Communications in Applied Analysis **17** (2013), no. N, xxx–xxx

THE DISTANCE CHOICE AND OPTIMAL PARAMETER SELECTION IN k-NN ALGORITHM FOR FX DATA

VINDYA I. KUMARI PATHIRANA[1] AND K. M. RAMACHANDRAN[2]

[1,2]Department of Mathematics and Statistics,
University of South Florida
Tampa, FL 33620-5700
[1] *E-mail:* vkumari@mail.usf.edu
[2]*E-mail:* ram@usf.edu

ABSTRACT. Foreign Exchange rate forecasting is a challenging area of study over the past. Various linear and non-linear methods have been used to forecast foreign exchange rates. As the FX data are nonlinear and highly correlated, forecasting through non-linear dynamical systems is becoming more and more relevant. Nearest Neighbor Algorithms is one of the most commonly used non-linear pattern recognition methods that outperform the available linear forecasting methods for the high frequency foreign exchange data. As the distance plays a key role in the k-NN algorithm, by choosing an appropriate distance we can improve the performance of the algorithm significantly. The most commonly used distance for k-NN forecasting in the past was Euclidean distance. Due to possible correlation among vectors at different time frames, distances based on deterministic vectors such as Euclidean, are not very effective when applying for financial data. Since Mahalanobis distance captures the correlations, we suggest to use this distance in the selection of neighbors. In this work, we used five different FX currencies to compare the performances of the algorithm with traditional Euclidean and Absolute distances with the proposed Mahalanobis distance. The performances were compared in two ways: (i) forecast accuracy and (ii) transforming their forecasts in to a more effective technical trading rule. The results are obtained with real FX trading data, and the results show that method introduced in this paper outperforms the other popular methods.
Furthermore, we have conducted a thorough investigation of optimal parameter choice with different distance measures. We adopt the concept of distance based weighting to the NN and compared the performances with traditional unweight NN algorithm based forecasting.

AMS (MOS) Subject Classification. 99Z00. 39A10, 39A99.

Received xxxx xx, 2013

1. **INTRODUCTION**

The foreign exchange (FX) market is a global market for currency trading. A preliminary global study by the Bank for International Settlements from the 2013 Triennial Central Bank Survey of Foreign Exchange and OTC Derivatives Markets Activity show that trading in foreign exchange markets averaged 5.3 trillion dollars per day in April 2013 [15].Thus, foreign exchange rates forecasting is one of the challenging and important applications of financial time series prediction. Being one of noisiest financial data, foreign exchange rates make it even more difficult for the data analyst to forecast. Also, the currency rates are nonlinear and highly correlated [6, 13]. Due to this nature of the data, forecasting through non-linear dynamical systems is becoming more and more relevant. Neighbor Algorithms is one of the most popular such non-linear pattern recognition algorithm, which dates back to an unpublished report by Fix and Hodges in 1951, [3]. Like any other technical analysis method, nearest neighbor prediction model is also completely rely on the historical data. When applying for foreign exchange rate forecasting, its main goal is to investigate the past behavior of the currency rates so that it can fully capture the dependency of the future exchange rates and that of the past. We look for the repetitions of specific price patterns such as major trends, critical or turning points.

Nearest Neighbor algorithms are examples of instant-based learning. The idea of Nearest Neighbor (or k -Nearest Neighbor) algorithm is to select fixed number of observations which are closest to the desired point (value). The term 'nearest' is determined by a distance metric. Even though the nearest neighbor (NN) algorithm outperforms available linear forecasting methods, it also has issues which need to be addressed. Choosing an appropriate distance metric, deciding the number of nearest neighbors and embedding dimension have become the major challenges when applying k-NN algorithm.

In section 2, we will give some background material on distance measures, and error measures. Section 3 will deal with selection of Embedding Dimension (m) and Number of Nearest Neighbors (k). Comparison of forecast accuracy and trading decisions based on Mahalanobis and Euclidean distance based methods will be done in section 4. Conclusion will be given in section 5.

2. **BACKGROUND**

2.1. k-**Nearest Neighbor Algorithm and the choice of Distance.**

k-Nearest neighbor (k-NN) algorithm rank the data and chose the k closest of them based on the distance between the query vector and the historical values.
Consider the finite time series $\{x_t\}_{t=1}^{n} = \{x_1, x_2, ..., x_n\}$. First, we divide the time series data in to two separate parts; for $T < n$, a training (in-sample) set $\{x_1, x_2, ..., x_T\}$ and a testing (or out-of-sample) set $\{x_{T+1}, x_{T+2}, ..., x_n\}$. In order to identify behavioral patterns

in the data, we transform the scalar time series in to time series vectors. We need to choose an embedding dimension and delay time to create vectors out of the training set. Embedding dimension is the number of time series we consider for a history vector and delay time is the time gap between two consecutive data values in a vector.
Let m and $\tau \in \mathbb{N}$ (set of positive integers) be the dimension and delay parameter respectively. Then a time series vector at time t can be written as;

$$x_t^{m,\tau} = (x_t, x_{t-\tau}, ..., x_{t-(m-1)\tau}) \quad \text{for} \quad 1 + (m-1)\tau \leq t \leq T \tag{2.1}$$

These m-dimensional vectors are often called as $m - histories$ and the m-dimensional space $\mathbb{R}^m$ is referred to be the phase space of the time series [6, 7].
Our primary goal is to use the most relevant vectors out of the training set in the forecasting algorithm to predict the exchange rate at time $t = T + 1$, which is a one-step-ahead forecasting. The most relevant vectors are the ones having similar dynamic behavior as the delay vector x_T^m. We compare the distance between the delay vector and all the other m-history vectors to choose the vectors with similar dynamic behavior [6, 7]. Then we look for the closest k vectors in the phase space $\mathbb{R}^m$ such that they minimize the distance function $d(x_T^m, x_i)$.

In k-NN algorithm, m and k are pre-determined constants. In the literature, the optimal values of m and k are quite ambiguous. There have been quite a lot argument and discussions about the optimal choice of m and k since the NN rule was first officially introduced by Cover and Hart in 1967 [3, 14]. In section 3 We will discuss the choice of m and k for Mahalanobis distance along with other distance choices.
For the forecasting we can incorporate variety of Statistical and time series predicting methods with NN algorithm for the forecasting. In the literature of k-NN forecasting, the most commonly used forecasting method is locally weighted simple linear regression [1, 6, 7]. So, to compare the performance of the chosen distance functions, we use the following locally adjusted linear regression model [1, 7]:

$$\hat{x}_{T+1} = \sum_{n=0}^{m-1} \hat{a}_n x_{T-m\tau} + \hat{a}_m \tag{2.2}$$

The coefficients were fitted by the linear regression of $x_{t_j+1}^m$on $x_{t_j}^m = (x_{t_j}, x_{t_j-\tau}, ..., x_{t_j-(m-1)\tau}$ for $j = 1, 2, ..., k$. Thus the estimated coefficients $\hat{a}_i$ are the values of a_i that minimize

$$\sum_{j=1}^{k} (x_{t_j+1} - a_0 x_{t_j} - a_1 x_{t_j-1} - ... - a_{m-1} x_{t_j-(m-1)\tau} - a_m)^2 \tag{2.3}$$

The data used in equation (2.3) are the only $k(m + 1)$ data values obtained from the k-neighbor vectors of size m and the corresponding next values, $x_{t_j+1}^m$ for $j = 1, 2, ..., k$ chosen neighboring vectors, not the entire data.

As the forecasting is completely depending on the selected k nearest neighbors, it is highly important to use a distance function which captures the behavior of the data accurately. Many researchers have pointed out the difficulty of choosing a distance measure for the NN algorithm that works well for different types of data. Over the past decades, the most common choice of distance was Euclidean distance [4, 17]. The way it?s defined, the Euclidean distance is unable to capture the trend of the highly volatile (hence random) and highly correlated foreign exchange data when choosing the neighbors for the NN algorithm. Apart from Euclidean distance, several other distance measures such as Manhattan, Minkowski, and Hamming distances have been used in the algorithm for various types of classification problems [8, 14].
Even though the asymptotic probability of error of the NN is independent of the choice of metric, classification performance of finite sample nearest neighbor algorithm is not independent of the distance function [8, 11]. As Nearest neighbor rule is highly sensitive to outliers, selecting irrelevant neighbors can cause increase in forecasting error.
In their work, Fukunaga & Hostetler showed that using a proper distance measure, the variance of the finite sample estimate can be minimized [8]. Short & Fukunaga investigate the relation between the distance function in k-NN and the error measure [14]. They conclude that the error can be minimized by using an appropriate distance metric without increasing the number of sample vectors.

2.2. **Distance.**

In general, the distance between two objects describes how far apart the objects are. Distance is a rule of assigning positive numbers between pair of objects (or points). As it is a concrete way of describing an element of some space is closer to or far away from another, distance concept has been widely used in the field of time series data clustering [2, 4]. In time series pattern recognition, an appropriate distance function can categorize data in to clusters by capturing the similarity or dissimilarity between the data.
In Mathematics, a distance function is usually called as a metric. It is a generalization of the concept of physical distance. Let X be an arbitrary set.
A function $d : \Omega \times \Omega \to \mathbb{R}$ is a metric (or a distance function) on if the following conditions are satisfied for all $x, y, z \in \Omega$.

(i) Non-negativity: $d(x, y) \geq 0$
(ii) Coincide axiom: $d(x, y) = 0$ if and only if $x = y$
(iii) Symmetry: $d(x, y) = d(y, x)$
(iv) Triangular inequality: $d(x, z) \leq d(x, y) + d(y, z)$

The pair $< \Omega, d >$ is called a metric space. In general, elements of the set Ω are called points of the metric space and $d(x, y)$ referred as the distance between points x, y.

Now, we will briefly discuss some distance measures commonly used in NN algorithm. Consider n-dimensional vectors $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ in $\mathbb{R}^m$.

The $Euclidean distance$ between x and y is define as

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.4}$$

Euclidean distance is a function which calculates the real straight line distance between two points. It is the most common distance of choice in NN algorithms. Even though it works well for low dimensional data, it performs poorly when the data are high dimensional. Also, Euclidean is not the best distance choice when the data are highly correlated as it does not account the correlation among the vectors.

$Manhattan \quad distance$ gets its name from the rectangular grid patterns of the streets in Manhattan [18]. The Manhattan distance between x and y in $\mathbb{R}^m$ is defined as

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.5}$$

As it looks at the absolute difference between the coordinates, the most common and appropriate name for this distance measure is absolute value. It is also recognized as a computationally simplified version of Euclidean distance. Manhattan distance is preferred to Euclidean distance in practice sometime, because the distance along each axis is not squared, a large difference in one of the dimensions will not affect the total outcome.

$Mahalanobis \quad distance$ was introduced by P. C. Mahalanobis in 1936 by considering the possible correlation among the data [9]. It is defined between two vectors x and y as:

$$d(x, y) = \sqrt{(x - y)' \sum^{-1} (x - y)} \tag{2.6}$$

Here, $\sum^{-1}$ is the inverse of variance-covariance matrix $\sum$ between x and y and $'$ denotes the matrix transpose. The major difference in Mahalanobis to any other distance measure is that it takes the covariance in to account. Due to this reason it is also called Statistical distance as well. Mahalanobis distance belongs to the class of generalized ellipsoid distance defined by

$$d(x, y) = \sqrt{(x - y)' M (x - y)} \tag{2.7}$$

Here M is a positive definite, symmetric matrix. In the case the Mahalanobis distance, the matrix M becomes the inverse of variance-covariance matrix. Obviously, this includes

Euclidean distances as a special case when M is the identity matrix.

When using Euclidean distance, the set of points equidistant from a given location is a sphere. The Mahalanobis distance stretches this sphere to correct for the respective scales of the different variables, and to account for correlation among variables [18]. As the axes of ellipsoidal sphere can assume any direction depending upon the data, this is more applicable in the area of time series pattern recognition. So unlike dimensional Euclidean distance, it is possible to express the correlation and weight between dimensions using Mahalanobis distance.

As the nature of Mahalanobis distance allows it to capture correlation among the data and also trend of the time series better compare to the other distances [4, 11], in this work, we proposed to use Mahalanobis distance in k-NN algorithm for FX data. We compare the performance of the Mahalanobis distance based k-NN algorithm with popular Euclidean and Manhattan distance based algorithm.

The performance of the Mahalanobis distance based K-nearest neighbor algorithm was compared with the other distance based algorithms in two ways:

(i) Forecast accuracy
(ii) Transforming their forecasts in to a technical trading rule

In the former case, our goal is to capture the deviation of the fitted values against the actual observations. In the latter case, we are interested in looking at the forecasts in financial point of view. For that we create trading signals, buy and sell using a technical trading rule [6, 7] and the performances were evaluated by the commonly used performance measures in practice.

2.3. **Measures of Forecasting Accuracy.**

Let x_t and $\hat{x}_t$ for $t = 1, 2, ..., n$ be the actual and fitted values respectively. To determine the forecast accuracy of the prediction model for number of out-of-sample predictions, the following error measures were used.

Mean square error (MSE): Mean square error defined by

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(\hat{x}_t - x_t)^2 \tag{2.8}$$

is the most common measurement of error used in Statistics to determine the difference between the true values and estimates. It is a scale dependent measure but gives a basis to compare the forecasts. Due to squaring, MSE gives disproportionate weight to larger errors.

Means absolute percentage error (MAPE): Means absolute percentage error is another widely used accuracy measure when the observations are non-negative. It gives a forecasting accuracy as a percentage so one can compare the error of fitted time series that differ in levels.

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}|\frac{\hat{x}_t - x_t}{x_t}| \tag{2.9}$$

Also, mean absolute percentage error does not affect by larger deviations as MSE does. It is zero for a forecasting model when there is a perfect fit. But there is no restriction of its upper bound.

Theil's U- statistic (U): We consider the following version of to compare the forecasting accuracy of our model.

$$U = \frac{\sqrt{\sum_{t=1}^{n}(\hat{x}_t - x_t)^2}}{\sqrt{\sum_{t}^{n}(\hat{x}_t)^2} + \sqrt{\sum_{t}^{n}(x_t)^2}} \tag{2.10}$$

This is a measure of the degree to which the forecasted values differ from the actual values. U statistic is independent of the scale of the variable and constructed in such a way that it necessarily lies between zero and one, with zero indicating a perfect fit.However, U statistic do not provide information on forecasting bias which is better captured by the mean square error.

Normalized Root Mean Square Error (NRMSE): Scale invariant forms of mean square error (MSE) are useful because often we want to compare errors in different scales. The non-dimensional version we consider here is the Normalized-Root-Mean-Squared Error (NRMSE) given by:

$$NRMSE = \frac{\sum_{t=1}^{n} e^2(t)}{\sigma} = \frac{\sqrt{\sum_{t=1}^{n}(\hat{x}_t - x_t)^2}}{\sigma} \tag{2.11}$$

Here is the standard deviation of the time series [1]. The Normalized Root Mean Square Error (also called the normalized root mean square deviation, NRMSD) is a frequently used measure of the difference between values predicted by a model and the values actually observed.

FX trading data contain many directional changes. So one of the natural questions that arise is which measure does better prediction at the points of directional change. Following gives a measure of accuracy of directions at the points of change of direction.

Sign Correction Proportion (SCP): This is a measure of forecasting accuracy in which the proportion of forecasts that correctly predict the direction of the series movement is considered. Directional change in times series is defined as:

$$d(i) = \begin{cases} 1 & ;\text{if} \quad x(t+1) - x(t) > 0 \\ -1 & ;\text{if} \quad x(t+1) - x(t) < 0 \end{cases} \tag{2.12}$$

To identify whether a model forecasts in the same direction as real data, we compare the directional change using the forecasted values by the model and actual data. Let δ be the function that takes values 1 if the forecast observations correctly predicts the direction of change and 0 otherwise. So δ can be defined as:

$$\delta(i) = \begin{cases} 1 & ;\text{if} \quad \hat{d}(i) = d(i) \\ 0 & ;\text{if} \quad otherwise \end{cases} \tag{2.13}$$

Here $\hat{d}(i)$ are the directional change values obtained by the forecasting model. The sign correction proportion (SCP) for n forecasted values is calculated by

$$SCP = \frac{1}{n}\sum_{i=1}^{n}\delta(i) \tag{2.14}$$

This is a widely used accuracy measure to observe how well a model can captures the direction of the time series [13]. Higher the SCP value, better the directional forecasting accuracy.

3. SELECTING EMBEDDING DIMENSION (m) AND NUMBER OF NEIGHBOURS (k)

3.1. Data.

The data used here are the daily exchange rates of Euro (EUR), British pound sterling (GBP), Swiss franc (CHF), Japanese Yen (JPY), and Canadian dollar (CAD) vis-$\acute{a}$-vis American dollar (USD) were used in this paper (ProQuest Statistical datasets). These are the daily spot rates of the currencies from $January$ 2006 to $December$ 2010.

3.2. Embedding Dimension.

The choice of embedding dimension for the time series is a key issue need to be addressed before start making trading signals. So first we conduct an empirical investigation to select a suitable value for m. Here we have considered all five exchange rates data sets we used in our work. In this empirical investigation, we want to figure out whether the choice of m is data dependent and also distance dependent. So the forecasting accuracy was compared using all the error measures mentioned in section 3.3 by varying the value of m along with different distance choice. 80% of the data was considered as the training set and rest of the

20% was taken as the testing set. The value of k was set to be 2% of the training sample for this part of the work [6]. The training window size was kept fix by removing the oldest data value at each time step.

TABLE 1. U-statistic with Mahalanobis distance

Currency	$m = 3$	$m = 4$	$m = 5$	$m = 6$
EUR	0.00389801	0.00392875	0.00430221	0.00456411
GBP	0.00345430	0.00348754	0.00359710	0.00362095
JPY	0.00690517	0.00717526	0.00810838	0.00771370
CHF	0.00555915	0.00572837	0.00586630	0.00590578
CAD	0.00597986	0.00642774	0.00612342	0.00643873

Table 1 shows the U-Statistic values for different embedding dimension, m with Mahalanobis distance. According to our observation, larger m values do not improve the accuracy of the model. It clearly indicates that for almost all the data sets, the embedding dimension, $m = 3$ gives the minimum error when comparing U-Statistics. These observations do not change significantly even when we replace Mahalanobis distance with Euclidean or Absolute distance. Also the obtained results are similar for the other error measures as well. As can be seen from the graph below, normalized root mean square error also suggests choosing 3 as the embedding dimension.

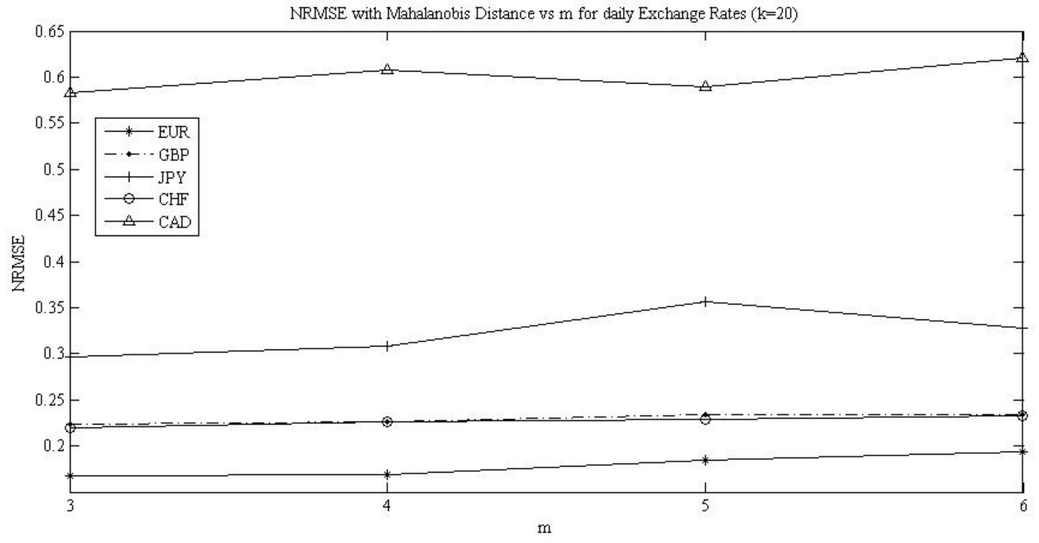

Figure 3.1: Normalized Root Mean Square Error vs. Embedding Dimension with Mahalanobis Distance and Euclidean Distance respectively

The tables 2(A) & (B) given below summarize the obtained results for the choice of embedding dimension for all 5 currencies with U-Statistic and normalized root mean square error. (This was published in the proceedings of Nonlinear Dynamical Systems and Application, $Volume$ 6)

TABLE 2(A). Optimal Choice of Embedding Dimension with U-Statistic.

Currency	Mahalanobis distance	Euclidean distance	Absolute distance
EUR	3	3	3
GBP	3	3	3
JPY	3	3	3
CHF	3	3	3
CAD	3	4	3

TABLE 2(B). Optimal Choice of Embedding Dimension with Normalized Root Mean Square Error.

Currency	Mahalanobis distance	Euclidean distance	Absolute distance
EUR	3	3	3
GBP	3	3	3
JPY	3	3	3
CHF	3	3	3
CAD	3	3	3

Almost all the data sets agree with the conclusion of $m = 3$ being optimal. Similar results were obtained comparing mean square error (MSE) and mean absolute percentage error (MAPE).

3.3. **Number of Nearest Neighbors, k.**

Next, we wanted to figure out the effect of number of nearest neighbors (k) in the forecasting algorithm before comparing the performance of each distance choice. In time series data forecasting, there isn't a uniform guideline to select the neighborhood size [6, 7]. Especially for Financial data, the approaches used were diverse, so one cannot come up with a unique method [6, 7]. The approach used here is the commonly used Casdagli's (1991) algorithm [1, 2, 7].

In Casdagli's work [1, 2], the k was determined empirically by testing the algorithm for several value of k between $2(m + 1)$ and $T - (m - 1)$. His approach was to minimize normalized root mean square error (NRMSE) by changing the value of k. He also varied the embedding dimension m and studied the behavior of NRMSE as a function of k. His choice of distance was Euclidean distance.

In this work, we have chosen the neighborhood size k after comparing not only NRMSE, also all the accuracy measures discussed in section 2.2. We did not restrict our distance choice to just one function. We tested the data for $k(k > m + 1)$ using Mahalanobis, Euclidean and Absolute distances. Also this method was applied to all five currencies to select an appropriate value for k for further analysis.

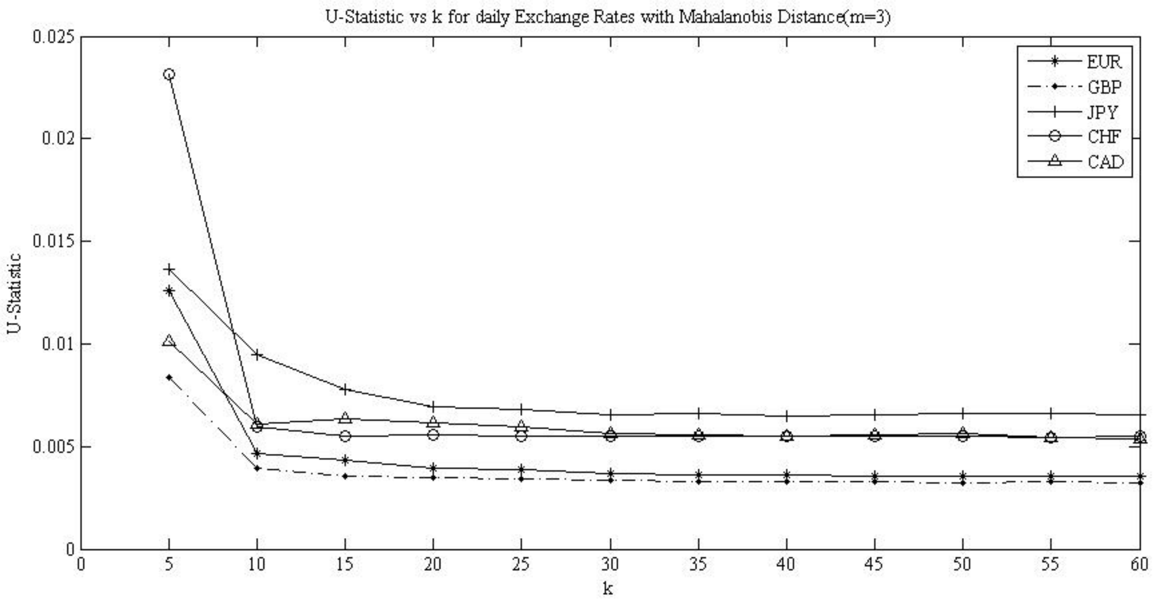

Figure 3.2(a) U-Statistic vs. k with Mahalanobis Distance

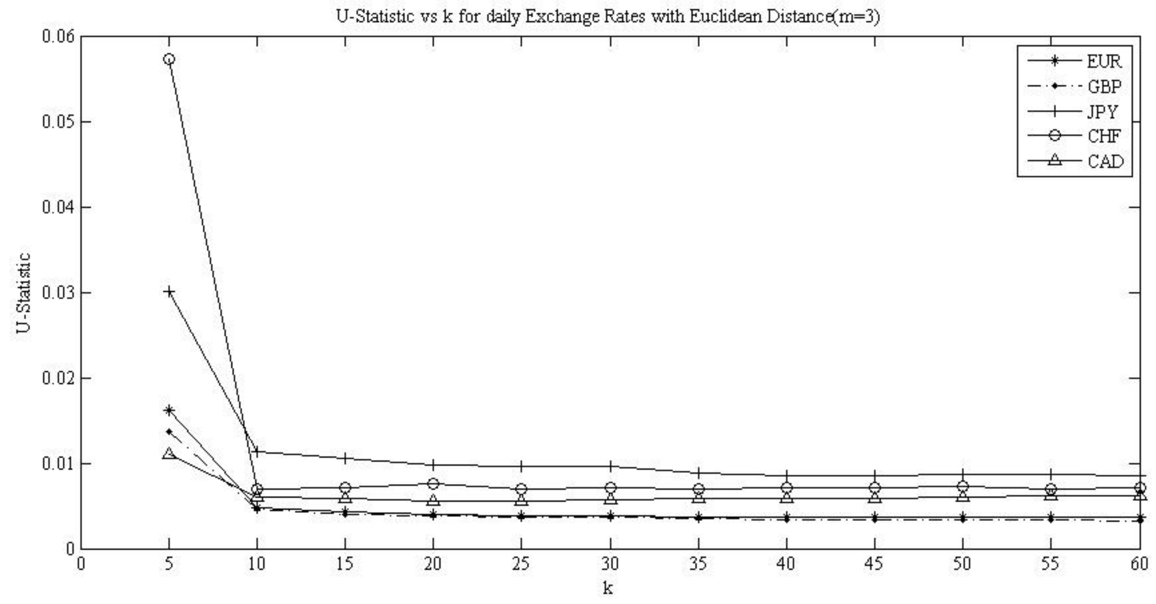

Figure 3.2(b) U-Statistic vs. k with Euclidean Distance

Even though the numerical values are slighting different, all the data sets behaves exactly the same way as k changes. We started with $k = 5$ as k has to be greater than $m + 1$ which is 4 in our case since $m = 3$. It can be seen clearly from the graphs above that for values of k from 5 to 15, the change in U-Statistic were significant for all the currencies. Thereafter, even the error measure decreases, the difference is much smaller. Especially, after $k = 20$ or 25 the changes are negligible.

We did not want to increase k further as can be seen from the table, for some cases the error measure even started to increase slightly at some point. Also, as k-NN method is a data reduction technique, the larger values of does not justify our main goal. This may even result in considering the data as neighbors which are not that much effective on the forecasting as father we go the data are less relevant to the most recent available vector. The other accuracy measures also support the conclusion that an intermediate value of neighborhood size will be an appropriate choice for k.

We wanted to extend our search for an optimum choice of k for different choices of m, so instead of restricting m to be 3, we considered $m = 4, 5$ and 6 and compared the forecasting accuracy as a function of k. Similar analysis were performed as of $m = 3$ and behavior of error measures along with the distance choice were investigated. The obtained results were not much of different to those of $m = 3$.

Even though with a higher m value we need slightly larger k, earlier analysis of m indicates that increasing m does not improve the forecasting accuracy. Considering all these facts, to compare the performance of Mahalanobis distance based k-nearest neighbor algorithm with other distance choices, the key parameters m and k of the algorithm were set to be 3 and 20 respectively.

4. COMPARISON OF DISTANCE MEASURES WITH RESPECT TO FORECASTS ACCURACY AND BUY OR SELL DECISIONS.

4.1. Forecasting Accuracy.

As the key parameters for the k-NN been selected the key parameters, next step is to compare the performance of proposed Mahalanobis distance based k-NN algorithm with traditional Euclidean and Absolute distance based algorithms. For the first part, we considered all the accuracy measures mentioned in section 3.3 and compared the performances of each algorithm based on how accurate their forecasts are. The following tables give the summarized results for the currencies EUR, GBP, JPY, CHF, and CAD with the choice of $m = 3$ and $k = 20$.

In table 3 and 4 below, we report the forecasting accuracy for Mahalanobis, Euclidean and absolute distance using U-Statistic and normalized root mean square error respectively. The observations were similar when comparing the other error measures as well.

TABLE 3. U-Statistics.

Currency	Mahalanobis distance	Euclidean distance	Absolute distance
EUR	0.00389801	0.00405640	0.00401522
GBP	0.00345430	0.00377776	0.00374190
JPY	0.00690517	0.00971223	0.01331857
CHF	0.00555915	0.00755257	0.00771336
CAD	0.00597986	0.00648498	0.00619930

TABLE 4. Normalized Root Mean Square Error.

Currency	Mahalanobis distance	Euclidean distance	Absolute distance
EUR	0.16793291	0.17332175	0.17155159
GBP	0.22321725	0.24333251	0.24104249
JPY	0.29681857	0.42612478	0.58432940
CHF	0.21983992	0.29457650	0.30101833
CAD	0.58313581	0.58857894	0.59790435

From the results above, it is clear that Mahalanobis distance outperforms the other distance measures for all the currencies. Except for normalized root mean square for CAD/USD rates, all the other values for Mahalanobis distance are significantly smaller compared to other distances. Especially in the cases of JPY/USD and CHF/USD rates, the error measures of the proposed algorithm are much smaller than the traditional k-NN forecasting with Euclidean and Absolute distances.

When comparing sign correction proportion (SCP) measures, the obtained results do not support our previous observations. As can be seen from the table below, 3 out of 5 currencies indicate that Euclidean distance based method has a higher correction proportion than Mahalanobis based method.

TABLE 5. Sign correction proportion (SCP) with Mahalanobis and Euclidean distances

	Mahalanobis distance	Euclidean distance
EUR	0.448	0.52
GBP	0.496	0.516
JPY	0.524	0.5
CHF	0.576	0.548
CAD	0.584	0.592

The above results shows that Mahalanobis distance performs better only for JPY and CHF data sets. This was a bit surprising as all the other accuracy measures support Mahalanobis based algorithm.

Due to this reason, we wanted to conduct a thorough investigation of directional forecasting accuracy. So we calculated mean absolute deviation (MAD) and mean square error (MSE) only for the places where the model does not forecast in the same direction as the actual values. In this way, we can measure how forecasted values are deviated from the actual values even though they are not in the same direction as actual values.

TABLE 6. Man Absolute Deviation (MAD) with Mahalanobis and Euclidean distance measures.

	Mahalanobis distance	Euclidean distance
EUR	0.010498383	0.011265078
GBP	0.01103014	0.012782458
JPY	0.00896291	0.023209785
CHF	0.007168656	0.012757111
CAD	0.007185595	0.009144574

TABLE 7. Man Square error (MSE) with Mahalanobis and Euclidean distance measures.

	Mahalanobis distance	Euclidean distance
EUR	0.000146502	0.000164365
GBP	0.000169709	0.000204144
JPY	0.000302811	0.001211702
CHF	0.000115296	0.000458595
CAD	0.000161336	0.000335663

This analysis shows that MAD and MSE values are smaller for Mahalanobis distance compare to Euclidean distance even with a lower SCP value. The question arise here is that can a model have a lower sign correction proportion and still have a smaller MAD? Whats really happening here is even though the Euclidean based method captures the direction of the real data somewhat better than Mahalanobis; forecasted values can be way off the actual values which is giving a higher deviation.

These results indicate that a model may be able to predict more accurately in the same direction as the actual time series still having larger deviation with the actual values. So we can conclude that even though the Mahalanobis distance based method seems to perform slightly weaker comparing SCP measure, it forecasts are much closer to actual data than those of Euclidean distance based method.

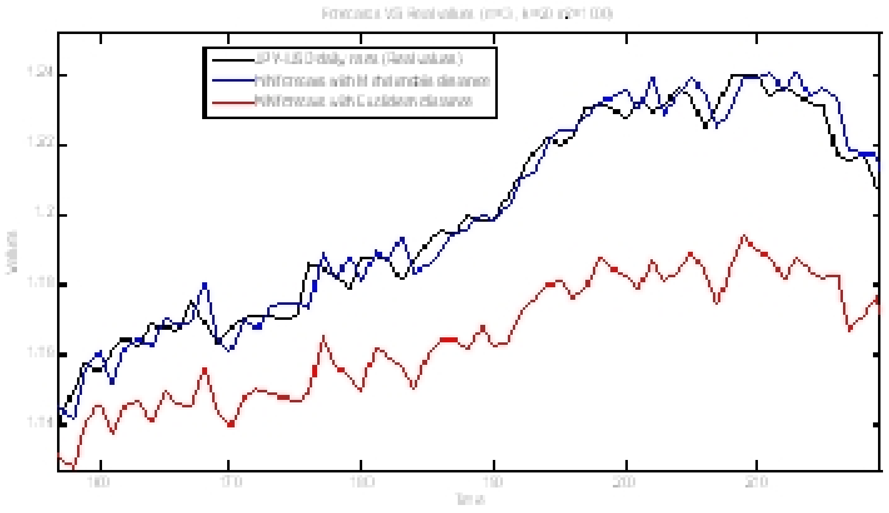

Figure 4.1: Forecasted and real values- JPY/USD rates

Figure 4.1 shows the forecasted rates using Mahalanobis distance (blue color) and Euclidean distance (red color) with the real data for JPY/USD currency market form $t = 150$ to $t = 210$.

As can be seen from the graph, the deviation of Mahalanobis distance based forecasts follow the real FX rates pretty well compare to the Euclidean distance based algorithm.

Even though the accuracy measures verify how well a model forecasts, it is highly recommended that we give an economic value for the predicted financial time series. In the next subsection, we evaluate the economic significance of Mahalanobis distance based forecasting algorithm and compare it with standard Euclidean distance based decisions.

4.2. **Trading Decisions.**

As in any other financial market, in FX market also a trader's main goal is to make more money out of foreign currency fluctuations. The primary goal of foreign exchange rate forecasting has to be making proper trading signals: buy and sell at each time step so that the trader makes more money. To satisfy this main aspect, first we need to transform forecasted values in to trading signals. The forecasts were transformed into a simple technical trading strategy using the trading rule used by Fernandez-Rodriguez, Sosvilla-Rivero, and Andrada-Felix in their work [6, 7]. Let $\hat{r}_t$ given by

$$\hat{r}_t = \ln(\hat{x}_{t+1}) - \ln(1 + i_t^{'}) - \ln(1 + i_t) \tag{4.1}$$

be the estimated return from a foreign currency position over the period $(t, t+1)$ based on the forecasted FX rate at time t. Here x_t represents the spot exchange rate at time $t, \hat{x}_{t+1}$, is the forecasted value for x_{t+1} is the domestic (US) daily interest rate and $i^{'}$ is the foreign country daily interest rate. The trading signals at time t are made based on the estimated return $\hat{t}_t$. The positive returns are executed as long positions (buy) and the negative returns are executed as short position (sell) [6, 7]. So the trading decision can be given as

$$\hat{z}_t = \begin{cases} 1 & ;\text{if} \quad \hat{r}_t > 0 \\ -1 & ;\text{if} \quad \hat{r}_t < 0 \end{cases} \tag{4.2}$$

Based on estimated return, we calculate $estimatedtotal(logaccess)return$ of the trading strategy over the time period $(1, n)$ as

$$\hat{R}_n = \sum_{t=1}^{n} \hat{z}_t r_t \tag{4.3}$$

Here r_t is the actual return at time given by

$$r_t = \ln(x_{t+1}) - \ln(x_t) - \ln(1 + i_t^{'}) - \ln(1 + i_t)$$

We also consider the popular performance measure: $Sharpe\ ratio$ to compare the results along with the estimated total return. The Sharpe ratio, S_R used here is the mean daily total return of the trading strategy over its standard deviation,

$$S_R = \frac{\mu_{\hat{R_n}}}{\sigma_{\hat{R_n}}} \tag{4.4}$$

Higher values of Sharpe ratio indicate that the model is performing better.

TABLE 8. Estimated return

Currency	Mahalanobis distance	Euclidean distance	Absolute distance
EUR	0.52991777	0.47275687	0.46008299
GBP	4.16807227	4.13638609	4.05861762
JPY	0.67755404	0.22975657	0.48608330
CHF	5.42108879	5.16084868	5.16742874
CAD	4.38589604	3.76747181	4.01797807

The estimated total return for the technical trading strategy under different distance measures are given in table 8. The final conclusion of distance choice is pretty much same as that of error measures. Our proposed distance choice outperforms the traditional distance functions

TABLE 9. Sharpe ratio

Currency	Mahalanobis distance	Euclidean distance	Absolute distance
EUR	0.27890809	0.24686318	0.26614195
GBP	2.41593434	2.29776019	2.06031102
JPY	0.18429771	0.01026936	0.13113608
CHF	1.67419376	1.44768129	1.42328537
CAD	1.26400087	0.89502219	1.03678437

The Sharpe ratio also supports our conclusion of choosing Mahalanobis as the distance of choice in in the method as can be seen from the table.
For all these data sets, we have used same number of neighbors and same forecasting technique with each distance measure. From our results, we can clearly see that choosing an appropriate distance in NN algorithm can improve the forecasting significantly. As forecasted values directly effect on trading decision, more accurate forecasting will result in better trading.

4.3. **Further Analysis on k-nearest Neighbor.**

As the k neighboring vectors play a key role in nearest neighbor forecasting, we further investigated on chosen nearest neighbors with different distance measures to observe how well a distance function captures the dynamic behavior of a delay vector. The following figures illustrate a delay vector and its nearest neighbors with Mahalanobis distance and Euclidean distance functions.

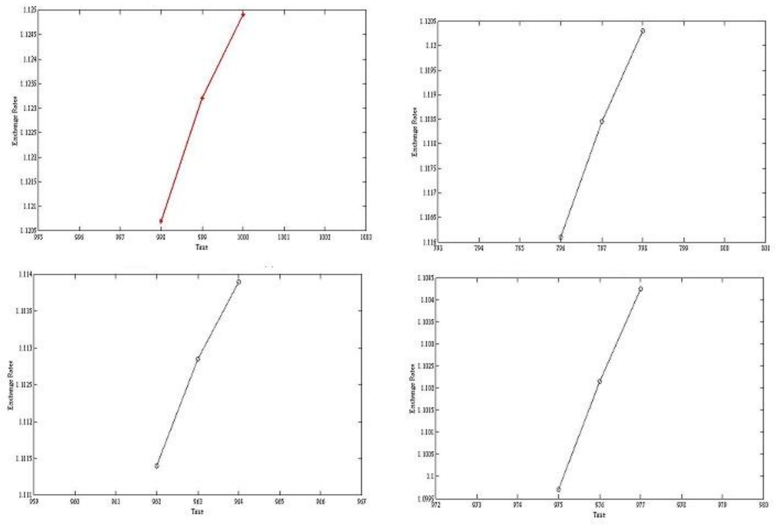

Figure 4.2 (a): A delay vector and its with Mahalanobis distance

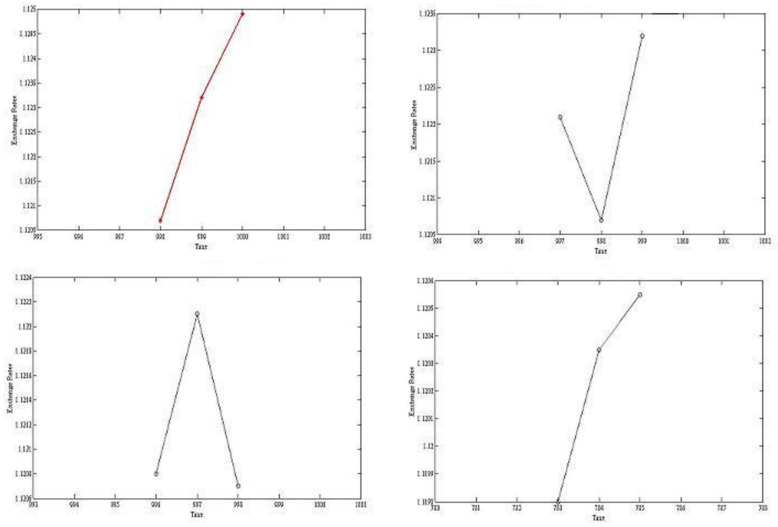

Figure 4.2 (b): A delay vector and its with Euclidean distance

In each figures above, the graph on the top left corner shows the same delay vector for JPY/USD data with $m = 3$ The other graphs are the 3 closets vectors selected by Mahalanobis distance (4.2(a)) and Euclidean distance (4.2(b)). As can be seen from the graphs, Mahalanobis distance captures the time series vectors which are much similar to the delay vector compare to Euclidean distance. These observations support our claim Mahalanobis distance as a better choice for k-NN algorithm.

Even though Mahalanobis distance captures relevant time series vectors as neighbors, when a delay vector has data values in different directions, the selected neighbors do not exactly follow the same pattern. The results are similar with other distance choices are well. The following figure is an example of such a case.

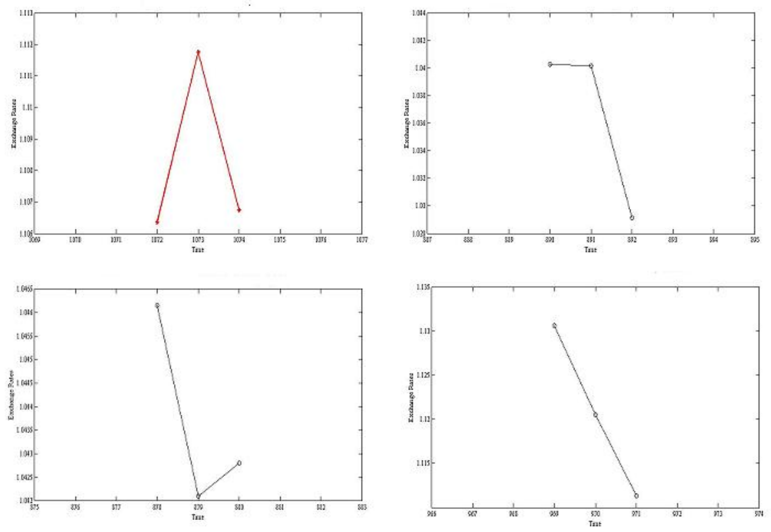

Figure 4.3: A delay vector and it's NN with Mahalanobis distance when there's a directional change

As can be seen from the figure above, for a delay vector with data in different directions, even the Mahalanobis distance captures neighbors which do not behave as same as the delay vector. This is an issue in many time series data forecasting methods. When there are directional changes in data, forecasting methods do not performs well in practice. Due to this reason, the importance of detecting change points in data prior to forecasting have been a major part of time series data analysis [ref]. This is a minor setback of NN forecasting method as well. Even though a proper distance measure captures the dynamic behavior of the data, when there is a change point, the algorithm does not perform as good as the other places. To address this, we will work on combining change point detection methods with Mahalanobis distance based k- nearest neighbor algorithm as a part of our future work.

4.4. k-Nearest Neighbor and weighted regression.

In the standard nearest neighbor algorithm, the selected neighbors are given equal weights towards forecasting. We wanted to see if we assign weights for selected k neighbors according to their distance, how well the model performs against the traditional method. So we were interested in adopting the concept of ?distance-based weighing? with k-NN forecasting. Weighting the data can be viewed as giving more importance to relevant instances

and discarding irrelevant instances. As nearest neighbor algorithm is already replicating relevant instance, we wanted to see whether we need further weighing in the forecasting part.
We used the popular tri-cube weight function defined by;

$$w(u) = \begin{cases} (1-|u|^3)^3, & \text{for } |u| < 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

as our choice of weighting in this paper. Here u is defined as follows:

$$u(i) = \frac{d(i)}{d(k)}$$

where $d(i)$ is distance to the i^{th} nearest neighbor and $d(k)$ is distance to the k^{th} nearest neighbor where k is the farther neighbor considered. Obviously, farther the neighbor, smaller the weight. We compare the forecasting accuracy of locally weighted $k-NN$ and standard $k-NN$ algorithm with $m = 3$ with different values of k. We used both Mahalanobis and Euclidean distance for this analysis. The table below gives U statistics values for k-NN algorithm with Mahalanobis distance.

TABLE 10. U Statistic with and without weighted regression Mahalanobis distance.

Currency	Mahalanobis $(k = 20)$	Mahalanobis weight $(k = 20)$	Mahalanobis weight $(k = 25)$	Mahalanobis weight $(k = 30)$	Mahalanobis weight $(k = 35)$	Mahalanobis weight $(k = 40)$
EUR	0.00389801	0.00482134	0.00447082	0.00429696	0.00415463	0.00406373
GBP	0.00345430	0.00390652	0.00369875	0.00360442	0.00353554	0.00347328
JPY	0.00690517	0.01172117	0.010751872	0.009973787	0.009461335	0.00888432
CHF	0.00555915	0.006070793	0.005971333	0.005872418	0.005811847	0.005759591
CAD	0.00597986	0.008575087	0.007683319	0.007121021	0.00676352	0.006640292

First column in Table 9 gives the values for U-Statistics without weighted regression for $k = 20$. With the same choice of k, when we introduce weights, the algorithm does not performs as good as before. To see what is really happening here, we increased the value of and performed the same analysis. Even with $k = 20$ we don?t see results as good without weighting. When we increased the number of nearest neighbors, further, weighted regression started to give better results which are almost as good as regular local regression method. This idea does not support our primary goal of data reduction. If we use Mahalanobis distance with less number of data we can have better performance in forecasting compare to weighted regression. So if we use an appropriate distance to select neighbors for NN algorithm, we can treat the selected neighbors equally during the forecasting.

5. CONCLUDING REMARKS.

As finite sample nearest neighbor algorithm dependent on the distance function, choosing an appropriate distance measure we can obtain better performances. We compared the accuracy of the k-NN forecasting for Foreign exchange data with traditional Euclidean and Absolute distance based algorithm with our proposed Mahalanobis distance based algorithm. In this work, we observed that the proposed Mahalanobis distance based method outperform Euclidean distance as well as the absolute distance based methods both in terms of better fit and in terms of the trading rule. Also, we have conducted a thorough analysis of choice of embedding dimension and neighborhood size for nearest neighbor algorithm using five different currencies with different distance choices. The observations suggest that smaller value of m such as 3 and sufficiently large enough value of k give significantly better results.

References

[1] M. Casdagli, Nonlinear forecasting, *Chaos and statistics, Santa Fe Institute working paper No.91-05-022* (1991).

[2] M. Casdagli, chaos and deterministic versuss stochastic nonlinear modeling *Journal of Royal Statistical Society, Series B Vol. 54, No 2* (1992), 303–328 .

[3] T. M. Cover, P. E. Hart, Nearest Neighbor Pattern Classification, *IEEE Trans. Information Theory, IT-13 (1)* (1967), 21–27.

[4] L. Gyorfi and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer-Verlag, New York, 1996.

[5] S. A. Dudani, The Distance-Weighted k-Nearest-Neighbor Rule, *IEEE Transactions on Systems, Man, and Cybernetics* (April 1976).

[6] F. Fernández-Rodríguez, S. Sosvilla-Rivero, J. Andrada-Félix, Technical Analysis in Foreign Exchange Markets: Linear versus Nonlinear Trading Rules, *FEDEA Int. Economics and Finance DEFI* (2000), working paper No. 00-02.

[7] F. Fernández-Rodríguez, S. Sosvilla-Rivero, J. Andrada-Félix, Exchange-rates forecasting via simultaneous nearest-neighbor methods: evidence from the EMS, *International Journal of forecasting* (1999), 15, 383–392.

[8] K. fukunaga and L. D. Hostetler, Optimization of k-Nearest-Neighbor Density Estimates , *IEEE Transactions on information theory, Vol. IT-19* (May 1973), 320–326.

[9] Y. Kawahara, T. Yairi, and K. Machida, Change-Point Detection in time-Series Data Based on Subspace Identification, in: *Proc. of the 7th IEEE Int'l Conf. on Data Miningl* (2007), 559–564.

[10] Songbliu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama, Change point detection in time series by relative density ratio estimation, in: *Strucctural, Syntactic, and Statstical Pattern recognition*(Joint IAPR International Workshop, SSPR&SPR 2012), Hiroshima, Japan, November 7-9, 2012.

[11] P. C. mahalanobis, On the Generalized Distance in Statistics, in: *Proceedings of the national Institute of Science of India 12* (1936), 49–55.

[12] Vindya I. Kumari Pathirana and K. M. Ramachandran, k-Nearest Neighbor Algorithm with Mahalanobis Distance for FX Trading, in: *proceedings of dynamic Systems and Applications*, 6 (2012), 324–328.

[13] A. A. Skabar, Direction-of-Change Financial Time Series Forecasting using Bayesian Learning for MLPs, in: *proceedings of the World Congress on Engineering* (2008), Vol II.

[14] D. Robert Short and K. Fukunaga, The optimal Distance Measure for Nearest neighbor Classification, *IEEE transactions on information theory* (1981), Vol. IT-27, No. 5, 622–627.

[15] triennial Central bank Survey, *foreign exchange turnover in April 2013*, preliminary global results, http://www.bis.org/publ/rpfx13fx.pdf.

[16] Walters-Williams, and Y. Li, Comparative Study of Distance Functions for Nearest Neighbor, in *Advanced Techniques in Computing and Software Engineeringl*, (2010), (Ed: Elleithy) 123–456.

[17] J. Wang, P. Neskovic, and L. N. Cooper, Improving Nearest Neighbor Search Algorithm with Simple Adaptive Distance measure., *Pattern Recognition Letters* (2007), 28 123–456.

[18] T. Uemiya, T. Matsumoto, D. koizumi, M. Shishibori, K. Kita, Fast Multidimensional Nearest Neighbor Search Algorithm Based on Ellipsoid Distanc, *International Journal of Advanced Intelligence* (2009).