PREDICTION OF STOCK MARKET INDEX CHANGES

İzzet Şirin and H. Altay Güvenir

Department of Computer Engineering and Information Science,
Bilkent University, Bilkent, 06533 Ankara
sirin@cs.bilkent.edu.tr, guvenir@cs.bilkent.edu.tr

In *Adaptive Intelligent Systems, Proceedings of the BANKAI Workshop, Brussels, Belgium, 12-14 October 1992*, S.W.I.F.T. (Ed.), Elsevier Science Publishers B. V., Amsterdam, (1993), 149–160.

BU-CEIS-9201

Prediction of Stock Market Index Changes

İzzet Şirin, H. Altay Güvenir

Computer Engineering and Information Science Department
Bilkent University, Ankara 06533 TURKEY

Abstract

Systems for inducing concept descriptions from examples are valuable tools for assisting in the task of knowledge acquisition for expert systems. In this research three machine learning techniques are applied to the problem of predicting the daily changes in the index of Istanbul Stock Market, given the price changes in other investment instruments such as foreign currencies and gold, also changes in the interest rates of government bonds and bank certificate of deposit accounts. The techniques used are *instance-based learning* (IBL), *nested-generalized exemplars* (NGE), and *neural networks* (NN). These techniques are applied to the actual data comprising the values between January 1991 and July 1992. The most important characteristic of this data is the large amount of noise inherent in its domain. In this paper we compare these three learning techniques in terms of efficiency, ability to cope with noisy data, and human friendliness of the learned concepts.

1 Introduction

In expert system construction process the main bottleneck is knowledge acquisition. Two families of systems for learning from examples, based on ID3 [8] and AQ [5] algorithms, have been especially successful. These basic algorithms assumes no noise in the domain, searching for a concept description that classifies training data perfectly. However, application to *real–world* domains requires methods for handling noisy data.

Financial markets form such a real-world domain. Investment management is complex, yet promising domain for the application of machine learning techniques [2]. In this paper, we compare three machine learning techniques are applied to the problem of predicting the daily changes in the index of Istanbul Stock Market, given the changes in other investment instruments. The input is the daily changes in the exchange rates of US Dollar (USD) and Deutsch Mark (DM), price changes in Republic Gold coin (RG) and 1 gr. of solid gold (SG), changes in the interest rate of government bonds (GB) and three-month Certificate of Deposit bank account (CD). Given the changes from previous

day to current day, the learning techniques are tested to predict the change in the index of the Istanbul Stock Market between current day and the following business day. These techniques are applied to the actual data comprising the values of an eighteen months period, between January 1991 and July 1992. The most important characteristic of this data is the large amount of noise inherent in its domain.

There are several requirements that a learning system should meet, if it is to prove useful in variety of real–world situations. The first of these requirements is accuracy. The induced rules should be able to classify new examples accurately, even in the presence of noise. However, due to the high amount of noise in this particular domain, we cannot expect to obtain the accuracy which is possible in other domains. The second requirement is the simplicity of the learned rules; this requirement is applicable to inductive learning systems. For the sake of comprehensibility, the induced rules should be as short as possible. The resulting description should be comprehensive as single chunks of information, directly interpretable in natural language. However, when noise is present, the requirement of accuracy can lead to complex rules. Thus to induce short description, one must usually relax the requirement that the induced rules be consistent with all the training data. The choice of how much to relax involves a trade–off between accuracy and simplicity [3].

The techniques used in this research are *instance-based learning* (IBL), *nested-generalized exemplars* (NGE), and *neural networks* (NN). These techniques learn class descriptions from examples. The first two of them generate human readable descriptions. Neural networks are black-box systems where the representation of the learned knowledge is parallel and distributed over multiple units of the network.

The next section defines the problem domain selected. The following three sections describe *Instance-Based Learning* (IBL), the *Nested-Generalized Exemplars* (NGE) techniques and the neural networks. These sections also give the results of the application of the techniques to problem domain. Using these results the last section compares the techniques in terms of accuracy, ability to cope with noisy data, and human friendliness of the learned concepts.

2 Prediction of Stock Market Index

The prediction of the changes of index in a stock market, is a complicated process as most of the economical activities. There are many external effects which have to be considered in prediction process. Some of them are;

- Alternative markets (i.e. gold, foreign exchange markets),
- Macro economical conjuncture,
- Political effects,
- Sectoral conjuncture,
- International affairs,
- Psychological effects.

However, it is not possible to consider all these effects in computer processing, since most of these factors are qualitative rather than quantitative. Hence, in this work a greatly simplified model of stock market is employed. In our model, the stock market index is affected by six factors. They are the daily changes in the exchange rates of US Dollar (USD) and Deutsch Mark (DM), price changes in Republic Gold coin (RG) and 1 gr. of solid gold (SG), changes in the interest rate of government bonds (GB) and three-month Certificate of Deposit bank account (CD). The data contains the values of an eighteen months period, between January 1991 and July 1992, which 395 data points. Each data point comprises the changes in six input factors and the classification of the index change. Since our techniques learn concept description, we defined the index changes as three disjoint concepts: *index–will–increase*, *index-will-not-change*, and *index-will-decrease*. A 0.9% or more increase in the market index is represented by the concept of index–will–increase. Similarly, -0.9% or more decrease is represented by the concept of index-will-decrease. The choice of $\mp 0.9\%$ is to avoid bias towards any of the concepts, since for the break points, the distribution of the number of examples points for each class is most similar: 124 increase, 135 no change, and 136 decrease. Out of 395 examples points, 316 examples (80 %) are used in the training phase and 79 of them are used for testing.

3 Instance-Based Learning

In this section we present IB3 algorithm, the noise tolerant version of the instance-based learning technique [1]. IBL algorithms store in memory only those instances that have informative value. The primary output of IBL algorithms is a *concept description* (CD). This is a function that maps instances to concepts. An instance–based concept description includes a set of stored instances and some information concerning their past performance during the training process (e.g. number of correct and incorrect classification predictions). The final set of instances can change after each training process. However, IBL algorithms do not construct extensional concept descriptions (or do not make generalization). Instead, concept descriptions are determined by how IBL algorithm's *similarity* and *classification* functions use the current set of saved instances. The similarity and classification functions determine how the set of saved instances in the concept description are used to predict values for the category attribute. Therefore, IBL concept descriptions contain these two functions along with the set of instances.

Three components of IBL algorithms are:

1. *Similarity function*: computes the similarity between training instance and instances in concept description.

2. *Classification Function*: yields the classification for training instance by using result of the similarity function and performance record of the concept description.

3. *Concept Description Updater*: maintains records on classification performance and decides which instances should be included in the concept description.

```
CD← ∅
for each x in Training set do
    for each y ∈ CD do
        Sim[y] ← Similarity(x,y)
    if ∃{ y ∈ CD | acceptable(y)} then
        y_max ← some acceptable y ∈ CD with maximal Sim[y]
    else
        i ← randomly selected value in [1, |CD| ]
        y_max ← some y in CD that i–th most similar i instance to x
    if class(x) = class(y_max) then
        classification ← correct
    else
        classification ← incorrect
        CD ← CD ∪ {x}
    for each y in CD do
        if Sim[y] ≥ Sim[y_max] then
            Update y's classification record
            if y's record is significantly poor then
                CD ← CD – {y}
```

Fig. 1. The IB3 Algorithm.

IBL algorithms assume that similar instances have similar classifications. This leads to their local bias for classifying novel instances according to their most similar neigbour's classification. They also assume that, without prior knowledge, attributes will have equal relevance for classification decisions (e.g. by having equal weight in similarity function).

3.1 Description of the IB3 algorithm

IB3 is the noise tolerant version of the IBL algorithms. It employs *wait and see* evidence gathering method to determine which of the saved instances are expected to perform well during classification. IB3 algorithm is shown in Fig. 1.

In all IBL algorithms, the similarity between instances x and y is computed as:

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

IB3 maintains a classification record (i.e. number of correct and incorrect classification attempts) with each saved instance. A classification record summarizes an instances's classification performance on subsequently presented training instances and suggests how it will perform in the future.

IB3 employs a significance test (i.e. acceptable and significantly poor) to determine which instances are good classifier and which ones are believed to be noisy.

IB3 accepts an instance if its classification accuracy is significantly greater than its class's observed frequency and removes the instance from concept description if its accuracy is significantly less. Confidence intervals are used to determine whether an instance

```
 RG       SC       USD      DM       GB       CD       Class
 -------  -------  -------  -------  -------  -------  -------
 0.00000  0.00000 -0.64433  0.21739   0.00000  0.00000 : 1   (31/74)=0.418919
-0.19047 -0.50697 -0.07072 -0.22148   0.00000  0.00000 : 2   (29/68)=0.426471
-0.25253 -0.42409  0.00000  0.29806   0.00000  0.00000 : 2   (14/26)=0.538462
 0.28818  0.86289  0.88593  0.19084 -15.36599  0.11230 : 0   (10/24)=0.416667
 0.72289  0.64102  0.36429  0.29282   0.00000  0.00000 : 1   (12/30)=0.400000
-0.77519 -0.60711  0.00000  0.33898   0.00000  0.00000 : 0   (12/30)=0.400000
```

Fig. 2. Some instances stored by the IB3 algorithm.

is acceptable, mediocre, or noisy. Confidence intervals are constructed around both the current classification accuracy of the instance and current observed relative frequency of its class. We chose 85% as acceptance and 55% as confidence level for dropping in prediction of stock market index.

3.2 Results of the IB3 Algorithm

We have tested the IB3 algorithm on our problem of predicting the stock market index changes. Out of the 316 training examples (80% of all data) 80 instances are stored in the memory. For each class two instances stored in the memory with highest confidante values are given in Fig. 2. Each line in the figure represents an instance. The first six values are the feature values and the seventh is the class of the instance. The last value indicates the number of correct predictions vs. the number of references made to that instance. The algorithm was trained and tested on 10 different orderings of the data. The average success in 10 test runs is 46%. Considering that there are three classes, it is 13 points above the random prediction.

4 Nested Generalized Exemplars

In this section, we presented an incremental learning algorithm based on the theory of learning from examples, called Nested Generalized Exemplars (NGE) [7]. NGE is a variation of learning model called *exemplar–based learning*, which was originally proposed as a model of human learning by Medin and Schaffer [4]. In the simplest form of the exemplar–based learning, every example is stored in memory, with no change in representation (or without generalization), as in IB3 algorithm presented in the previous section.

NGE adds generalization on top of the simple exemplar–based learning. It adopts the position that exemplars, once stored, should be generalized. The learner compares a new example to those it has seen before and finds the most similar, according to a similarity metric, which is inversely related to the distance metric (Euclidean distance in n–space). The term *exemplar* is used to denote an example stored in memory. Over time, exemplars may be modified (due to generalization) from their original forms.

Once a theory moves from a symbolic space to a Euclidean space, it becomes possible to nest generalization one inside the other. This is where the term *Nested* comes from.

Take a few (minimum two) examples as initial exemplar(s).
for each example E in training set do
  for each (Hyperrectangle) $H_k \in$ CD do
    $D_k \leftarrow Distance(E, H_k)$
  Take the two hyperrectangles (H_{min1}, H_{min2}) with minimum D_k
  if H_{min1}.class = E. class then
    increment(H_{min1}.correct)
    increment(H_{min1}.reference)
    Generalize H_{min1} with E in all feature dimensions.
  else if H_{min2}.class = E.class then
    increment(H_{min1}.reference)
    increment(H_{min2}.correct)
    increment(H_{min2}.reference)
    Generalize H_{min2} with E in all feature dimensions.
  else
    increment(H_{min2}.reference)
    store E as new exemplar
    adjust all features weights as follows
    if E_{f_i} matches with $H_{min1_{f_i}}$ then
      $w_{f_i} \leftarrow w_{f_i}$ (1 + Δ_f)
    else
      $w_{f_i} \leftarrow w_{f_i}$ (1 - Δ_f)

Fig. 3. The EACH Algorithm.

Its generalizations, which take the form of hyperrectangles in E^n, can be nested to an arbitrary depth, where inner rectangles act as *exceptions* to the outer ones.

4.1 Description of EACH

EACH (Exemplar-Aided Constructor of Hyperrectangles) is a particular implementation of the NGE technique [7]. The EACH algorithm is given in Fig. 3. EACH uses numeric slots for feature values of exemplar. The generalizations in EACH take the form of *hyperrectangles in Euclidean n–space*, where the space is defined by the feature values for each example. Therefore, generalization process replaces the slot values with more general values (i.e. replacing range of values [a, b] with another range [c, d], where $c \leq a$ and $d \geq b$).

The EACH algorithm given in Fig. 3. compares the class of a new example with the most similar (shortest distance) exemplar in the memory. The distance between and an example and an exemplar is computed according to the following formula:

$$Distance(E, H_k) = w_{H_k} \sqrt{\sum_{i=1}^{n} (w_{f_i} \frac{d_i f_i}{max_i - min_i})^2}$$

RG	SC	USD	DM	GB	CD	Class	#C/#R
[0.00, 0.78]	[-0.53, 0.40]	[-0.94, 0.65]	[-0.94, 1.17]	[0.00, 1.79]	[0.00, 0.85]	1	30/59
[-2.34, 5.16]	[-1.06, 2.18]	[-0.15, 3.01]	[-0.21, 20.31]	[0.00, 0.00]	[0.00, 0.00]	1	27/60
[-0.38, 0.23]	[0.00, 0.53]	[-0.15, 0.28]	[-0.54, 0.43]	[0.00, 11.78]	[-0.35, 0.00]	0	5/7
[-0.94, 2.47]	[-6.70, 2.13]	[-1.97, 2.92]	[-12.75, 2.88]	[-11.91, 16.50]	[0.00, 0.22]	2	89/246

Fig. 4. Some exemplars stored by the EACH algorithm.

where w_{H_k}: weightof$Hyperrectangle_k(H_k.reference/H_k.correct)$
w_{f_i}: weight of the$feature_i$
max_i, min_i: maximum and minimum feature values, respectively

$$d_i f_i = \begin{cases} E_{f_i} - H_{k_{upper}} & \text{if } E_{f_i} > H_{k_{upper}} \\ H_{k_{lower}} - E_{f_i} & \text{if } E_{f_i} < H_{k_{lower}} \\ 0 & \text{otherwise.} \end{cases}$$

If the they are the same (i.e. it has made a correct prediction) the exemplar is generalized to include the new example, if it is not already contained in the exemplar. However, if the closest example has a different class then that of the example, then the second closest example is tried in the similar way. The idea behind the second minimum is apply the *second chance* heuristic. This heuristic is useful to reduce the number of exemplars in the memory. If none the closest two exemplars have the same class as the example, then the algorithm modifies the weights of feature so that the weights of the features caused the wrong prediction is increased (in terms of distance), and weights those which affected against the predictions are decreased, using a global feature adjustment rate Δ_f. A typical value for Δ_f is 0.05. However, there is no general rule for its value; it is domain dependent.

4.2 Results of the EACH Algorithm

The EACH algorithm was tested on our problem of predicting the stock market index changes. After training with randomly selected 10 different sets of examples, on the average 112 exemplars were saved in the memory. Few of the exemplars are given in Fig. 4. Each line in the figure represents an exemplar. The first six values are the feature ranges and the seventh is the class of the exemplar. The last value indicates the number of correct predictions vs. the number of references made to that exemplar.

The algorithm was trained and tested on 10 different orderings of the data. The average success in 10 test runs is 44%. Considering that there are three classes, it is 11 points above the random prediction.

4.3 Modifications to the EACH

Rule induction process is an important issue for semi–automated knowledge acquisition. The induced rules should be compact and human friendly, such that they can be evaluated

Sort (descending) the H_i's according to their reliability (inverse of w_{H_i}).
if H_i overlaps with H_j and $H_i.class \neq H_j.class$ then
  shrink hyperrectangle which has large w_H(less reliable one) value.
Remove the *redundant* hyperrectangles
  (i. e. H_i is in H_j and both have same classes.
  This can happen due to the generalization).
Mark the *exceptions* hence, no more processing is done for exceptions.
For each $class_i$ do disjunctive generalization in feature level
  For each (which is not exception and not used yet) H_k in $class_i$
    if f_j's are overlap then
      generalize the f_j of more reliable exemplar
    else
      add f_j of less reliable exemplar to the disjunct list of
      the feature j of more reliable exemplar and mark
      the less reliable exemplar as used.
List the rules.
Fig. 5. Second phase of the modified EACH algorithm.

and criticized by experts. Especially in noisy domains (such as the one used here) EACH generates too many exemplars. In order to achieve comprehensibility of the learned rules we made some modifications to the EACH algorithm. To reduce number of exemplars we introduced disjunctive generalization at the feature level (note that, EACH handles disjunctive generalization at concept level).

New algorithm consists of two phases. The first phase is same as the original EACH. The second Phase of the algorithm tries to reduce number of exemplars (generated in first phase) by using disjunctive generalization at feature level. For example, suppose H_i and H_j are two exemplars of the same class.

H_i: if f_1 in [1 .. 5] and f_2 in [3] then C_1
H_j: if f_1 in [10] and f_2 in [1] then C_1

In the second phase H_i and H_j disjunctively generalized into a single exemplar

H_{ij}: if (f_1 in [1 .. 5] or [10]) and (f_2 in [3] or [1]) then C_1).

The second phase of the modified EACH algorithm is given in Fig. 5.

In the stock market index prediction problem, the second phase of the modified EACH algorithm reduced the number of exemplars from 112 down to 4. The resulting set of rules are given in the appendix.

5 Artificial Neural Networks

Artificial neural network (ANN) models have been studied for many years in the hope of achieving human–like performance in many fields [6]. These models are composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural nets. Computational elements or nodes are connected via weights that are typically adapted during the training phase to improve performance. There has been a recent resurgence in the field of ANN caused by new net topologies and

algorithms. ANN models attempt to achieve good performance via dense interconnection of simple computational elements.

5.1 Back–propagation Training Algorithm

There are many models for ANN implementations. We used three–layer feed–forward net for our domain and we used *back–propagation* training algorithm. The back-propagation algorithm is a generalization of LMS (Least Mean Squares) algorithm. It uses a gradient search technique to minimize a cost function equal to the mean square difference between desired and actual net outputs. The net is trained by initially selecting small random weights and internal thresholds and then presenting all training data repeatedly. Weights are adjusted after every trial using side information specifying the correct class until the weights converge and the cost functions is reduced to an acceptable value. An essential component of the algorithm is the iterative method that propagates error terms (required to adapt weights) back from nodes in the output layer to nodes in previous layers.

The generally good performance found for the back–propagation algorithm is somewhat surprising considering that it is a gradient search technique that may find a local minimum in the LMS cost function, instead of the desired global minimum.

To improve performance and reduce the occurrence of the local minima extra hidden units can be added to the network. Also the weight adjustment parameters momentum (α) and learning rate (η) can be tried. One difficulty noted with the back–propagation algorithm is that in many cases it requires large number of epochs (one pass through all training examples).

5.2 Results of Neural Network with Back–Propagation

Network has 6 input units, 2 hidden units and 3 output units (one for each category; 0 : inactive 1 : active). We experimentally determined the number of hidden units as 2 and the learning rate as 0.7 and momentum as 0.3. The neural network was trained with randomly selected 10 different sets of training examples. It successfully identified on average 38% of the 79 test examples.

6 Conclusion

In this work, we have tested three machine learning algorithms on the problem of predicting the daily changes in the index of Istanbul Stock Market, given the price changes in other investment instruments. As far as the success rates of these algorithms over the test data is concerned, the IB3 algorithm performed better than the other two techniques.

One of the most important advantages of IBL algorithms is their simplicity. IBL algorithms have relatively relaxed concept bias. They incrementally learn piecewise–linear approximations of concepts. In contrast, algorithms that learn decision trees or rules approximate concepts with hyperrectangle representations. IBL algorithms can record faster learning rates than these other algorithms when their bias is not satisfied by target concepts in application domain. This occurs when the target concept's boundary is not parallel to the attributes dimensions.

Instance–based algorithms also have some disadvantages. IB3's learning performance is highly sensitive to the number of irrelevant attributes used to describe instances. Its storage requirements increase exponentially and its learning rate decrease exponentially with increasing dimensionality. Hence, IB3 is a poor learning algorithm for applications involving multiple irrelevant attributes. IB3's learning performance is also highly sensitive to the choice of the confidence intervals.

Another issue is comprehensibility of the knowledge acquired by the IBL algorithms. Instance–based representation does not summarize conceptual structure.

Although the EACH algorithm performed slightly worse than the IB3 on the test data, the modified EACH was able to produce compact and human friendly rules. This is important especially if the knowledge acquired by a learning system is to be evaluated and criticized by experts.

Since EACH algorithm maintains different weights for each feature (dimension), it can cope with the problem of irrelevant features. The generalization process is an important component of the EACH algorithms. It helps the construction of the more compact description of the concepts.

If the application domain noise free (or at least less amount of noise), fewer hyper-rectangles are generated, and they can be easily interpreted. Even if too many rectangles present they can be interpreted easily by combining the disjunction at feature level. The extended EACH algorithm further disjunctively generalizes these rectangles into more human friendly rules.

NGE's learning performance highly depend on the noise level of the main. Although, it tries to reduce the effect of the noise in some sense by creating new hyperrectangles and reducing the reliability of the hyperrectangles, it is not sufficient for extensively noisy domains (such as our domain). Too many hyperrectangles increase the failure probability in prediction. Another weakness of the EACH is static feature adjustment rate. Changing the weights for noisy examples can cause the system to forget what it learned during the previous training. This is another weakness of the EACH algorithm, which is the sensitivity to the order of the examples and value of feature adjustment rate.

The EACH constructs axis–parallel rectangles. In some domains boundaries of the concept descriptions are not axis-parallel. Although this construction helps in some domains, in such domains it causes performance degradation.

The neural network with back-propagation performed worst among the techniques tested in this work. Although the neural networks worked well in many noisy domains, for our problem they did not prove useful. This failure was due to converging in local minima. The disadvantage of the back-propagation algorithm is that it requires large number of passes through all the training examples.

We tried to determine the correlation between these investment instruments and stock market. Due to the high amount of noise in the data, none of the learning techniques performed at any acceptable rate. In fact, this is because of the nature of the economical activities. There are many factors affecting these activities. For example, during that period Golf War affected the whole world economy, especially the Turkish economy. Another important event for Turkish economy, was the general election held in October 1991. Taking these exceptional events into account, the performance of these techniques can be considered reasonable.

References

[1] D. W. Aha, D. Kibler and M. K. Albert, Instance–Based Learning Algorithms. *Machine Learning* **6** 37–66, 1991.

[2] G. Mani, The DIME System: A Preliminary Report, *International Journal of Intelligent System in Accounting Finance and Management* **1** 29–39, 1992.

[3] W. Iba, J. Wogulis and P. Lngley, Trading of simplicity and coverage in incremental concept learning. *Proceedings of Fifth International Conference on Machine Learning* 73–79, 1988.

[4] D. Medin and M. Schaffer, Context Theory of Classification Learning, *Psychological Review* **85** 207–238, 1978.

[5] R. S. Michalski, On the quasi–minimal solution of the general covering problem. *Proceedings of the Fifth International Symposium on Information processing* 125–128. 1969.

[6] R. H. Nielsen, *Neorocomputing.* Addison-Wesley.

[7] S. Salzberg, A Nearest Hyperrectangle Learning Method, *Machine Learning*, **6** 251–276, 1991.

[8] J. R. Quinlan, Inductions of Decision Tress. *Machine Learning* **1**, 81–106, 1986

Appendix: Generated Rules

The rules generated by the modified EACH can be seen below. Note that the original EACH produced 132 rules for the same input data, which was too many to be used by human experts.

All the feature weights were initialized to 1.0, and the feature adjustment rate was set to 0.02.

The numbers in parenthesis show the the number of successful predictions made by the rule and the number of references made to the rule, respectively. For example, (20/35) means that the rule was referenced 35 times and 20 of them were successful predictions.

In this context, CF represents the coverage factor of (*#correct/#examples*) of a rule, that is, it indicates the ratio of the data points correctly classified by the rule over all data points.

Here are the rules (only 4) generated for 3 classes:

```
IF USD in [-1.97 .. -0.94] or [0.13 .. 0.16] or [-2.32 .. -2.22] or [-4.34 .. -3.64] or
          [-0.86 .. -0.60] and
    DM in [-13.87 .. -4.24] or [-1.77 .. -0.63] or [6.63] and
    RG in [-4.06 .. -2.28] or [-0.45 .. -0.40] and
    SG in [-6.70 .. -2.23] or [-1.05 .. -0.93] and
    GB in [-29.94 .. -18.06] or [-6.51 .. 6.48] or [27.85] or [6.93 .. 10.54] and
    CD in [-0.74 .. 0.0] or [0.43 .. 1.61] or [2.42]
THEN Index will increase CF= 0.1975(78/170)

IF USD in [-2.32 .. -1.94] or [0.68 .. 0.94] or [-0.49 .. -0.18] or [0.24 .. 0.37] or
          [-4.34 .. -3.64] and
    DM in [-13.87 .. -4.24] or [-0.94 .. -0.19] or [0.87 .. 1.17]  or [-36.67] and
    RG in [-1.93 .. -1.69] or [1.22 .. 1.70] or [-4.0 .. -2.24] or [-0.86 .. -0.40] or
          [0.39 .. 0.93] or [2.69] and
    SG in [-6.70 .. -2.33] or [-1.05 .. -0.93] or [1.21 .. 1.47] or [4.79] and
    GB in [-6.51 .. 6.93] or [7.17 .. 13.86] or [-19.27 .. -17.51] and
    CD in [-0.24 .. 5.30]
THEN Index will not change CF= 0.2684(106/204)

IF USD in [-0.45] and
    DM in [-0.33 .. -0.23] and
    RG in [-0.45 .. -0.40] and
    SG in [-1.05 .. -0.93] and
    GB in [0.0] and
    CD in [0.0]
THEN Index will not change CF= 0.0025( 1/ 2)

IF USD in [-3.64 .. 2.93] or [3.49] and
    DM in [-17.70 .. 16.60] and
    RG in [-1.69 .. 0.93] or [1.42] or [1.70 .. 1.83] or [-2.24] or [4.17] and
    SG in [-6.70 .. 3.69] and
    GB in [-29.94 .. -19.63] or [33.47] and
    CD in [-0.74 .. 2.87]
THEN Index will decrease CF= 0.3165(125/245)
```