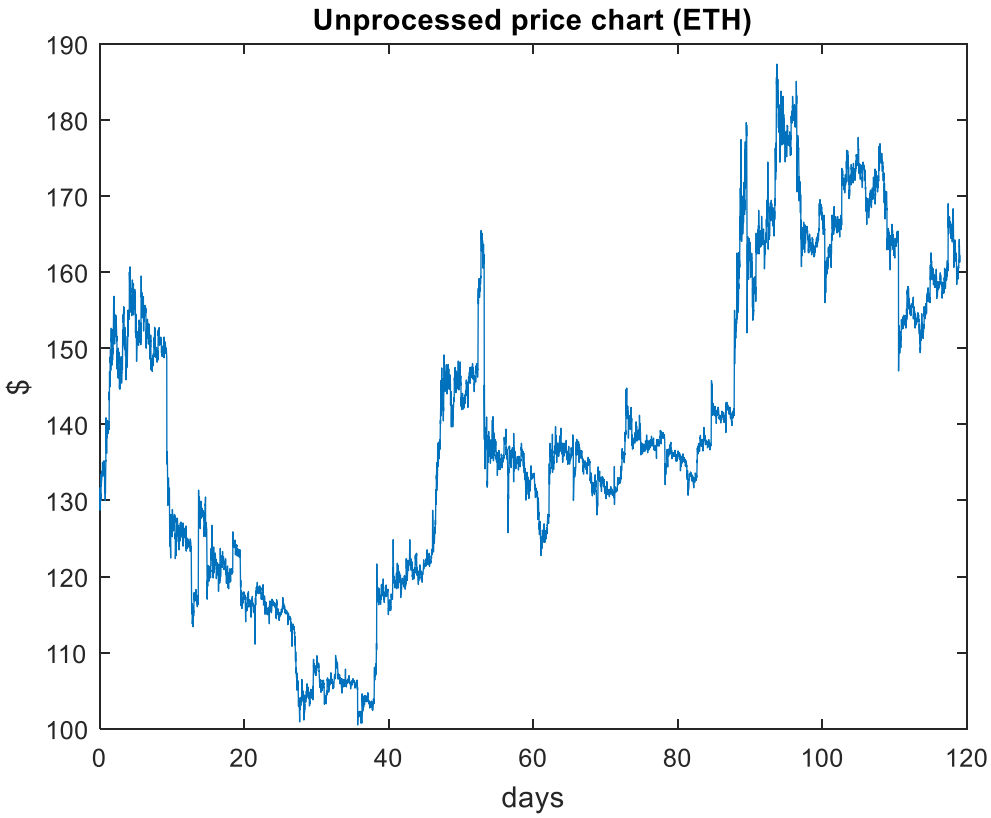
EEE492 PROJECT REPORT

Table of Contents

1) INTRODUCTION

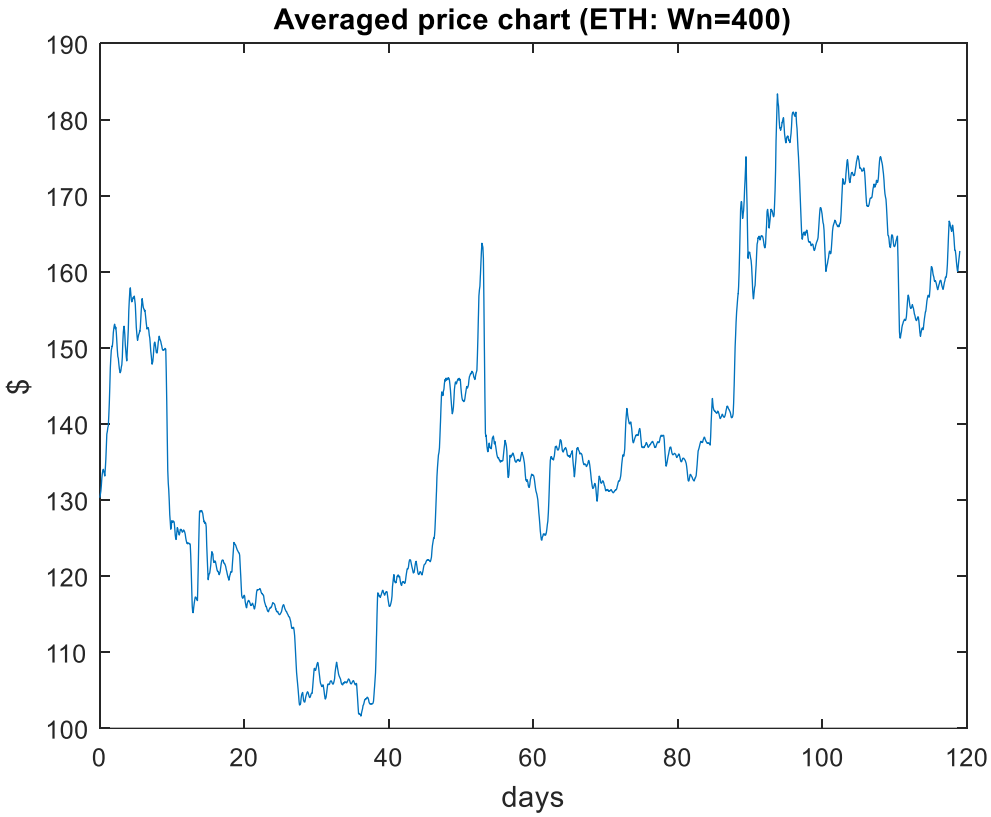
The purpose of this Project is to create a classification tool for cryptocurrency price data. This classifier uses peak analysis to detect the peaks of the price data following a moving average filtering process. Two crypto-coins are used in this project, which are Ethereum (ETH) and LiteCoin (LTC). 4-month-period data for both coins are used. The time window for both data is the same and is from 2019-01-01-00:00 to 2019-05-05-00:00 [1].

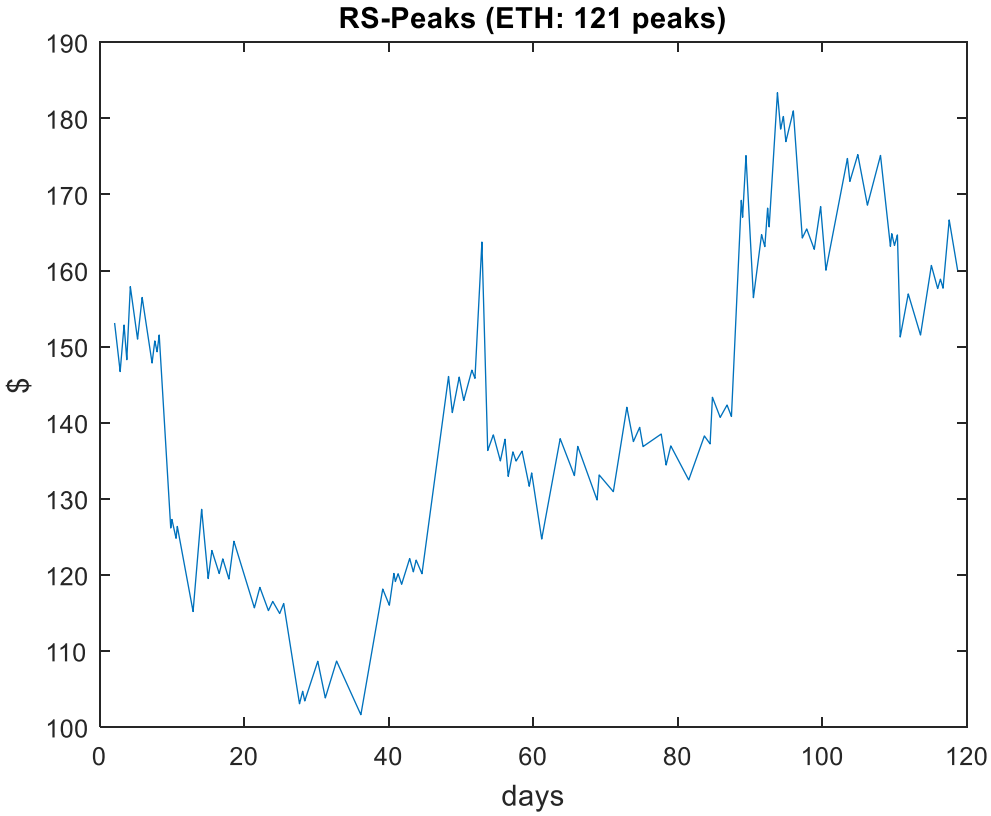
2) PRE-PROCESSING

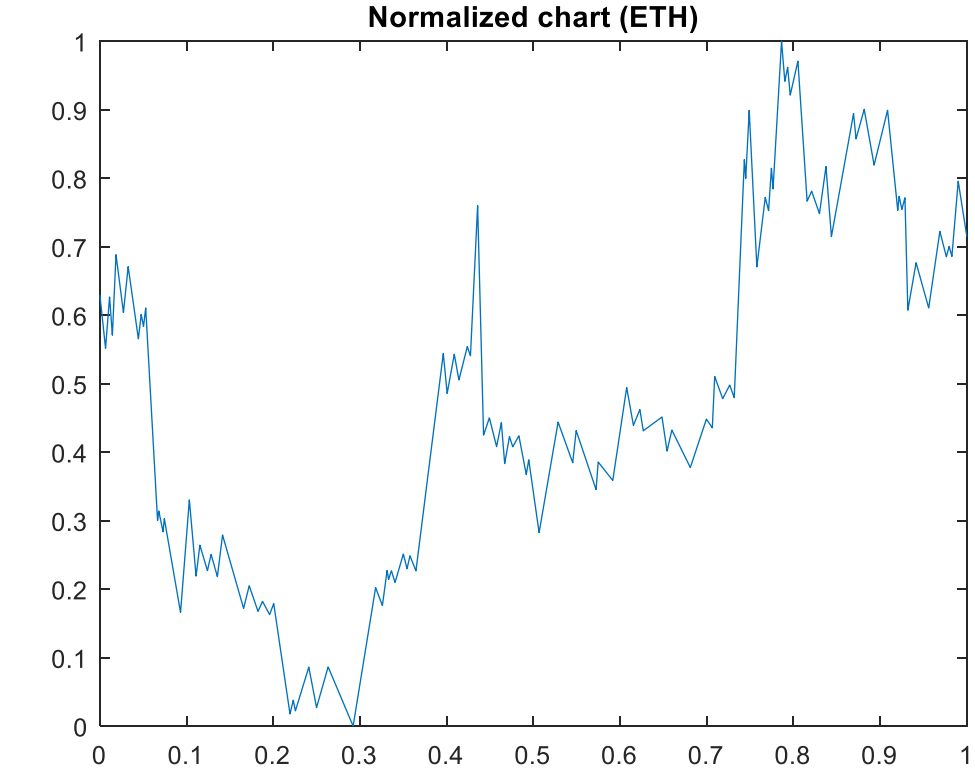


Pre-processing is handled with the function **'RSpeaks.'** The window size for the moving averaging filter is determined by the interval of our trading window. The peak analysis is done by QRS analysis with the **'findpeaks'** function that comes with signal-processing toolbox of MATLAB. This function allows us to find the peaks as R and S points. The R points correspond to peaks occurring due to increasing price movements and the S points correspond to peaks occurring due to decreasing price movements. The RS-peaks found then are normalized with unity-based normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$




RS-Peaks (ETH: 121 peaks)
$
days
190
180
170
160
150
140
130
120
110
100
0
20
40
60
80
100
120


Normalized chart (ETH)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

After the data is processed, the function **'createFeatures'** creates feature vectors which are derived of the magnitude and the time difference of the R and S points on the chart. We divide magnitude by time to calculate the angle of the line we are using.

3) CLASSIFIER

After the normalization, linearized movement parts are used to derive feature vectors for both the angles and the durations of movements. Each feature vector corresponds to the next price movement (label). For feature vectors of durations, notation is:

$$x_{time} = \begin{bmatrix} t_0 - t_{-1} \\ t_{-1} - t_{-2} \\ t_{-2} - t_{-3} \\ t_{-3} - t_{-4} \end{bmatrix} \qquad y_{time} = t_{next} - t_0$$

$$x_{angle} = \begin{bmatrix} (m_0 - m_{-1})/(t_0 - t_{-1}) \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \qquad y_{angle} = \frac{m_{next} - m_0}{t_{next} - t_0}$$

After the features are created, two methods can be employed for price movement prediction.

a) Intraprediction

This method is applied with the **'intraprediction'** function. The set of feature vectors is divided into two sets, namely: 'trainVectors,' which are the vectors that are before the 'present moment' (i.e. their labels are already known), and 'testVectors' which are in the future and their labels are assumed to be unknown. As the algorithm moves forward in time, labels of the 'testVectors' are predicted. After the prediction, vectors in the set 'testVectors' are transferred to the set 'trainVectors' along with their real labels.

The prediction process employs K-Nearest-Neighbors algorithm. Euclidian distance between the vector whose label to be predicted and each vector in the training set is computed:

$$d_i = d(x, x_i) = \|x, x_i\|$$

K nearest neighbors are found from the K minimum distances. Prediction is then performed by taking weighted or non-weighted average of the labels of the nearest neighbors. If weighted option is employed, reciprocal kernels are used to compute the weights:

$$K_i = \frac{1}{d_i}$$

$$w_i = \frac{K_i}{\sum_{j=1}^{K} K_j}$$

$$y = \sum_{i=1}^{K} w_i y_i$$

If non-weighted option is employed, the prediction is:

$$y = \frac{1}{K} \sum_{i=1}^{K} y_i$$

The function then returns the predictions and mean additive absolute errors for each K-value fed into the function.

b) Interprediction

This method is applied with the **'interprediction'** function. The method is basically the same as the intraprediction method. However, instead of dividing the set of vectors into two, all the vectors are assumed to be in the future, and their labels are predicted by using the feature vectors of an entirely different cryptocurrency, whose price movements are assumed to be in the past.

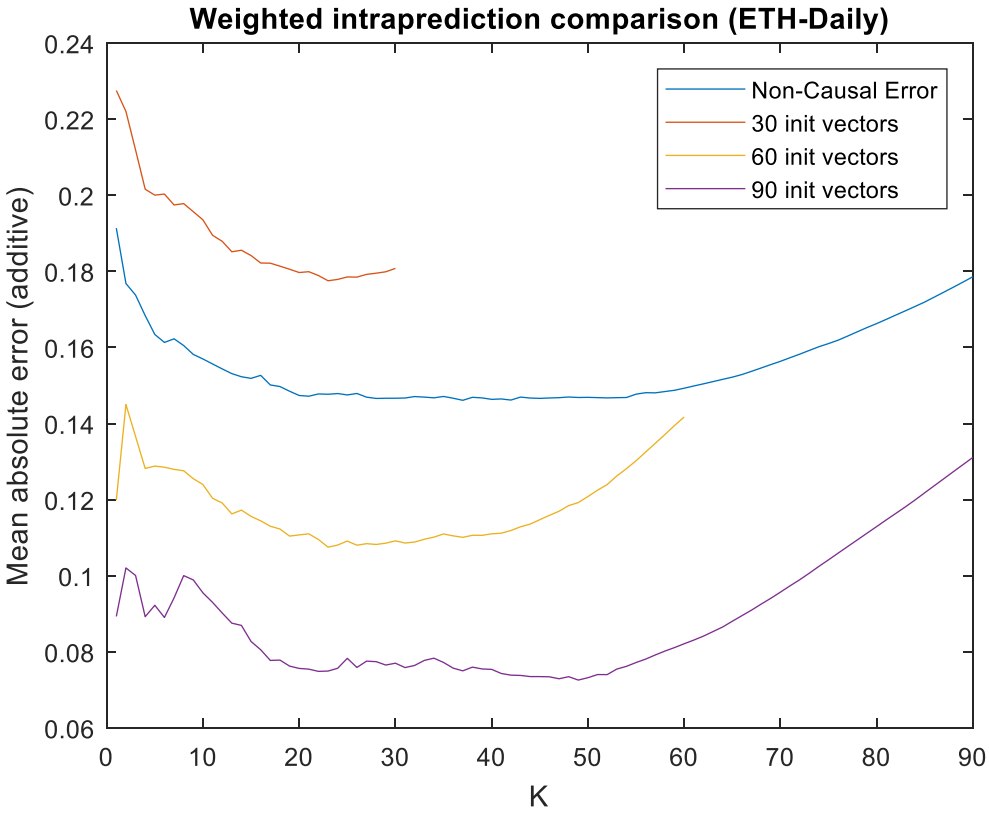
For both of the methods, the returned errors are compared with the 'Non-causal errors' of the currency whose labels are predicted. Non-causal (Standard) errors are computed with the function 'getStandardErrors.' Each vector goes in KNN algorithm and is compared with every other vector (regardless of past or future). If the mean errors returned from the prediction methods are less than non-causal errors, then there is an implication of causality within the cryptocurrency in question.

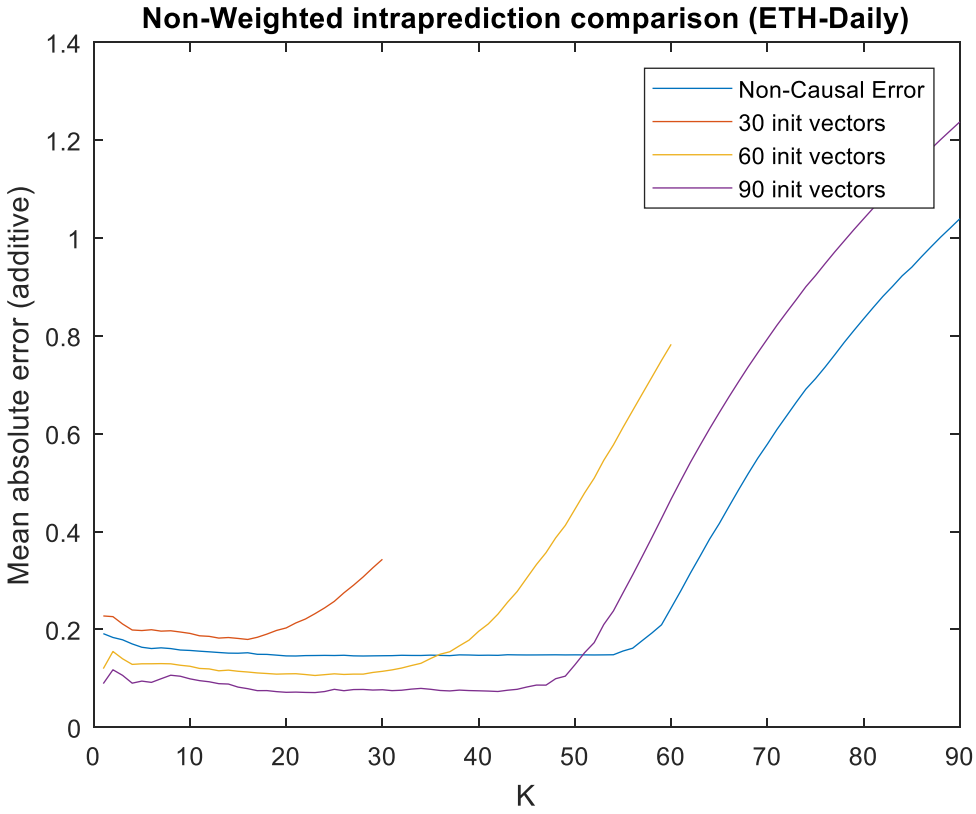
4) EXPERIMENTS AND RESULTS

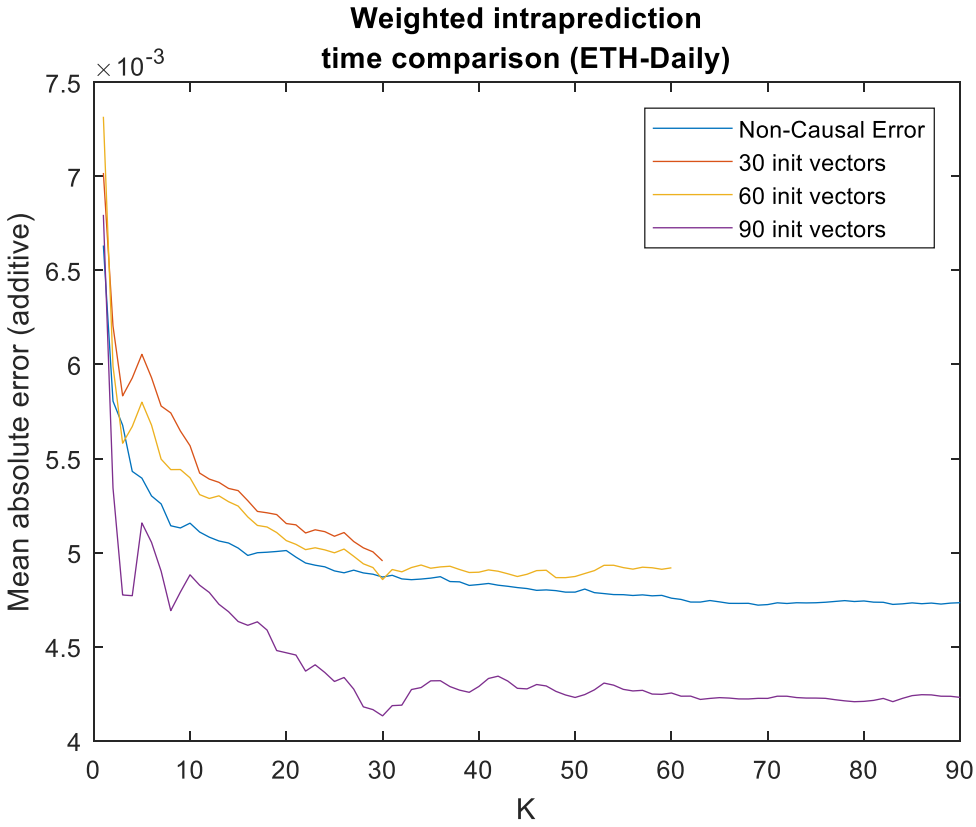
Results of four experiments are given in this section:

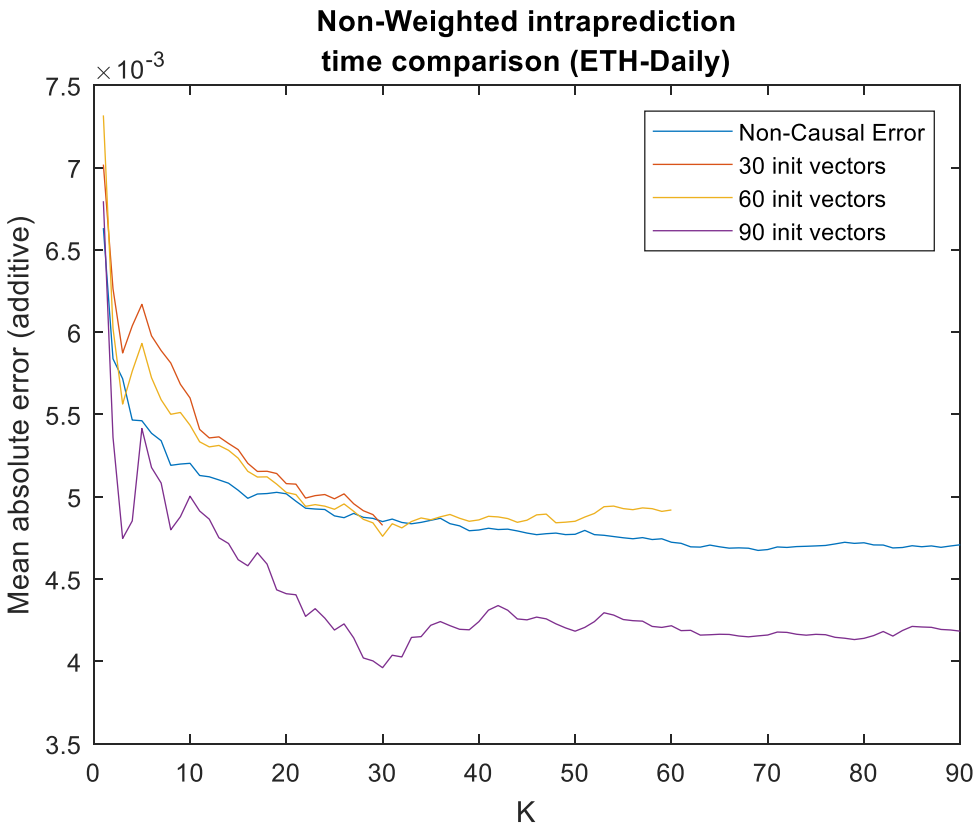
a) Experiment 1

The 4-month-period of price changes of the coin 'Ethereum (ETH)' is used for this experiment. Window length of the moving averaging filter is set to 400, so we'd end up with 121 RS-peaks after pre-processing, so that the average length of each linearized price movement is about a day. The feature vector size is set to 4, so 115 feature vectors are created. Intraprediction method is employed with 30, 60, 90 vectors in the set trainVectors. The experiment is run for both weighted and non-weighted options. Both the angles and the durations are predicted for every case. The error comparison for angles and durations is provided in the charts below:



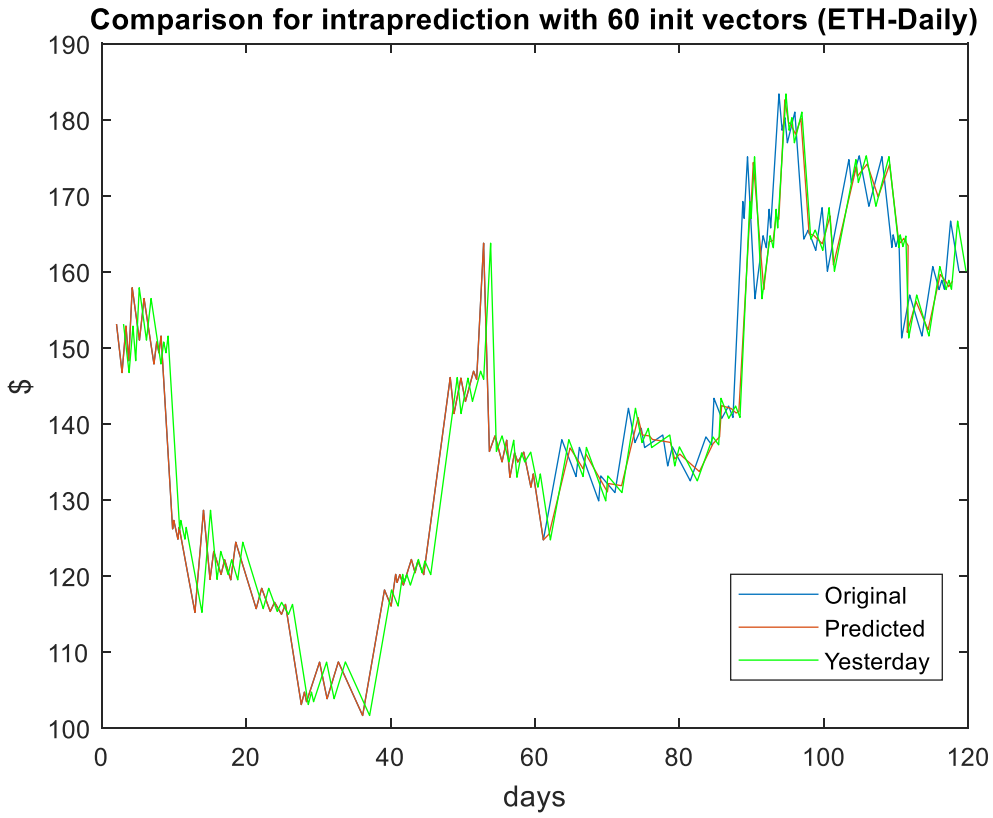

Non-Weighted intraprediction comparison (ETH-Daily)
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
10
20
30
40
50
60
70
80
90


Weighted intraprediction
time comparison (ETH-Daily)
×10-3
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors
4
4.5
5
5.5
6
6.5
7
7.5
0
10
20
30
40
50
60
70
80
90


Non-Weighted intraprediction
time comparison (ETH-Daily)
×10^-3
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors
7.5
7
6.5
6
5.5
5
4.5
4
3.5
0
10
20
30
40
50
60
70
80
90

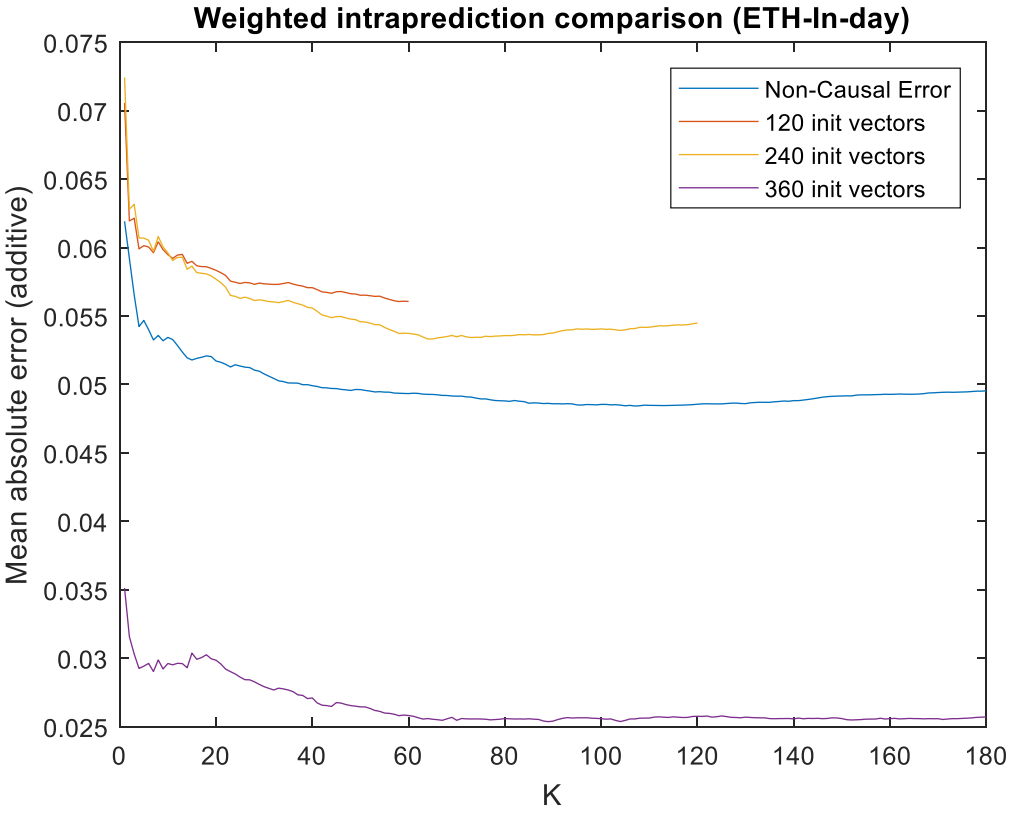
The price chart is reconstructed for the case with 60 vectors in training set and the weighted option. The result is provided in the chart below along with the result of prediction which predicts by setting yesterday's to today's, such that:

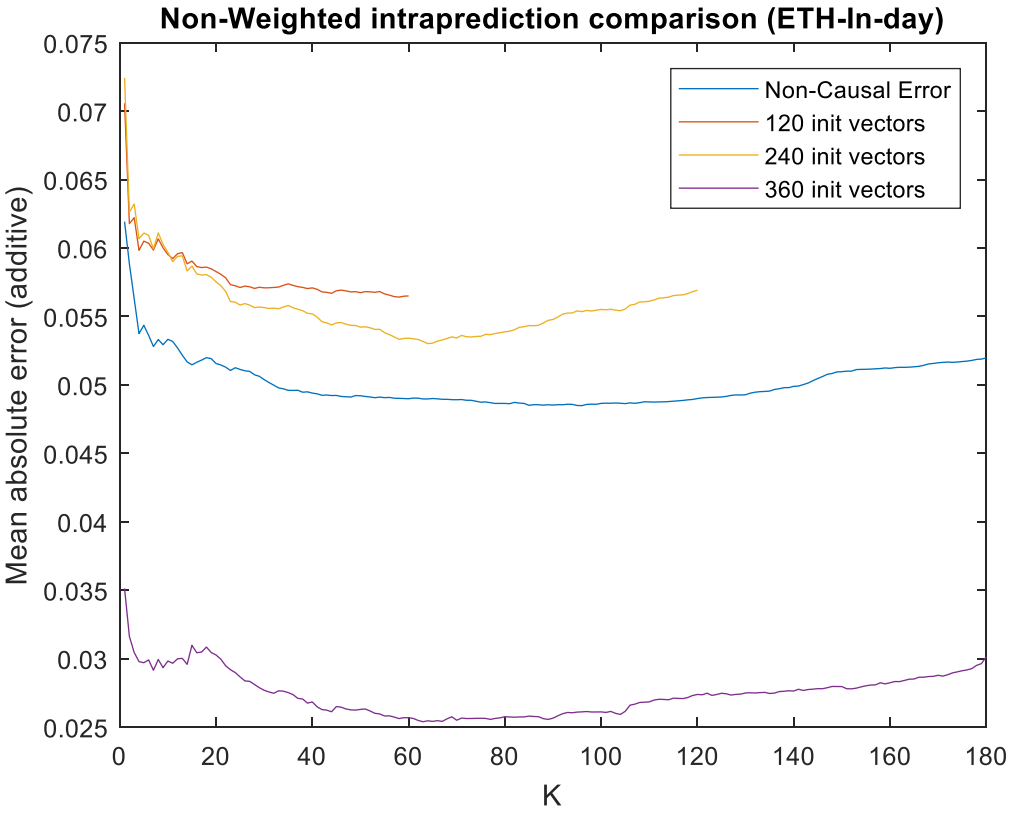
$$P'(t) = P(t - \Delta t_{ave}), \quad \Delta t_{ave} \cong 24h$$

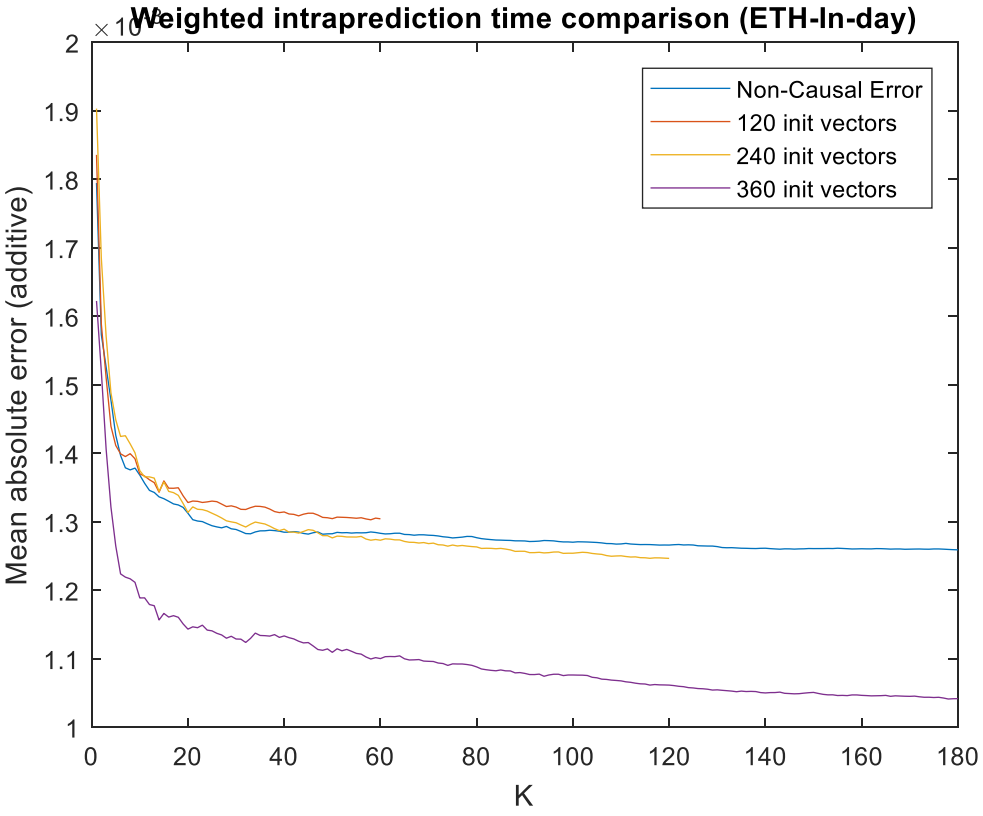


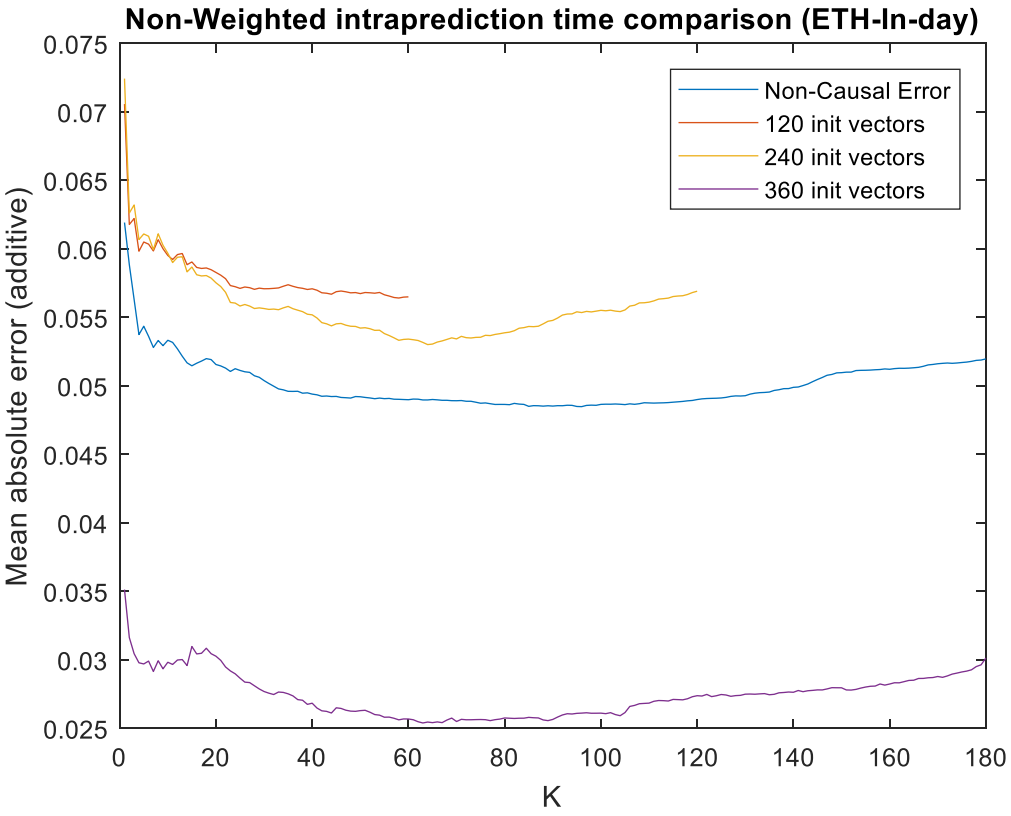
b) Experiment 2

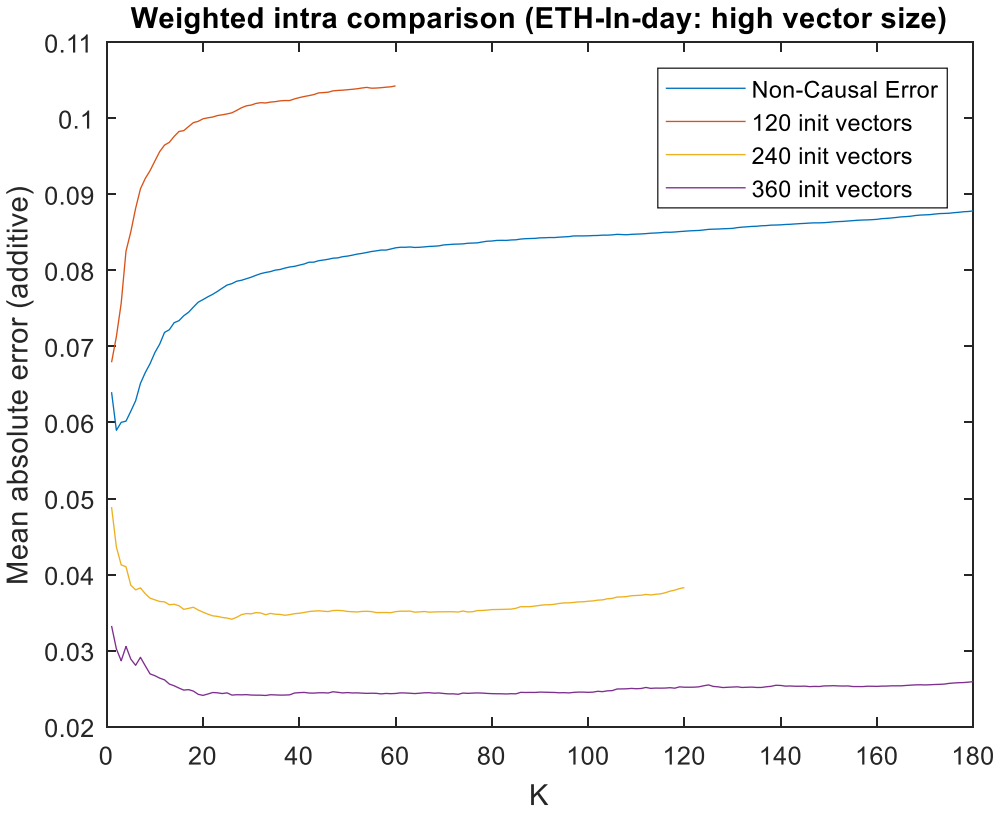
The same 4-month-period of price changes of the coin 'Ethereum (ETH)' is used for this experiment. Window length of the moving averaging filter is set to 40, so we'd end up with 490 RS-peaks after pre-processing, so that the average length of each linearized price movement is about a six hours. The experiment is run two times: one for the vector size of 4, and one for the vector size of 16. Intraprediction method is employed with 120, 240, 360 vectors in the set trainVectors. The experiment is run for both weighted and non-weighted options. Both the angles and the durations are predicted for every case. The error comparison for both angles and times are provided in the charts below:

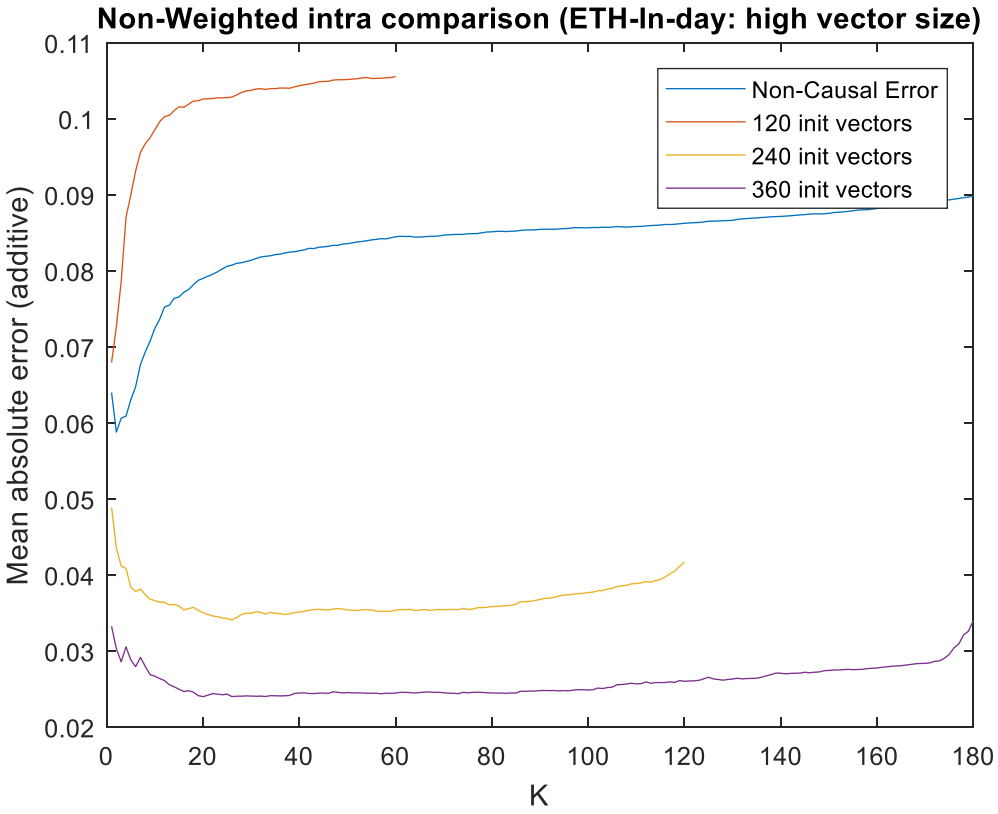

Weighted intraprediction comparison (ETH-In-day)
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors
Mean absolute error (additive)
K
0.075
0.07
0.065
0.06
0.055
0.05
0.045
0.04
0.035
0.03
0.025
0
20
40
60
80
100
120
140
160
180

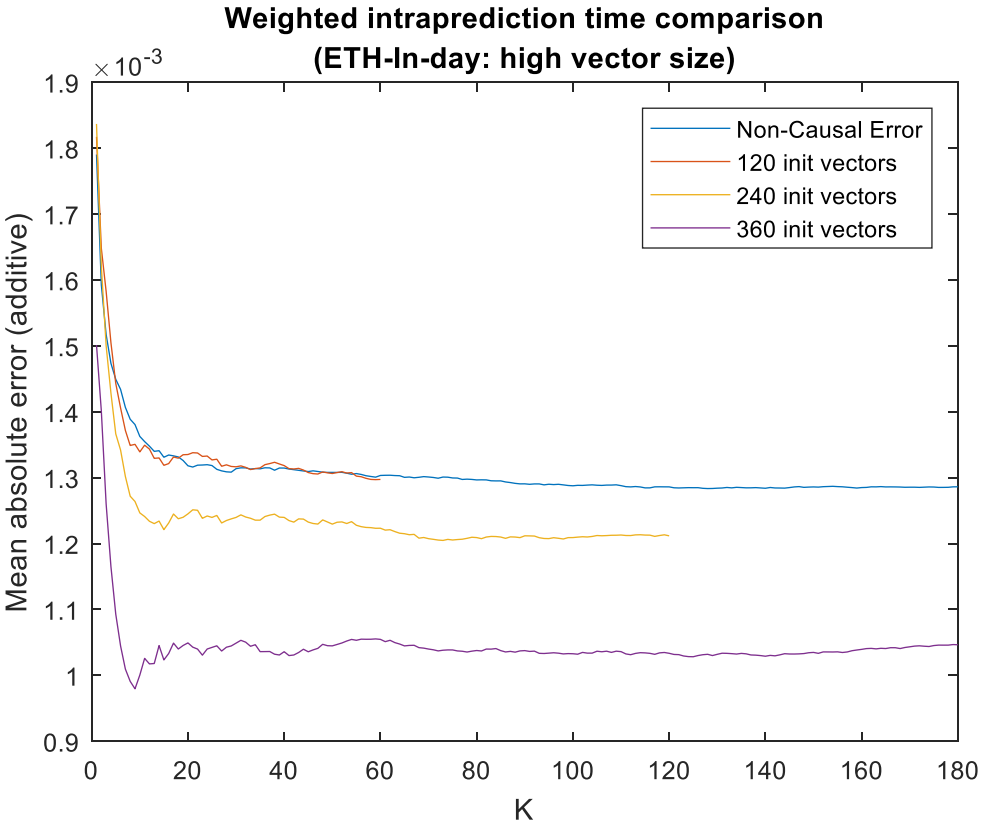

Non-Weighted intraprediction comparison (ETH-In-day)
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors
Mean absolute error (additive)
K
0.075
0.07
0.065
0.06
0.055
0.05
0.045
0.04
0.035
0.03
0.025
0
20
40
60
80
100
120
140
160
180

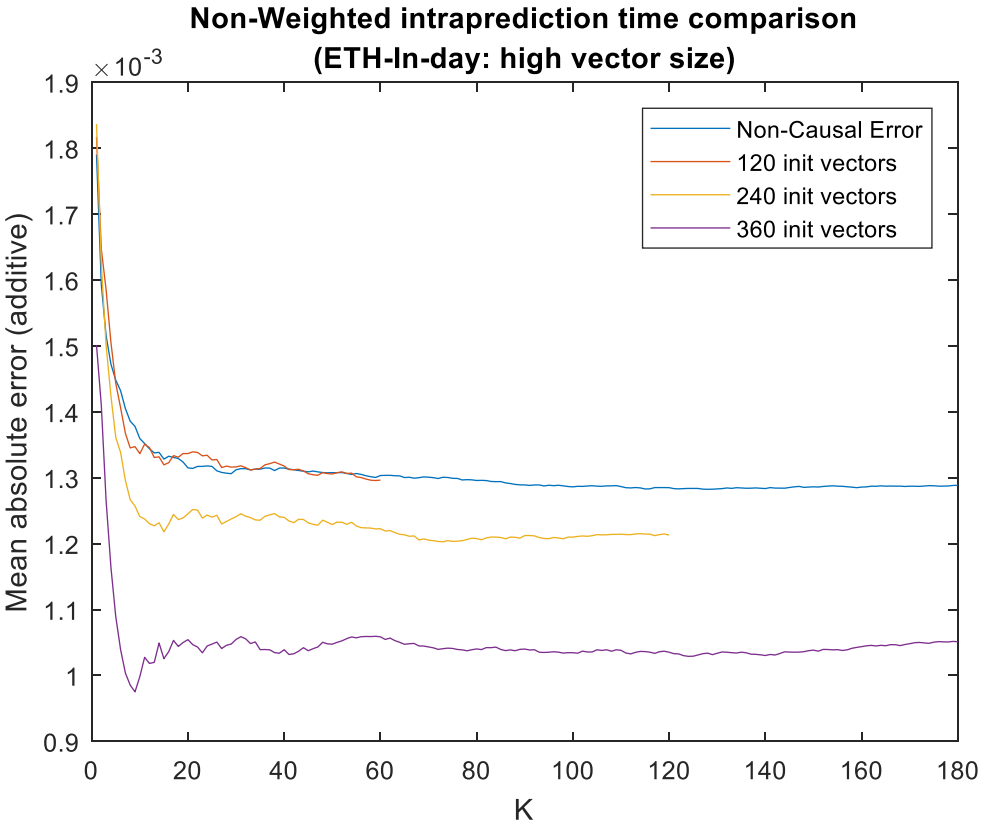

Weighted intraprediction time comparison (ETH-ln-day)
×10^4
Mean absolute error (additive)
K
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors

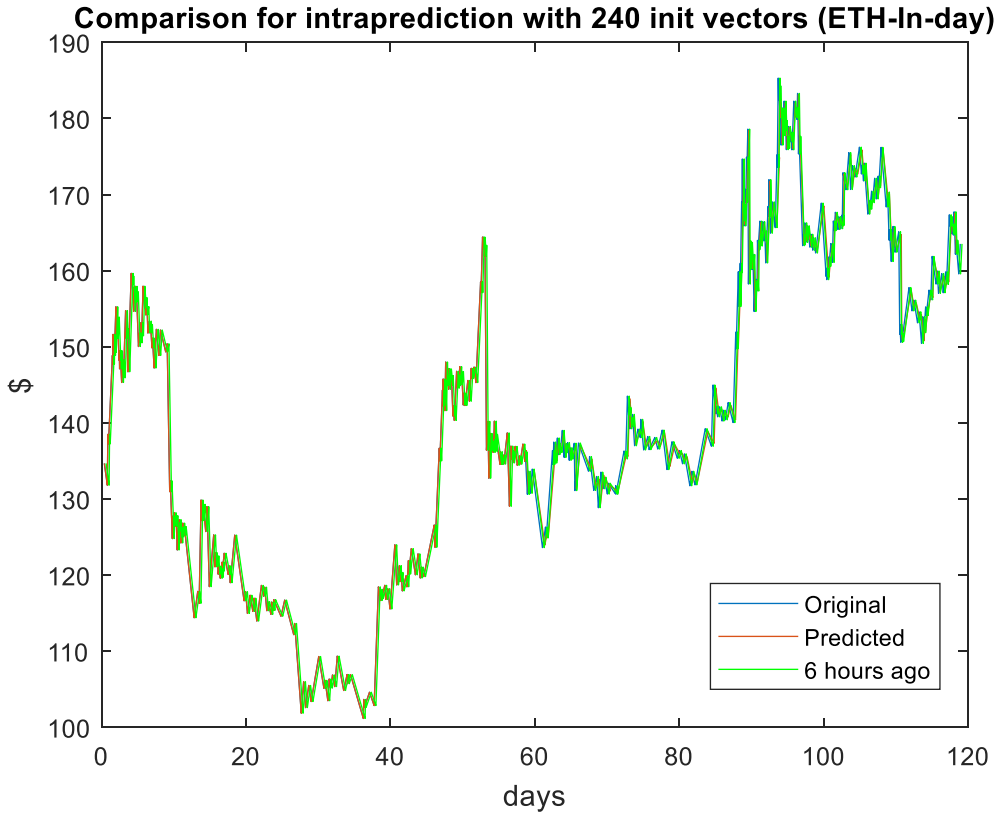

Non-Weighted intraprediction time comparison (ETH-ln-day)
Mean absolute error (additive)
K
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors


Weighted intra comparison (ETH-In-day: high vector size)
Mean absolute error (additive)
K
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors
0.11
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0
20
40
60
80
100
120
140
160
180


Non-Weighted intra comparison (ETH-In-day: high vector size)
Mean absolute error (additive)
K
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors
0.11
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0
20
40
60
80
100
120
140
160
180

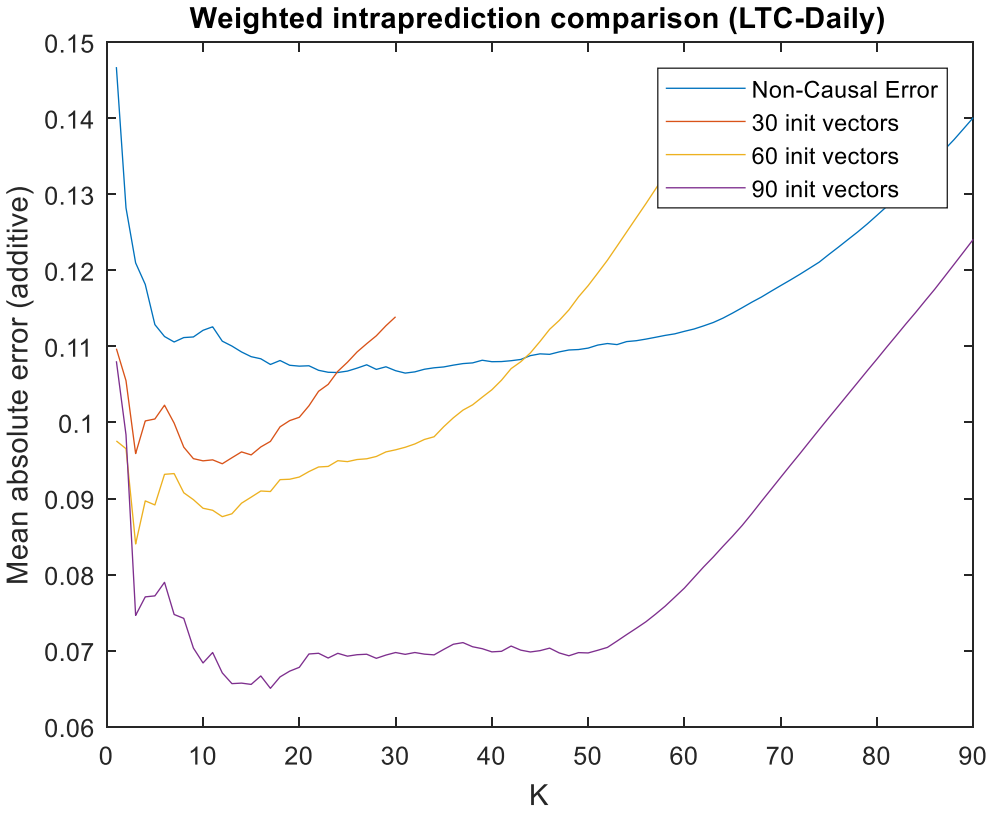

Weighted intraprediction time comparison
(ETH-ln-day: high vector size)
×10^-3
Mean absolute error (additive)
K
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors

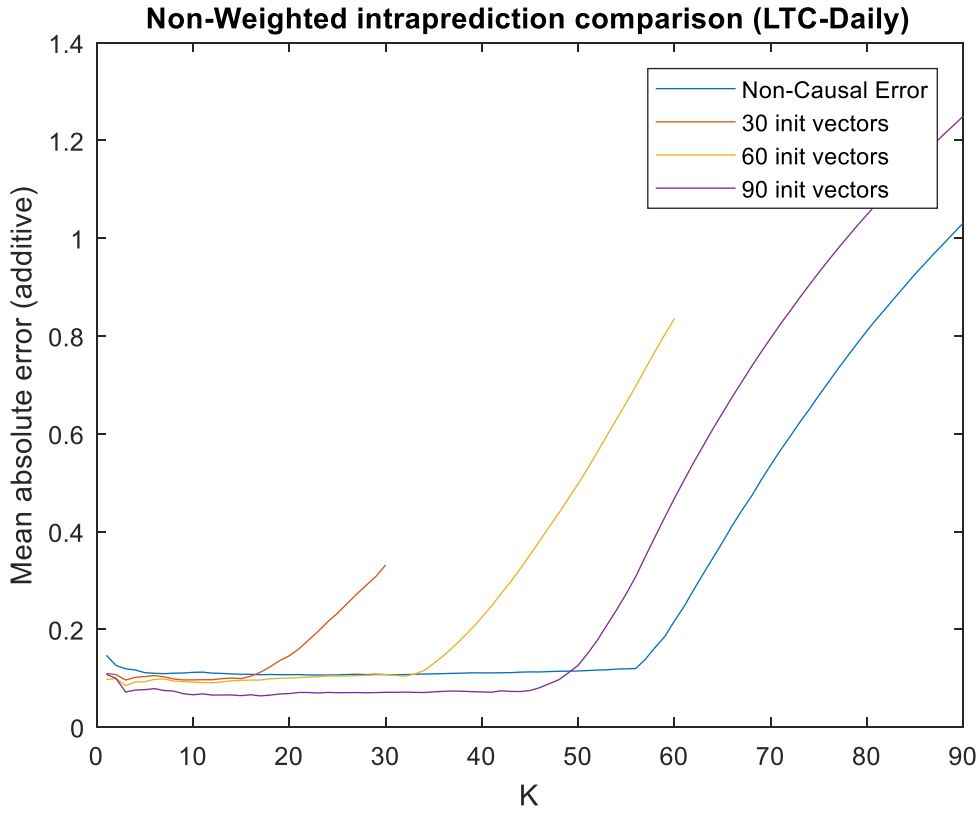

Non-Weighted intraprediction time comparison
(ETH-ln-day: high vector size)
×10^-3
Mean absolute error (additive)
K
Non-Causal Error
120 init vectors
240 init vectors
360 init vectors

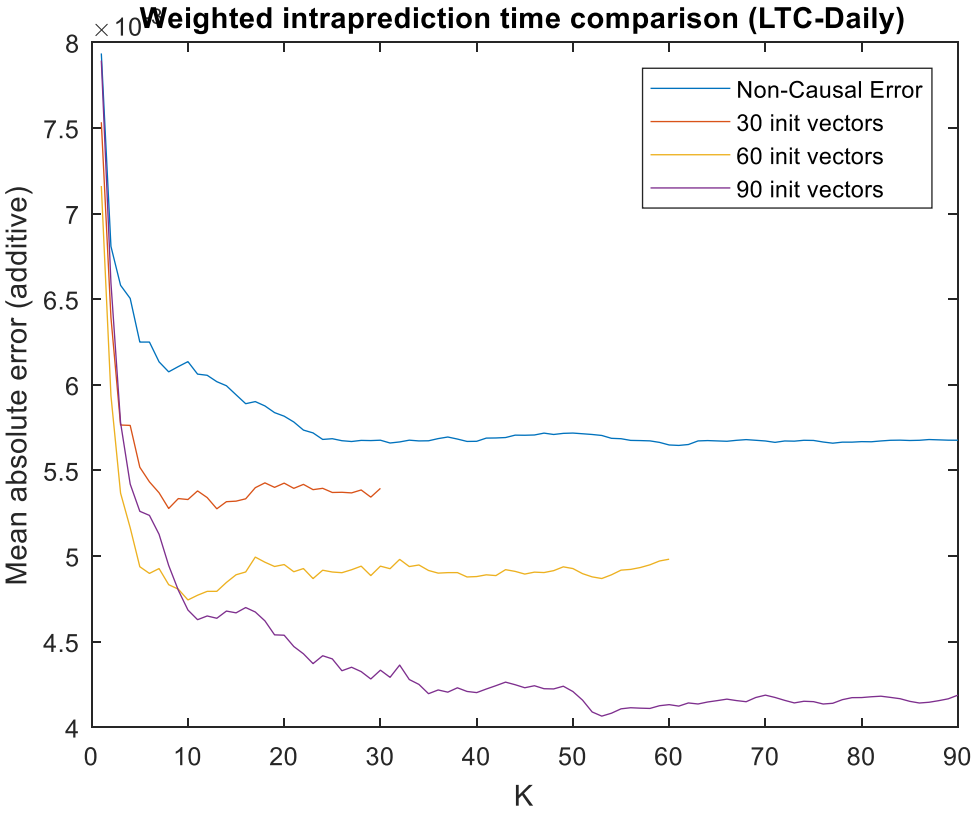


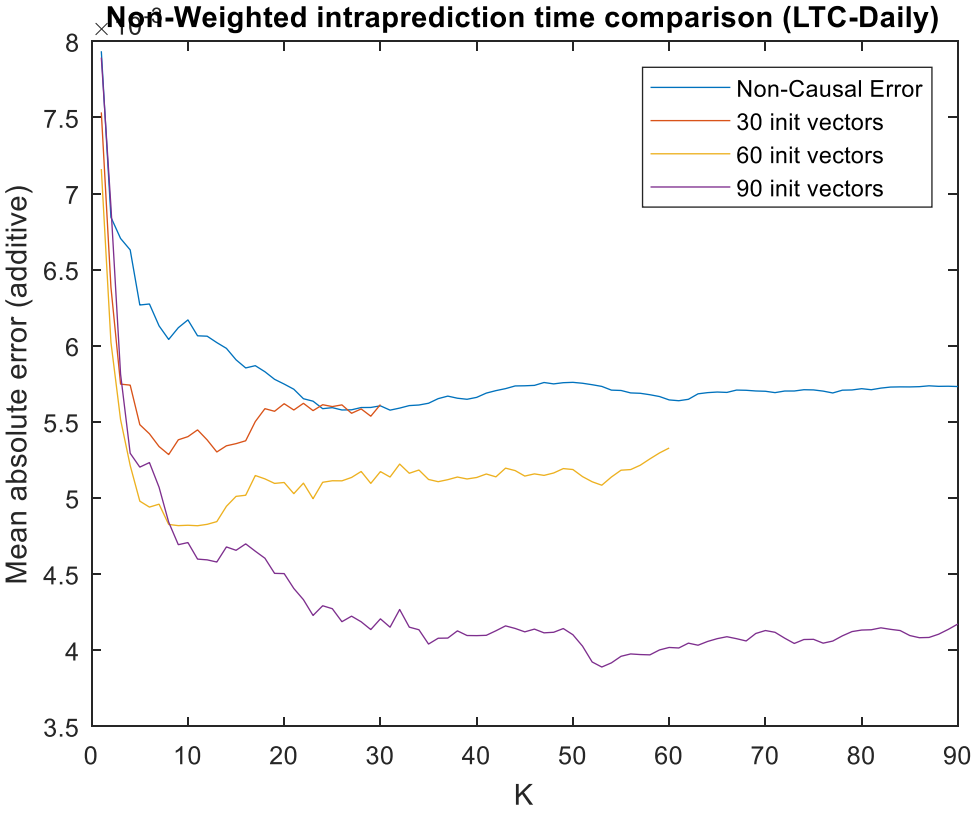
c) Experiment 3

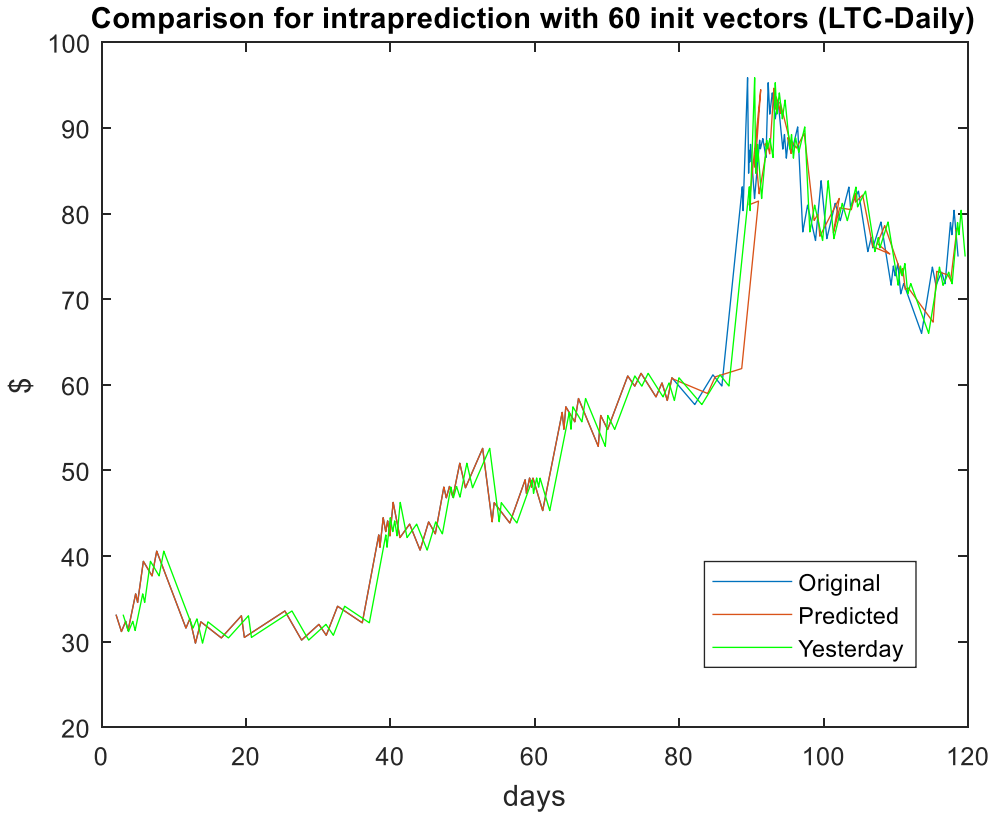
The same procedure in the experiment 1 is used, but for the 4-month-period of price changes of the coin 'Litecoin (LTC).' Window length of the moving averaging filter is set to 150, so we'd end up with 120 RS-peaks after pre-processing, so that the average length of each linearized price movement is about a day. The feature vector size is set to 4, so 114 feature vectors are created. Intraprediction method is employed with 30, 60, 90 vectors in the set trainVectors. The experiment is run for both weighted and non-weighted options. Both the angles and the durations are predicted for every case. The error comparison for angles and durations is provided in the charts below:


Weighted intraprediction comparison (LTC-Daily)
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors
0.15
0.14
0.13
0.12
0.11
0.1
0.09
0.08
0.07
0.06
0
10
20
30
40
50
60
70
80
90


Non-Weighted intraprediction comparison (LTC-Daily)
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
80
90

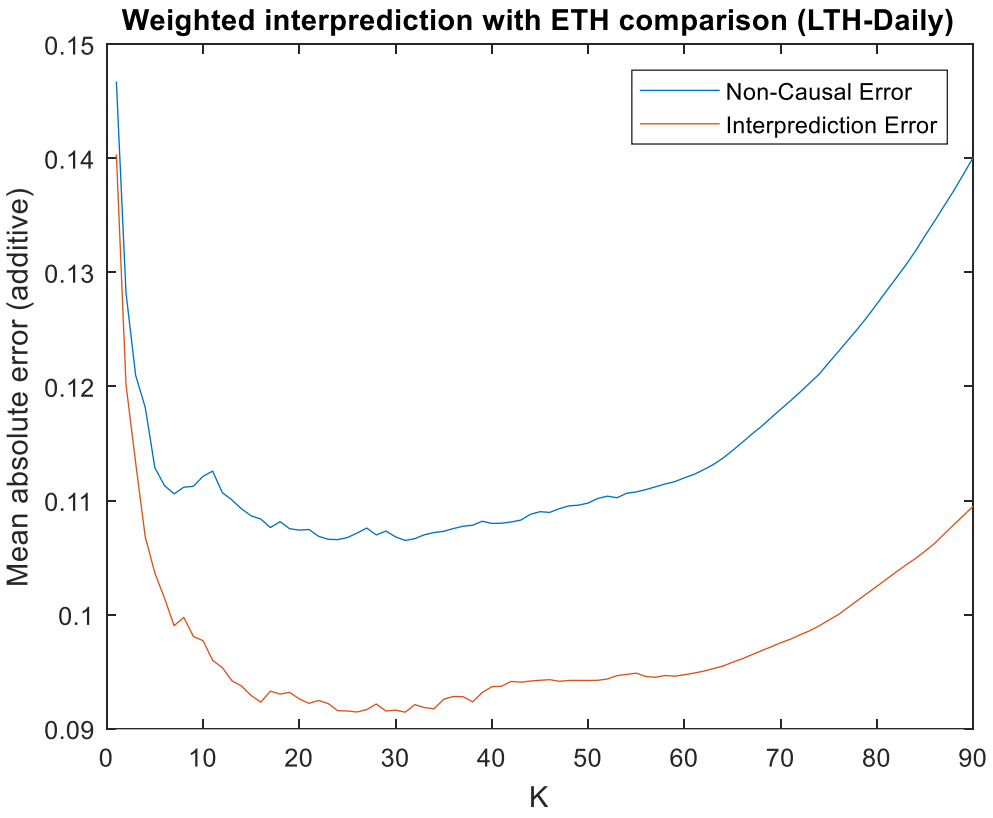

Weighted intraprediction time comparison (LTC-Daily)
×10^4
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors

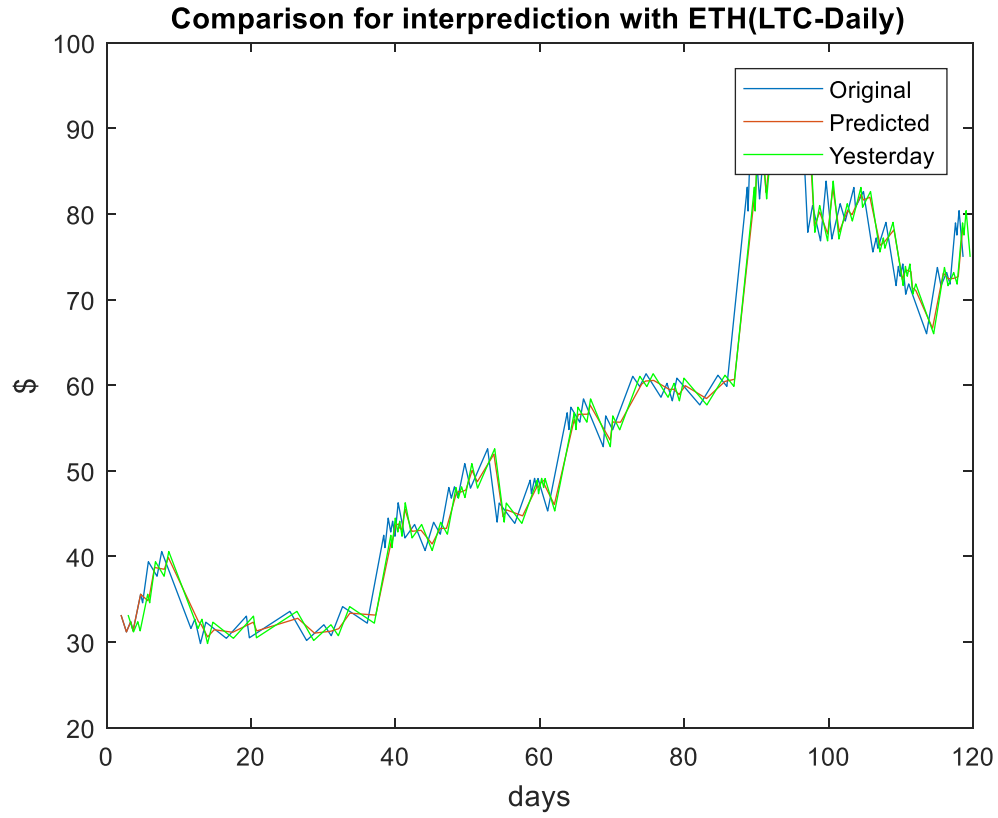

Non-Weighted intraprediction time comparison (LTC-Daily)
×10^4
Mean absolute error (additive)
K
Non-Causal Error
30 init vectors
60 init vectors
90 init vectors



d) Experiment 4

The same 4-month-period of price changes of both ETH and LTC are used in this experiment. Interprediction method is employed to predict the price changes of LTC by using the data of ETH. The averaging filter window length is set to 400 for ETH and 150 for LTC. The vector size is 4. The weighted option is used. Error comparison for the angles predicted is given below:


Weighted interprediction with ETH comparison (LTH-Daily)
Non-Causal Error
Interprediction Error
Mean absolute error (additive)
K
0.15
0.14
0.13
0.12
0.11
0.1
0.09
0
10
20
30
40
50
60
70
80
90


Comparison for interprediction with ETH(LTC-Daily)
Original
Predicted
Yesterday
$
days
100
90
80
70
60
50
40
30
20
0
20
40
60
80
100
120

5) DISCUSSION

In Experiment 1, Weighted prediction for angles returns lower mean errors than non-causal errors for 60 and 90 vectors in the training set for a good range of K: errors stay close to the minimum values for about half of K-values. This implies there is somewhat-strong causality within the used 4-month-period of ETH, if the trading window is about 1 day long. When there are 30 vectors in the training set, the returned errors are higher than non-causal errors. This can be the result of insufficient amount (30) of initial vectors in the training set. In the case of non-weighted option, the results don't vary much until K-value reaches around the half (~60)of the size of the vectors n total. After K passes that value, the error grows rapidly due to the fact that far-away neighbors have high contribution to the prediction. The fact that the lowest mean errors are acquired with 90 as the initial vector count can be attributed to the fact that the finally predicted value is much closer to the initial present time than the other options. This situation implies that the autocorrelation of ETH decreases as the distance in time grows, and the prediction results converge to those of basic prediction of setting yesterday's value to today.

In Experiment 2 with the vector size of 4, when the quarter (120) or the half (240) of vectors are used for the initial training set, the returned errors are higher than non-causal errors. However, when three quarters (360) are used for training, the returned errors are much lower than non-causal errors. Implication is that autocorrelation value drops faster than the previous experiment as distance in time grows; then the autocorrelation is more dependent on the indices of feature vectors, rather than time. Weighted and non-weighted cases show the same behavior as in experiment 1.

In experiment 2 with the vector size of 16, the mean non-causal error starts to grow monotonously just after K=2 for a case of angle prediction. Time prediction does not suffer such an issue; from which, we can conclude that angle prediction is more sensitive to the size of feature vectors than duration prediction.

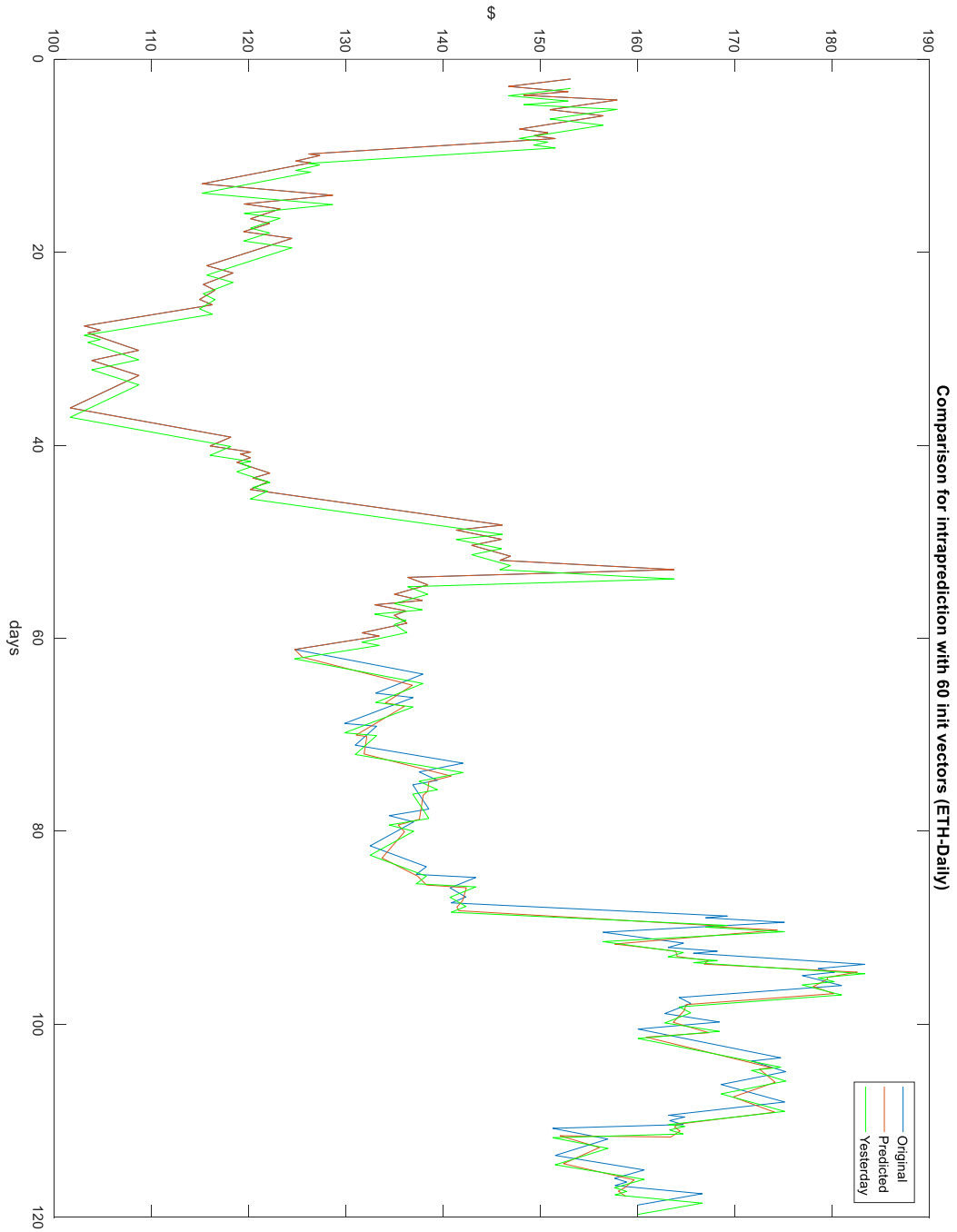
In experiment 3, the results are very similar to the results of experiment 1. However, the mean error for the case of low initial training vectors is lower than non-causal errors until K = 24. This implies that LTC has higher autocorrelation than ETH between its distant data points.

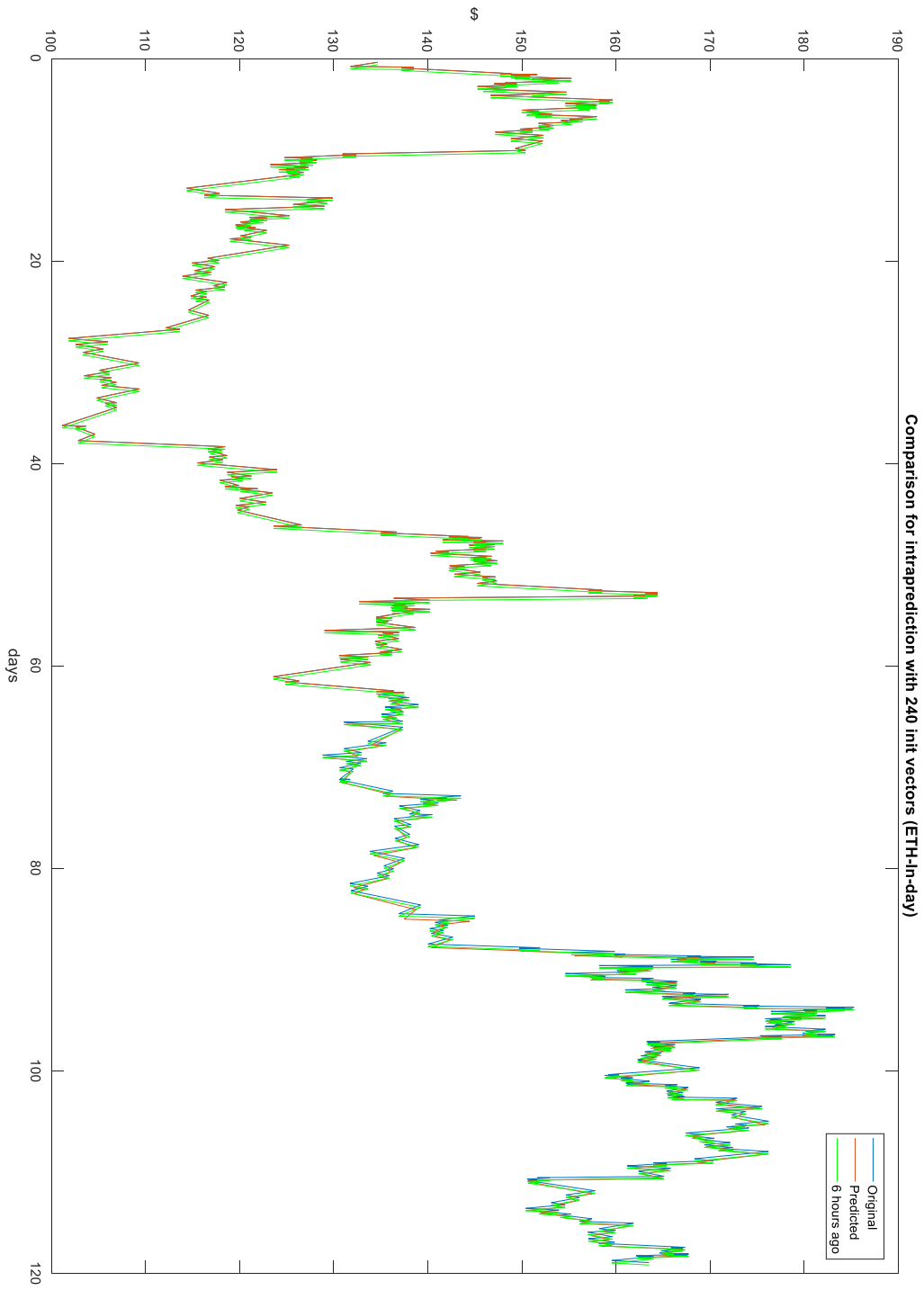
In experiment 4, behavior of the predicted errors shows similarities to the case of 30 initial vectors in experiment 3. This implies that LTC and ETH has somewhat high correlation.

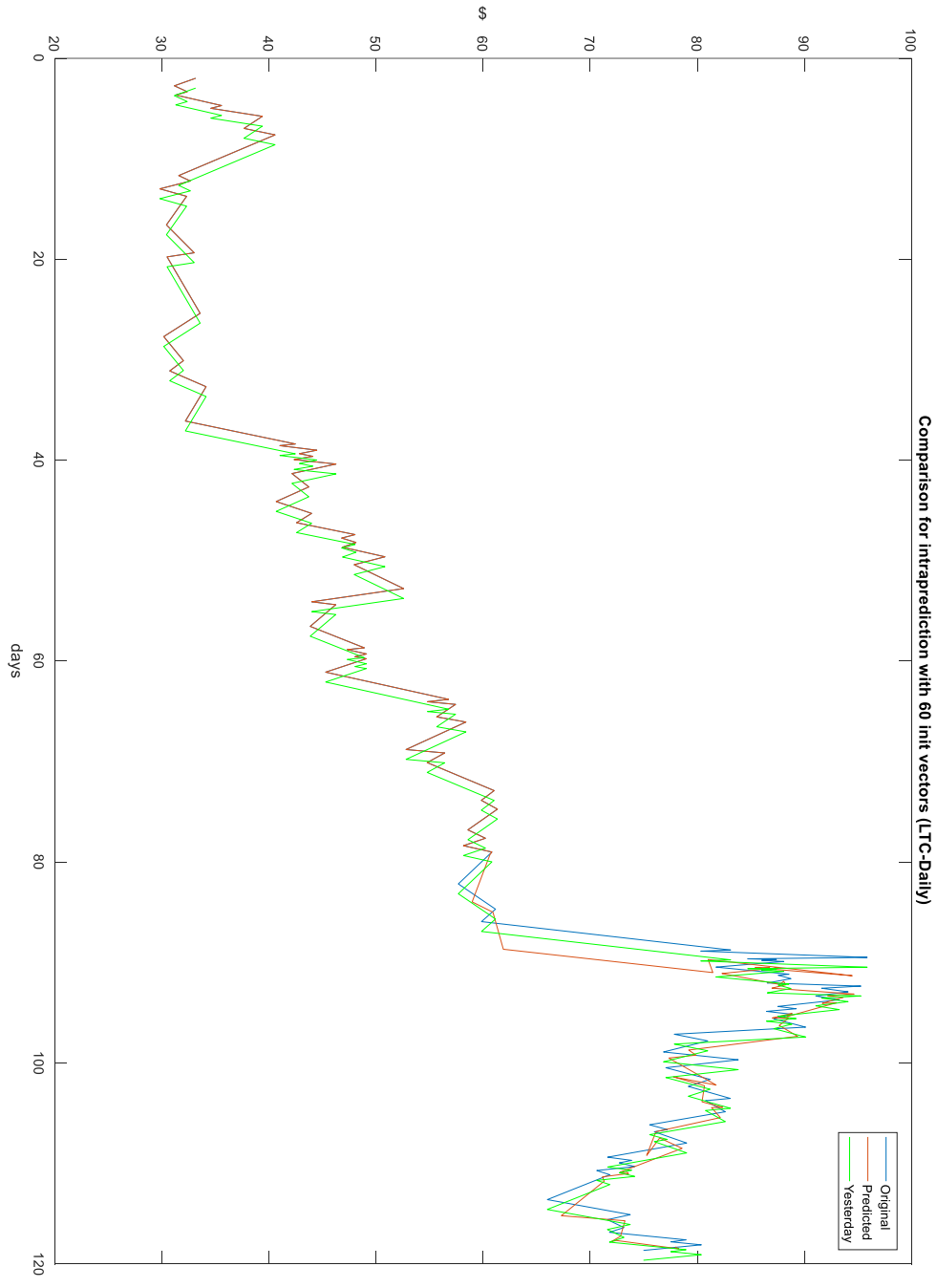
6) WORKS CITED

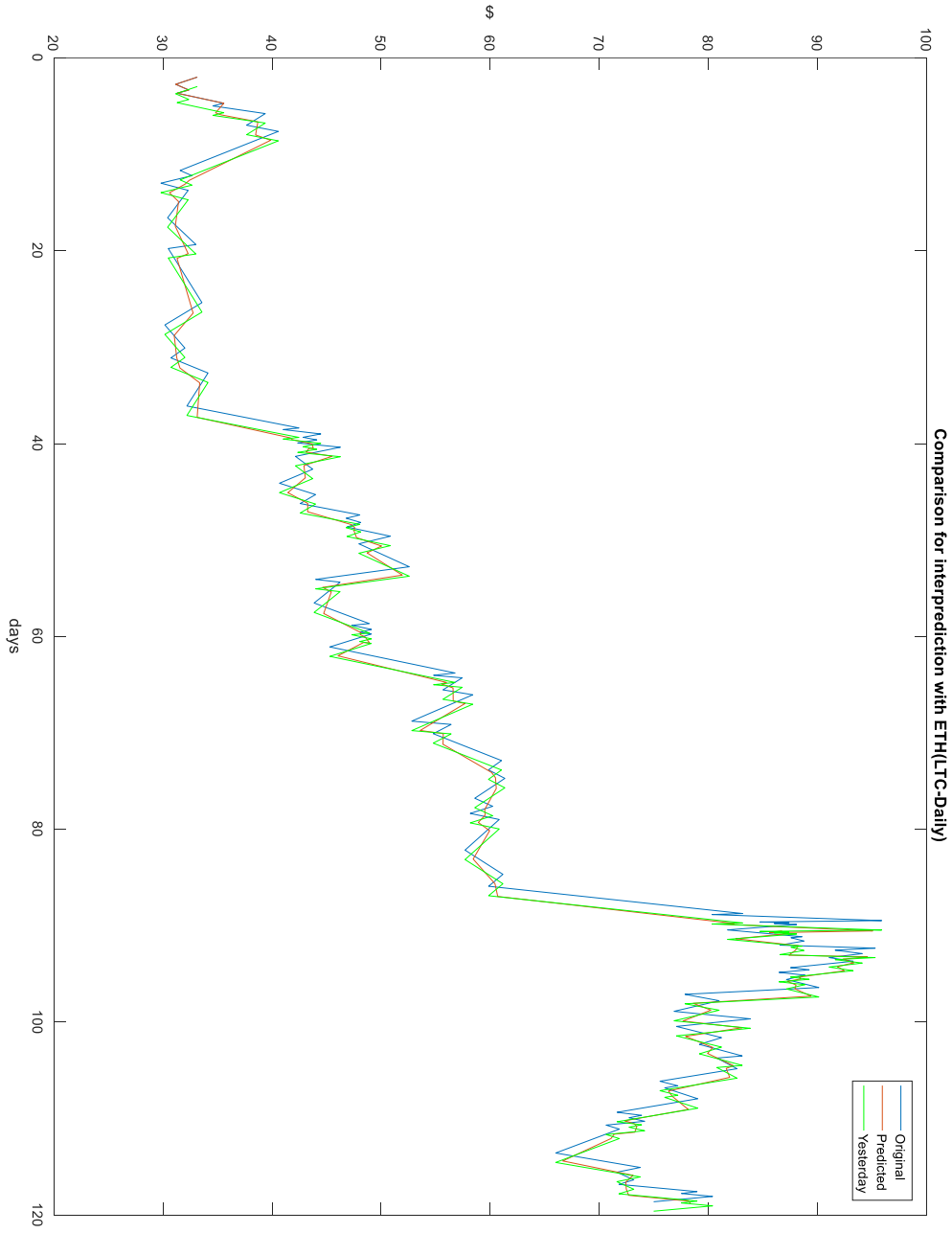
[1] "US & UK Exchange Data," 2019. [Online]. Available: https://www.cryptodatadownload.com/data/northamerican/. [Accessed 19 May 2019].

7) APPENDIX 1: RECONSTRUCTED PRICE CHARTS




Comparison for intraprediction with 240 init vectors (ETH-ln-day)
$
days
Original
Predicted
6 hours ago
100
110
120
130
140
150
160
170
180
190
0
20
40
60
80
100
120


Comparison for intraprediction with 60 init vectors (LTC-Daily)
$
days
Original
Predicted
Yesterday
20
30
40
50
60
70
80
90
100
0
20
40
60
80
100
120


Comparison for interprediction with ETH(LTC-Daily)
$
days
Original
Predicted
Yesterday
20
30
40
50
60
70
80
90
100
0
20
40
60
80
100
120

8) APPENDIX 2: MATLAB CODES

Function 'RSpeaks'

```
function [priceav, priceRS, priceN, tRS, tN, pricemin, pricemax,
timemin, timemax] = RSpeaks(price, t, Wn, peakProminence,
normalize)
priceN = [];
tN = [];

priceav = movmean(price, Wn); % Moving window averaged price
[~, locsR] = findpeaks(priceav, 'MinPeakProminence',
peakProminence); % Local maxima
[~, locsS] = findpeaks(-priceav, 'MinPeakProminence',
peakProminence); % Local minima

locsRS = sort([locsR; locsS]); % RS peak locations combined
tRS = t(locsRS); % Time points of RS peaks
priceRS = priceav(locsRS); % Price values of RS peaks
pricemin = min(priceRS);
pricemax = max(priceRS);
timemin = min(tRS);
timemax = max(tRS);

if normalize == 1 % Unity-based normalization
    priceN = (priceRS - min(priceRS))/(max(priceRS) -
min(priceRS));
    tN = (tRS - min(tRS))/(max(tRS) - min(tRS));
end

end
```

Function 'getStandardErrors

```
function [ standardErrors, predictions] = getStandardErrors(
featureVectors, labels, K, weighted )

predictions = zeros(length(labels), length(K));
standardErrors = zeros(1,length(K));

% Main algorithm
for k = 1:length(K)

    error = 0;
    for l = 1:length(labels)

        inputVector = featureVectors(l,:);
        inputLabel = labels(l);
        notInputVectors = featureVectors;
        notInputVectors(l,:) = [];
        notInputLabels = labels;
        notInputLabels(l) = [];

        distances = zeros(length(labels)-1,1);

        for m = 1:length(labels)-1
            distances(m) = norm(inputVector-
notInputVectors(m,:));
        end
        % Find nearest neighbors
        [~, sortedIndices] = sort(distances);
        nearests = distances(sortedIndices(1:K(k)));

        if weighted == 1 % Weighted summation
            Ki = 1./nearests;
            wi = Ki./sum(Ki);
            predictions(l,k) =
sum(wi.*notInputLabels(sortedIndices(1:K(k))));
        else % Non-weighted summation
            predictions(l,k) =
mean(notInputLabels(sortedIndices(1:K(k))));
        end
        error = error + abs((predictions(l,k)-inputLabel));
    end
    standardErrors(k) = error/length(labels);
end

end
```

Function 'Intraprediction'

```
function [ predictions, meanErrors ] = intraPrediction(
featureVectors, labels, K, trainSize, weighted)

K = K(K~=0);
predictions = zeros(length(labels)-trainSize, length(K));
meanErrors = zeros(1,length(K));

% Main algorithm
for k = 1:length(K)
    error = 0;
    for l = trainSize+1:length(labels)
        % Get sample
        inputVector = featureVectors(l,:);
        inputLabel = labels(l);

        % Find euclidian distances
        distances = zeros(l-1,1);
        for m = 1:l-1
            distances(m) = norm(inputVector-
featureVectors(m,:));
        end

        % Find nearest neighbors
        [~, sortedIndices] = sort(distances);
        nearests = distances(sortedIndices(1:K(k)));

        if weighted == 1 % Weighted summation
            Ki = 1./nearests;
            wi = Ki./sum(Ki);
            predictions(l-trainSize,k) =
sum(wi.*labels(sortedIndices(1:K(k))));
        else % Non-weighted summation
            predictions(l-trainSize,k) =
mean((labels(sortedIndices(1:K(k)))));
        end
        error = error + abs((predictions(l-trainSize,k)-
inputLabel));
        %trainVectors = [trainVectors; inputVector];
        %trainLabels = [trainLabels; inputLabel];
    end
    meanErrors(k) = error/(length(labels)-trainSize);
end
end
```

function interPrediction

```
function [ predictions, meanErrors ] = interPrediction(
trainVectors, testVectors, trainLabels, testLabels, K, weighted
)

K = K(K~=0);
predictions = zeros(length(testLabels), length(K));
meanErrors = zeros(1,length(K));
% Main algorithm
for k = 1:length(K)

    error = 0;
    for l = 1:length(testLabels)
        % Get sample
        inputVector = testVectors(l,:);
        inputLabel = testLabels(l);

        % Find euclidian distances
        distances = zeros(length(trainLabels),1);
        for m = 1:length(trainLabels)
            distances(m) = norm(inputVector-trainVectors(m,:));
        end

        % Find nearest neighbors
        [~, sortedIndices] = sort(distances);
        nearests = distances(sortedIndices(1:K(k)));

        if weighted == 1 % Weighted summation
            Ki = 1./nearests;
            wi = Ki./sum(Ki);
            predictions(l,k) =
sum(wi.*trainLabels(sortedIndices(1:K(k))));
        else % Non-weighted summation
            predictions(l,k) = mean(((sortedIndices(1:K(k)))));
        end
        error = error + abs((predictions(l,k)-
inputLabel)/inputLabel);
    end
    meanErrors(k) = error/(length(testLabels));
end
end
```

Script of Experiment 1

```
% Script0:


% Load price data
load 'edata.mat';

% Define parameters
eWn = 400; % 400 window length creates ~120 RS-peaks,
corresponding to daily trade
Mn = 4;
K = [1:1:30, zeros(1,60); 1:1:60, zeros(1,30); 1:1:90];
trainSize = [30, 60, 90];

% Process data
[epav, epRS, epN, etRS, etN, epmin, epmax, etmin, etmax] =
RSpeaks(eprice0, et0, eWn, 1, 1);

eslopes = diff(epN)./diff(etN);
eangles = atan(eslopes);
etimes = diff(etN);

% Create features
[eangleFeatureVectors, eangleLabels] = createFeatures(eangles,
Mn);
[etimeFeatureVectors, etimeLabels] = createFeatures(etimes, Mn);

% Get standard errors
[eangleStErrors, ~] = getStandardErrors( eangleFeatureVectors,
eangleLabels, K(3,:), 1);
[eangleStErrorsNw, ~] = getStandardErrors( eangleFeatureVectors,
eangleLabels, K(3,:), 0);

[etimeStErrors, ~] = getStandardErrors( etimeFeatureVectors,
etimeLabels, K(3,:), 1);
[etimeStErrorsNw, ~] = getStandardErrors( etimeFeatureVectors,
etimeLabels, K(3,:), 0);

% Predictions
[eanglePredictions1, eangleMeanErrors1] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(1,:),
trainSize(1), 1);
[etimePredictions1, etimeMeanErrors1] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(1,:),
trainSize(1), 1);
```

```
[eanglePredictions2, eangleMeanErrors2] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(2,:),
trainSize(2), 1);
[etimePredictions2, etimeMeanErrors2] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(2,:),
trainSize(2), 1);

[eanglePredictions3, eangleMeanErrors3] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(3,:),
trainSize(3), 1);
[etimePredictions3, etimeMeanErrors3] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(3,:),
trainSize(3), 1);

[eanglePredictions1Nw, eangleMeanErrors1Nw] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(1,:),
trainSize(1), 0);
[etimePredictions1Nw, etimeMeanErrors1Nw] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(1,:),
trainSize(1), 0);

[eanglePredictions2Nw, eangleMeanErrors2Nw] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(2,:),
trainSize(2), 0);
[etimePredictions2Nw, etimeMeanErrors2Nw] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(2,:),
trainSize(2), 0);

[eanglePredictions3Nw, eangleMeanErrors3Nw] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(3,:),
trainSize(3), 0);
[etimePredictions3Nw, etimeMeanErrors3Nw] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(3,:),
trainSize(3), 0);

figure; plot(K(3,:),eangleStErrors);
hold on; plot(K(1,1:30),eangleMeanErrors1(1:30));
hold on; plot(K(2,1:60),eangleMeanErrors2(1:60));
hold on; plot(K(3,:),eangleMeanErrors3);
title('Weighted intraprediction angle comparison (ETH-Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

figure; plot(K(3,:),eangleStErrorsNw);
hold on; plot(K(1,1:30),eangleMeanErrors1Nw(1:30));
```

```
hold on; plot(K(2,1:60),eangleMeanErrors2Nw(1:60));
hold on; plot(K(3,:),eangleMeanErrors3Nw);
title('Non-Weighted intraprediction angle comparison (ETH-
Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

figure; plot(K(3,:),etimeStErrors);
hold on; plot(K(1,1:30),etimeMeanErrors1(1:30));
hold on; plot(K(2,1:60),etimeMeanErrors2(1:60));
hold on; plot(K(3,:),etimeMeanErrors3);
title('Weighted intraprediction time comparison (ETH-Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

figure; plot(K(3,:),etimeStErrorsNw);
hold on; plot(K(1,1:30),etimeMeanErrors1Nw(1:30));
hold on; plot(K(2,1:60),etimeMeanErrors2Nw(1:60));
hold on; plot(K(3,:),etimeMeanErrors3Nw);
title('Non-Weighted intraprediction time comparison (ETH-
Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

% Recreate chart for weighted prediction with 60 initial vectors
index = find(eangleMeanErrors2 == min(eangleMeanErrors2),1);

timeRecreated = zeros(size(etN));
priceRecreated = zeros(size(epN));

timeRecreated(1:trainSize(2)+Mn+1) = etN(1:trainSize(2)+Mn+1);
priceRecreated(1:trainSize(2)+Mn+1) = epN(1:trainSize(2)+Mn+1);

for k=1:length(eanglePredictions2(:,index))
    timeRecreated(k+trainSize(2)+Mn+1) = etN(k+trainSize(2)+Mn)
+ etimePredictions2(k,index);
    priceRecreated(k+trainSize(2)+Mn+1) = epN(k+trainSize(2)+Mn)
+ etimePredictions2(k,index)*eanglePredictions2(k,index);
end

tave = mean(diff(etRS));
```

```
denormalizedTime = timeRecreated*(etmax-etmin)+etmin;
denormalizedPrice = priceRecreated*(epmax-epmin)+epmin;
figure;
plot(etRS/(24*60),epRS);
hold on; plot(denormalizedTime(1:end-1)/(24*60),
denormalizedPrice(1:end-1));
hold on; plot((etRS+tave)/(24*60),epRS,'g');
title('Comparison for intraprediction with 60 init vectors (ETH-
Daily)');
xlabel('days');
ylabel('$');
legend('Original','Predicted','Yesterday');
```

Script of Experiment 2

```
% Script1:


% Load price data
load 'edata.mat';

% Define parameters
eWn = 40; % 400 window length creates ~120 RS-peaks,
corresponding to daily trade
Mn = 4;
K = [1:1:60, zeros(1,120); 1:1:120, zeros(1,60); 1:1:180];
trainSize = [120, 240, 360];

% Process data
[epav, epRS, epN, etRS, etN, epmin, epmax, etmin, etmax] =
RSpeaks(eprice0, et0, eWn, 1, 1);

eslopes = diff(epN)./diff(etN);
eangles = atan(eslopes);
etimes = diff(etN);

% Create features
[eangleFeatureVectors, eangleLabels] = createFeatures(eangles,
Mn);
[etimeFeatureVectors, etimeLabels] = createFeatures(etimes, Mn);
0
% Get standard errors
[eangleStErrors, ~] = getStandardErrors( eangleFeatureVectors,
eangleLabels, K(3,:), 1);
0.5
[eangleStErrorsNw, ~] = getStandardErrors( eangleFeatureVectors,
eangleLabels, K(3,:), 0);
1
[etimeStErrors, ~] = getStandardErrors( etimeFeatureVectors,
etimeLabels, K(3,:), 1);
1.5
[etimeStErrorsNw, ~] = getStandardErrors( etimeFeatureVectors,
etimeLabels, K(3,:), 0);
2
% Predictions
[eanglePredictions1, eangleMeanErrors1] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(1,:),
trainSize(1), 1);
2.5
```

```
[etimePredictions1, etimeMeanErrors1] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(1,:),
trainSize(1), 1);
3
[eanglePredictions2, eangleMeanErrors2] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(2,:),
trainSize(2), 1);
3.5
[etimePredictions2, etimeMeanErrors2] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(2,:),
trainSize(2), 1);
4
[eanglePredictions3, eangleMeanErrors3] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(3,:),
trainSize(3), 1);
4.5
[etimePredictions3, etimeMeanErrors3] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(3,:),
trainSize(3), 1);
5
[eanglePredictions1Nw, eangleMeanErrors1Nw] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(1,:),
trainSize(1), 0);
5.5
[etimePredictions1Nw, etimeMeanErrors1Nw] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(1,:),
trainSize(1), 0);
6
[eanglePredictions2Nw, eangleMeanErrors2Nw] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(2,:),
trainSize(2), 0);
6.5
[etimePredictions2Nw, etimeMeanErrors2Nw] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(2,:),
trainSize(2), 0);
7
[eanglePredictions3Nw, eangleMeanErrors3Nw] =
intraPrediction(eangleFeatureVectors, eangleLabels, K(3,:),
trainSize(3), 0);
7.5
[etimePredictions3Nw, etimeMeanErrors3Nw] =
intraPrediction(etimeFeatureVectors, etimeLabels, K(3,:),
trainSize(3), 0);
%%
figure; plot(K(3,:),eangleStErrors);
hold on; plot(K(1,1:60),eangleMeanErrors1(1:60));
hold on; plot(K(2,1:120),eangleMeanErrors2(1:120));
```

```
hold on; plot(K(3,:),eangleMeanErrors3);
title('Weighted intraprediction angle comparison (ETH-In-day)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '120 init vectors', '240 init
vectors', '360 init vectors');

figure; plot(K(3,:),eangleStErrorsNw);
hold on; plot(K(1,1:60),eangleMeanErrors1Nw(1:60));
hold on; plot(K(2,1:120),eangleMeanErrors2Nw(1:120));
hold on; plot(K(3,:),eangleMeanErrors3Nw);
title('Non-Weighted intraprediction angle comparison (ETH-In-
day)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '120 init vectors', '240 init
vectors', '360 init vectors');

figure; plot(K(3,:),etimeStErrors);
hold on; plot(K(1,1:60),etimeMeanErrors1(1:60));
hold on; plot(K(2,1:120),etimeMeanErrors2(1:120));
hold on; plot(K(3,:),etimeMeanErrors3);
title('Weighted intraprediction time comparison (ETH-In-day)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '120 init vectors', '240 init
vectors', '360 init vectors');

figure; plot(K(3,:),etimeStErrorsNw);
hold on; plot(K(1,1:60),etimeMeanErrors1Nw(1:60));
hold on; plot(K(2,1:120),etimeMeanErrors2Nw(1:120));
hold on; plot(K(3,:),etimeMeanErrors3Nw);
title('Non-Weighted intraprediction time comparison (ETH-In-
day)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '120 init vectors', '240 init
vectors', '360 init vectors');


% Recreate chart for weighted prediction with 60 initial vectors
index = find(eangleMeanErrors2 == min(eangleMeanErrors2),1);

timeRecreated = zeros(size(etN));
priceRecreated = zeros(size(epN));

timeRecreated(1:trainSize(2)+Mn+1) = etN(1:trainSize(2)+Mn+1);
```

```
priceRecreated(1:trainSize(2)+Mn+1) = epN(1:trainSize(2)+Mn+1);

for k=1:length(eanglePredictions2(:,index))
    timeRecreated(k+trainSize(2)+Mn+1) = etN(k+trainSize(2)+Mn)
+ etimePredictions2(k,index);
    priceRecreated(k+trainSize(2)+Mn+1) = epN(k+trainSize(2)+Mn)
+ etimePredictions2(k,index)*eanglePredictions2(k,index);
end

tave = mean(diff(etRS));
denormalizedTime = timeRecreated*(etmax-etmin)+etmin;
denormalizedPrice = priceRecreated*(epmax-epmin)+epmin;
figure; plot(etRS/(24*60),epRS);
hold on; plot(denormalizedTime(1:end-1)/(24*60),
denormalizedPrice(1:end-1));
hold on; plot((etRS+tave)/(24*60),epRS,'g');
title('Comparison for intraprediction with 240 init vectors
(ETH-In-day)');
xlabel('days');
ylabel('$');
legend('Original','Predicted','6 hours ago');
```

Script of Experiment 3

```
% Script2:


% Load price data
load 'ldata.mat';

% Define parameters
lWn = 150; % 400 window length creates ~120 RS-peaks,
corresponding to daily trade
Mn = 4;
K = [1:1:30, zeros(1,60); 1:1:60, zeros(1,30); 1:1:90];
trainSize = [30, 60, 90];

% Process data
[lpav, lpRS, lpN, ltRS, ltN, lpmin, lpmax, ltmin, ltmax] =
RSpeaks(lprice0, lt0, lWn, 1, 1);

lslopes = diff(lpN)./diff(ltN);
langles = atan(lslopes);
ltimes = diff(ltN);

% Create features
[langleFeatureVectors, langleLabels] = createFeatures(langles,
Mn);
[ltimeFeatureVectors, ltimeLabels] = createFeatures(ltimes, Mn);

% Get standard errors
[langleStErrors, ~] = getStandardErrors( langleFeatureVectors,
langleLabels, K(3,:), 1);
[langleStErrorsNw, ~] = getStandardErrors( langleFeatureVectors,
langleLabels, K(3,:), 0);

[ltimeStErrors, ~] = getStandardErrors( ltimeFeatureVectors,
ltimeLabels, K(3,:), 1);
[ltimeStErrorsNw, ~] = getStandardErrors( ltimeFeatureVectors,
ltimeLabels, K(3,:), 0);

% Predictions
[langlePredictions1, langleMeanErrors1] =
intraPrediction(langleFeatureVectors, langleLabels, K(1,:),
trainSize(1), 1);
[ltimePredictions1, ltimeMeanErrors1] =
intraPrediction(ltimeFeatureVectors, ltimeLabels, K(1,:),
trainSize(1), 1);
```

```
[langlePredictions2, langleMeanErrors2] =
intraPrediction(langleFeatureVectors, langleLabels, K(2,:),
trainSize(2), 1);
[ltimePredictions2, ltimeMeanErrors2] =
intraPrediction(ltimeFeatureVectors, ltimeLabels, K(2,:),
trainSize(2), 1);

[langlePredictions3, langleMeanErrors3] =
intraPrediction(langleFeatureVectors, langleLabels, K(3,:),
trainSize(3), 1);
[ltimePredictions3, ltimeMeanErrors3] =
intraPrediction(ltimeFeatureVectors, ltimeLabels, K(3,:),
trainSize(3), 1);

[langlePredictions1Nw, langleMeanErrors1Nw] =
intraPrediction(langleFeatureVectors, langleLabels, K(1,:),
trainSize(1), 0);
[ltimePredictions1Nw, ltimeMeanErrors1Nw] =
intraPrediction(ltimeFeatureVectors, ltimeLabels, K(1,:),
trainSize(1), 0);

[langlePredictions2Nw, langleMeanErrors2Nw] =
intraPrediction(langleFeatureVectors, langleLabels, K(2,:),
trainSize(2), 0);
[ltimePredictions2Nw, ltimeMeanErrors2Nw] =
intraPrediction(ltimeFeatureVectors, ltimeLabels, K(2,:),
trainSize(2), 0);

[langlePredictions3Nw, langleMeanErrors3Nw] =
intraPrediction(langleFeatureVectors, langleLabels, K(3,:),
trainSize(3), 0);
[ltimePredictions3Nw, ltimeMeanErrors3Nw] =
intraPrediction(ltimeFeatureVectors, ltimeLabels, K(3,:),
trainSize(3), 0);

figure; plot(K(3,:),langleStErrors);
hold on; plot(K(1,1:30),langleMeanErrors1(1:30));
hold on; plot(K(2,1:60),langleMeanErrors2(1:60));
hold on; plot(K(3,:),langleMeanErrors3);
title('Weighted intraprediction angle comparison (LTC-Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

figure; plot(K(3,:),langleStErrorsNw);
hold on; plot(K(1,1:30),langleMeanErrors1Nw(1:30));
```

```
hold on; plot(K(2,1:60),langleMeanErrors2Nw(1:60));
hold on; plot(K(3,:),langleMeanErrors3Nw);
title('Non-Weighted angle intraprediction comparison (LTC-
Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

figure; plot(K(3,:),ltimeStErrors);
hold on; plot(K(1,1:30),ltimeMeanErrors1(1:30));
hold on; plot(K(2,1:60),ltimeMeanErrors2(1:60));
hold on; plot(K(3,:),ltimeMeanErrors3);
title('Weighted intraprediction time comparison (LTC-Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

figure; plot(K(3,:),ltimeStErrorsNw);
hold on; plot(K(1,1:30),ltimeMeanErrors1Nw(1:30));
hold on; plot(K(2,1:60),ltimeMeanErrors2Nw(1:60));
hold on; plot(K(3,:),ltimeMeanErrors3Nw);
title('Non-Weighted intraprediction time comparison (LTC-
Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', '30 init vectors', '60 init vectors',
'90 init vectors');

% Recreate chart for weighted prediction with 60 initial vectors
index = find(langleMeanErrors2 == min(langleMeanErrors2),1);

timeRecreated = zeros(size(ltN));
priceRecreated = zeros(size(lpN));

timeRecreated(1:trainSize(2)+Mn+1) = ltN(1:trainSize(2)+Mn+1);
priceRecreated(1:trainSize(2)+Mn+1) = lpN(1:trainSize(2)+Mn+1);

for k=1:length(langlePredictions2(:,index))
    timeRecreated(k+trainSize(2)+Mn+1) = ltN(k+trainSize(2)+Mn)
+ ltimePredictions2(k,index);
    priceRecreated(k+trainSize(2)+Mn+1) = lpN(k+trainSize(2)+Mn)
+ ltimePredictions2(k,index)*langlePredictions2(k,index);
end

tave = mean(diff(ltRS));
```

```
denormalizedTime = timeRecreated*(ltmax-ltmin)+ltmin;
denormalizedPrice = priceRecreated*(lpmax-lpmin)+lpmin;
figure; plot(ltRS/(24*60),lpRS);
hold on; plot(denormalizedTime(1:end-1)/(24*60),
denormalizedPrice(1:end-1));
hold on; plot((ltRS+tave)/(24*60),lpRS,'g');
title('Comparison for intraprediction with 60 init vectors (LTC-
Daily)');
xlabel('days');
ylabel('$');
legend('Original','Predicted','Yesterday');
```

Script of Experiment 4

```
% Load price data
load 'edata.mat';
load 'ldata.mat'

% Define parameters
eWn = 400;
lWn = 150;
Mn = 4;
K = [1:1:90];
%trainSize = [30, 60, 90];

% Process data
[lpav, lpRS, lpN, ltRS, ltN, lpmin, lpmax, ltmin, ltmax] =
RSpeaks(lprice0, lt0, lWn, 1, 1);
[epav, epRS, epN, etRS, etN, epmin, epmax, etmin, etmax] =
RSpeaks(eprice0, et0, eWn, 1, 1);

eslopes = diff(epN)./diff(etN);
eangles = atan(eslopes);
etimes = diff(etN);

lslopes = diff(lpN)./diff(ltN);
langles = atan(lslopes);
ltimes = diff(ltN);

% Create features
[eangleFeatureVectors, eangleLabels] = createFeatures(eangles,
Mn);
[etimeFeatureVectors, etimeLabels] = createFeatures(etimes, Mn);

[langleFeatureVectors, langleLabels] = createFeatures(langles,
Mn);
[ltimeFeatureVectors, ltimeLabels] = createFeatures(ltimes, Mn);

% Get standard errors
[langleStErrors, lol] = getStandardErrors( langleFeatureVectors,
langleLabels, K, 1);
[langleStErrorsNw, ~] = getStandardErrors( langleFeatureVectors,
langleLabels, K, 0);

[langlePredictions, langleMeanErrors] =
interPrediction(eangleFeatureVectors, langleFeatureVectors,
eangleLabels, langleLabels, K, 1);
```

```
[ltimePredictions, ltimeMeanErrors] =
interPrediction(etimeFeatureVectors, ltimeFeatureVectors,
etimeLabels, ltimeLabels, K, 1);

[langlePredictionsNw, langleMeanErrorsNw] =
interPrediction(eangleFeatureVectors, langleFeatureVectors,
eangleLabels, langleLabels, K, 0);
[ltimePredictionsNw, ltimeMeanErrorsNw] =
interPrediction(etimeFeatureVectors, ltimeFeatureVectors,
etimeLabels, ltimeLabels, K, 0);

figure; plot(K,langleStErrors);
%hold on; plot(K,langleStErrorsNw);
hold on; plot(K,langleMeanErrors);
%hold on; plot(K,langleMeanErrorsNw);
title('Weighted interprediction with ETH comparison (LTH-
Daily)');
xlabel('K');
ylabel('Mean absolute error (additive)');
legend('Non-Causal Error', 'Interprediction Error');

% Recreate chart
index = find(langleMeanErrors == min(langleMeanErrors),1);

timeRecreated = zeros(size(ltN));
priceRecreated = zeros(size(lpN));

timeRecreated(1:Mn+1) = ltN(1:Mn+1);
priceRecreated(1:Mn+1) = lpN(1:Mn+1);

for k=1:length(langlePredictions(:,index))
    timeRecreated(k+Mn+1) = ltN(k+Mn) +
ltimePredictions(k,index);
    priceRecreated(k+Mn+1) = lpN(k+Mn) +
ltimePredictions(k,index)*langlePredictions(k,index);
end

%figure; plot(ltN,lpN); hold on; plot(timeRecreated,
priceRecreated);
tave = mean(diff(ltRS));
denormalizedTime = timeRecreated*(ltmax-ltmin)+ltmin;
denormalizedPrice = priceRecreated*(lpmax-lpmin)+lpmin;
figure; plot(ltRS/(24*60),lpRS);
hold on; plot(denormalizedTime(1:end-1)/(24*60),
denormalizedPrice(1:end-1));
hold on; plot((ltRS+tave)/(24*60),lpRS,'g');
title('Comparison for interprediction with ETH(LTC-Daily)');
```

```
xlabel('days');
ylabel('$');
legend('Original','Predicted','Yesterday');
```