ARTIFICIAL NEURAL NETWORK BASED DECISIVE PREDICTION MODELS ON HIGH FREQUENCY FINANCIAL DATA

A DOCTOR OF PHILOSOPHY THESIS

in

Modeling and Design of Engineering Systems (MODES)

(Main Field of Study: Industrial Engineering)

Atılım University

by

ADİL GÜRSEL KARAÇOR

JANUARY 2017

ARTIFICIAL NEURAL NETWORK BASED DECISIVE PREDICTION MODELS ON HIGH FREQUENCY FINANCIAL DATA

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

ADİL GÜRSEL KARAÇOR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

MODELING AND DESIGN OF ENGINEERING SYSTEMS (MODES)

PhD PROGRAM

(MAIN FIELD of STUDY: INDUSTRIAL ENGINEERING)

JANUARY 2017

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. İbrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

_____

Prof. Dr. Abdulkadir Erden

Program Chair

This is to certify that we have read the thesis "Artificial Neural Network Based Decisive Prediction Models on High Frequency Financial Data" submitted by "Adil Gürsel Karaçor" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Assoc. Prof. Dr. T. Erman Erkan

Supervisor

Examining Committee Members

Prof. Dr. Serkan Eryılmaz _____

Assoc. Prof. Dr. Diyar Akay _____

Assoc. Prof. Dr. Fatih Emre Boran _____

Assoc. Prof. Dr. T.Erman Erkan _____

Asst. Prof. Dr. Uğur Baç _____

Date: 19.01.2017

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Adil Gürsel Karaçor

Signature:

ABSTRACT

ARTIFICIAL NEURAL NETWORK BASED DECISIVE PREDICTION MODELS ON HIGH FREQUENCY FINANCIAL DATA

Karaçor, Adil Gürsel

PhD., Modeling and Design of Engineering Systems (Industrial Engineering)

Supervisor: Assoc.Prof.Dr. Turan Erman Erkan

January 2017, 88 pages

High frequency financial data are somewhat hard to model or predict, if not totally impossible, as stochastic processes and many other random factors are involved. In this thesis; a novel Artificial Intelligence model is designed and developed for financial time series prediction and decision making. Possibility to enhance prediction accuracy for foreign exchange rates is investigated in two ways: first applying an outside the box approach by bringing about methodology and techniques to facilitate the use of predictive models in engineering design to model price graphs by exploiting their visual properties together with principles of chaos theory, and secondly employing the most efficient methods to detect patterns to classify the direction of movement. The approach that exploits the visual properties of price graphs makes use of density regions along with high and low values describing the shape just as in Machine Vision. Mainly Artificial Neural Networks are used in modeling. However, other state-of-the-art methods; Extreme Gradient Boosting and Support Vector Machine are too used for comparison. The designed system is also software coded as a real-time trading robot. Comparable prediction results and profits are achieved in tests and simulations.

Keywords: Predictability, Artificial Neural Network, quantitative analysis, real-time trading robot, foreign exchange currency

ÖZ

YÜKSEK FREKANSLI MALİ PİYASA VERİLERİ ÜZERİNDE YAPAY SİNİR AĞI TABANLI KARAR VERİCİ TAHMİN MODELLERİ

Karaçor, Adil Gürsel

Doktora, Modeling and Design of Engineering Systems (Endüstri Mühendisliği)

Tez Yöneticisi: Doç. Dr. Turan Erman Erkan

Ocak 2017, 88 sayfa

Yüksek frekanslı mali piyasa verilerinin tahmin edilmeleri ve modellenmeleri, tamamen imkansız olmasa da, stokastik prosesler ve çok sayıda rastgele faktör araya girdiğinden, çok zordur. Bu çalışmada; mali zaman serisi tahmini ve karar alınmasına yardımcı, yeni bir Yapay Zeka modeli tasarlanmış ve geliştirilmiştir. Döviz çiftlerinin tahmin başarı yüzdelerinin artırılması iki şekilde irdelenmiştir: ilk olarak mühendislik tasarımlarıyla tahmin modellerinin ortaya çıkmasını sağlayacak metodolojinin kullanımı ile, kaos teorisi ve fiyat grafiklerinin görsel özelliklerinin devreye sokulması, ikinci olarak da en güncel ve güçlü yöntemlerin kullanılması. Görsel özellikleri kullanan yaklaşımda, aynı Yapay Görmedeki gibi, yoğunluk bölgeleri ile birlikte şekli tanımlayan yüksek ve düşük fiyat değerlerinden faydalanılmaktadır. Modellemede esas olarak Yapay Sinir Ağları uygulanmakta olup; diğer popüler yöntemlerden, İleri Gradyant Artırımı ve Destek Vektör Makinesi de karşılaştırma amaçlı olarak kullanılmıştır. Tasarlanan system ayrıca yazılımla gerçek zamanlı alım-satım robotu olarak da kodlanmıştır. Testler ve simulasyonlarda tatmin edici sonuçlar elde edilmiştir.

Anahtar Kelimeler: Tahmin edilebilirlik, Yapay Sinir Ağları, teknik analiz, gerçek zamanlı alım-satım robotu, döviz

To My Family

ACKNOWLEDGMENTS

I would like to extend my sincere appreciation and special thanks to my supervisor Assoc.Prof.Dr. Turan Erman Erkan for his guidance and support throughout the research. Additional thanks go to Assoc.Prof.Dr. Diyar Akay for his helpful inputs and to Assist.Prof.Dr. Uğur Baç for his valuable advices. The technical assistance of Çağatay Öztürk is also greatly appreciated. The last but not the least, I would like to express my gratitude to my wife Özlem for her continuous support and patience during the period of realization of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	-	Artificial Neural Network
AUC	-	Area Under Curve
FOREX	-	Foreign Exchange
MLP	-	Multi Layer Perceptron
MSE	-	Mean Squared Error
NMSE	-	Normalized Mean Squared Error
RBF	-	Radial Basis Function
ROC	-	Receiver Operating Characteristic
SVM	-	Support Vector Machine
XGB	-	Extreme Gradient Boosting

CHAPTER I

INTRODUCTION

1.1 Thesis Outline

This thesis is organized in 7 chapters. A brief introduction is given about the principles of chaos, financial time series, and their resemblance, in Chapter I. Statement of the problem and some issues regarding time series prediction are given in Chapter II. In Chapter III, three state-of-the-art methods used in this thesis are explained in detail. In Chapter IV a profitability analysis among financial instruments is done. Being found that the most predictable group of instruments, a predictability enhancement work on foreign exchange currencies is carried out in Chapter V. In Chapter VI, combining all the knowledge accumulated in the thesis, a real time trading robot i.e. profitable trading system is attempted. Finally concluding remarks are given in Chapter VII.

1.2 Non-Linear Dynamics and Chaos

Price movements of financial instruments, whether they are random, chaotic, or of any other stochastic process, are definitely non-linear. Prior to going further into predictability and profitability analysis, it should be a good idea to define some terminology about non-linear dynamics.

First of all, a dynamic system can be defined as a deterministic mathematical prescription for evolving the state of a system forward in time where time may be either a continuous or discrete variable [1]. A *dissipative* system is a dynamic system, in which the phase space volume contracts along a trajectory.

The term non-linear is defined as the opposite of linear. In a linear system the variables appear in the first degree, they are not multiplied by one another; they are only multiplied by constants, and are combined only by addition or subtraction, while non-linear systems can occur in various forms such as division, multiplication and powers. The vast majority of systems we come across in real life are actually non-linear.

Collection of all possible states of a dynamic system is called the phase space. A map is a discrete function in the phase space that gives the next state of the system as a function of its current states. In a similar fashion, a flow is a continuous function that describes the time derivative of the state variables as a function of the present state values.

A state $\mathbf{x}^*$ is an equilibrium point of the system if once x(t) is equal to $\mathbf{x}^*$, it remains equal to $\mathbf{x}^*$ for all future times. When the system is exactly on the periodic orbit it will move on it forever and pass through the same points periodically. Both flows and maps can have equilibrium points and periodic orbits. Equilibrium points and periodic orbits can be either

stable (even if the system slightly deviates from the equilibrium point or periodic orbit, it returns), unstable (if the system slightly deviates from the equilibrium point or periodic orbit, it does not come back) or a saddle (if the system slightly deviates from the equilibrium point or periodic orbit in some direction, it returns; if it deviates in some other direction it diverges; such equilibrium points or periodic orbits are also considered unstable).

As non-linear financial dynamics resemble to chaos, let us explain some more terminology on chaotic systems.

Until the last few decades of 20th century, the term chaos only meant disordered formless matter, complete disorder, utter confusion, randomness, or uncertainty. However, since then it has gained a special scientific meaning, and started being used designate deterministic chaos which is a specific type of behavior that can be observed in non-linear dynamic systems and can be expressed by a set of discrete-time or continuous-time equations. Although its mathematical description is deterministic, a chaotic system has unpredictability in the long run. Another interesting point about chaotic dynamics is that the phase trajectories strongly depend on initial conditions. In addition, a dissipative chaotic system neither converges to a stable point or a stable periodic orbit, nor diverges to infinity; instead it wanders around in a fractal region. Such kind of chaotic behavior can be observed in various systems in different areas ranging from chemistry to electronics. However, it is not usually desired for a non-linear system to exhibit chaotic behavior, this is where control comes into the picture: due to its unpredictable nature chaos can give rise to problems. On the other hand, by taking advantage of certain properties of chaotic behavior, much can be achieved with little control effort. The founders of the OGY method; Ott, Grebogi and Yorke took advantage of the fact that there are quite a number of unstable limit cycles and equilibrium points within the strange attractor and the system does come close enough to one of these points of choice at certain times.

As the term strange attractor is mentioned, it needs to be explained. A dissipative chaotic system is such a system that neither converges to a stable point or a stable periodic orbit, nor diverges to infinity; instead it wanders around in a fractal region. Such a system contains many unstable or saddle type equilibrium points and so called strange attractors. A strange attractor actually consists of an infinitely long single trajectory that does not cross itself, and does not repeat itself periodically. Yet any trajectory attracted to a strange attractor sticks with it forever, see Lorenz attractor example in Figure 1.1.

The global stability of a linear, time invariant system is determined by the eigenvalues of the system, which all have to be negative. Similarly, in a non-linear system the Lyapunov exponent indicates the exponential divergence or convergence rate (positive exponents correspond to divergence and negative ones to convergence) of the system when it slightly deviates from the equilibrium point or trajectory.

Chaotic systems have at least one positive Lyapunov exponent, which can be thought of as the source of the sensitive dependence on initial conditions, because even small deviations are amplified and nearby trajectories diverge from one another due to the positive Lyapunov exponent.

In a dissipative system, trajectories starting from a finite phase volume tend to occupy smaller and smaller phase volume as time evolves such that in the limit as time goes to infinity the phase volume they occupy becomes zero. It should be noted that only dissipative chaotic

systems can have strange attractors with zero phase volume, but due to their fractal structure, these strange attractors are distributed over a finite phase volume.

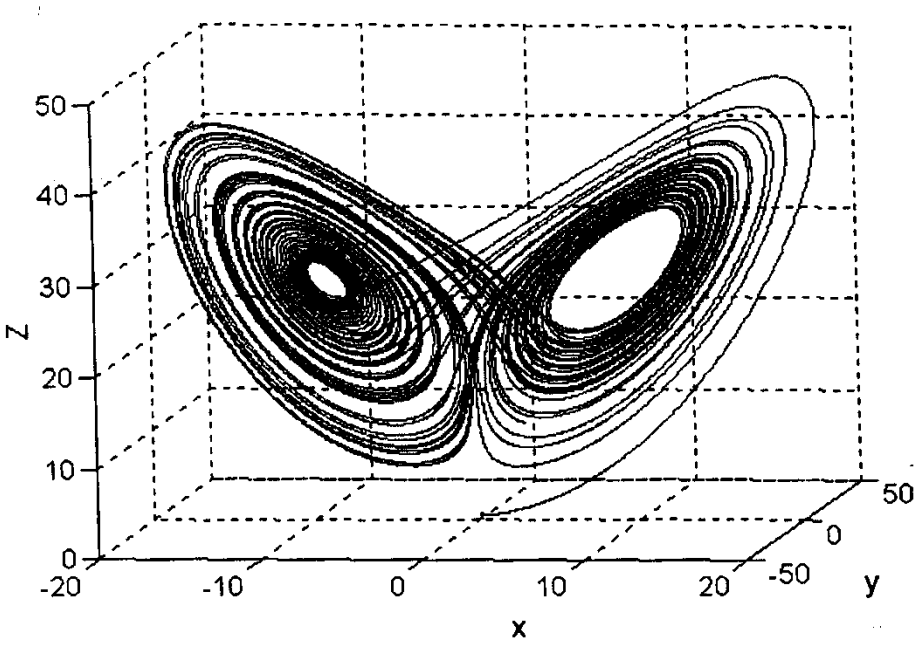


Figure 1.1 Lorenz Attractor [2]

Embedded into the strange attractor of a dissipative chaotic system there usually are many (sometimes infinite) unstable (saddle-type) equilibrium points and/or periodic orbits. The chaotic system, once it enters the strange attractor, keeps on moving in it and passes from time to time through a close neighborhood of these embedded saddle-like equilibrium points and/or periodic orbits [2].

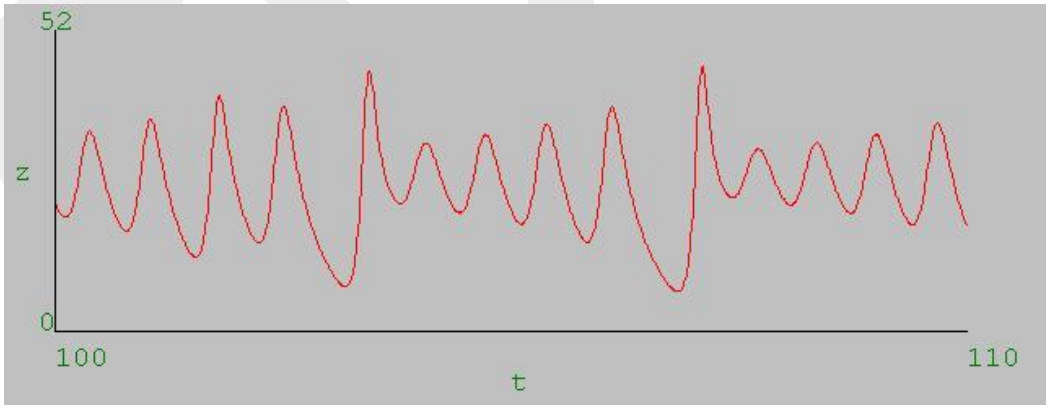


Figure 1.2 z-state output sample of the Lorenz System in chaotic fashion [2]

Movements of financial instruments resemble a lot to chaos, i.e. movements about and in between *saddle points* (or lines in time series domain - see Figure 1.2). As a matter of fact, financial markets are non-linear in nature and exhibit chaotic-like behavior, hence the methods and tools that aim to model them should also be non-linear, and be able to recognize *strange* fluctuations. Principles of chaos should also be taken into account.

1.3 Financial Markets, Data Representation and Instruments

1.3.1 Financial Markets

Huge amount of money flows into the markets all around the world for the trade of various financial instruments such as stocks, commodities, foreign exchange (forex), futures, and so on every day. For example, according to experts and professionals, average daily turnover in forex markets alone is in excess of 5 trillion US dollars [3]. Investment and trading decisions, whether they tend to be long term or short term, are mainly based on predicting the future movements of the financial instrument(s) in question. There has been an ongoing debate among researchers on whether financial markets are predictable or not for a long time. Some think that financial market movements are nothing but random walk, and some findings support that claim: for example, VIX futures prices were found to be unpredictable by Konstantinidi & Skiadopoulos [4]. Furthermore, some researchers even claimed that none of the conventional predictive models proposed in the literature on stock prediction seems capable of systematically predicting stock returns in long range of time horizons, and speculators do not earn significant profits in commodity and interest rate futures markets in aggregate [5-7]. On the other hand, many researchers disagree with this random walk approach. Some of them claim speculators can gain profits on commodity and currency futures [8-12]. Some others believe that financial instruments, commodities in particular, are predictable, at least to a certain extent [13, 14]. The debate has not been settled yet; however, it is fair to say that financial instruments are definitely hard to model or predict, if not totally unpredictable. Obviously, it would be valuable information for the investor if he or she could know in advance which way an instrument would go. In this thesis an investigation is carried out in order to give the investor a huge advantage by trying to answer the following questions: would a certain forex rate be likely to go up or down in the following hours, and what kind of a trading strategy would be profitable? Technical (quantitative) analysis i.e. past price values of the instruments is solely used, and fundamental analysis is not in the scope of this study.

1.3.2 Financial Time Series

Financial data are typically represented by bar charts with Open (O), High (H), Low (L), and Close (C) denoting corresponding price values, see Figure 1.3. Data graphs are made up of series of OHLC bars [15-18]. A sample financial data graph is given in Figure 1.4.

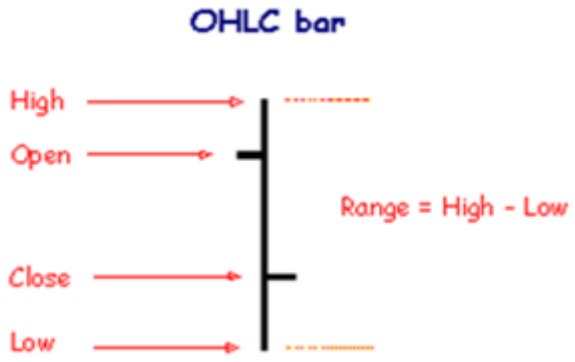


Figure 1.3 Financial data representation

1.3.3 Financial Instruments Used

Three types of instruments are considered in this stage: stocks, currencies (forex), and commodities. Four individual instruments are chosen to represent each type of instrument; namely Australian Dollar against US Dollar (AUDUSD); Euro against Canadian Dollar (EURCAD); Euro against US Dollar (EURUSD); and US Dollar against Japanese Yen (USDJPY), representing forex, BRENT crude oil; LIGHT crude oil; silver (XAGUSD); and gold (XAUUSD), representing commodities, and finally Amazon.com Inc (AMZN); Cisco Systems Inc. (CSCO); General Motors (GM); and Coca Cola Company (KO), representing stocks. The selected financial instruments are very commonly traded in the market, and often mentioned in the literature as well [19-27].

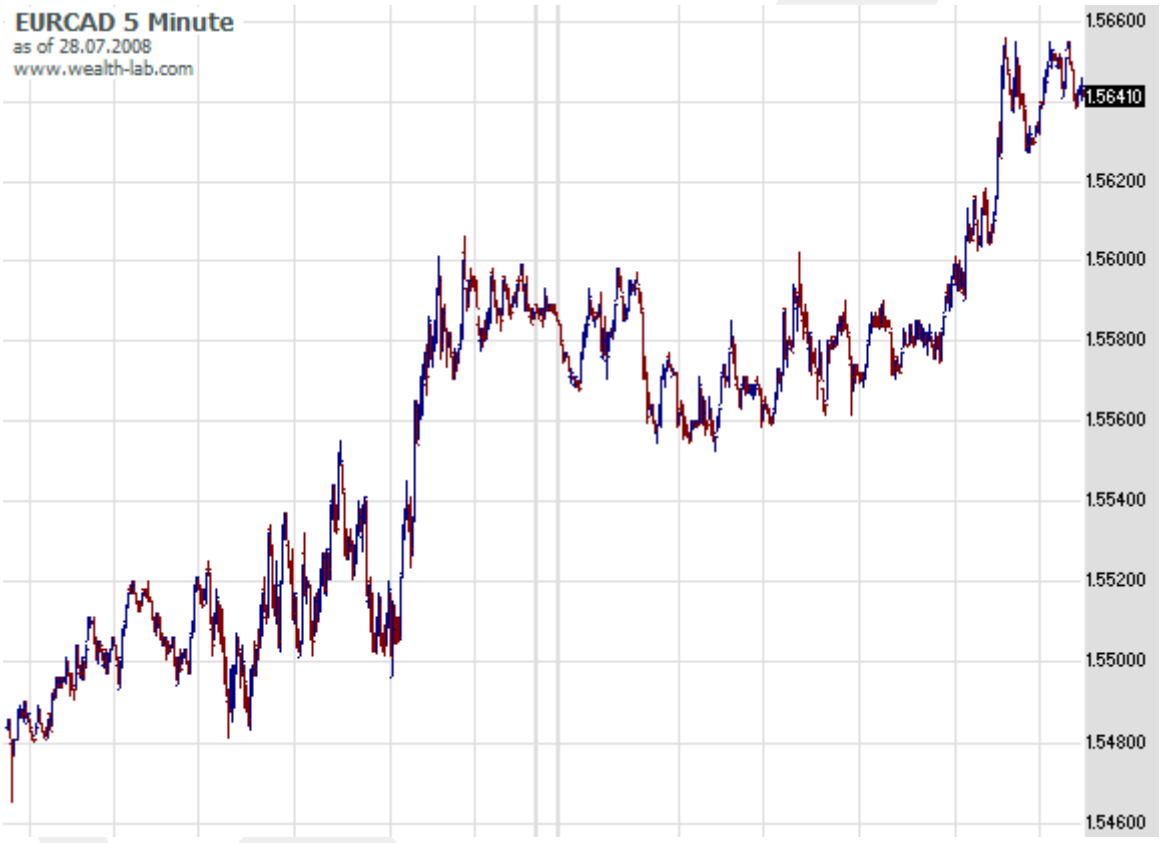


Figure 1.4 Sample financial time series data graph

CHAPTER II

STATEMENT OF THE PROBLEM

2.1 Modeling and Prediction

Considering the non-linearity of financial price movements, it would be wise to use non-linear modeling tools. Usage of Artificial Neural Networks (ANNs) is quite popular for modeling, prediction, and decision making over financial data, and ANNs are regarded as an excellent tool for the purpose [28-34]. In this thesis ANNs are used as a tool for measuring and comparing predictability. ANNs using RBF (ANN-RBFs) are also quite popular and have good performance in financial time series prediction. As per other methods and tools used in this thesis; similar to ANNs, usage of Support Vector Machines (SVMs) in financial markets is quite common [35-38]. On the other hand usage of eXtreme Gradient Boosting (XGB) method is very rare [39].

In the predictability analysis stage, ANN-RBFs are trained to predict three different types of instruments: currency pairs, commodities, and stocks. As mentioned above there are many research papers on predictability of financial instruments; however, predictability comparison of these instruments in literature is very rare. In fact the only study we came across was the one comparing real estate returns with stock returns [40]. The initial stage of this study is to determine which financial instrument at which frequency is more predictable than the others first of all. After that, a further effort will be made to improve predicting performances of the most predictable financial instruments at right frequencies, in order to build a profitable trading algorithm by exploiting all this knowledge.

2.2 Data Collection and Organization

The data are obtained from public datasets and the author's private data accounts. The data used in predictability enhancement stage is a subset of the data used in predictability analysis stage yet with a different processing and feature engineering, the data used in profitability stage is also another subset, which all is explained in separate subsections where necessary.

The data used in the predictability analysis stage consist of 1-minute (1m), 5-minute (5m), 15-minute (15m), 30-minute (30m), 1-hour (1h), and 4-hour (4h) intraday frequencies covering the period between 1996 and 2016 with approximately 65000 data points for each instrument and frequency. For forex data only the period between 2001 and 2016 is used because being one of the main currencies, Euro was not effectively traded before 2000's. This corresponds to a few months for 1-minute data, and 20 years for 4-hour data. The data were taken from authors' private data accounts. High frequency data are preferred, because in lower frequencies more and more non-technical fundamental factors might start to affect price movements.

One advantage of working with financial data is having good quality data most of the times, as trading transactions are based on these. Therefore our data is of very good quality as well in general. There are very few empty values and outliers – an example case is shown in Table 2.1. Financial data is time series, and it should be dealt with accordingly. The missing (NULL) values are filled in as average of 2 prior and 2 posterior values in the time series. The outlier values i.e. the values that fall outside a typical moving average range by a threshold are also updated just the same way as the missing values are filled. The processed version (missing values filled, outlier eliminated) of the sample data is given in Table 2.2.

Table 2.1 Sample of empty values and outliers in the data

	A	B	C	D	E	F	G
1	EURCAD	Time	Open	High	Low	Close	Target (Next High)
2	23.05.2008	00:05:00	1.5484	1.5486	1.5483	1.5485	1.548599958
3	23.05.2008	00:10:00	1.5486	1.5486	1.5481	1.5481	1.548099995
4	23.05.2008	00:15:00	1.548	1.5481	1.5465	1.5477	1.548099995
5	23.05.2008	00:20:00	1.5478	1.5481	1.5477	1.5481	1.548099995
6	23.05.2008	00:25:00	1.5481	1.5481	1.5481	1.5481	1.548799992
7	23.05.2008	00:30:00	1.5481	1.5488	1.5481	1.5488	NULL
8	23.05.2008	00:35:00	NULL	NULL	NULL	NULL	NULL
9	23.05.2008	00:40:00	1.5488	1.5488	1.5486	1.5487	1.549000025
10	23.05.2008	00:45:00	1.5488	1.549	1.5485	1.5489	1.549000025
11	23.05.2008	00:50:00	1.5488	1.549	1.5486	1.5486	1.548699975
12	23.05.2008	00:55:00	1.5485	1.5487	1.5485	1.5487	1.548599958
13	23.05.2008	01:00:00	1.5486	1.5486	1.5483	1.5484	1.558300028
14	23.05.2008	01:05:00	1.5483	1.5583	1.5481	1.5481	1.548099995
15	23.05.2008	01:10:00	1.5481	1.5481	1.548	1.5481	1.548499942
16	23.05.2008	01:15:00	1.5482	1.5485	1.5482	1.5484	1.548900008
17	23.05.2008	01:20:00	1.5485	1.5489	1.5485	1.5488	1.548799992
18	23.05.2008	01:25:00	1.5487	1.5488	1.5485	1.5485	1.548499942
19	23.05.2008	01:30:00	1.5485	1.5485	1.5483	1.5483	1.548300028
20	23.05.2008	01:35:00	1.5483	1.5483	1.5481	1.5481	1.548400044
21	23.05.2008	01:40:00	1.5482	1.5484	1.5482	1.5482	1.548599958
22	23.05.2008	01:45:00	1.5482	1.5486	1.5481	1.5486	1.548699975
23	23.05.2008	01:50:00	1.5487	1.5487	1.5483	1.5483	1.549200058

2.3 Issues about Financial Time Series Data Analysis

There exists a trap in most of financial time series data to be avoided, especially in forex where only small perturbations occur in unit time. Let us consider the previous example where the missing data is filled and the outlier is updated. The target to be predicted in that data is the next high value i.e. the highest value that price will reach in the next 5 minutes. The correlations are given in Table 2.3. Everything seems perfectly correlated here, even with the target; however, this is not useful information at all. The changes in the forex price are so small in 5 minutes that all values are highly correlated with each other in terms of magnitude [41].

To define the problem more clearly, if the target is formulated differently e.g. next high but percentage change from the previous close, we get a different correlation matrix as in Table 2.4. This time input parts are still highly correlated with each other, yet they fail to explain the target.

Table 2.2 Missing values filled and outlier value updated

	A	B	C	D	E	F	G
1	EURCAD	Time	Open	High	Low	Close	Target (Next High)
2	23.05.2008	00:05:00	1.5484	1.5486	1.5483	1.5485	1.548599958
3	23.05.2008	00:10:00	1.5486	1.5486	1.5481	1.5481	1.548099995
4	23.05.2008	00:15:00	1.548	1.5481	1.5465	1.5477	1.548099995
5	23.05.2008	00:20:00	1.5478	1.5481	1.5477	1.5481	1.548099995
6	23.05.2008	00:25:00	1.5481	1.5481	1.5481	1.5481	1.548799992
7	23.05.2008	00:30:00	1.5481	1.5488	1.5481	1.5488	1.548675001
8	23.05.2008	00:35:00	1.54845	1.548675	1.548325	1.548625	1.548799992
9	23.05.2008	00:40:00	1.5488	1.5488	1.5486	1.5487	1.549000025
10	23.05.2008	00:45:00	1.5488	1.549	1.5485	1.5489	1.549000025
11	23.05.2008	00:50:00	1.5488	1.549	1.5486	1.5486	1.548699975
12	23.05.2008	00:55:00	1.5485	1.5487	1.5485	1.5487	1.548599958
13	23.05.2008	01:00:00	1.5486	1.5486	1.5483	1.5484	1.548474967
14	23.05.2008	01:05:00	1.5483	1.548475	1.5481	1.5481	1.548099995
15	23.05.2008	01:10:00	1.5481	1.5481	1.548	1.5481	1.548499942
16	23.05.2008	01:15:00	1.5482	1.5485	1.5482	1.5484	1.548900008
17	23.05.2008	01:20:00	1.5485	1.5489	1.5485	1.5488	1.548799992
18	23.05.2008	01:25:00	1.5487	1.5488	1.5485	1.5485	1.548499942
19	23.05.2008	01:30:00	1.5485	1.5485	1.5483	1.5483	1.548300028
20	23.05.2008	01:35:00	1.5483	1.5483	1.5481	1.5481	1.548400044
21	23.05.2008	01:40:00	1.5482	1.5484	1.5482	1.5482	1.548599958
22	23.05.2008	01:45:00	1.5482	1.5486	1.5481	1.5486	1.548699975
23	23.05.2008	01:50:00	1.5487	1.5487	1.5483	1.5483	1.549200058

Table 2.3 Correlations among variables and target

	A	B	C	D	E	F
1		*Open*	*High*	*Low*	*Close*	*Target (Next High)*
2	Open	1				
3	High	0.999704	1			
4	Low	0.999687	0.999581	1		
5	Close	0.999398	0.999718	0.999721	1	
6	Target (Next High)	0.999111	0.999498	0.999326	0.999664	1
7						

Table 2.4 Correlation matrix when Target differently formulated

	A	B	C	D	E	F
1		*Open*	*High*	*Low*	*Close*	*Target (Next High - percent from previous close)*
2	Open	1				
3	High	0.999704	1			
4	Low	0.999687	0.999581	1		
5	Close	0.999398	0.999718	0.999721	1	
6	Target (Next High - percent from previous close)	-0.0054	-0.00282	-0.00956	-0.0073	1
7						

In order to get around this problem we need feature engineering, which is basically done by creating new variables from the originals, and by eliminating the unnecessary variables to better define the target. As an example we can normalize the original Open, High, and Low values by dividing them by the close value and call them O, H, and L respectively. Then by adding some more normalized past Open, High, Low, and Close values we get a new data set as shown in Table 2.5, and a corresponding correlation matrix as shown in Table 2.6. As can

be seen in Table 2.6, the target is much better explained by the new derivative variables this time. How the feature engineering is done for our predictive models is shown in more detail, in the relevant chapters and sections ahead.

Table 2.5 Derivative features added

	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	High	Low	Close	O	H	L	O-1	H-1	L-1	C-1	H-3	L-3	H-5	L-5	Target
2	1.5486	1.5483	1.5485	0.999935	1.000065	0.999871	1	1	1	1	1	1	1	1	0.006459
3	1.5486	1.5481	1.5481	1.000323	1.000323	1	1.000194	1.000323	1.000129	1.000258	1	1	1	1	0
4	1.5481	1.5465	1.5477	1.000194	1.000258	0.999225	1.000581	1.000581	1.000258	1.000258	1	1	1	1	0.025841
5	1.5481	1.5477	1.5481	0.999806	1	0.999742	0.999935	1	0.998966	0.999742	1.000323	1.000129	1	1	0
6	1.5481	1.5481	1.5481	1	1	1	0.999806	1	0.999742	1	1.000323	1	1	1	0.045217
7	1.5488	1.5481	1.5488	0.999548	1	0.999548	0.999548	0.999548	0.999548	0.999548	0.999548	0.998515	0.999871	0.999677	-0.00807
8	1.548675	1.548325	1.548625	0.999887	1.000032	0.999806	0.999661	1.000113	0.999661	1.000113	0.999661	0.999403	0.999984	0.999661	0.0113
9	1.5488	1.5486	1.5487	1.000065	1.000065	0.999935	0.999839	0.999984	0.999758	0.999952	0.999613	0.999613	0.999613	0.998579	0.019374
10	1.549	1.5485	1.5489	0.999935	1.000065	0.999742	0.999935	0.999935	0.999806	0.999871	0.999935	0.999483	0.999483	0.999225	0.006457
11	1.549	1.5486	1.5486	1.000129	1.000258	1	1.000129	1.000258	0.999935	1.000194	1.000048	0.999822	0.999677	0.999677	0.006459
12	1.5487	1.5485	1.5487	0.999871	1	0.999871	1.000065	1.000194	0.999935	0.999935	1.000065	0.999935	1.000065	0.999613	-0.00646
13	1.5486	1.5483	1.5484	1.000129	1.000129	0.999935	1.000065	1.000194	1.000065	1.000194	1.000387	1.000065	1.000178	0.999952	0.004839
14	1.548475	1.5481	1.5481	1.000129	1.000242	1	1.000323	1.000323	1.000129	1.000194	1.000581	1.000323	1.000452	1.000323	0
15	1.5481	1.548	1.5481	1	1	0.999935	1.000129	1.000242	1	1	1.000388	1.000258	1.000581	1.000258	0.025835
16	1.5485	1.5482	1.5484	0.999871	1.000065	0.999871	0.999806	0.999806	0.999742	0.999806	1.000129	0.999935	1.000387	1.000129	0.032289
17	1.5489	1.5485	1.5488	0.999806	1.000065	0.999806	0.999613	0.999806	0.999613	0.999742	0.99979	0.999548	0.999935	0.999806	0
18	1.5488	1.5485	1.5485	1.000129	1.000194	1	1	1.000258	1	1.000194	0.999742	0.999677	1.000065	0.999871	0
19	1.5485	1.5483	1.5483	1.000129	1.000129	1	1.000258	1.000323	1.000129	1.000129	1.000129	0.999935	1.000113	0.999871	0
20	1.5483	1.5481	1.5481	1.000129	1.000129	1	1.000258	1.000258	1.000129	1.000129	1.000517	1.000258	1	0.999935	0.019382
21	1.5484	1.5482	1.5482	1	1.000129	1	1.000065	1.000065	0.999935	0.999935	1.000388	1.000194	1.000194	1	0.025833
22	1.5486	1.5481	1.5486	0.999742	1	0.999677	0.999742	0.999871	0.999742	0.999742	0.999935	0.999806	1.000194	0.999935	0.006459
23	1.5487	1.5483	1.5483	1.000258	1.000258	1	0.999935	1.000194	0.999871	1.000194	1	0.999871	1.000323	1.000129	0.06813

Table 2.6 Correlation matrix with extra derivative variables

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		*Open*	*High*	*Low*	*Close*	*O*	*H*	*L*	*O-1*	*H-1*	*L-1*	*C-1*	*H-3*	*L-3*	*H-5*
2	Open	1													
3	High	0.999704	1												
4	Low	0.999687	0.999581	1											
5	Close	0.999398	0.999718	0.999721	1										
6	O	0.022107	0.004362	0.003778	-0.0126	1									
7	H	0.023958	0.022944	0.005201	-0.00081	0.713903	1								
8	L	0.014462	-0.00354	0.014051	-0.00958	0.692808	0.254298	1							
9	O-1	0.005008	-0.00699	-0.00716	-0.01811	0.666293	0.4684	0.463429	1						
10	H-1	0.015769	0.003757	-0.00091	-0.01173	0.792429	0.652016	0.457576	0.860239	1					
11	L-1	0.010695	-0.00566	-0.00172	-0.01692	0.795574	0.473874	0.643001	0.851394	0.756754	1				
12	C-1	0.018671	0.001552	0.0008	-0.01485	0.965807	0.690314	0.661968	0.688712	0.821479	0.822291	1			
13	H-3	-0.00332	-0.01145	-0.01383	-0.02087	0.505421	0.396272	0.297477	0.726801	0.663807	0.596541	0.52406	1		
14	L-3	-0.00702	-0.01754	-0.01517	-0.02473	0.510091	0.302524	0.404121	0.729091	0.604704	0.664645	0.529282	0.889576	1	
15	H-5	-0.01304	-0.01977	-0.02138	-0.02696	0.400764	0.302698	0.235669	0.578074	0.518042	0.476746	0.414757	0.795784	0.759145	1
16	L-5	-0.01639	-0.02504	-0.02304	-0.03065	0.410551	0.235449	0.321537	0.584976	0.482832	0.530455	0.424415	0.766548	0.799104	0.928077
17	Target	-0.00528	-0.0027	-0.00944	-0.00718	0.054749	0.188383	-0.09565	0.046059	0.129203	-0.01824	0.054926	0.071824	-0.01575	0.051868

CHAPTER III

STATE-OF-THE-ART METHODS AND TOOLS USED IN MODELING

3.1 Artificial Neural Networks

First of all, whenever the term *neural network* is used within this thesis, it actually refers to an artificial neural network. The concepts that belong to artificial neural networks will be inspected in this section. Artificial Neural Network is the structure behind the *deep learning* concept which is extremely popular nowadays.

Some top of the Artificial Intelligence (AI) and Machine Learning (ML) tools and methods are employed in this thesis. ANNs have gained much popularity in the recent years, mainly due to their success in recognition and classification tasks via deep learning. Therefore ANNs are chosen as the main modeling tool in this thesis. Similar to ANNs, SVMs are accepted as one of the best tools for classification and hence included in our work. Also a third modeling tool is chosen, a decision tree based method namely XGB, that has become one of the most popular tools for classification and prediction in data science community in the last few years.

3.1.1 Basic Concepts of Neural Networks

In fact, biological neural networks are much more complicated than the mathematical models we use. Biological neural networks consist of *biological neurons* while artificial neural networks consist of so-called mathematical neurons (nodes), which contain activation functions. A biological neuron is shown in Figure 3.1.

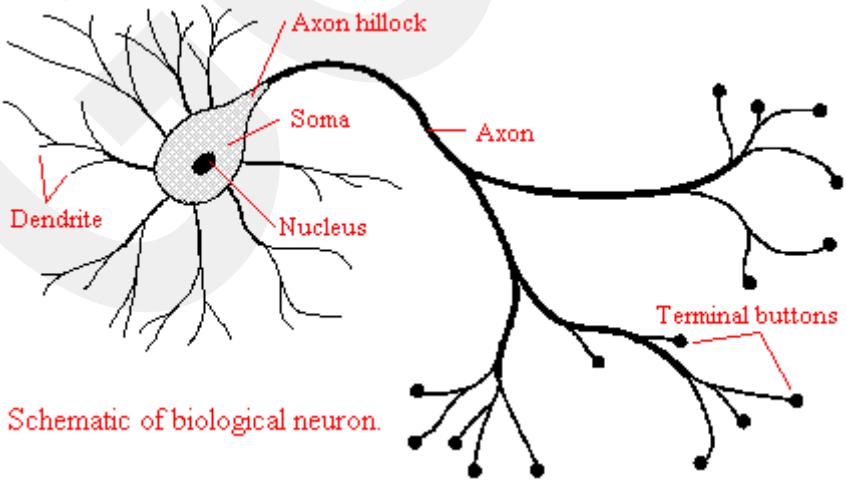


Figure 3.1 A biological neuron [2]

The functioning of biological neural networks can be explained as follows: each neuron is a unit that uses biochemical reactions to receive, process and transmit information. A neuron's

dendrites are connected to a thousand neighboring neurons. When one of those neurons fires, a charge (positive or negative) is received by one of the dendrites. The strengths of all the received charges are added together, and then the aggregate input is passed to the soma. The soma and the enclosed nucleus do not play a significant role in the processing of incoming and outgoing data. Their primary function is to perform the continuous maintenance required to keep the neuron functional. The part of the soma that does concern itself with the signal is the axon hillock. If the aggregate input is greater than the axon hillock's threshold value, the neuron fires, and an output signal is transmitted down the axon. The strength of the output is constant, regardless of whether the input was just above the threshold, or a hundred times as great. The output strength is unaffected by the many divisions in the axon; it reaches each terminal button with the same intensity it had at the axon hillock. This uniformity is critical in an analogue device such as a brain where small errors can snowball, and where error correction is more difficult than in a digital system. Each terminal button is connected to other neurons across a small gap called a synapse. The physical and neuro-chemical characteristics of each synapse determine the strength and polarity of the new input signal [42].

On the other hand, an artificial neuron, as shown in Figure 3.2, is designed to simulate a real neuron with inputs entering the node and then multiplied by corresponding weights (w1, w2, ... , wn) to indicate the strength of the *synapse*. Equation for a single node is given in Equation 3.1.

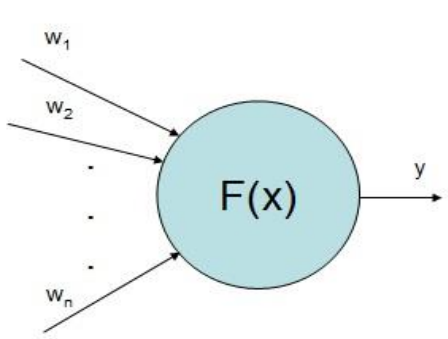


Figure 3.2 Artificial neuron

$$y = F\left(\sum_j \sum_i (w_j\, x_i) + \theta\right) \tag{3.1}$$

Here; y is the output, w_j is the j^{th} weight connected to the node, x_i is the i^{th} input, F(x) is the activation function, and θ is the bias term i.e. a value between [0, 1]. A bias value allows us to shift the activation function to the left or right, which may be critical for successful learning.

F(x) is called *activation function*, and there are numerous activation functions. Generally some sort of threshold function is used as activation functions: a hard limiting threshold function (signum), or a linear or piece-wise-linear function, or a smoothly limiting threshold. Types of these functions are shown in Figure 3.3. Usually sigmoid or hyperbolic tangent functions are used for smoothly limiting thresholds. The sigmoid (S-shaped) is described as:

$$F(x) = \frac{1}{1+e^{-x}} \qquad \text{x: } (-\infty, \infty) \tag{3.2}$$

While hyperbolic tangent function is in the following form:

$$F(x) = \frac{e^{-x} - e^{x}}{e^{x} + e^{-x}} \qquad \text{x: } (-\infty, \infty) \tag{3.3}$$

And the Gaussian:

$$F(x) = e^{-\frac{(x-c)^2}{r}} \qquad \text{x: } (-\infty, \infty) \tag{3.4}$$

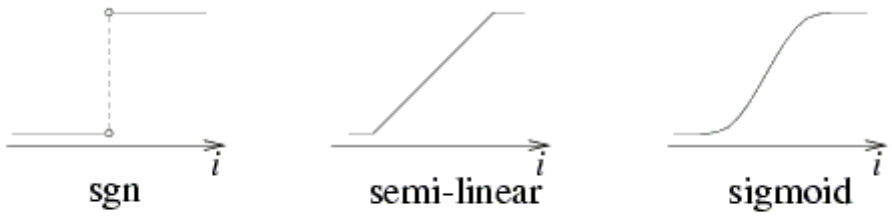


Figure 3.3 Various Activation Functions of a node

An artificial neural network is a network of many processors (*nodes*) each possibly having a small amount of local memory. The nodes have *activation functions*, which describe the output behavior of that node. Activation functions can be of various types such as sigmoid, hyperbolic tangent, Gaussian, etc. as explained above. Communication channels that usually carry numeric data, encoded by activation functions, connect the nodes. The nodes operate only on their local data and on the inputs they receive via the connections. A simple layout of a multilayer neural network is given in Figure 3.4.

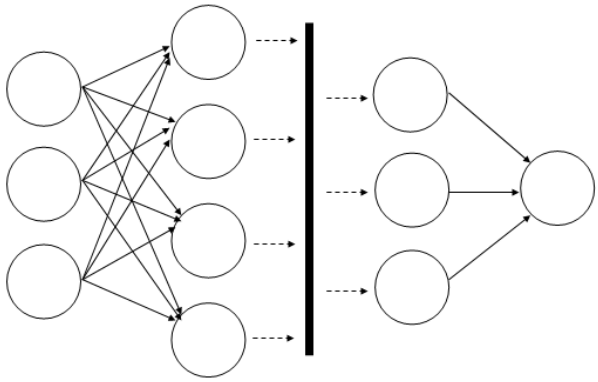


Figure 3.4 General structure of a multi-layer artificial neural network

Input Layer is the part, which provides external input to the network. Hidden Layer(s) is the part, which receives from the input layer or another hidden layer and provides inputs to the output layer or the next hidden layer. Output Layer receives inputs from a hidden layer and produces the output(s) of the network. Weight is a connection between two nodes with a value that is dynamically changed during a neural networks learning process [43].

Some neural networks are models of biological neural networks and some are not, but historically much of the inspiration comes from the desire to produce artificial systems capable of sophisticated, perhaps *intelligent* computations similar to those that the human

brain routinely performs. However, much is yet to be achieved, as an artificial neural network is, at least for the time being, no match for the human brain, which is a collection of billions of interconnected neurons within itself.

Most neural networks have some sort of a *training* rule whereby the weights of connections are adjusted on basis of data. In other words, neural networks *learn* from examples and exhibit some capability for generalization beyond the training data. There are two types of learning: *supervised* and *unsupervised*. In supervised learning the learning algorithm is provided with a set of inputs for the algorithm along with the corresponding correct outputs, and learning involves the algorithm comparing its current actual output with the correct or desired outputs, and modifies things accordingly. In contrast, unsupervised learning signifies a type of learning where the system is not told the *right answer*, i.e. it is not trained on pairs consisting of an input and the desired output. Instead, the system is given the input patterns and is left to find interesting patterns, regularities, or relations among them. As an example, a boy learning to skate can be classified as in an unsupervised learning situation while a student learning the multiplication table at school is learning in a supervised manner.

Neural networks normally have great potential for parallelism, since the computations of the components are largely independent of each other. In practice, neural networks are especially useful for classification and function approximation/mapping problems which are somewhat ambiguous and have to be tolerant to some imprecision, which have reasonable amount of training data available, but to which hard and fast rules (such as those that might be used in an expert system) cannot easily be applied. Almost any finite-dimensional vector function on a compact set can be approximated to arbitrary precision by neural networks if one has enough data and computing resources. Of course ANNs are not limited to regression, they can swiftly manage classification tasks too.

3.1.2 Areas of Application

In principle, neural networks can compute any computable function. They essentially are function approximators, pattern matchers, and categorizers. They do very little outside of these basic functions although these tasks can be employed in a wide variety of powerful and complex applications ranging from job satisfaction performance evaluation [44], solving capacitated P-median problem [45] to prediction of aircraft accident occurrence [46]. ANNs are trained in order that they learn a set of input-output data that represent usually a very complex or even undefined function. With sufficient number of hidden layers and neurons, they can model any given input-output relationship [47]. All nodes (artificial neurons) are interconnected, thus form a massive parallelism, and each connection has a weight that changes as the ANN is trained, and also each node has an activation function. There are numerous activation functions, ranging from simple linear functions to various nonlinear ones. The nonlinearity of activation functions enables the ANN to learn even the most complex patterns. A quite amazing example is as follows: neural networks have been put to use in tests at NASA's Dryden Flight Research Centre in Edwards, California using a modified F-15 aircraft. In this application a neural network was allowed to study normal flight operations. The neural network learned how a properly flying aircraft should behave. Then if the aircraft suffers some type of damage, the flight control system enables the neural net and allows the network to correct mismatches between data on the plane's airspeed, bearing, and the forces on its body versus what the network thinks the data should be if the plane were flying normally. This way the pilot could continue to fly a damaged aircraft by controlling the plane as if it were undamaged. The neural network does the job of transforming the pilot’s

actions from normal operation into the necessary operations given that the plane is damaged in some way. The network was tested in high performance manoeuvres, such as tracking a target or performing a 360-degree roll. The neural net managed to keep disabled planes under control even at supersonic speeds [48].

There are several requirements and conditions a problem must satisfy if it is to be an acceptable candidate for a neural network solution. First and foremost the problem must be tolerant of some level of imprecision. All artificial intelligence techniques sacrifice some small measure of precision in favor of speed and tractability. This imprecision may be very small, much less than one percent or it may be relatively large such as thirty percent. Neural network error rates tend to be below two percent, but for certain applications error rates can go as low as a very small fraction of one percent. Any application that has zero tolerance for imprecision, cannot be solved with any artificial intelligence technique including neural networks.

Another requirement is that abundant high quality data must exist for both training and testing purposes. A neural network must be able to observe the problem at hand. It must also be put to test on that problem once it is trained but before it is put into service. This may require quite a lot of training and testing data depending on the complexity of the problem.

Consequently; complex, ambiguous and nearly unpredictable systems which tend to be difficult to compute or model, yet have enough resources for training data, are very suitable candidates for neural network modelling. Prediction of financial instruments fits well into this frame.

3.1.3 Training

As mentioned previously, there are two types of learning as far as neural networks are concerned: supervised and unsupervised. Supervised learning requires the programmer to give the network a sufficient number of data sets consisting of input(s) and the corresponding correct output. This way, the network can compare what it has output against what it should output, and it can correct itself. On the other hand, unsupervised learning provides input but no correct output. A network using this type of learning is only given inputs and the network must organize its connections and outputs without direct feedback and try to find possible regularities and relations. This type of learning is well suited to data extraction and analysis in which a pattern is known to exist in some data but the type and location of this pattern is unknown. Financial instrument prediction is based on data gathered from the source. These data can be used to train the neural network in a supervised manner.

Learning involves the change in the weight matrix such that,

$$w_{k+1} = w_k + 2\mu e_k x_k \tag{3.5}$$

where μ is the learning rate parameter i.e. a sufficiently small constant, e_k is the error – the difference between the desired response and the actual system response – at iteration step k, x_k is the input value to the weight i at iteration k, and w_k is the value of weight i at iteration k. This method is known as Least Mean Square (LMS) used in backpropagation algorithm. Here, weights are updated according to the error encountered. LMS algorithm employs *gradient descent*. Gradient descent is the most widely used method for ANN training. The reason for this popularity is that only simple computation is necessary to apply this method,

and gradient can be computed with local information. The principle in this method is simple: the values for the weights are changed in a direction opposite to the direction of the gradient. The gradient of a surface indicates the direction of the maximum rate of change. Therefore, provided that the weights are changed in the opposite direction of the gradient, the system state will approach points where the surface is flatter. The weight values that correspond to the point of minimum error are the optimal weights.

Although there are many learning algorithms such as Hebb rule, Adaline, decision tree, and genetic algorithms, *backpropagation* is the most widely used method for neural network training because it is easy to implement and to understand and works reasonably well for most problems.

3.1.4 Types of ANNs

There is virtually no limit to the type and variety of ANNs; however, in general neural networks are classified according to two factors: the topology or shape of the network and the learning method used. For instance, the most widely used topology is the feed-forward network and the most common learning method is the backpropagation of errors. Backpropagation is a form of supervised learning in which a network is given input and then the network's actual output is compared to the correct output. The network's connections are adjusted to minimize the error between the actual and the correct output. Feed forward networks that use backpropagation learning are so common that these networks are commonly referred to as *backpropagation networks* although this terminology is not correct. *Multi-layer feed-forward* refers to the topology and pattern of information flow in the network. Backpropagation refers to a specific type of learning algorithm. It is possible to use a feed forward architecture without backpropagation, or to use back propagation with another type of structure. In both cases, it has become commonly accepted to call this combination of topology and learning method simply a backpropagation network.

Another common network structure is the *recurrent* or *feedback network*. Recurrent networks are usually similar in shape to the feed forward network although data may pass backwards through the net or between nodes in the same layer (see Figure 3.5). Networks of this type operate by allowing neighboring neurons to adjust other nearby neurons either in a positive or negative direction.

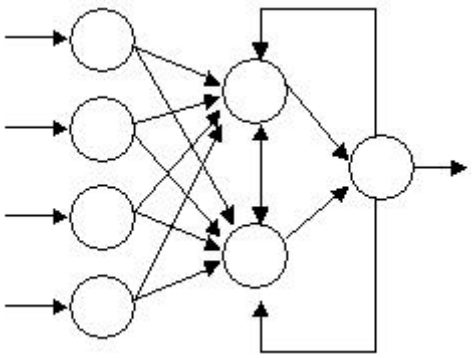


Figure 3.5 Recurrent/Feedback Network

This allows the network to reorganize the strength of its connections by not only the comparison of actual output against correct output but also by the interaction of neighboring

nodes. Recurrent networks are generally slower to train and to implement and also derive mathematical models than feed forward networks although they present several interesting possibilities including the idea of unsupervised learning. As discussed earlier, in unsupervised learning the network is only given input with no output and neurons are allowed to work in order to extract meaningful information from the data. This is especially useful when trying to analyze data searching for some pattern but no specific pattern is known to exist beforehand.

A third network structure, also based on the feed forward architecture, is the *functional link network*. This type of network, as shown in Figure 3.6, duplicates the input signal with some type of transformation on the input. In a functional link network, additional inputs will also be fed to the network, which are in some form of the original inputs. These additional inputs may be various products of the original inputs, or they may be high and low values from the whole input set, or they may be any combination of mathematical functions that are deemed to contain value for this set of input. In Figure 3.6 a functional link network is shown with four actual inputs and two additional functional links, which in this example are products of the first two and second two inputs. In this network the functional link is directly connected to the output layer although the functional link may be directed toward the hidden layer. The idea behind this type of network is to give the network as much information as possible about the original input set by also giving it variations of the input set.

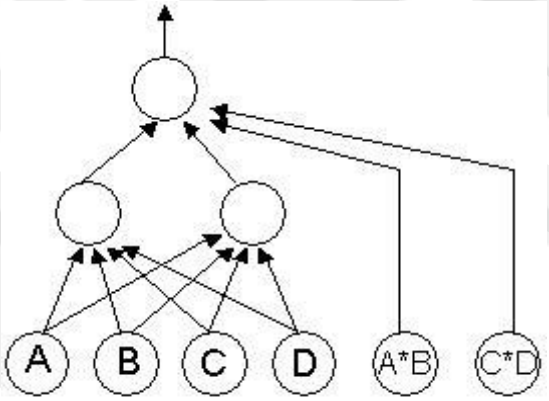


Figure 3.6 Functional Link network

Another type of neural network is Radial Basis Functions networks (ANN-RBFs). Like other types of neural networks, ANN-RBFs can learn arbitrary mappings: the primary difference is in the hidden layer. RBF hidden layer units have a *receptive field*, which has a *center*: that is, a particular input value at which they have a maximal output. Their output tails off as the input moves away from this point or just the other way round in the case of a multiquadric node function. This hidden layer function could be a Gaussian, Cauchy, multiquadric, etc. ANN-RBFs have the advantage that one can add extra nodes with centers near parts of the input that are difficult to classify. Figure 3.7 shows a traditional RBF network, in which each of n components of the input vector u feeds forward to m basis functions whose outputs are linearly combined with weight vector w into the network output f(u).

Once the network structure is designed, the input-output relation should be *taught* the network in order to obtain the desired model. In our case, which is the supervised learning problem, the relation between input and output of the system, is learned from the examples supplied by a *supervisor*. The set of the examples is referred to as the *training set*, and it contains pairs of

dependent and independent variables. Let y be the dependent variable and u is the independent variable with the relation,

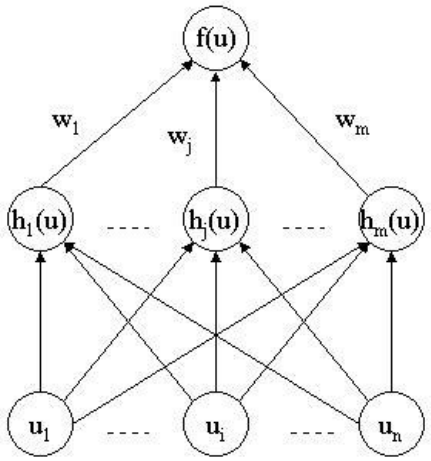


Figure 3.7 Radial Basis Function Neural Network

$$y = f(\boldsymbol{u}) \tag{3.6}$$

In (3.6) y is a scalar and u is a vector. Assume that there is a training set {[ui, yi]}, i=1,...,P to be used approximate the function f. The ANN-RBF uses a linear model,

$$F(u) = \sum_{j=1}^{m} w_j h_j(u) \tag{3.7}$$

where w_j is the weight from the j^{th} function to the output and, h is a Radial Basis Function.

In the hidden layer node, for a scalar input, the radial function can be either a Gaussian, Cauchy or multiquadric as mentioned earlier. A multiquadric radial function can be expressed as in (3.8),

$$h(u) = \frac{\sqrt{r^2+(u-c)^2}}{r} \tag{3.8}$$

where c is the center and r is the radius of the function. Whereas a Gaussian function is given in (3.4). Graphs of both functions are shown in Figure 3.8.

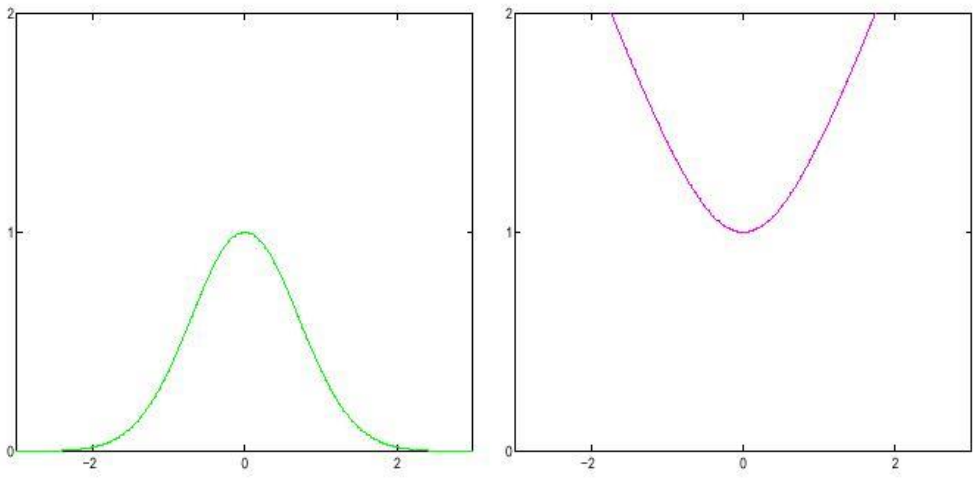
With the model expressed in (3.7) and a P-pattern training set {[ui,yi]}P, than the least squares should minimize the sum-squared-error with respect to the weights of the model [49]:

$$S = \sum_{i=1}^{P} (y_i - f(u_i)^2 \tag{3.9}$$

3.1.5 Drawbacks of ANNs

Aside from the fact that the system to be modelled must be error tolerant to a certain extent, there is another somewhat disturbing characteristic of neural networks. Let us explain: a neural network may solve a practical problem, but it is not obvious how it does it. Once the architecture of the network is decided and the training data are fed, the network learns and does everything by itself, without any human intervention. After the learning phase, the

network may start to give output that makes sense; however, the designers and users of the network do not have much knowledge about how that output is generated and why it is generated. The knowledge of a neural network resides entirely in its synaptic table, the table that holds the weights for the connections in the network. This table typically holds quite a lot of numbers, but is of little meaning to the onlooker. That is why this type of methods and tools are referred to as black box. In a scientific work, knowledge must be accessible. On the other hand, as regards predicting future forex movements, investors would not be too curious to know why the ANN predicted in which direction a certain currency pair would go, as long as it predicts correctly [50].



(a) (b)

Figure 3.8 RBF Activation Functions (a) Gaussian, (b) Multiquadric

3.2 Support Vector Machines

3.2.1 SVM Basics

SVMs are similar to ANNs, and are state of the art classifiers. SVM maps datasets to higher dimensional spaces in order to more easily (linearly) classify them (see Figure 3.9). Therefore it is a very strong classification tool. SVM is implemented using the kernel Adatron algorithm. The kernel Adatron maps inputs to a high-dimensional feature space, and then optimally separates data into their respective classes by isolating those inputs which fall close to the data boundaries. Therefore, the kernel Adatron is especially effective in separating sets of data which share complex boundaries. Gaussian kernel functions (Equation 3.4) are used in this study [51].

3.2.2 SVM Parameters

C: Penalty parameter C of the error term (default = 1.0).

Kernel: Specifies the kernel type to be used in the algorithm.

- Linear

- Poly - polynomial

- Gaussian – see Equation 3.4 (default)

- Hyperbolic tangent – see Equation 3.3

- Precomputed

- Callable – If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (no. of samples, no. of samples).

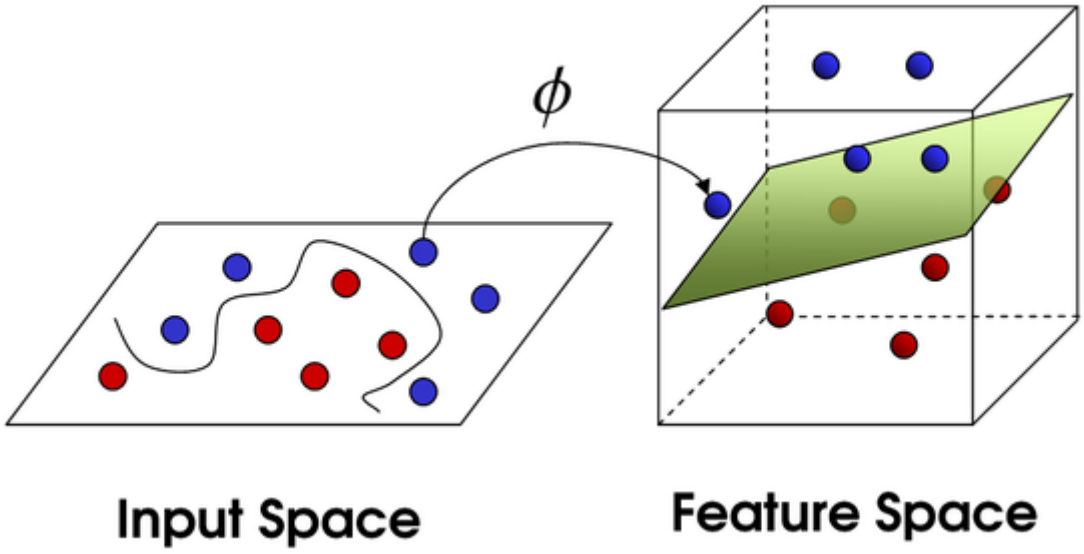


Figure 3.9 SVM classification

Degree: Degree of the polynomial kernel function ('poly'). This parameter is ignored by all other kernels (default = 3).

Gamma: Kernel coefficient for 'gaussian', 'poly' and 'hyperbolic tangent'. If gamma is 'auto' then 1/(no. of features) will be used instead (default = 'auto').

Coef0: Independent term in kernel function. It is only significant in 'poly' and 'hyperbolic tangent' (default = 0.0).

Probabilty: Whether to enable probability estimates (default = false).

Shrinking: Whether to use the shrinking heuristic (default = true).

Tol: Tolerance for stopping criterion (default = 1e-3).

Max_iter: Hard limit on iterations within solver, or -1 for no limit (default = -1).

Random_state: The seed of the pseudo random number generator to use when shuffling the data for probability estimation and can be used for generating reproducible results and also for parameter tuning (default = none).

3.3 Extreme Gradient Boosting

3.3.1 XGB Basics

XGB is another state of the art machine learning method, and is based on decision trees. A decision tree example showing how it classifies the XOR problem is shown in Figure 3.10.

Truth Table

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Binary Decision Tree

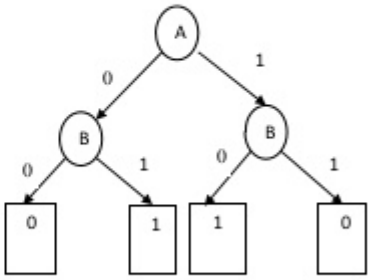


Figure 3.10 Decision tree for XOR classification

XGB classifier is chosen as one of the prediction and classification methods. This is mainly because it has gained much popularity in the recent years, due to its overwhelming success in data science competitions [39]. Despite its reputation, XGB method has not yet achieved popularity in the literature accordingly. Another great advantage of XGB is that, it trains very fast. Decision trees tackle classification tasks in general; however, XGB can handle both classification and regression problems.

It is basically a decision tree and C++ based model, and has libraries and interfaces in R, Python, and Julia languages. The model is basically like this:

$$\sum_{k=1}^{K} f_k \tag{3.10}$$

where each f_k is a prediction (probability) from a decision tree. The model is a collection of decision tree probabilities. With all the decision trees, prediction is made by:

$$y_i = \sum_{k=1}^{K} f_k\,(x_i) \tag{3.11}$$

where x_i is the feature vector for the i^{th} data point. In order to train the model, a loss function is needed; depending upon the task at hand, XGB typically uses three different loss functions. Root mean squared error for regression:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (3.12)$$

Logarithmic loss (logloss) for binary classification:

$$L = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \quad (3.13)$$

where N denotes number of samples/instances, y binary variable, and p is the classification probability at instance i.

Multi-logloss for multi-class classification:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \quad (3.14)$$

where N is the number of instances, M is the number of different labels, y_{ij} is the binary variable with the expected labels and p_{ij} is the classification probability output by the classifier for the i-instance and the j-label.

Another important part of the model is regularization. A good regularization term controls the complexity of the model and prevents overfitting:

$$\Omega = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \quad (3.15)$$

where T is the number of leaves, γ is the relaxation term (constant), λ is the L2 regularization term (constant), and w_j^2 is the score on the j^{th} leaf. With loss function and the regularization term together, the objective of the model is attained:

$$Obj = L + \Omega \quad (3.16)$$

where loss function controls the predictive power, and the regularization controls the simplicity. Just like ANN learning, gradient descent is used to optimize the objective function.

Unlike ANN and SVM, thanks to its tree structure, XGB is not a black box method. That is to say the results achieved by XGB classifier, can be more easily explained. Of course this is an advantage for the XGB method.

3.3.2 XGB Parameters

XGB has a number of parameters used in building, training and optimizing thus supporting robust machine learning. These parameters are explained as follows:

Eta / learning rate: Shrinks weights at each step, defines the speed of learning, similar to ANN learning (default = 0.3).

Min_child_weight: Defines the minimum sum of weights of all observations required in a child. It is used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree. On the other hand, too high values can lead to under-fitting (default = 1).

Max_depth: It is the maximum depth of a tree, and defines model complexity in a way. Too high values may lead to over-fitting (default = 6).

Gamma: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split (default = 0).

Subsample: Denotes the fraction of observations to be randomly samples for each tree. Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting (default = 1).

Colsample_bytree: Denotes the fraction of columns to be randomly samples for each tree (default = 1).

Colsample_bylevel: Denotes the subsample ratio of columns for each split, at each level (default = 1).

Lambda: L2 regularization term on weights. It is used to handle the regularization part of XGB. Though many data scientists do not use it often, it should be explored to reduce overfitting (default = 1).

Alpha: L1 regularization term on weights. It can be used in case of very high dimensionality so that the algorithm runs faster when implemented (default = 0).

Scale_pos_weight: A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence (default = 1).

Objective_function: Defines the loss function to be minimized.

- reg:linear – used for regression.

- binary:logistic - logistic regression for binary classification, returns predicted probability.

- multi:softmax - multiclass classification using the softmax objective, returns predicted class. One also needs to set an additional num_class (number of classes) parameter defining the number of unique classes.

- multi:softprob - same as softmax, but returns predicted probability of each data point belonging to each class.

Eval_metric: The metric to be used for validation data. Defaults vary according to the objective functions.

- rmse - root mean square error

- mae - mean absolute error

- logloss - negative log-likelihood

- error - Binary classification error rate (0.5 threshold)

- mirror - Multiclass classification error rate

- mlogloss - Multiclass logloss

- auc - Area under the curve

Seed: It is the random number seed, and can be used for generating reproducible results and also for parameter tuning (default = 0) [52].

3.4 Experiment to Identify the Type of Ann For Building Models

A comparison study of different types of ANNs is carried out in the context of this thesis, as they have varying performances on different data. Three instruments chosen for the experiment are: CSCO from stocks, EURUD from currencies being the mostly traded one, and XAGUSD from commodities. The experiment is designed as follows: these three instruments are trained at all frequencies, according to model inputs and outputs, using three different ANN topologies. The three ANN topologies are; ANN-RBF, multilayer perceptron ANN (ANN-MLP), and recurrent ANN (ANN-REC). The same sets of training and test data were used for all topologies. Training and testing for this experiment is carried out in the same fashion as comparing predictabilities of different instruments. The Normalized Mean Squared Error (NMSE) and correlation coefficient (R) performances of the three topologies over the test set are given in Table 3.1. NMSE is calculated by the formula below:

$$\text{NMSE} = \frac{\sum_{j=0}^{P}\sum_{i=0}^{N}(d_{ij} - y_{ij})^2}{\sum_{j=0}^{P}\left[\frac{N\sum_{i=0}^{N} d_{ij}^{\,2} - (\sum_{i=0}^{N} d_{ij})^2}{N}\right]} \tag{3.17}$$

where P = number of output processing elements (neurons),

N = number of exemplars in the data set,

y_{ij} = network output for exemplar i at processing element j,

d_{ij} = desired output for exemplar i at processing element j.

NMSE is actually mean square error divided by variance of desired output. Being a normalized value, it could easily be used for comparing different instruments of different prices and of different frequencies. Since it is an error term, values closer to zero denote better predictability [53].

Another statistically meaningful variable we use for predictability performance is the correlation coefficient R. R is used to measure how well one variable fits on another, linear regression wise. In our case, these variables are predicted against desired, in other words, ANN outputs vs. actual maximum values of the financial instruments in the next 8 periods of corresponding frequencies. R value is calculated by the formula below:

$$R = \frac{\frac{\sum_i (x_i - \bar{x})(d_i - \bar{d})}{N}}{\sqrt{\frac{\sum_i (d_i - \bar{d})^2}{N}} \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{N}}} \tag{3.18}$$

where P = number of output processing elements (neurons),

N = number of exemplars in the data set,

x = network output,

d_i = desired output.

The size of the mean square error (MSE) can be used to determine how well the network output fits the desired output, but it does not necessarily reflect whether the two sets of data move in the same direction. For instance, by simply scaling the network output, we can change the MSE without changing the directionality of the data. The correlation coefficient R solves this problem. By definition, the correlation coefficient between a network output x and a desired output d is defined by formula (3.18). The correlation coefficient is confined to the range [-1 1]. When R = 1 there is a perfect positive linear correlation between x and d, i.e. they vary accordingly, which means that they vary by the same amount. When R = -1, there is a perfect linear negative correlation between x and d, i.e. they vary in opposite ways (when x increases, d decreases by the same amount). When R =0 there is no correlation between x and d, i.e. the variables are called uncorrelated. Intermediate values describe partial correlations [54].

The total of approximately 65000 data points for each instrument is time-wise randomly split: approximately 75% was used for training, 15% for testing, and the remaining 10% for cross validation [55]. Each network is trained until no substantial improvement is brought about in the cross validation set, and the weights are then saved. One of the main issues concerning machine learning is over-learning or over-fitting problem, in which the system memorizes a certain data set rather than learning it, because of too many training epochs or too complex ANN structure. The performance of the system can be excellent on that data set; however, it performs poorly on different data due to the lack of generalization. The opposite of this issue is the under-learning problem. Therefore, sufficient number of training epochs is crucial for an optimum system performance. The criterion of training until improvement in the cross validation set stops is a trade off in this study. The inputs to ANN contain only past data of the financial instruments. The output is the prediction of the actual result i.e. the maximum price value into the next eight periods for each frequency (e.g. 8x1-minute, 8x5-minutes, 8x15-minutes, and so on).

According to Table 3.1, the best performing ANN topology is ANN-RBF, in terms of both NMSE and R.

Table 3.1 ANN topology performance comparison

		NMSE			R		
Frequency	**Instrument**	**RBF-ANN**	**MLP-ANN**	**REC-ANN**	**RBF-ANN**	**MLP-ANN**	**REC-ANN**
1m	**csco**	0.80224	0.91322912	1.3665	0.495934	0.4270939	0.05665
1m	**eurusd**	0.895819	0.89217794	1.0446	0.32357	0.3178204	0.1139
1m	**xagusd**	0.9258	0.94220455	1.1835	0.272667	0.2715276	0.09419
5m	**csco**	0.89849	0.99797156	1.0111	0.326634	0.3325848	0.2118
5m	**eurusd**	0.916606	0.91964578	0.9543	0.288946	0.2836154	0.2235
5m	**xagusd**	0.935662	0.94941301	0.9876	0.255305	0.2480923	0.1917
15m	**csco**	0.978234	0.99980437	1.08955	0.167856	0.1396122	0.08557
15m	**eurusd**	0.924395	0.92679357	0.9999	0.275004	0.2706413	0.2338
15m	**xagusd**	0.853869	0.8675471	0.9238	0.382474	0.3658165	0.2745
30m	**csco**	0.954717	0.96035662	1.004	0.214785	0.2017414	0.1746
30m	**eurusd**	0.902811	0.9100532	0.9347	0.312319	0.3003872	0.2836
30m	**xagusd**	0.788396	0.7825817	0.8974	0.462418	0.4691003	0.3448
1h	**csco**	0.938382	0.94525936	0.9547	0.226632	0.2158823	0.2021
1h	**eurusd**	0.925988	0.92564248	1.1356	0.272134	0.2727817	0.0459
1h	**xagusd**	0.821388	0.79527502	0.9342	0.425614	0.4524943	0.3113
4h	**csco**	0.916473	0.9155791	0.9948	0.28789	0.288276	0.25443
4h	**eurusd**	0.829866	0.83359682	0.9558	0.41161	0.4125939	0.2993
4h	**xagusd**	0.906547	0.89864533	0.97561	0.29725	0.287439	0.26107
Average		**0.895316**	**0.90976537**	**1.01931444**	**0.316613**	**0.30875**	**0.203484**

CHAPTER IV

PREDICTABILITY ANALYSIS

4.1 Data Used

In this chapter, the possibility to give the investor a better starting point by trying to answer the following question is investigated: which financial instrument is quantitatively more predictable? The answer to this question will also be useful in the following chapters where we investigate the possibilities to enhance predictability and achieve profitability.

As mentioned earlier, the data used in predictability analysis consist of 1-minute (1m), 5-minute (5m), 15-minute (15m), 30-minute (30m), 1-hour (1h), and 4-hour (4h) intraday frequencies covering the period between 1996 and 2016 with approximately 65000 data points for each instrument and frequency. This corresponds to a few months for 1-minute data, and 20 years for 4-hour data. The data are taken from authors' private data accounts. High frequency data are preferred, because in lower frequencies more and more non-technical factors might start to affect price movements. Three types of instruments are considered in this chapter: stocks, currencies (forex), and commodities. Four individual instruments were chosen to represent each type of instrument; namely Australian Dollar against US Dollar (AUDUSD); Euro against Canadian Dollar (EURCAD); Euro against US Dollar (EURUSD); and US Dollar against Japanese Yen (USDJPY), representing forex, BRENT crude oil; LIGHT crude oil; silver (XAGUSD); and gold (XAUUSD), representing commodities, and finally Amazon.com Inc (AMZN); Cisco Systems Inc. (CSCO); General Motors (GM); and Coca Cola Company (KO), representing stocks as mentioned earlier.

4.2 Feature Selection

The total of fourteen features for the predictability analysis model is chosen, which contains only past data (past high/low values, etc.) of the financial instruments. The output is the prediction of the actual result i.e. the maximum price value into the next eight periods (8x1-minute, 8x5-minutes, 8x15-minutes, and so on).

4.3 Prediction Method and Training

ANNs are preferred in this study as a prediction tool, considering ANNs high predictive modeling power. As explained in Chapter 3, ANN-RBF is chosen as topology according to the results of the experiment.

Data points for each instrument are time-wise randomly split: approximately 75% is used for training, 15% for testing, and the remaining 10% for cross validation. Each network is trained until no substantial improvement is brought about in the cross validation set, and the weights are then saved. Cross validation is used in order to avoid one of the main issues concerning machine learning, i.e. over-fitting. The criterion of training until improvement in the cross

validation set stops is a trade off in this study. The inputs to ANNs contain only past data of the financial instruments. The output is the prediction of the actual result i.e. the maximum price value into the next eight periods for each frequency (e.g. 8x1-minute, 8x5-minutes, 8x15-minutes, and so on) as mentioned earlier.

ANNs to model each instrument and frequency are trained 5 times starting with randomly different initial weights, and performances on test sets are averaged to obtain Normalized Mean Squared Error (NMSE) and correlation coefficient R values. Computation and physical meanings of these variables are explained in the previous chapter. A total of 72 different artificial neural networks representing 12 different instruments at 6 different frequencies are trained five times each, and their prediction performances are recorded on average.

4.4 Results and Conclusion

All recorded and calculated performance comparison values in terms of R and NMSE for all financial instruments and all frequencies are given in Table 4.1 and Table 4.2 which hopefully could enlighten us on predictability.

Table 4.1 Detailed Performance Comparisons

Instrument	4h		1h		30m		15m		5m		1m	
	R	*NMSE*	*R*	*NMSE*	*R*	*NMSE*	*R*	*NMSE*	*R*	*NMSE*	*R*	*NMSE*
AMZN	0.35	0.88	0.33	0.89	0.25	0.94	0.25	0.94	0.28	0.92	0.27	0.93
CSCO	0.28	0.92	0.26	0.94	0.22	0.96	0.15	0.98	0.33	0.89	0.45	0.80
GM	0.29	0.91	0.30	0.91	0.36	0.87	0.40	0.84	0.41	0.84	0.53	0.73
KO	0.22	0.96	0.21	0.96	0.18	0.97	0.22	0.96	0.21	0.96	0.07	1.01
AUDUSD	0.42	0.82	0.40	0.84	0.46	0.79	0.32	0.90	0.28	0.92	0.25	0.94
EURCAD	0.51	0.74	0.55	0.70	0.35	0.88	0.24	0.94	0.70	0.52	0.80	0.36
EURUSD	0.41	0.83	0.28	0.92	0.31	0.90	0.27	0.92	0.29	0.91	0.36	0.87
USDJPY	0.39	0.85	0.32	0.90	0.36	0.87	0.26	0.93	0.31	0.90	0.25	0.94
BRENT	0.19	0.96	0.22	0.95	0.21	0.96	0.21	0.96	0.40	0.84	0.36	0.87
LIGHT	0.16	0.97	0.13	0.98	0.05	1.02	0.10	1.00	0.39	0.87	0.36	0.87
XAGUSD	0.41	0.84	0.44	0.81	0.46	0.79	0.37	0.86	0.25	0.94	0.27	0.93
XAUUSD	0.40	0.84	0.45	0.80	0.36	0.87	0.36	0.87	0.38	0.85	0.27	0.93

As can be seen in Tables 4.1 and 4.2, there is a clear distinction between currencies and other instruments both R and NMSE wise. In other words, currencies are the easiest to predict among the instruments in question. The group of instruments which is the hardest to predict is stocks; however, with a narrow margin against commodities. We can also say that 15-minute data are the least technically predictable among the other frequencies. Nevertheless, the predictability of 30-minute data is not so much better than that of 15-minute data, again both R and NMSE wise. 1-minute data are the most quantitatively predictable in terms of NMSE, on the other hand 5-minute data are the most quantitatively predictable in terms of R. However, neither the predictability performance of 1-minute data nor the predictability performance of 5-minute data is much better than those of 1-hour or 4-hour data. Therefore from the investor's point of view, trading decisions based on 1-hour or 4-hour data could even be more profitable, considering commissions and spread margins are usually more disadvantageous in higher frequencies like 5-minute or 1-minute.

Judging by the R values in Tables 4.1 and 4.2, one could argue that the prediction performances are somewhat low; however, predicting financial instruments technically is

indeed such a hard task and enhancement in prediction performances is handled in the next chapter. In this chapter, at least it is demonstrated that it would be possible to give some hope to the investor and assist him/her in choosing more predictable instruments.

Table 4.2 Overall Performance Comparisons

		Stocks	Currencies	Commodities	All Instruments
4h	***R***	0.28643812	0.43200822	0.29237317	0.33693984
	NMSE	0.91880484	0.81166616	0.90663130	0.87903410
1h	***R***	0.27377488	0.39063054	0.31002520	0.324810208
	NMSE	0.92536862	0.83727909	0.88463085	0.88242618
30m	***R***	0.25179278	0.37118151	0.26792855	0.29696761
	NMSE	0.93196670	0.85952219	0.90406936	0.89851942
15m	***R***	0.25615607	0.27153008	0.26154911	0.26307841
	NMSE	0.92765705	0.92566387	0.92063667	0.92465253
5m	***R***	0.30625493	0.39743173	0.35640030	0.35336232
	NMSE	0.90142611	0.81463511	0.87568382	0.86391501
1m	***R***	0.33082840	0.41394639	0.31386376	0.35287952
	NMSE	0.86390559	0.77781108	0.90010259	0.84727309
Average	***R***	0.28420753	0.37945474	0.30035668	0.32133965
	NMSE	0.91152149	0.83776292	0.89862577	0.88263672

CHAPTER V

PREDICTABILITY ENHANCEMENT

5.1 A Different Approach in Prediction

In this chapter an investigation is carried out in order to give the investor a huge advantage by trying to answer the following question: would a certain FOREX rate be likely to go up or down in the following hours? The answer to this question will also be useful in the following chapter where we investigate the possibilities to achieve profitability. Technical (quantitative) analysis i.e. past price values of the instruments is solely used, and fundamental analysis is not in the scope of this thesis.

Since we are to use past price movements only, we need to come up with something creative to make a *difference*. Bearing this in mind, we propose the usage of *visual features* related to the shapes of FOREX price movements to classify trends. We are inspired by the fact that one can somewhat determine the *differences* among the price movements by visually inspecting their graphs. The proposed approach; called herein *Finance Vision* method, is similar to Machine Vision which is the technology that employs image processing methods to recognize patterns, in the way that, in this chapter, the FOREX price movements are treated like images to get certain features, just as the saying goes: "a picture is worth a thousand words" [56, 57]. These features are then used for training XGB, ANN and SVM models to recognize future price trends. Experiments show that comparable recognition rates are obtained.

5.2 Data Used

As shown in the previous chapter; following an extensive predictability comparison, the most predictable instrument is found to be FOREX. Similarly, the most predictable frequencies are turned out to be 1-minute and 5-minute. However, neither the predictability performance of 1-minute data nor the predictability performance of 5-minute data is much better than those of 1-hour or 4-hour data. Therefore, from the investor's point of view, trading decisions based on 1-hour or 4-hour data could even be more profitable, considering commissions and spread margins are usually more disadvantageous in higher frequencies like 5-minute or 1-minute. Hence; as an optimum frequency, 1-hour data is chosen for predictability enhancement, covering the 6-year period between 2011 and 2016 with approximately 36000 data points for each currency pair before preprocessing. Four individual FOREX pairs are investigated; namely Australian Dollar against US Dollar (AUDUSD); Euro against Canadian Dollar (EURCAD); Euro against US Dollar (EURUSD); and US Dollar against Japanese Yen (USDJPY).

5.3 Feature Selection

In predictability analysis (Chapter 4), fourteen features containing past high and low values of the financial instruments are used. In order to enhance the performance, we need to introduce

new features, and most of these would be density based visual features in our case. We propose ten additional features to have a total of twenty four features. Twenty three out of the twenty four features used for our prediction model are related to visual attributes of price movements, and one is related to periodicity (day of week + hour of day/24). Nine of the visual features are related to density. Density is visualized as distribution of total counts of open, high, low and close values within 8 different price level bands over a certain period i.e. past 377 hours. Density levels also show similarity to chaotic behavior in terms of movements about and in between *saddle regions*. The remaining fourteen features comprise of various normalized high and low values to complete the overall shape of the financial instrument [58]. Density levels are illustrated in Figure 5.1 below.

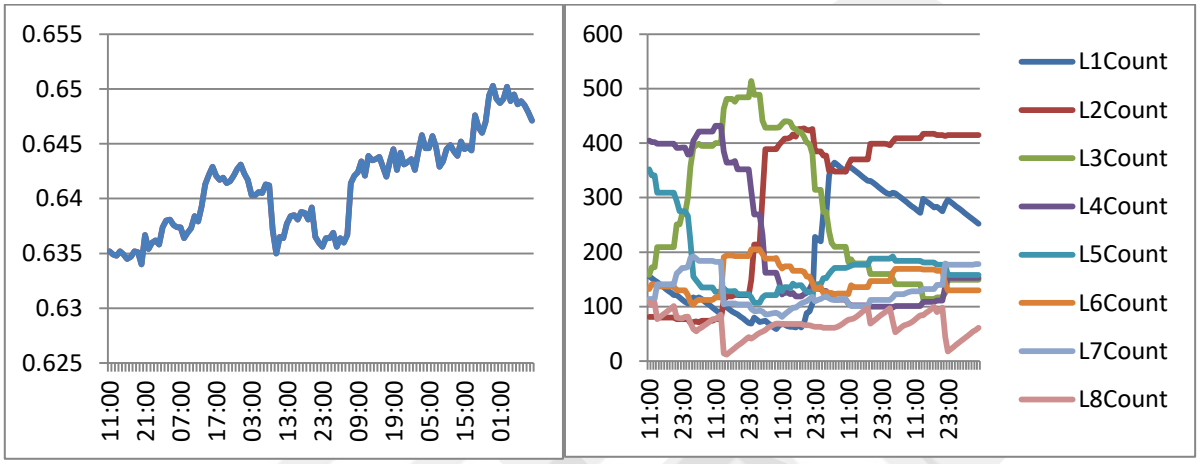


Figure 5.1 (a) Sample FOREX price movement (b) Associated density levels

5.4 Output Selection

Although in predictability analysis, the output is chosen as the highest value into the next 8 hours, in this chapter a further correlation analysis is done between the inputs and the alternative outputs as; highest values into the next 3, 5, 8, and 13 hours. Results of this analysis are given in Table 5.1. Here, the total correlation value denotes the sum of absolute correlation values between all 24 input features and the corresponding output. There are only slight differences in the values; however, the period with the highest value i.e. 5 hours ahead is chosen as the output.

Table 5.1 Correlation analysis of output alternatives

High value period	Total abslolute correlation with inputs
3 hours ahead	0.130290125
5 hours ahead	0.131632675
8 hours ahead	0.130441884
13 hours ahead	0.129039732

Consequently, the output is the binary classification value denoting whether the FOREX instrument would go up a certain high threshold value (1) or go down another low threshold value (0) in the next five hours. However, this is not only a high or low value above threshold: when going up it should not go down the low threshold, and vice versa. The remaining in between values are tagged as no trend, and excluded from model training.

5.5 Classifiers

5.5.1 Artificial Neural Network

ANNs are chosen as main predictors and classifiers in this thesis and are explained in 3.1 in detail. In the experiment to identify the type of ANN for building models (Chapter 3), ANN-RBF is chosen. On the other hand, for 1-hour data and FOREX, Multi-Layer Perceptron (MLP) ANN is found to perform better than ANN-RBF (see Table 3.1). Therefore, for predictability enhancement, an ANN model is constructed with a multi-layered feed forward network, having 2 hidden layers with 50 and 8 neurons, respectively. A test is carried out to determine the optimum number of hidden layers for ANN-MLP, where the total number of neurons kept constant and different networks are trained with 1, 2, 3, and 4 hidden layers. ANN-MLPs are trained for all four FOREX pairs and their average performances are given in Table 5.2. According to the outcome of this test, the 2-hidden layer ANN-MLP structure has the optimum performance. Hyperbolic tangent is chosen as activation function. The networks are trained by back propagation gradient descent algorithm.

Table 5.2 ANN-MLP performances by number of hidden layers

ANN Structure	R	NMSE
1 hidden layer	0.302269	0.486505
2 hidden layers	0.378598	0.478978
3 hidden layers	0.336188	0.480796
4 hidden layers	0.241799	0.489439

5.5.2 Support Vector Machine

SVMs are used as an alternative method to ANNs and are explained in 3.2. Default parameter values are used in model training.

5.5.3 Extreme Gradient Boosting

XGB is used as another alternative method to ANNs and is explained in 3.3. The best parameter values for the XGB classifier are given in Table 5.3 below. For the remaining parameters, defaults are used.

Table 5.3 XGB Training parameters

Parameter	Value
learning rate	0.34
no. of estimators (trees)	377
max depth	8
objective function	logistic

5.6 Training, Testing, Validation and Results

The data are randomized in order to achieve a fair distribution. Only 50% of the data is chosen to train the models, such that only data that go above certain fluctuation (up and down) threshold levels are kept and the rest are eliminated. 75% of data are used for training,

15% for testing, and the remaining 10% for cross validation. All models are trained until the results stop improving in the validation set. The model training and testing data cover the 5-year period between 2011 and 2015. The remaining, out of the range, 1-year data (2016) is used for validation [59]. A statistical analysis between model building and validation data is made. The size of the validation data is approximately 1/12th of the model building data; hence, the validation data is repeated for this analysis to get equal data lengths. The ANOVA analysis for this comparison is given in Table 5.4. The correlation between the two data sets is 0.0249. Judging by the correlation and the analysis given in Table 5.4, it is fair to say that model building and validation periods have much different characteristics. Neurosolutions software [60] is used for ANN and SVM training and Python software [61] for XGB.

To understand the effects of the density based visual features better, three different feature sets are used in the training and testing of the models: the first set comprises of full density and all other features, second set is the one containing fewer features with density features are removed, and finally the third set contains additional features containing technical indicators commonly used in financial time series [62-65], such as moving averages, Relative Strength Index (RSI), ARMAX, NARMAX, GARCH volatility, etc. in place of the density features, keeping the total number of features constant.

During the training process how the mean squared training error changes for an ANN model illustrated in Figure 5.2, as an example. Also an example of training termination is given in Figure 5.3. A statistical R-value and classification performance analysis is made on the test data in order to compare the produced outputs with the actual values that indicated whether predictions succeed or not. Results of this analysis are given in Table 5.5, 5.6, and 5.7 for the model with full density based visual features, with density features removed, and with density features removed but other extra features introduced instead respectively. Overall performance comparisons on average classification scores are illustrated through the Figures 5.10 to 5.13. Finally in Table 5.8, an R value analysis is given of all FOREX pairs on average, and overall comparisons for all models are illustrated through the figures 5.14 to 5.17 for the out of range validation set.

Table 5.4 ANOVA analysis between model building and validation data

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Model building data	79151	67344.22	0.850832	0.014946
Validation data	79151	59143.31	0.747221	0.000346

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	424.8518	1	424.8518	55563.93	0	3.841518
Within Groups	1210.39	158300	0.007646			
Total	1635.242	158301				

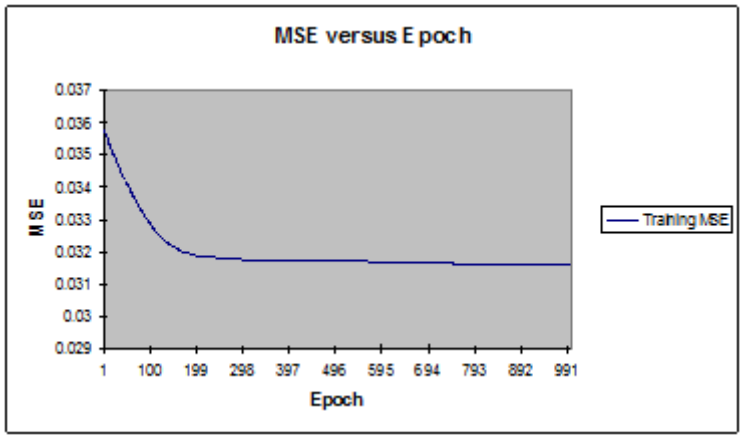


Figure 5.2 MSE for ANN training

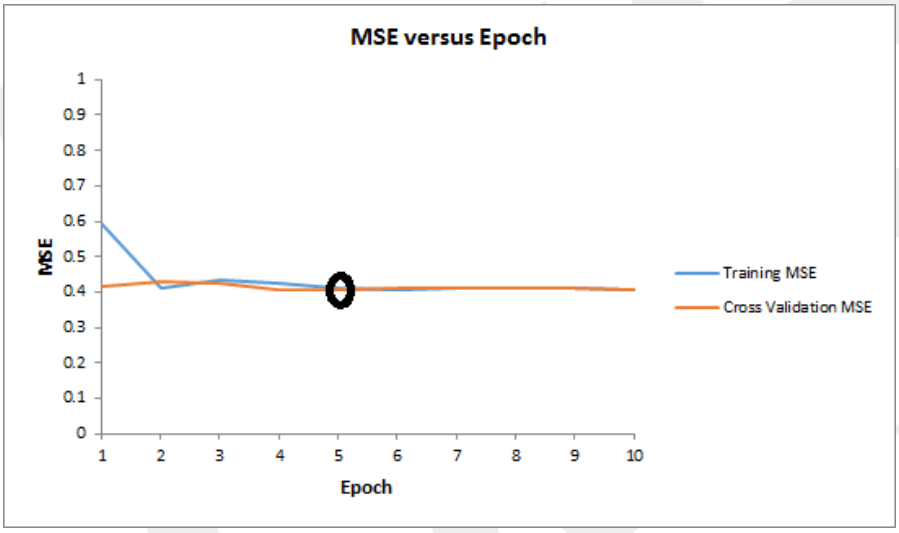


Figure 5.3 Timing example for training termination

A statistically meaningful variable we used for predictability performance comparison is the correlation coefficient R. R is used to measure how well one variable fits on another, linear regression wise. In our case, these variables were predicted against desired, in other words, model outputs vs. actual classification values denoting movement directions of the FOREX pairs in the following 5 hours. R value is calculation is explained in Chapter 3.

The performances of all three classifiers are also evaluated in terms of sensitivity, specificity, accuracy, and precision. These parameters are statistical measures for classification. Values close or equal to 100% are desirable [66-69]. They are related with true positive (TP), true negative (TN), false positive (FP) and false negative (FN) values, as explained below:

TP: Number of cases belonging to a certain class that are correctly classified.

TN: Number of cases not belonging to a certain class that are correctly classified.

FP: Number of cases belonging to a certain class that are incorrectly classified.

FN: Number of cases not belonging to a certain class that are incorrectly classified.

These parameters are calculated by the following equations:

$$Sensitivity = \frac{TP}{TP+FN} \quad (5.1)$$

$$Specifity = \frac{TN}{TN+FP} \quad (5.2)$$

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (5.3)$$

$$Precision = \frac{TP}{TP+FP} \quad (5.4)$$

Another popular method for measuring classification performance is Receiver Operating Characteristic (ROC) curve. In statistics, a ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the fall-out or probability of false alarm and can be calculated as (1 − specificity). The ROC curve is thus the sensitivity as a function of fall-out. ROC analysis is used in many areas and is increasingly used in machine learning and data science research nowadays. As a score for model comparison, Area Under Curve (AUC) is used with ROC. AUC values closer to 1 indicate better classification [70]. ROC curves and AUC scores are given for the AUDUSD models using full density based visual features in Figures 5.4, 5.5, and 5.6; as well as for the models without density based features in Figures 5.7, 5.8, and .9.

Average training times in seconds of the three classifiers are given in Table 5.9.

Table 5.5 Performance comparison for full visual density based features

	XGB				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.78878505	0.802207	0.795819	0.7836583	0.590827
EURCAD	0.83746407	0.863375	0.838635	0.827683	0.698691
EURUSD	0.62103673	0.630012	0.623518	0.612748	0.458639
USDJPY	0.65473571	0.673684	0.664537	0.642535	0.493787
	ANN				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.67291312	0.728733	0.699373	0.733519	0.415513
EURCAD	0.79364731	0.832647	0.803744	0.8547348	0.570436
EURUSD	0.53283632	0.593846	0.547354	0.6046932	0.294438
USDJPY	0.58354845	0.637453	0.604749	0.6544773	0.335256
	SVM				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.80723982	0.835847	0.821685	0.8282266	0.683576
EURCAD	0.83746459	0.850938	0.846453	0.842834	0.749882
EURUSD	0.60999436	0.657484	0.635482	0.629374	0.512497
USDJPY	0.67458321	0.716359	0.708465	0.711286	0.578643

Table 5.6 Performance comparison for no density features

	XGB				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.60186916	0.673175	0.639235	0.62585	0.275713
EURCAD	0.70027383	0.767734	0.723863	0.71989	0.404757
EURUSD	0.51836091	0.583741	0.553908	0.54552	0.189921
USDJPY	0.52539831	0.593896	0.553672	0.54355	0.210928
	ANN				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.55712452	0.562285	0.560484	0.405607	0.187504
EURCAD	0.64983635	0.653709	0.643948	0.510039	0.273645
EURUSD	0.50938732	0.515673	0.515112	0.355638	0.137481
USDJPY	0.52536849	0.538292	0.529931	0.369984	0.157392
	SVM				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.69130435	0.745841	0.717742	0.742991	0.477888
EURCAD	0.81536348	0.846376	0.828393	0.845362	0.632083
EURUSD	0.55029322	0.600846	0.585434	0.598693	0.348462
USDJPY	0.60667591	0.654438	0.619974	0.643946	0.380293

Table 5.7 Performance comparison for no density but other extra features

	XGB				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.6728972	0.7249576	0.700178	0.689655	0.398406
EURCAD	0.7389458	0.812746	0.797838	0.787926	0.523949
EURUSD	0.6047511	0.574944	0.603922	0.556482	0.315367
USDJPY	0.6029374	0.640133	0.637492	0.638325	0.347458
	ANN				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.5691861	0.6434231	0.600604	0.6850945	0.256393
EURCAD	0.6649503	0.749375	0.674835	0.7659401	0.345537
EURUSD	0.5294856	0.585942	0.544947	0.5839463	0.223944
USDJPY	0.5595754	0.628354	0.589407	0.6466588	0.213957
	SVM				
	Sensitivity	**Specifity**	**Accuracy**	**Precision**	**R**
AUDUSD	0.7852447	0.7805332	0.782696	0.752274	0.600139
EURCAD	0.8328467	0.846473	0.820125	0.817352	0.725844
EURUSD	0.5493644	0.648312	0.54837	0.600247	0.458353
USDJPY	0.6448465	0.696902	0.689432	0.595734	0.523949

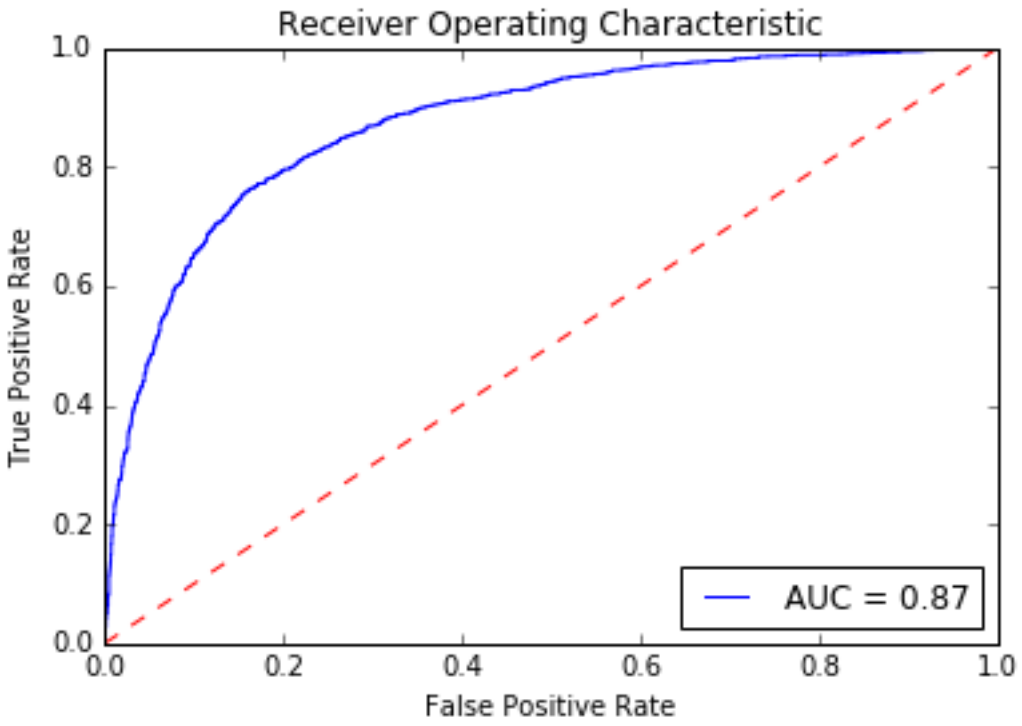


Figure 5.4 ROC curve for XGB model with full visual density based features

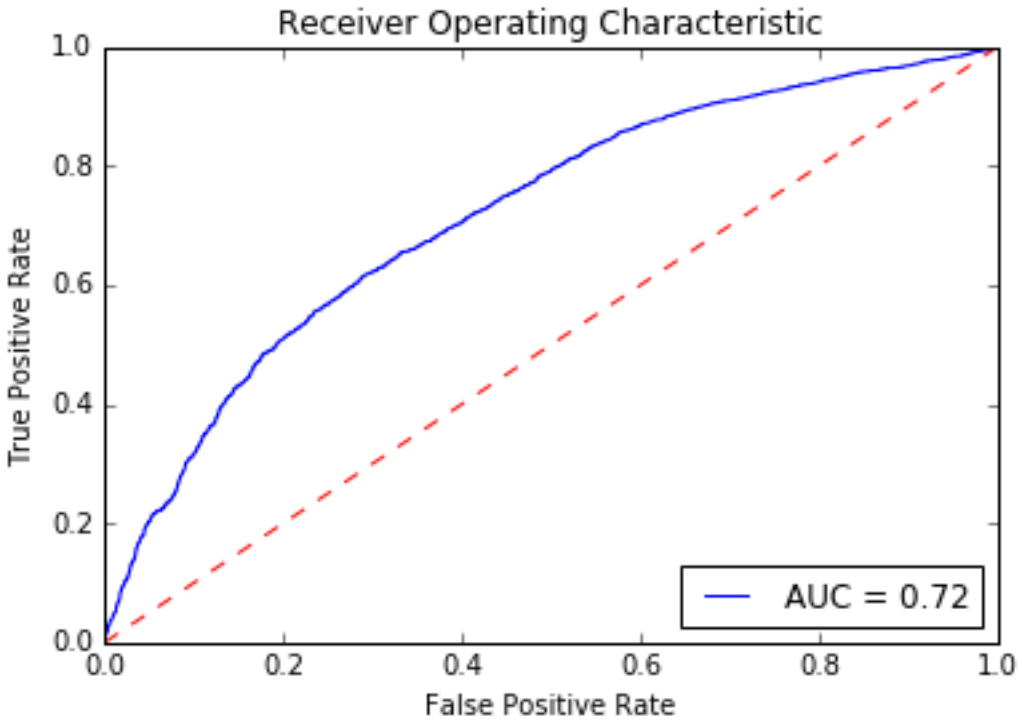


Figure 5.5 ROC curve for ANN model with full visual density based features

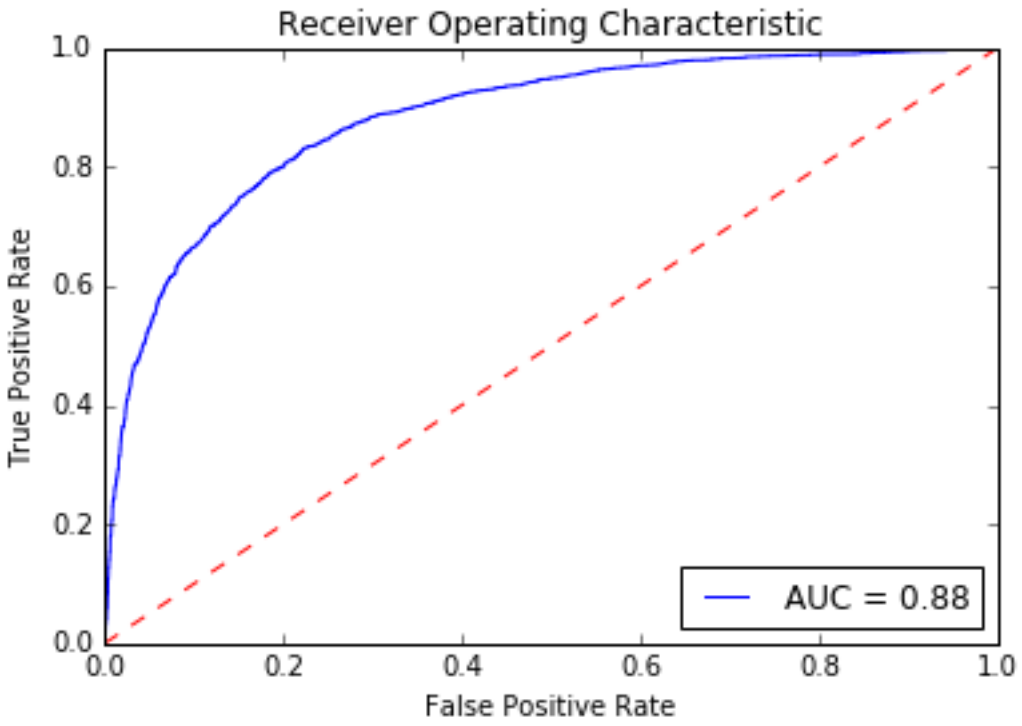


Figure 5.6 ROC curve for SVM model with full visual density based features

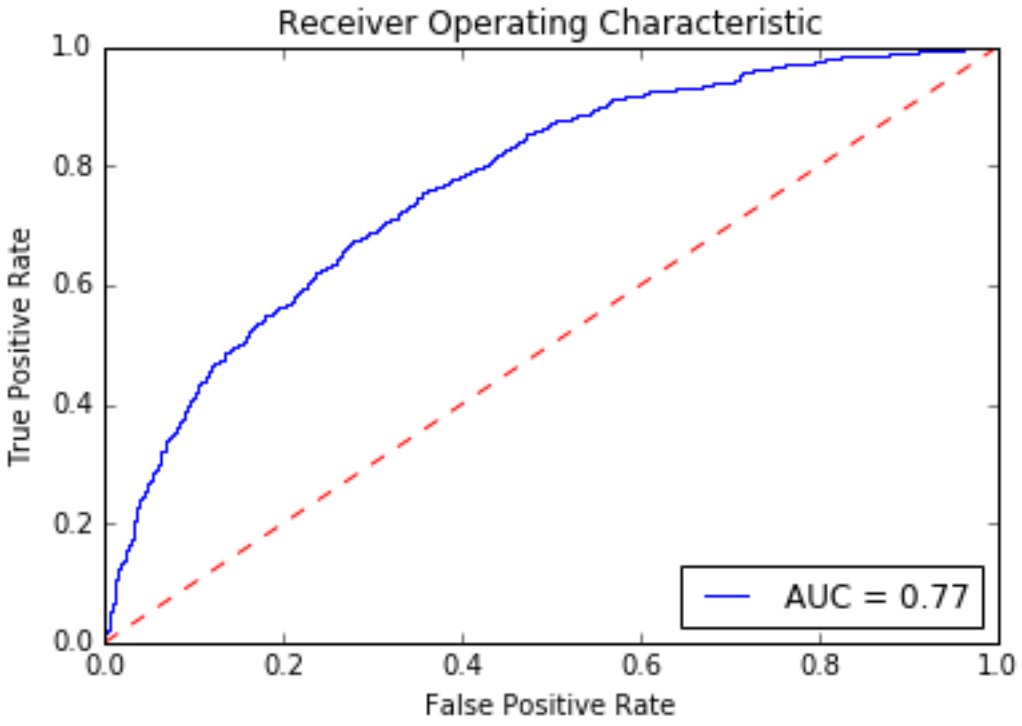


Figure 5.7 ROC curve for XGB model with no density features

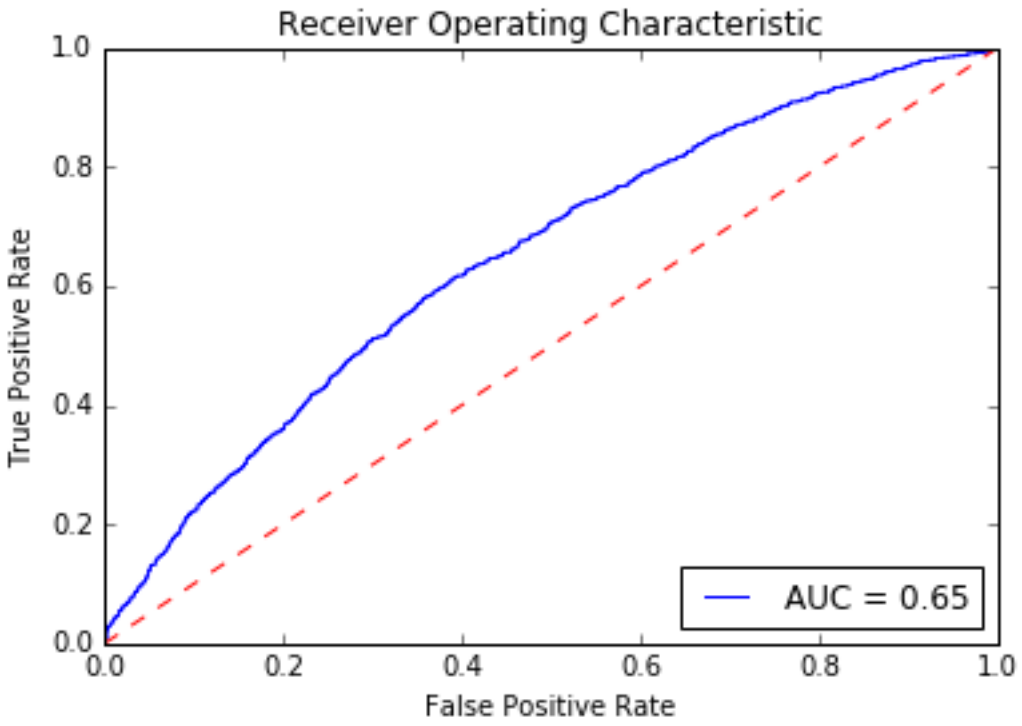


Figure 5.8 ROC curve for ANN model with no density features

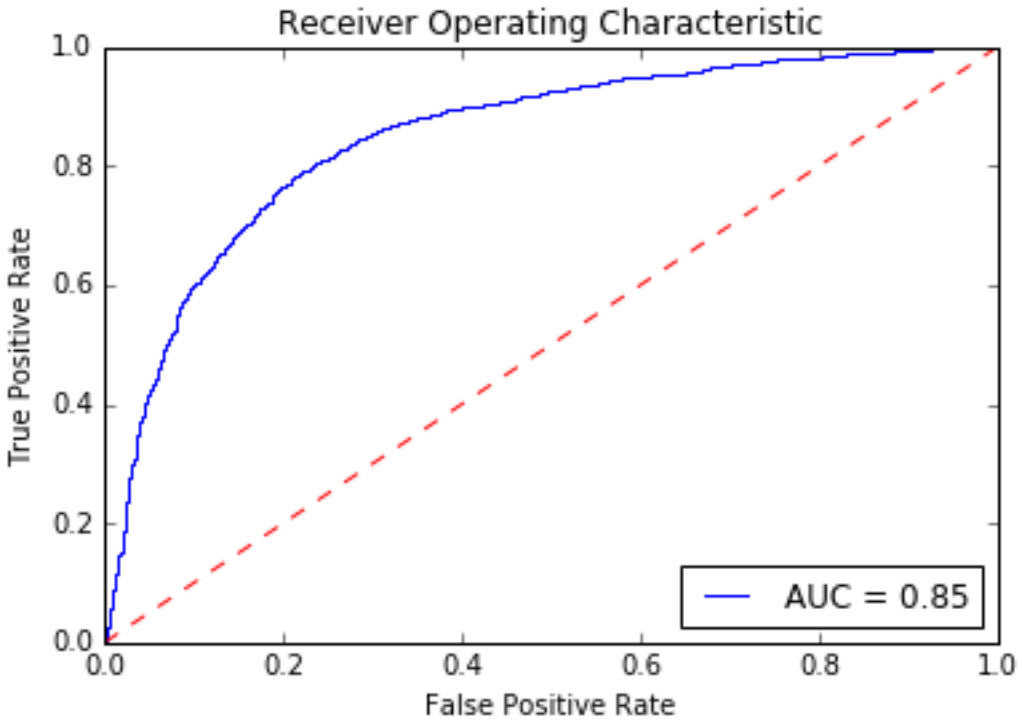


Figure 5.9 ROC curve for SVM model with no density features

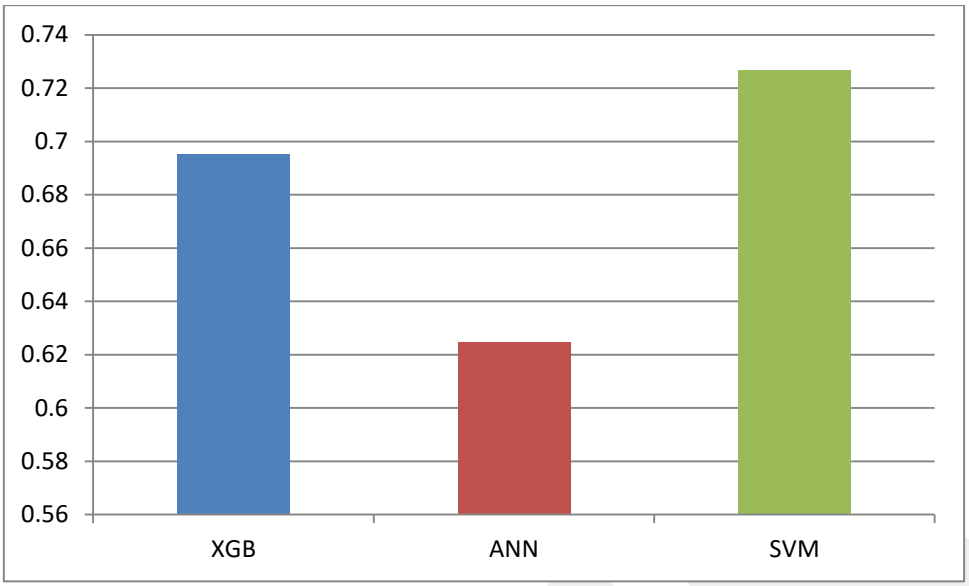


Figure 5.10 Performance comparison for full visual density based features

Figure 5.11 Performance comparison for no density features

Figure 5.12 Performance comparison for no density but other extra features

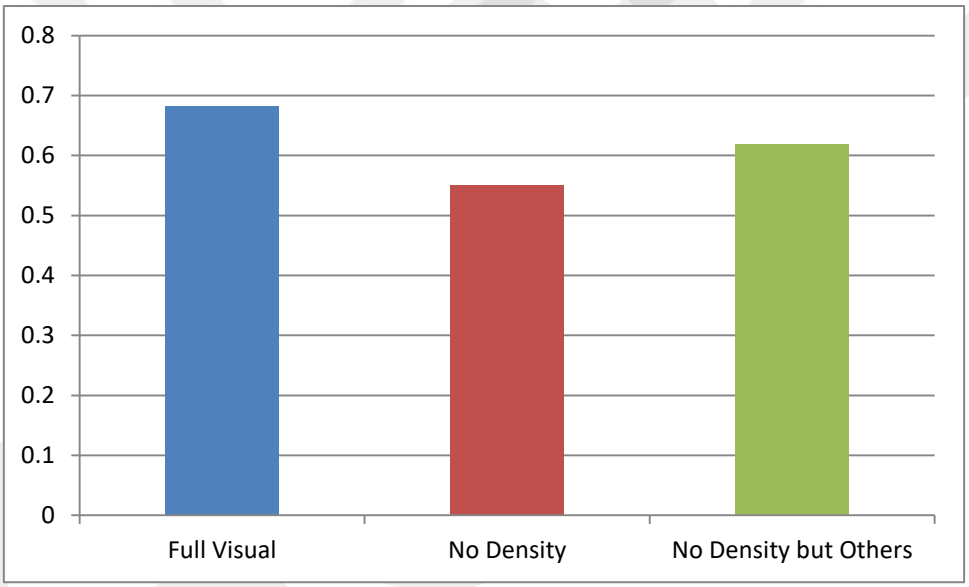


Figure 5.13 Overall comparison among feature sets

Table 5.8 R values on validation data

	XGB	ANN	SVM
Full	0.157639	0.219886	0.055405
No density	0.089221	0.190187	0.019189
No density but other	0.147648	0.173114	-0.00287

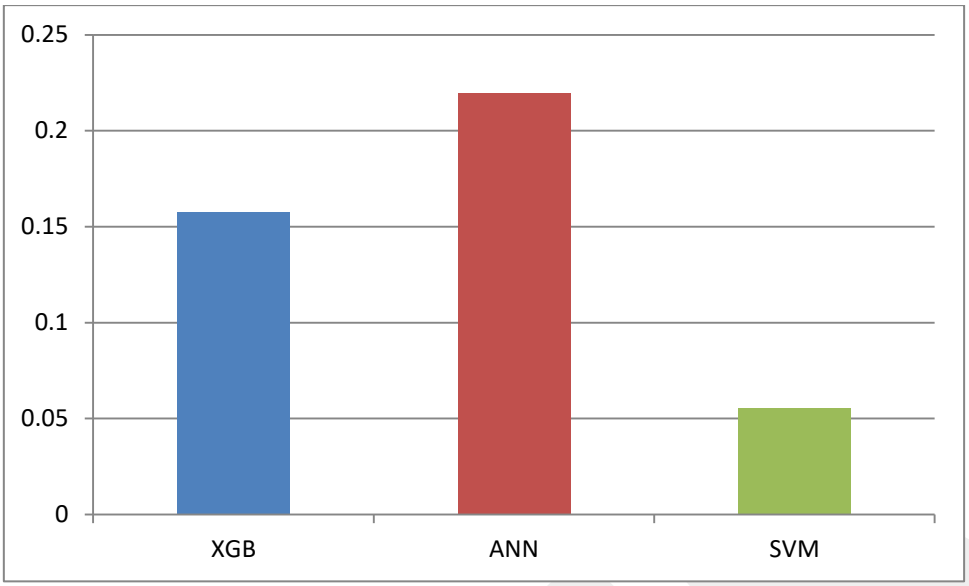


Figure 5.14 Performance comparison for full visual density based features on validation data

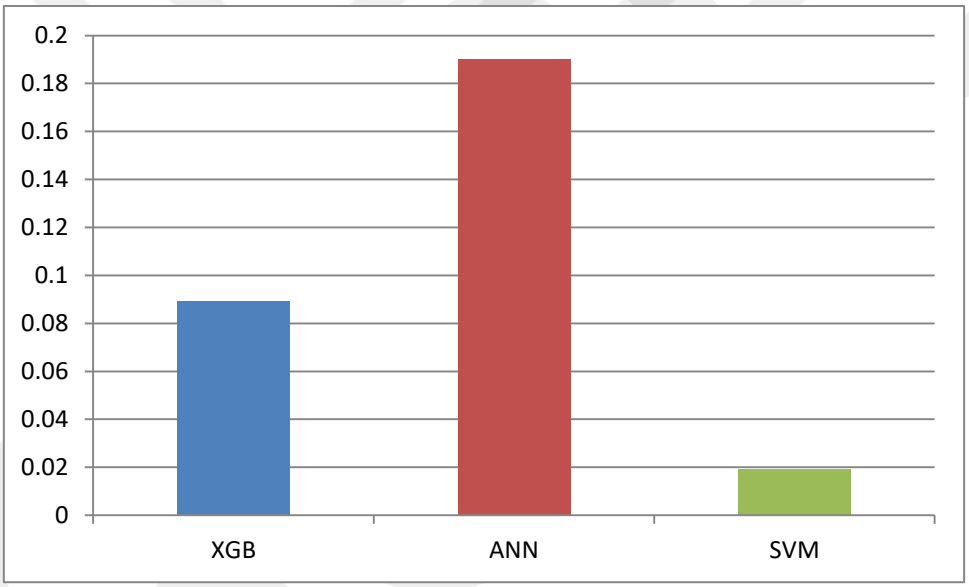


Figure 5.15 Performance comparison for no density features on validation data

Figure 5.16 Performance comparison for no density but other extra features on validation data

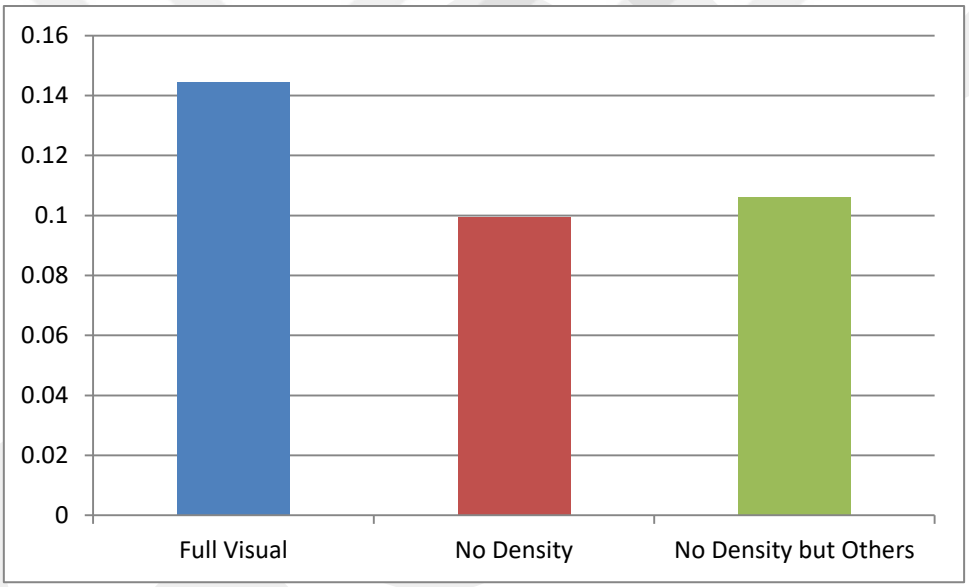


Figure 5.17 Overall comparison among feature sets on validation data

Table 5.9 Average training times in seconds

XGB	4.16
ANN	720
SVM	2700

5.7 Discussion and Conclusion

As can be seen in Tables 5.5, 5.6, 5.7, and Figures 5.4 through 5.9, 5.13 and 5.17; there is a clear distinction between performances of density based and other features both R and classification criteria wise. In other words, density based visual features simply outperform the others by boosting the prediction and classification success. The performance of no density but extra features added instead is better than that of no density features. Of course this is understandable as the former has more features. The best performing method is SVM, and XGB performs better than ANN. The most predictable FOREX pair turns out to be EURCAD, while AUDUSD comes second, and EURUSD is the least predictable. This is also interesting as EURUSD is the most traded FOREX currency pair all over the world. As per training times shown in Table 6.8, the performance of XGB classifier is outstanding, and the worst training time performance belongs to the SVM classifier.

XGB method is not a black box unlike ANN and SVM, that is to say the results achieved by XGB classifier, can be more easily explained. An analysis on feature importances of the XGB model is illustrated in Figure 5.18. As the illustration implies, 5 of the most important 8 features out of the total 24 are density variables. This also shows the superiority of the density based visual features.

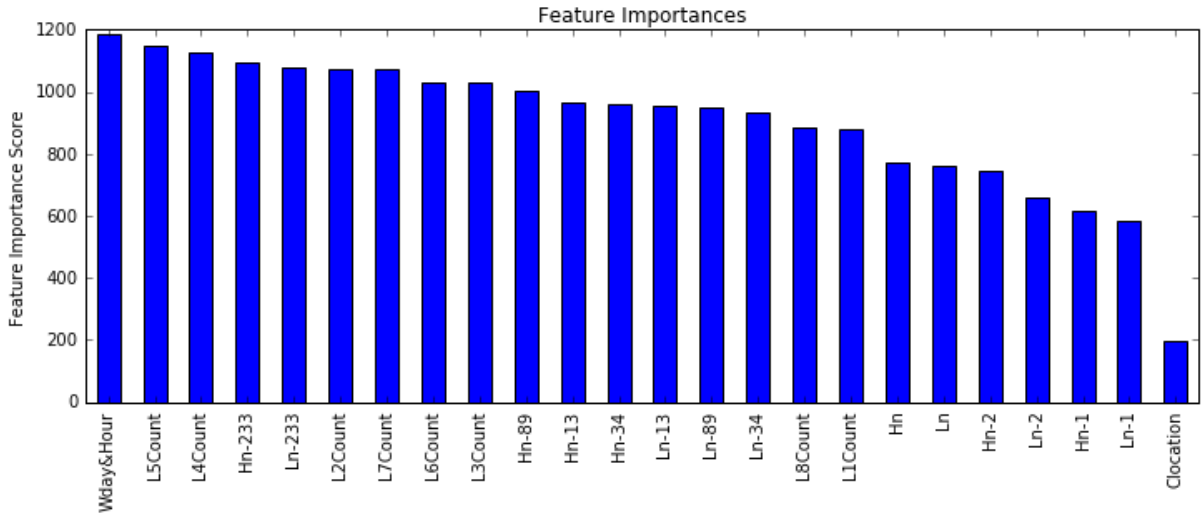


Figure 5.18 Feature importances of XGB classifier

On the other hand, as seen in Table 5.8, there is a sharp decrease in the performances of all methods over the validation data. This may be due to some overfitting because of intertwining past and future data belonging to the model building period. It could also stem from the ever changing behavior of financial markets. However, density based visual features still perform better than the others. As per the performances of different methods, the list is reversed: ANN performs best, XGB comes second, and SVM third over the validation period.

In the next chapter; development of a trading strategy depending on the density based visual features is attempted. This work includes profitability along with predictability, using buy, sell, take profit and stop loss conditions.

Consequently, by looking at prediction and classification performances of density based visual features, it is fair to say that they could open another window to Finance Vision.

CHAPTER VI

PROFITABILITY ANALYSIS

6.1 Challenges in Building Profitable Systems over Financial Instruments

On the collected data of chosen financial instruments; predictability comparison and predictability enhancement is completed so far, in the previous chapters. Consequently FOREX is found to be the most predictable financial instrument and 1-hour is suggested to be the most promising time period for a profitability analysis. A predictability enhancement work is done in Chapter 5 using this 1-hour frequency. In this chapter we attempt to come up with a profitable system using all these findings and optimizing buy, sell, take profit, stop loss, etc. conditions.

6.1.1 Dynamic and Chaotic Nature of Financial Markets

As discussed earlier, financial markets are highly non-linear and exhibit chaotic-like behavior. Therefore any trading system relying on a linear prediction model is most likely to fail. A non-linear but static model is also likely to fail, because the structure of the markets and the patterns observed also change in time. Such a static model could yield profitable results for a certain time frame, and yet run into losses in time, and become not profitable at all [71].

Employing AI and machine learning, as it is done in this thesis, could be a way around this problem, as new data comes in, the models could be re-trained and re-optimized at certain periods. It is not quite reasonable to claim that a profitable system would stay profitable for the next ten years for example. Nevertheless, all measures should be taken as to build a *robust* model with as much *general validity* as possible. These measures should be taken in both machine learning, especially through decent tests, out of range validation and also during optimization of buy/sell conditions.

6.1.2 Commissions, Spreads, Slippages and Hidden Trading Costs

If there were no commissions, ask-bid price differences (*spreads* in FOREX jargon) and other extra costs, trading financial instruments profitably would be much easier. One should aim much higher profits than these costs in order to have a sound and profitable system. Otherwise small profits would be swept away by the hidden costs. This is a challenging task as the higher the profit targets, the harder it is to reach them.

That is why, even if the *paper trading* seems to be profitable in theory, in the real world trading case it is usually the opposite. Within this profitability analysis, we try to account for all hidden costs in optimizations and simulations.

6.2 Data Used

As discussed earlier; following an extensive predictability comparison, the most predictable instrument is found to be FOREX. Similarly, the most predictable frequencies are turned out to be 1-minute and 5-minute. However, neither the predictability performance of 1-minute data nor the predictability performance of 5-minute data is much better than those of 1-hour or 4-hour data. Therefore, from the investor's point of view, trading decisions based on 1-hour or 4-hour data could even be more profitable, considering commissions and spread margins are usually more disadvantageous in higher frequencies like 5-minute or 1-minute with lower profit targets. Hence; as an optimum frequency, 1-hour data is chosen for profitability analysis just as predictability enhancement. Profitability analysis is done on one FOREX pair. In fact, in 1-hour data the most predictable currency pair is EURCAD, and the second is AUDUSD as shown in Table 3.1. However, spread costs for EURCAD is more than twice as high as AUDUSD [72]. Therefore, for the reasons explained in 6.1.2 AUDUSD is chosen for the profitability analysis. The data cover the five and a half year period between 2011 and 2016 with approximately 33000 data points. The 5 year period between 2011 and 2015 is used for training and testing the ANN model. The 3 year period between 2013 and 2015 is used for optimizing the trading system. As per out of range validation data for the trading system, the 6 month period between January and July 2016 is chosen.

6.3 Training the Prediction Model

The main idea behind building a profitable trading system is to use the best model found in predictability enhancement analysis, and add the trading strategy on top of that. Considering both training/testing and out of range validation period performances, ANN is chosen as the most stable prediction/classification tool. An ANN model is constructed with a multi-layered feed forward network, having 2 hidden layers with 50 and 8 neurons, respectively. Hyperbolic tangent is chosen as activation function. The network is trained by back propagation gradient descent algorithm using density based Finance Vision features as in Chapter 5. On the other hand, its output is arranged to vary between [-1, 1], "-1" denoting down, and "1" up trend, instead of [0, 1].

6.4 Platform

6.4.1 User Interface

MetaTrader 4 platform is used for building; optimization and simulation of the trading system (see Figure 6.1). It is one of the most popular trading platforms, and is used for trading FOREX, analyzing financial markets with the help of many built-in technical indicators, and coding and running trading strategies live with the help of its *Expert Advisors*. It has Windows, Mac OS, and Linux versions for personal computers, as well as iPhone, iPad, and Android versions for smartphones and tablets [73, 74].

6.4.2 Coding Platform

MetaTrader tool has an integrated programming language called mql4, in order to provide developers and traders with a coding platform for realizing their trading strategies live. It is a C and C++ like language which has a library with buy, sell, take profit, stop loss, etc. commands [75]. It does not have an ANN library, therefore the ANN models built with Neurosolutions software has to be migrated to mql4. This process is not very easy and

involves some reverse engineering. A sample part of an Expert Advisor coded for our model is shown in Figure 6.2.

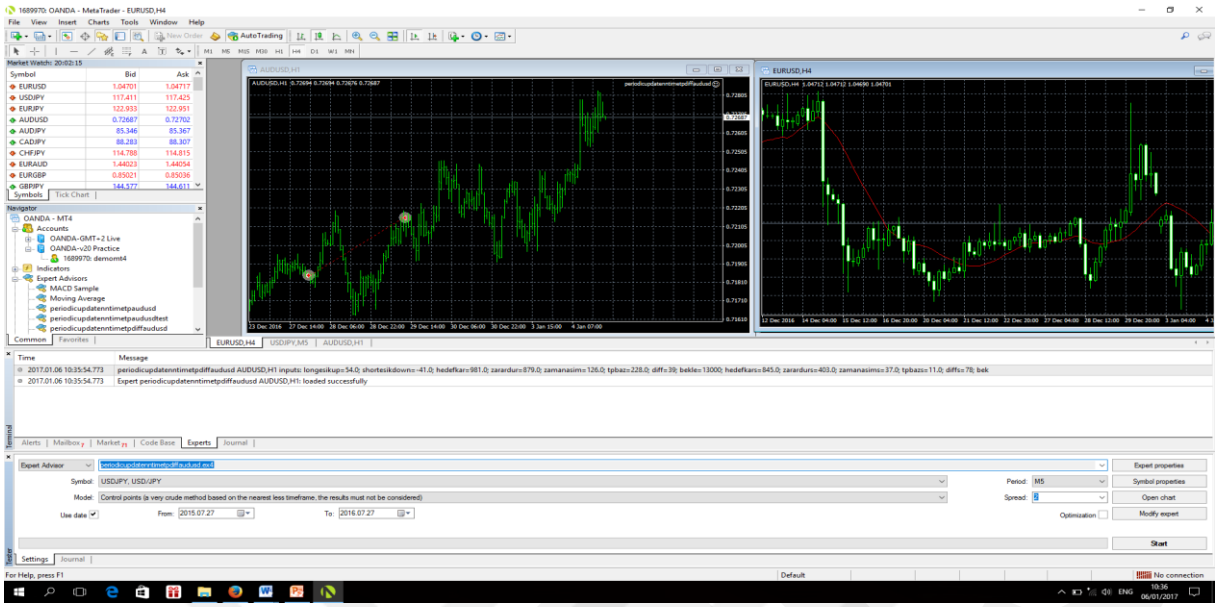


Figure 6.1 Metatrader 4 trading platform

6.5 Trading Strategy

6.5.1 Trading Basics

Long Position: A certain amount of a financial instrument (a FOREX pair in our case) is bought at a certain price, with the hope to be sold at a higher price.

Short Position: A certain amount of a financial instrument (a FOREX pair in our case) is sold at a certain price, with the hope to be sold at a lower price to close the position. It is opposite of long position.

Take Profit: Closing an open position with a certain amount of profit, for a long position at a higher price and at a lower price than the opening for a short position.

Stop Loss: Closing an open position with a certain amount of loss in order not to incur more loss, for a long position at a lower price and at a higher price than the opening for a short position.

Leverage: This is usually associated with FOREX markets, meaning one can use several times his/her capital to open positions. The leverage could be up to 50, or with some brokers, even up to 100 or 200 times one's capital. It is actually borrowing money. However, too high leverage margins usually have a *gambling* effect, and may cause loss of all capital [76]. In order to avoid making things more complicated by going into *money management* issues, leverage is not used in our simulations or optimizations.

If the price of an instrument is predicted to go up, a long position should be opened. On the other hand, if the price of an instrument is predicted to go down, a short position should be opened. FOREX market gives this flexibility to open both long and short positions. In other

markets, opening short positions is not that easy and is usually subject to certain restrictions. Using both long and short positions is an advantage for the trader since the markets sometimes have up and other times down trends.

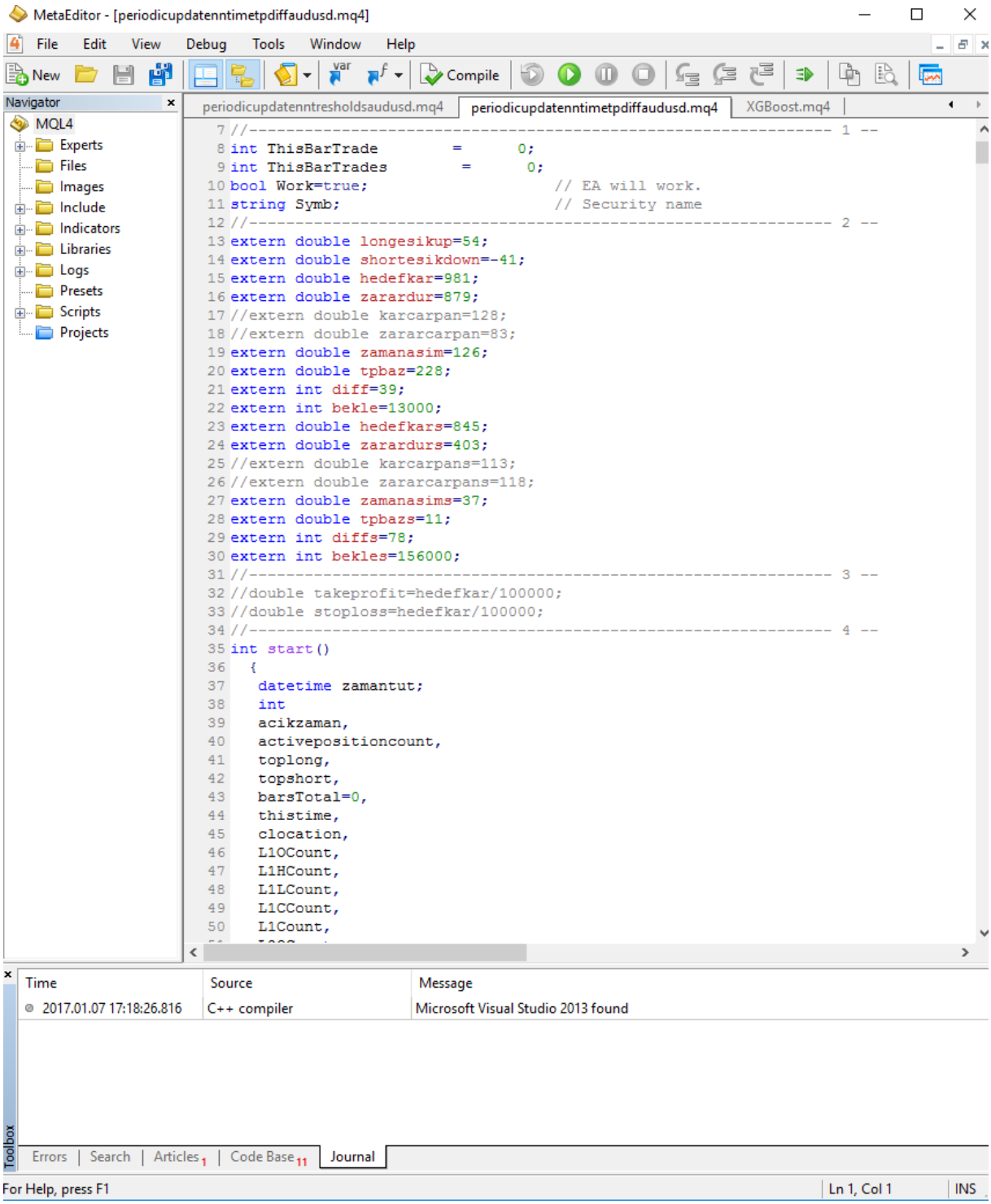


Figure 6.2 mql4 coding platform

6.5.2 Logic behind the Strategy

If our enhanced prediction model was perfect and could make precise predictions at all times, only thing to do would be enter long positions when the model output is positive, enter short positions when the output is negative, and close the positions when a certain profit level is reached. However, in practice a model that works at all times without a need to be tuned is not quite reasonable or possible. Therefore there is a need for optimizing buy and sell thresholds of the model output, as well as optimizing take profit and stop loss, etc.

6.5.3 Optimization Parameters

As we intend to open both long and short positions; there should be two different optimization values for each variable: one value for long positions, and another for short positions. Optimization parameters are explained as follows:

ANN Output Threshold: The value that varies in the range [-1, 1] is used as a threshold for the decision to open either a long or a short position. The values are magnified to the range [-100, 100] for the sake of easier optimization.

ANN Output Change: If the ANN output changes more than a certain amount, the position is closed. This change amount is in negative direction for a long positon, and in positive direction for a short position.

Take Profit Target: The highest targeted profit amount. In a long position it is above the position opening price, while in a short position it is below it.

Stop Loss: Maximum amount of loss to be incurred. If this amount of loss is experienced the position is closed. In a long position it is below the position opening price, while in a short position it is above it.

Position Time Out: Maximum amount of time in hours for a position to last. When this limit is reached the position is closed whether it is profitable or not.

Base Profit: This value is the minimum amount of profit allowed before reaching the Take Profit Target. Take Profit is arranged dynamically in our trading strategy such that; if a certain smaller level of profit is reached earlier this can be realized and the position is closed. This amount is greater than or equal to the Base Profit and smaller than the Take Profit Target, and is proportional to the Position Time Out. This strategy is related with the *opportunity costs* and higher profits in unit time.

Waiting Time after Loss: In the FOREX market, the price movements can occur in long trends (up or down) at times. There is no point in trying to open a long position when in a long term down trend, and trying to open a short position in a strong up trend. Also ANN output might be stuck in a wrong signal somehow. Therefore if the last trade is a loss, a waiting time of several minutes/hours is applied before opening a new position.

The strategy is coded in mql4 as a *trading robot*, alias Expert Advisor in MetaTrader terminology. The trained ANN weights and parameters are hard coded. All the computations can be inspected in the approximately 1500 lines of code which is given in Appendix A.

6.6 Optimization, Simulation, Tests and Results

Optimization is run for the 3 year period between 2013 and 2015 as mentioned earlier, with the parameters explained in 6.5.3. Instead of an exhaustive one, a rough optimization is carried out, in which, instead of using every *tick* of data, only control points are used. A detailed and very much fine-tuned optimization would lead to over-fitting; however, a crude optimization would yield better generalization. Among the profitable combinations, the most *robust* one is picked. For robustness these are considered: a high *profit factor*, a low *maximal drawdown*, and balanced average win or loss amounts for each trade. The performance report of the picked trading strategy is given in Figure 6.3 for the optimization period and Figure 6.4 for the out of range validation period. Again the validation period is between January and July 2016 as mentioned before. For the simulation of the trading strategy in the out of range validation period, every tick data is used which is the most precise method in MetaTrader. To test the robustness of the strategy, the same model with the exact same parameters is also run on a different currency pair, i.e. EURCAD for the validation period, and the corresponding report is given in Figure 6.5.

Strategy Tester Report

periodicupdatenntimetpdiffaudusd

OANDA-v20 Practice (Build 1031)

Symbol	AUDUSD (AUD/USD)				
Period	1 Hour (H1) 2013.01.02 01:00 - 2015.12.31 00:00 (2013.01.01 - 2015.12.31)				
Model	**Control points (a very crude method, the results must not be considered)**				
Parameters	longesikup=54; shortesikdown=-41; hedefkar=981; zarardur=879; zamanasim=126; tpbaz=228; diff=39; bekle=13000; hedefkars=845; zarardurs=403; zamanasims=37; tpbazs=11; diffs=78; bekles=156000;				
Bars in test	19485	Ticks modelled	460298	Modelling quality	n/a
Mismatched charts errors	0				
Initial deposit	10000.00			Spread	20
Total net profit	3977.30	Gross profit	7960.20	Gross loss	-3982.90
Profit factor	2.00	Expected payoff	11.27		
Absolute drawdown	58.40	Maximal drawdown	370.70 (2.73%)	Relative drawdown	2.73% (370.70)
Total trades	353	Short positions (won %)	67 (67.16%)	Long positions (won %)	286 (73.08%)
		Profit trades (% of total)	254 (71.95%)	Loss trades (% of total)	99 (28.05%)
	Largest	profit trade	112.60	loss trade	-98.30
	Average	profit trade	31.34	loss trade	-40.23
	Maximum	consecutive wins (profit in money)	11 (378.60)	consecutive losses (loss in money)	5 (-280.60)
	Maximal	consecutive profit (count of wins)	392.60 (8)	consecutive loss (count of losses)	-280.60 (5)
	Average	consecutive wins	3	consecutive losses	1

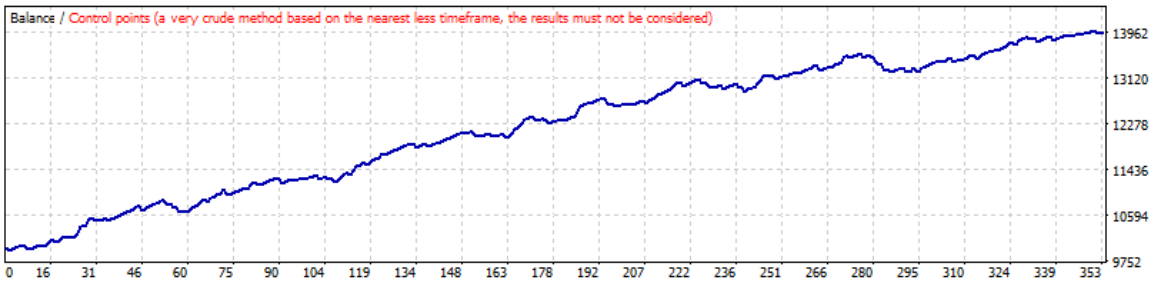


Figure 6.3 Strategy test report for the optimization period

Strategy Tester Report

periodicupdatenntimetpdiffaudusd

OANDA-v20 Practice (Build 1031)

Symbol		AUDUSD (AUD/USD)			
Period		1 Hour (H1) 2016.01.04 02:00 - 2016.07.26 23:00 (2016.01.01 - 2016.07.27)			
Model		Every tick (the most precise method based on all available least timeframes)			
Parameters		longesikup=54; shortesikdown=-41; hedefkar=981; zarardur=879; zamanasim=126; tpbaz=228; diff=39; bekle=13000; hedefkars=845; zarardurs=403; zamanasims=37; tpbazs=11; diffs=78; bekles=156000;			
Bars in test	4526	Ticks modelled	13384696	Modelling quality	90.00%
Mismatched charts errors	19				
Initial deposit	10000.00			Spread	20
Total net profit	679.50	Gross profit	2028.20	Gross loss	-1348.70
Profit factor	1.50	Expected payoff	5.86		
Absolute drawdown	34.20	Maximal drawdown	412.70 (3.89%)	Relative drawdown	3.89% (412.70)
Total trades	116	Short positions (won %)	12 (75.00%)	Long positions (won %)	104 (69.23%)
		Profit trades (% of total)	81 (69.83%)	Loss trades (% of total)	35 (30.17%)
	Largest	profit trade	82.70	loss trade	-88.20
	Average	profit trade	25.04	loss trade	-38.53
	Maximum	consecutive wins (profit in money)	15 (329.90)	consecutive losses (loss in money)	8 (-392.10)
	Maximal	consecutive profit (count of wins)	442.60 (12)	consecutive loss (count of losses)	-392.10 (8)
	Average	consecutive wins	4	consecutive losses	2

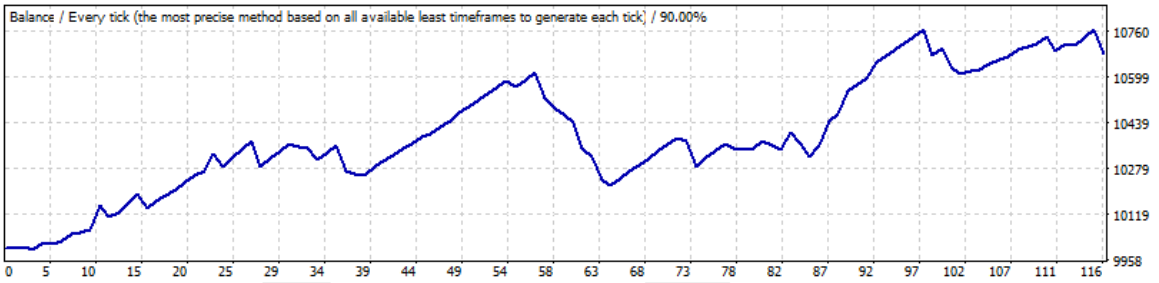


Figure 6.4 Strategy test report for the validation period

From the Figures 6.3 and 6.4, the performance for the out of range validation period is quite similar to that of optimization period. A monthly approximately 1.13% net profitable trading strategy is achieved in out-of-range validation period, while a 1.1% at the optimization period. The maximal drawdown is 2.73% in the optimization period, and it is 3.89% in the validation period. As per the third simulation test, in which a different currency pair (EURUSD) is used with the same parameters and ANN weights. The ANN is built with AUDUSD data, the trading strategy is optimized for AUDUSD, and all these are used on a totally different currency. The results are shown in Figure 6.5. There is a 0.17% monthly net profit, and the maximal drawdown is still under 3.14%.

For the optimization period, the success rate of all trades is 71.95%, while this rate is 69.83% in the validation period for AUDUSD and 66.96% for EURCAD (no parameters are trained or optimized for EURCAD).

Strategy Tester Report

periodicupdatenntimetpdiffaudusd

OANDA-v20 Practice (Build 1031)

Symbol		EURCAD (EUR/CAD)			
Period		1 Hour (H1) 2016.01.04 00:00 - 2016.07.26 23:00 (2016.01.01 - 2016.07.27)			
Model		Every tick (the most precise method based on all available least timeframes)			
Parameters		longesikup=54; shortesikdown=-41; hedefkar=981; zarardur=879; zamanasim=126; tpbaz=228; diff=39; bekle=13000; hedefkars=845; zarardurs=403; zamanasims=37; tpbazs=11; diffs=78; bekles=156000;			
Bars in test	4528	Ticks modelled	19706872	Modelling quality	90.00%
Mismatched charts errors	0				
Initial deposit	10000.00			Spread	30
Total net profit	103.93	Gross profit	1966.85	Gross loss	-1862.92
Profit factor	1.06	Expected payoff	0.90		
Absolute drawdown	192.73	Maximal drawdown	327.99 (3.14%)	Relative drawdown	3.14% (327.99)
Total trades	115	Short positions (won %)	16 (50.00%)	Long positions (won %)	99 (69.70%)
		Profit trades (% of total)	77 (66.96%)	Loss trades (% of total)	38 (33.04%)
	Largest	profit trade	137.39	loss trade	-87.44
	Average	profit trade	25.54	loss trade	-49.02
	Maximum	consecutive wins (profit in money)	8 (164.15)	consecutive losses (loss in money)	3 (-128.46)
	Maximal	consecutive profit (count of wins)	274.16 (3)	consecutive loss (count of losses)	-134.16 (2)
	Average	consecutive wins	3	consecutive losses	1

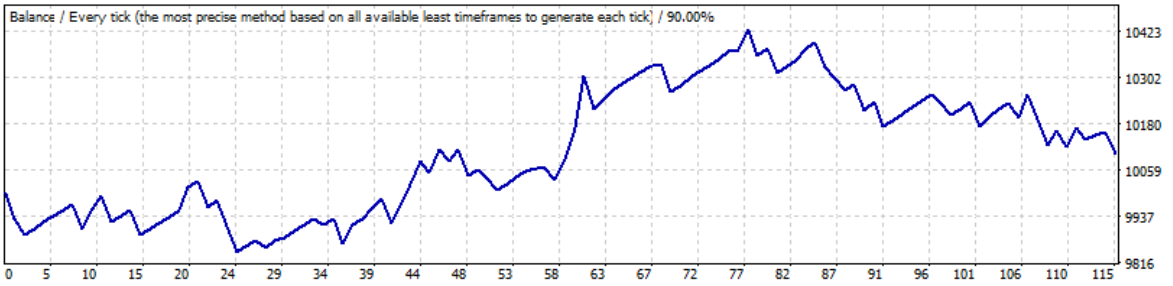


Figure 6.5 Strategy test report for EURCAD with AUDUSD parameters

CHAPTER VII

DISCUSSION AND CONCLUSION

In this thesis, a predictability analysis is made by comparing the predictability performances of different financial instruments first. In this analysis FOREX is found to be the most predictable financial instrument. Then predictability is enhanced with the introduction of Finance Vision features. Finally a profitable trading strategy is attempted using all the prior knowledge built in this thesis. The optimization, simulation, and tests are executed on AUDUSD FOREX pair; also a robustness simulation test on EURCAD is carried out.

As it can be seen in Figure 6.4, the performance in out of range validation period is quite similar to that of optimization period (see Figure 6.3 as well), naturally a little worse. A monthly approximately 1.13% net profitable trading strategy is achieved in out-of-range validation period, which is even slightly better than 1.1% of the optimization period. Another interestingly positive feature of our strategy is the maximal drawdown which is less than 4% in the validation period. Although no leverage is used in our simulations, the strategy is very appropriate for the usage of leverage, having such low drawdown levels. This can further raise profits.

As per further robustness of the trading strategy, another interesting simulation test is carried out. This time a different currency pair (EURUSD) is used with the same parameters and ANN weights. The ANN is built with AUDUSD data, the trading strategy is optimized for AUDUSD, and all these are used on a totally different currency. The same validation period is used for this simulation, and the results are not bad at all as shown in Figure 6.5. There is even a little net profit (0.17% monthly), and more importantly maximal drawdown is still under 4%. This is a sound proof that our trading strategy is indeed robust, and should yield good profits for other currencies as well, after customized training and optimization. With the addition of more currencies to trade and also applying reasonable leverage, much higher levels of profit seem achievable.

One should bear in mind; however, financial markets are of very dynamic nature, and all models and strategies must be updated or fine-tuned from time to time. Nevertheless we have shown that reasonable profits are achievable even in the chaotic financial markets.

REFERENCES

1. Ott, E., C. Grebogy, and J. Yorke. (1990) "Controlling Chaos", *Physical Review Letters*, 64, No.11, 1196-1199.

2. Karacor, A.G. (2002). "Extraction of Local Linear Model of a Chaotic System via Artificial Neural Networks" Bogazici University, Systems and Control Engineering, *Master Thesis*.

3. Chandler M. (2013). http://www.economonitor.com/blog/2013/09/bis-daily-fx-turnover-averages-5-3-trillion/, downloaded date: 15.08,2016.

4. Konstantinidi E., Skiadopoulos G. (2011). Are VIX futures prices predictable? An empirical investigation. *International Journal of Forecasting* 27, 543–560.

5. Bossaerts P., Hillion P. (1999). Implementing Statistical Criteria to Select Return Forecasting Models: What Do We Learn? *Review of Financial Studies*, 12(2), 405−428.

6. Goyal A., Welch I. (2008). A Comprehensive Look at the Empirical Performance of Equity Premium Prediction. *Review of Financial Studies*. 21 Issue 4, 1455-1508.

7. Hartzmark M. L. (1987). Returns to individual traders of futures: Aggregate results. *Journal of Political Economy*, 95, 1292–1306.

8. Kho B. C. (1996). Time-varying Risk Premia, Volatility, and Technical Trading Rule Profits: Evidence from Foreign Currency Futures Markets. *Journal of Financial Economics*, 41, 249–290.

9. Strozzi F., Zaldivar J.M. (2005), Non-linear Forecasting in High-frequency Financial Time Series. *Physica A*, 353 463–479.

10. Taylor S. J. (1992). Rewards Available to Currency Futures Speculators: Compensation for Risk or Evidence of Inefficient Pricing? *Economic Record*, 68(Supplement), 105–116.

11. Wang C. (2004). Futures Trading Activity and Predictable Foreign Exchange Market Movements. *Journal of Banking and Finance*, 28, 1023–1041.

12. Yoo J., Maddala G. S. (1991). Risk Premia and Price Volatility in Futures Markets. *Journal of Futures Markets*, 11, 165–177.

13. Campbell J. Y., Thompson S. (2008). Predicting the Equity Premium Out of Sample: Can Anything Beat the Historical Average? *Review of Financial Studies*. 21, Issue 4, 1509-1531.

14. Zunino L., Tabak B.M., Serinaldi F., Zanin M., Perez D.G., Rosso O.A. (2011). Commodity Predictability Analysis with a Permutation Information Theory Approach. *Physica A*, 390, 876–890.

15. Taylor, S. J. (2008). Modelling financial time series. *World Scientific Publishing*.

16. Azoff, E. M. (1994). Neural network time series forecasting of financial markets. *John Wiley & Sons, Inc.*

17. Tsay, R. S. (2005). Analysis of financial time series (Vol. 543). *John Wiley & Sons*.

18. Mills, T. C., & Markellos, R. N. (2008). The econometric modelling of financial time series. *Cambridge University Press*.

19. Caporale, G. M., Gil-Alana, L., & Plastun, A. (2016). Searching for inefficiencies in exchange rate dynamics. *Computational Economics*, 1-28.

20. Teneng, D. (2013). Outperforming the naïve Random Walk forecast of foreign exchange daily closing prices using Variance Gamma and normal inverse Gaussian Levy processes, *Mathematical Methods in Economics 2013 Conference Paper.*

21. Calin, A. C. (2015). The impact of trade announcements on financial markets. an event study analysis. *Romanian Journal of Economic Forecasting*, 18(2), 81.

22. Parida, A. K., Bisoi, R., Dash, P. K., & Mishra, S. (2015). Financial time series prediction using a hybrid functional link fuzzy neural network trained by adaptive unscented kalman filter. In Power, *Communication and Information Technology Conference* (PCITC), 2015 IEEE (pp. 568-575). IEEE.

23. Calin, A. C. (2015). Eloquence is The Key–the Impact of Monetary Policy Speeches on Exchange Rate Volatility. *The Romanian Economic Journal*, (56), 3-18.

24. Hunter, J., & Ali, F. M. (2014). Money demand instability and real exchange rate persistence in the monetary model of USD–JPY exchange rate. *Economic Modelling*, 40, 42-51.

25. Inci, A. C., & Seyhun, H. N. (2015). Degree of Integration between Brent Oil Spot and Futures Markets: Intraday Evidence. *The Business & Management Review*, 6(4), 59.

26. Si, J., Mukherjee, A., Liu, B., Pan, S. J., Li, Q., & Li, H. (2014). Exploiting Social Relations and Sentiment for Stock Prediction. In *EMNLP* (14, 1139-1145).

27. Alanyali, M., Moat, H. S., & Preis, T. (2013). Quantifying the relationship between financial news and the stock market, *Scientific Reports 3*, Article number: 3578.

28. Cheng W., McClain B.W., Kelly C. (1997). Artificial Neural Networks Make Their Mark as a Powerful Tool for Investors. *Review of Business* 4 –9.

29. Dash R., Dash P.K., Bisoi R. (2014). A self-adaptive differential harmony search based optimized extreme learning machine for financial time series prediction. *Swarm and Evolutionary Computation* 19, 25-42.

30. Dutta S., Shekhar S. (1988). Bond-rating: a Non-conservative Application of Neural Networks. *Proceedings of the IEEE International Conference on Neural Networks 2* pp 443–450.

31. Lam M. (2004). Neural Network Techniques for Financial Performance Prediction: Integrating Fundamental and Technical Analysis. *Decision Support Systems*, 37, 567– 581.

32. Wedding D.K., Cios K.J. (1996). Time series forecasting by combining RBF networks, certainty factors, and the Box-Jenkins model. *Neurocomputing* 10, Issue 2, 149-168.

33. Xi L., Muzhou H., Lee M.H., Li J., Wei D., Hai H., Wu Y. (2014). A new constructive neural network method for noise processing and its application on stock market prediction. *Applied Soft Computing* 15, 57-66.

34. Yu L., Lai K.K., Wang S. (2008). Multistage RBF neural network ensemble learning for exchange rates forecasting. *Neurocomputing*, 71, Issues 16-18, 3295-3302.

35. Choudhury S. et.al. (2014). A real time clustering and SVM based price-volatility prediction for optimal trading strategy, *Neurocomputing*, 131, 419-426.

36. Fenghua W., Jihong X., Zhifang H., Xu G. (2014). Stock Price Prediction based on SSA and SVM, *Procedia Computer Science*, 31, 625-631.

37. Kumar D, Meghwani S. S., Thakur M. (2016). Proximal support vector machine based hybrid prediction models for trend forecasting in financial markets, *Journal of Computational Science*, 17, Part 1, 1-13.

38. Nayak R. K., Mishra D., Rath A. K. (2015). A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices, *Applied Soft Computing*, 35, 670-680.

39. Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). ACM.

40. Serrano J., Hoesli M. (2010), Are Securitized Real Estate Returns more Predictable than Stock Returns?, *Journal of Real Estate Finance & Economics*, 41, 170–192.

41. Armstrong, J. S. (2011). Illusions in regression analysis.

42. Ding, B. N. K. L. (1996). Neural network fundamentals with graphs, algorithms and applications. *Mac Graw-Hill*.

43. Hassoun, M. H. (1995). Fundamentals of artificial neural networks. *MIT Press*.

44. Aktepe A., Ersoz S. (2012). A Quantitative Performance Evaluation Model Based on a Job Satisfaction - Performance Matrix and Application in a Manufacturing Company. *International Journal of Industrial Engineering*, 19(6), 264 – 277.

45. Domínguez, E., & Munoz, J. (2008). A neural model for the p-median problem. *Computers & Operations Research*, 35(2), 404-416.

46. Chang, L. Y. (2005). Analysis of freeway accident frequencies: negative binomial regression versus artificial neural network. *Safety science*, 43(8), 541-557.

47. Hornik K., Stinchcombe M., White H. (1989). Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, 2, 359-366.

48. Demuth, H. B., Beale, M. H., De Jess, O., & Hagan, M. T. (2014). Neural network design. Martin Hagan, *Oklahoma State University Press*.

49. Orr, M., Hallam, J., Murray, A., & Leonard, T. (2000). Assessing rbf networks using delve. *International Journal of Neural Systems*, 10(05), 397-415.

50. Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, 49(11), 1225-1231.

51. Kecman, V. (2004). Support vector machines basics. School of Engineering, *University of Auckland Press*.

52. Chen, T., & He, T. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*.

53. Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, 29(4), 309-317.

54. Wang, W. C., Chau, K. W., Cheng, C. T., & Qiu, L. (2009). A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. *Journal of hydrology*, 374(3), 294-306.

55. Diao, R., Vittal, V., & Logic, N. (2010). Design of a real-time security assessment tool for situational awareness enhancement in modern power systems. *IEEE Transactions on Power Systems*, 25(2), 957-965.

56. Duda, R. O., Hart, P. E., & Stork, D. G. (1973). Pattern classification (Vol. 2). *New York: Wiley*.

57. Gonzalez, R. C., & Woods, R. E. (2007). Image processing. Digital image processing, 2, *Gatesmark Publishing*.

58. Gopalakrishnan, J. (1999). Market Profile Basics. Technical Analysis of Stocks and *Commodities-Magazine Edition-*, 17, 17-22.

59. LeBaron, B., & Weigend, A. S. (1998). A bootstrap evaluation of the effect of data splitting on financial time series. *IEEE Transactions on Neural Networks*, 9(1), 213-220.

60. NeuroSolutions Software, NeuroDimension Inc. (1994-2015), www.neurosolutions.com.

61. Rossum G. (1995). Python tutorial, Technical Report CS-R9526, *Centrum voor Wiskunde en Informatica (CWI)*

62. Lloyd, T. K. (2013). Using Moving Averages and Relative Strength Performance to Beat the Index: Relative Strength Index, Money Flow Index, Keltner Channels, and Standard Deviation, with Apple Exhibits. Successful Stock Signals for Traders and Portfolio Managers: Integrating Technical Analysis with Fundamentals to Improve Performance 1-14, *Wiley Publishing*.

63. Bouri, E. (2015). Oil volatility shocks and the stock markets of oil-importing MENA economies: A tale from the financial crisis. *Energy Economics*, 51, 590-598.

64. Billings, S. A. (2013). Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. *John Wiley & Sons*.

65. Lee, D., & Liu, D. (2014). Monte-Carlo Simulations of GARCH, GJR-GARCH and constant volatility on NASDAQ-500 and the 10 year treasury. *Duke University Technical Report*.

66. Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In *Australasian Joint Conference on Artificial Intelligence* (pp. 1015-1021). Springer Berlin Heidelberg.

67. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437.

68. Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A., & Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5), 412-424.

69. Gu, Q., Zhu, L., & Cai, Z. (2009). Evaluation measures of the classification performance of imbalanced data sets. In *International Symposium on Intelligence Computation and Applications* (pp. 461-471). Springer Berlin Heidelberg.

70. Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1), 29-36.

71. LeBaron, B. (1994). Chaos and nonlinear forecastability in economics and finance. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 348(1688), 397-404.

72. Paukste, A., & Raudys, A. (2013). Intraday forex bid/ask spread patterns-Analysis and forecasting. In *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2013 IEEE Conference* on (pp. 118-121). IEEE.

73. Wong, M., Chan, L., & Wong, W. K. (2014). Automated Gold Trading with MT4, *Finamatrix Journal* 1-6.

74. MetaTrader 4 Software, MetaQuotes Software Corp. (2001 – 2016).

75. Ibrahim, A. E. M. (2014). Evolutionary Approach to Forex Expert Advisor Generation. *Intelligent Information Management*, 2014.

76. Baxter, N. D. (1967). Leverage, risk of ruin and the cost of capital. the Journal of Finance, 22(3), 395-403.

APPENDIX A

EXPERT ADVISOR MQL4 CODE

```
//+------------------------------------------------------------------+
int ThisBarTrade       =     0;
int ThisBarTrades      =     0;
bool Work=true;                // EA will work.
string Symb;                   // Security name
//+------------------------------------------------------------------+
extern double longesikup=54;
extern double shortesikdown=-41;
extern double hedefkar=981;
extern double zarardur=879;
extern double zamanasim=126;
extern double tpbaz=228;
extern int diff=39;
extern int bekle=13000;
extern double hedefkars=845;
extern double zarardurs=403;
extern double zamanasims=37;
extern double tpbazs=11;
extern int diffs=78;
extern int bekles=156000;
//+------------------------------------------------------------------+
int start()
  {
   datetime zamantut;
   int
   acikzaman,
   activepositioncount,
   toplong,
   topshort,
   barsTotal=0,
   thistime,
   clocation,
   L1OCount,
   L1HCount,
   L1LCount,
   L1CCount,
   L1Count,
   L2OCount,
   L2HCount,
   L2LCount,
   L2CCount,
   L2Count,
   L3OCount,
```

L3HCount,
L3LCount,
L3CCount,
L3Count,
L4OCount,
L4HCount,
L4LCount,
L4CCount,
L4Count,
L5OCount,
L5HCount,
L5LCount,
L5CCount,
L5Count,
L6OCount,
L6HCount,
L6LCount,
L6CCount,
L6Count,
L7OCount,
L7HCount,
L7LCount,
L7CCount,
L7Count,
L8OCount,
L8HCount,
L8LCount,
L8CCount,
L8Count,
yapma,
beklen,
zarar,
i, // Counter
j, // Counter
k, // Counter
ticket, // for orders
tk,
type,
Ticket; // Order number
double
fullsyn1 [5] [1300]={},
bias [5] [89]={},
ampin [25]={},
offin [25]={},
inp5plus [24]={},
tophid1 [89]={},
tophid2 [13]={},
tophid3 [5]={},
hid1_o [89]={},
hid2_o [13]={},

```
hid3_o [5]={},
ampdes,
offdes,
tophido_1,
hido_o,
htresh,
ltresh,
priceoffset,
takeprofit,
stoploss,
takeprofits,
stoplosss,
level0,
level1,
level2,
level3,
level4,
level5,
level6,
level7,
level8,
cevir,
gun,
saat,
longesikdown,
shortesikup,
Lots,
Lot=0.1,                  // Amount of lots in a selected order
Free,                     // Current free margin
Price,                    // Price of a selected order
psL,                      // Long position size
psS;                      // Short position size
bool
result,
mantiksal1,
mantiksal2,
mantiksal3,
mantiksal4,
mantiksal5,
//------------------------------------------------------------------
// Preliminary processing
if(Bars < 400)                  // Not enough bars
  {
   Alert("Not enough bars in the window. EA doesn't work.");
   return;                          // Exit start()
  }
if(Work==false)                        // Critical error
  {
   Alert("Critical error. EA doesn't work.");
   return;                          // Exit start()
```

```
    }
   if(AccountBalance() < 1000)                  // Not enough liquidity
    {
     Alert("Not enough balance. EA doesn't work.");
     return;                                   // Exit start()
    }
//------------------------------------------------------------------
  // ANN weights and parameters
  // Hidden Layer 1
fullsyn1[1,1]=-0.0146664564137322; fullsyn1[1,601]=0.216290493698992;
fullsyn1[1,2]=-0.0711669000424945; fullsyn1[1,602]=0.329139929474163;
fullsyn1[1,3]=0.345517883451332;   fullsyn1[1,603]=-0.389133001100423;
fullsyn1[1,4]=-0.073832059191969;  fullsyn1[1,604]=-0.0960621176451493;
fullsyn1[1,5]=0.148424920684566;   fullsyn1[1,605]=-0.169480505756089;
fullsyn1[1,6]=-0.65875999422223;   fullsyn1[1,606]=0.175803227231937;
fullsyn1[1,7]=0.528314760542512;   fullsyn1[1,607]=-0.647855665203261;
fullsyn1[1,8]=-0.725759335691857;  fullsyn1[1,608]=-0.398280352740293;
fullsyn1[1,9]=-0.740396348672831;  fullsyn1[1,609]=0.43083398081944;
fullsyn1[1,10]=1.106104504032;     fullsyn1[1,610]=0.0789815394216854;
fullsyn1[1,11]=-0.763335020246181; fullsyn1[1,611]=0.0703734340478749;
fullsyn1[1,12]=-0.229090594349718; fullsyn1[1,612]=-0.739168520575758;
fullsyn1[1,13]=0.335633888826666;  fullsyn1[1,613]=-0.29833230645788;
fullsyn1[1,14]=-0.376217875929678; fullsyn1[1,614]=0.265350850328109;
fullsyn1[1,15]=-0.0587897773322813;fullsyn1[1,615]=0.230471507690646;
fullsyn1[1,16]=-0.0396232692057697;fullsyn1[1,616]=0.491669018257967;
fullsyn1[1,17]=0.123821962533918;  fullsyn1[1,617]=-0.155348166584146;
fullsyn1[1,18]=-0.698561807357946; fullsyn1[1,618]=0.765863247001089;
fullsyn1[1,19]=-0.187125976686881; fullsyn1[1,619]=0.186113661189999;
fullsyn1[1,20]=0.0104544124787464;fullsyn1[1,620]=0.689704364189622;
fullsyn1[1,21]=-0.226529386680883; fullsyn1[1,621]=0.211002716222924;
fullsyn1[1,22]=0.614824902613758;  fullsyn1[1,622]=0.563454922338771;
fullsyn1[1,23]=-0.428862033403305; fullsyn1[1,623]=-0.172826471422602;
fullsyn1[1,24]=0.638065603224938;  fullsyn1[1,624]=1.14528615275978;
fullsyn1[1,25]=0.130904266078688;  fullsyn1[1,625]=-0.770726602646178;
fullsyn1[1,26]=0.683123151723382;  fullsyn1[1,626]=0.478459675843312;
fullsyn1[1,27]=-0.862200468609748; fullsyn1[1,627]=-1.15395123833285;
fullsyn1[1,28]=0.384601314808283;  fullsyn1[1,628]=-0.576173525725872;
fullsyn1[1,29]=0.899302815843989;  fullsyn1[1,629]=0.153472810495492;
fullsyn1[1,30]=-1.3493098612622;   fullsyn1[1,630]=-0.250322406669262;
fullsyn1[1,31]=-0.323680366938641; fullsyn1[1,631]=-0.15634417482068;
fullsyn1[1,32]=0.206916676591443;  fullsyn1[1,632]=0.66822993925591;
fullsyn1[1,33]=-0.772384051310449; fullsyn1[1,633]=-0.693109521601391;
fullsyn1[1,34]=-0.448031000029809; fullsyn1[1,634]=0.174890949032929;
fullsyn1[1,35]=-0.413735901067136; fullsyn1[1,635]=0.045239732134945;
fullsyn1[1,36]=0.301462228075818;  fullsyn1[1,636]=0.231137592455589;
fullsyn1[1,37]=-0.282488010239654; fullsyn1[1,637]=0.333314459701708;
fullsyn1[1,38]=0.100441966431264;  fullsyn1[1,638]=0.320036846851611;
fullsyn1[1,39]=-0.405876238897479; fullsyn1[1,639]=-0.0417085455943657;
fullsyn1[1,40]=-0.453566461603768; fullsyn1[1,640]=-0.0804572055423375;
fullsyn1[1,41]=-0.662522219011767; fullsyn1[1,641]=0.27331431949712;
```

fullsyn1[1,42]=0.137469270594556; fullsyn1[1,642]=0.18163843564086;
fullsyn1[1,43]=-1.26639630679153; fullsyn1[1,643]=0.363974692417446;
fullsyn1[1,44]=-0.395527771241392; fullsyn1[1,644]=0.561122663055776;
fullsyn1[1,45]=-0.582113506584513; fullsyn1[1,645]=0.563021894316458;
fullsyn1[1,46]=-0.673693681642738; fullsyn1[1,646]=-1.00069346956339;
fullsyn1[1,47]=-0.51540448288114; fullsyn1[1,647]=-0.180260226283108;
fullsyn1[1,48]=0.111662301769145; fullsyn1[1,648]=-1.46117492933409;
fullsyn1[1,49]=1.10847015077332; fullsyn1[1,649]=0.0601308978172228;
fullsyn1[1,50]=-0.372965982634767; fullsyn1[1,650]=0.457480239067054;
fullsyn1[1,51]=-0.279018698168269; fullsyn1[1,651]=0.549467887176667;
fullsyn1[1,52]=-0.460484941015639; fullsyn1[1,652]=0.736744627279843;
fullsyn1[1,53]=0.320818581230108; fullsyn1[1,653]=-0.630306030047624;
fullsyn1[1,54]=-0.0763558310079742;fullsyn1[1,654]=-0.493544227146508;
fullsyn1[1,55]=-0.52409335892931; fullsyn1[1,655]=0.818569012805;
fullsyn1[1,56]=-0.91958619937617; fullsyn1[1,656]=0.378708767462267;
fullsyn1[1,57]=0.58425105990495; fullsyn1[1,657]=-0.407321231555032;
fullsyn1[1,58]=0.0875972805797586;fullsyn1[1,658]=0.25779244012356;
fullsyn1[1,59]=-0.142348087753657; fullsyn1[1,659]=0.594774153101272;
fullsyn1[1,60]=0.332253902147803; fullsyn1[1,660]=-0.374424812226255;
fullsyn1[1,61]=-0.136061176772433; fullsyn1[1,661]=0.67411040216611;
fullsyn1[1,62]=-0.30760442694838; fullsyn1[1,662]=0.51190055950111;
fullsyn1[1,63]=-0.163957465478289; fullsyn1[1,663]=1.01490078468057;
fullsyn1[1,64]=0.348887825203424; fullsyn1[1,664]=0.781466089354379;
fullsyn1[1,65]=1.38045443034685; fullsyn1[1,665]=0.0458476768018029;
fullsyn1[1,66]=0.512521482476284; fullsyn1[1,666]=0.605613334025957;
fullsyn1[1,67]=-0.359636240234209; fullsyn1[1,667]=-0.576024672201139;
fullsyn1[1,68]=0.110255103595431; fullsyn1[1,668]=-0.153724404238559;
fullsyn1[1,69]=-0.699148200321825; fullsyn1[1,669]=-0.321105025127311;
fullsyn1[1,70]=-0.539263806197724; fullsyn1[1,670]=0.134115178619773;
fullsyn1[1,71]=0.330477538755095; fullsyn1[1,671]=-0.182466417389213;
fullsyn1[1,72]=-0.817809421321597; fullsyn1[1,672]=-0.361871364681041;
fullsyn1[1,73]=0.518640223210088; fullsyn1[1,673]=-0.863124030585008;
fullsyn1[1,74]=0.0168423187178567; fullsyn1[1,674]=0.800787694288182;
fullsyn1[1,75]=-0.234592737555269; fullsyn1[1,675]=0.323452553924414;
fullsyn1[1,76]=0.843315038781581; fullsyn1[1,676]=-0.395492453238945;
fullsyn1[1,77]=-0.252107783109466; fullsyn1[1,677]=0.797757427375338;
fullsyn1[1,78]=-0.704940602825417; fullsyn1[1,678]=0.950435888859631;
fullsyn1[1,79]=0.521122271213206; fullsyn1[1,679]=-0.418559015301545;
fullsyn1[1,80]=0.394408901274088; fullsyn1[1,680]=-0.159323750744125;
fullsyn1[1,81]=-0.63963918494725; fullsyn1[1,681]=-0.522150862956707;
fullsyn1[1,82]=0.0867556472319533;fullsyn1[1,682]=-0.66321167289179;
fullsyn1[1,83]=-0.147343348582728; fullsyn1[1,683]=0.51718820136565;
fullsyn1[1,84]=-0.330509820089163; fullsyn1[1,684]=-0.499181220102092;
fullsyn1[1,85]=-0.192536942070995; fullsyn1[1,685]=0.285678325995825;
fullsyn1[1,86]=-0.284437996410236; fullsyn1[1,686]=0.381182391443872;
fullsyn1[1,87]=-0.279701314192974; fullsyn1[1,687]=0.202637786432149;
fullsyn1[1,88]=-0.214717595683182; fullsyn1[1,688]=0.416284043885546;
fullsyn1[1,89]=-0.0668262187199624;fullsyn1[1,689]=-0.798331755153581;
fullsyn1[1,90]=-0.70800478974005; fullsyn1[1,690]=0.704823030936109;
fullsyn1[1,91]=-0.260825011335104; fullsyn1[1,691]=0.26908407598347;

fullsyn1[1,92]=-0.427825587733344; fullsyn1[1,692]=-0.24779581095664;
fullsyn1[1,93]=0.471660403667379; fullsyn1[1,693]=0.245940448430253;
fullsyn1[1,94]=-0.064298827621017; fullsyn1[1,694]=0.309325796876973;
fullsyn1[1,95]=0.591643690841328; fullsyn1[1,695]=0.429997963071668;
fullsyn1[1,96]=0.409028637626471; fullsyn1[1,696]=0.58964283372325;
fullsyn1[1,97]=-0.684787022548667; fullsyn1[1,697]=-0.104149089911561;
fullsyn1[1,98]=0.000104691517216958;fullsyn1[1,698]=1.26646623780831;
fullsyn1[1,99]=0.619688971420129; fullsyn1[1,699]=-1.00667374665789;
fullsyn1[1,100]=0.139279217464472; fullsyn1[1,700]=-1.25394211804909;
fullsyn1[1,101]=0.26183318855503; fullsyn1[1,701]=0.38956690627001;
fullsyn1[1,102]=0.861252181269488; fullsyn1[1,702]=0.413784782923223;
fullsyn1[1,103]=-0.309113554044356; fullsyn1[1,703]=-0.0613262631685353;
fullsyn1[1,104]=0.141049653443743; fullsyn1[1,704]=0.0365023239833153;
fullsyn1[1,105]=-0.091833635283825; fullsyn1[1,705]=0.646217766971627;
fullsyn1[1,106]=-0.465681636670548; fullsyn1[1,706]=1.5819926631022;
fullsyn1[1,107]=0.0305354388011587; fullsyn1[1,707]=-0.201965063255038;
fullsyn1[1,108]=-0.417312320937023; fullsyn1[1,708]=0.0405325222979537;
fullsyn1[1,109]=0.0976041630393983; fullsyn1[1,709]=0.428577703473148;
fullsyn1[1,110]=-0.557406271513049; fullsyn1[1,710]=-0.451546716605116;
fullsyn1[1,111]=0.21160696558491; fullsyn1[1,711]=0.0908714412778617;
fullsyn1[1,112]=-0.331578184546424; fullsyn1[1,712]=-0.0455925799578273;
fullsyn1[1,113]=0.376498206748423; fullsyn1[1,713]=-0.297095166994571;
fullsyn1[1,114]=0.0421488955873396; fullsyn1[1,714]=0.0982460764400309;
fullsyn1[1,115]=0.767387186639802; fullsyn1[1,715]=-0.588380389464094;
fullsyn1[1,116]=0.250640509740094; fullsyn1[1,716]=0.569165923180164;
fullsyn1[1,117]=0.306261800980506; fullsyn1[1,717]=0.778513355340206;
fullsyn1[1,118]=0.66010004596741; fullsyn1[1,718]=0.234173613423616;
fullsyn1[1,119]=-0.557878076846868; fullsyn1[1,719]=0.305029513115516;
fullsyn1[1,120]=0.400195165168139; fullsyn1[1,720]=-0.28047404294158;
fullsyn1[1,121]=0.253744831790083; fullsyn1[1,721]=1.12690521383744;
fullsyn1[1,122]=0.58296204385944; fullsyn1[1,722]=-0.588951418971385;
fullsyn1[1,123]=0.0167964810324569; fullsyn1[1,723]=0.00109793908352134;
fullsyn1[1,124]=0.197901322960765; fullsyn1[1,724]=-1.37905001556789;
fullsyn1[1,125]=0.303786114314072; fullsyn1[1,725]=0.588628348124223;
fullsyn1[1,126]=-0.0204635724558447; fullsyn1[1,726]=-0.198217707404679;
fullsyn1[1,127]=-0.120994571093889; fullsyn1[1,727]=-0.319434185383442;
fullsyn1[1,128]=0.304519307702008; fullsyn1[1,728]=-1.21858848487834;
fullsyn1[1,129]=-0.33539992698457; fullsyn1[1,729]=0.515714681093814;
fullsyn1[1,130]=-0.927451100112714; fullsyn1[1,730]=1.02078414746926;
fullsyn1[1,131]=0.230249250084872; fullsyn1[1,731]=-0.0385651074731187;
fullsyn1[1,132]=0.524272417514263; fullsyn1[1,732]=0.346207784082531;
fullsyn1[1,133]=-0.385278774754735; fullsyn1[1,733]=0.181912235841625;
fullsyn1[1,134]=-0.381968560924217; fullsyn1[1,734]=-0.308456333957797;
fullsyn1[1,135]=-0.467437555715592; fullsyn1[1,735]=0.0511483802197303;
fullsyn1[1,136]=-0.177930358026452; fullsyn1[1,736]=-0.0857036824300762;
fullsyn1[1,137]=0.166978354033481; fullsyn1[1,737]=0.384037558732622;
fullsyn1[1,138]=-0.00928164403530187; fullsyn1[1,738]=-0.987757249525496;
fullsyn1[1,139]=-0.423994740968047; fullsyn1[1,739]=0.514573092535602;
fullsyn1[1,140]=0.401028590733978; fullsyn1[1,740]=-0.859321809492371;
fullsyn1[1,141]=-0.168466766778991; fullsyn1[1,741]=0.742009259081195;

fullsyn1[1,142]=0.96037645818277; fullsyn1[1,742]=-0.0948628938974369;
fullsyn1[1,143]=0.317211635002196; fullsyn1[1,743]=0.0518588054579657;
fullsyn1[1,144]=0.42725402691985; fullsyn1[1,744]=-0.577172213099407;
fullsyn1[1,145]=0.185814582544863; fullsyn1[1,745]=0.751759488380356;
fullsyn1[1,146]=-0.0601174391382033; fullsyn1[1,746]=0.161381751474716;
fullsyn1[1,147]=-1.48175913009031; fullsyn1[1,747]=-1.00199582486524;
fullsyn1[1,148]=0.285556716893968; fullsyn1[1,748]=-0.197316350463847;
fullsyn1[1,149]=0.794582674030082; fullsyn1[1,749]=-0.852224575720116;
fullsyn1[1,150]=0.4311966609326; fullsyn1[1,750]=-0.0762164839363415;
fullsyn1[1,151]=-0.511755702822544; fullsyn1[1,751]=0.980977095995744;
fullsyn1[1,152]=-0.281140121380295; fullsyn1[1,752]=0.196487554261623;
fullsyn1[1,153]=0.3466268297618; fullsyn1[1,753]=0.462535711837937;
fullsyn1[1,154]=0.122583762852632; fullsyn1[1,754]=-0.639865864012564;
fullsyn1[1,155]=0.688977520258354; fullsyn1[1,755]=-0.0372438984278895;
fullsyn1[1,156]=-0.243502134051069; fullsyn1[1,756]=0.00971547613140937;
fullsyn1[1,157]=0.284006952184848; fullsyn1[1,757]=0.594907675789379;
fullsyn1[1,158]=0.0871208961001822; fullsyn1[1,758]=-0.315349119251507;
fullsyn1[1,159]=0.247158063845026; fullsyn1[1,759]=-0.0630108992178135;
fullsyn1[1,160]=0.167229181918549; fullsyn1[1,760]=0.149555989273473;
fullsyn1[1,161]=-0.958004106593909; fullsyn1[1,761]=0.557220416985015;
fullsyn1[1,162]=0.265899366826053; fullsyn1[1,762]=0.724352976062963;
fullsyn1[1,163]=-0.9631036776855; fullsyn1[1,763]=0.0825904655157969;
fullsyn1[1,164]=-0.384778731410781; fullsyn1[1,764]=-0.250592612113586;
fullsyn1[1,165]=-0.521538688313618; fullsyn1[1,765]=0.245451329253657;
fullsyn1[1,166]=0.431309701216816; fullsyn1[1,766]=-0.597804717897552;
fullsyn1[1,167]=0.220674060334214; fullsyn1[1,767]=0.246345779140853;
fullsyn1[1,168]=0.595768797654101; fullsyn1[1,768]=0.0295903062497052;
fullsyn1[1,169]=0.180646647220418; fullsyn1[1,769]=1.21879415753868;
fullsyn1[1,170]=0.427961579402088; fullsyn1[1,770]=-0.37528369803051;
fullsyn1[1,171]=-0.0158505239466717; fullsyn1[1,771]=-0.460879404625736;
fullsyn1[1,172]=0.200180227878993; fullsyn1[1,772]=0.366697859950374;
fullsyn1[1,173]=0.515372278316978; fullsyn1[1,773]=-1.21004825885691;
fullsyn1[1,174]=0.0908028675279514; fullsyn1[1,774]=0.297083581509485;
fullsyn1[1,175]=0.614363872464309; fullsyn1[1,775]=0.697251946884478;
fullsyn1[1,176]=-0.53771977169823; fullsyn1[1,776]=1.53975978431635;
fullsyn1[1,177]=-0.169689825868885; fullsyn1[1,777]=-0.173832592537341;
fullsyn1[1,178]=0.14873533005937; fullsyn1[1,778]=-1.0066934277053;
fullsyn1[1,179]=-0.747671220857492; fullsyn1[1,779]=-0.073894174563417 3;
fullsyn1[1,180]=0.569926018191939; fullsyn1[1,780]=-0.640902137574075;
fullsyn1[1,181]=-0.598906833711716; fullsyn1[1,781]=0.213782406633383;
fullsyn1[1,182]=0.0221920648765748; fullsyn1[1,782]=0.160667939990604;
fullsyn1[1,183]=-0.258886087680672; fullsyn1[1,783]=0.0379523129632198;
fullsyn1[1,184]=-0.492173893873209; fullsyn1[1,784]=0.321699474914301;
fullsyn1[1,185]=0.135390960811422; fullsyn1[1,785]=-0.518048856376962;
fullsyn1[1,186]=-0.85829524861228; fullsyn1[1,786]=0.18272210417132;
fullsyn1[1,187]=0.363357279395552; fullsyn1[1,787]=0.095804803522692;
fullsyn1[1,188]=-0.13115165482404; fullsyn1[1,788]=0.557030593007724;
fullsyn1[1,189]=0.16345384912534; fullsyn1[1,789]=1.39840423787547;
fullsyn1[1,190]=-0.137929824395022; fullsyn1[1,790]=0.975073729842111;
fullsyn1[1,191]=-0.073865125918936; fullsyn1[1,791]=0.139706627294719;

fullsyn1[1,192]=0.259418955064479; fullsyn1[1,792]=0.625230913330223;
fullsyn1[1,193]=-0.065286304326919; fullsyn1[1,793]=-0.511970510642831;
fullsyn1[1,194]=0.666491214726959; fullsyn1[1,794]=0.472871270983571;
fullsyn1[1,195]=0.221711245165037; fullsyn1[1,795]=-0.390390073674333;
fullsyn1[1,196]=-0.344401599848131; fullsyn1[1,796]=0.0997347938767157;
fullsyn1[1,197]=-0.0960122363770261; fullsyn1[1,797]=-0.259378366236796;
fullsyn1[1,198]=1.13943801765303; fullsyn1[1,798]=0.0706358810301185;
fullsyn1[1,199]=0.757058116283133; fullsyn1[1,799]=-0.40464807471304;
fullsyn1[1,200]=-0.483451251179424; fullsyn1[1,800]=1.05500811525347;
fullsyn1[1,201]=0.0450771380057026; fullsyn1[1,801]=-0.0815461603541675;
fullsyn1[1,202]=-0.564386320388987; fullsyn1[1,802]=-0.0340824970895411;
fullsyn1[1,203]=-0.60733536443187; fullsyn1[1,803]=-0.208203171257859;
fullsyn1[1,204]=-0.258069825957086; fullsyn1[1,804]=0.0448968248728998;
fullsyn1[1,205]=-0.723940491947048; fullsyn1[1,805]=-0.759149791782913;
fullsyn1[1,206]=0.222328862276474; fullsyn1[1,806]=-0.0136697288887326;
fullsyn1[1,207]=-0.758440684917749; fullsyn1[1,807]=-0.605435456724516;
fullsyn1[1,208]=-0.513994664546175; fullsyn1[1,808]=-0.506227230797924;
fullsyn1[1,209]=0.49766402962111; fullsyn1[1,809]=-0.69787041865644;
fullsyn1[1,210]=-0.559648295873935; fullsyn1[1,810]=0.322783232136285;
fullsyn1[1,211]=0.790324985435969; fullsyn1[1,811]=-0.0886888669201331;
fullsyn1[1,212]=-0.335445802022864; fullsyn1[1,812]=0.260010163051898;
fullsyn1[1,213]=-0.214000733390111; fullsyn1[1,813]=0.0588802475409664;
fullsyn1[1,214]=0.226715958013449; fullsyn1[1,814]=-0.0790246773096773;
fullsyn1[1,215]=-0.499778250613028; fullsyn1[1,815]=0.0321289424115488;
fullsyn1[1,216]=0.0193822583584954; fullsyn1[1,816]=0.144859831209234;
fullsyn1[1,217]=-0.838994385226028; fullsyn1[1,817]=0.265595627001181;
fullsyn1[1,218]=0.0165506486762533; fullsyn1[1,818]=1.66240766323767;
fullsyn1[1,219]=-0.536188229483703; fullsyn1[1,819]=0.930588657077301;
fullsyn1[1,220]=-0.114851283632153; fullsyn1[1,820]=0.261776573684144;
fullsyn1[1,221]=0.73132671383335; fullsyn1[1,821]=-1.13939011365827;
fullsyn1[1,222]=0.539729849949008; fullsyn1[1,822]=0.79149730129361;
fullsyn1[1,223]=-0.0710070371812559; fullsyn1[1,823]=0.0173844162506433;
fullsyn1[1,224]=0.833293995774493; fullsyn1[1,824]=-1.10883720191177;
fullsyn1[1,225]=-0.506832636523929; fullsyn1[1,825]=0.134384823825546;
fullsyn1[1,226]=-0.767616059028047; fullsyn1[1,826]=0.884365455239214;
fullsyn1[1,227]=0.560507014489989; fullsyn1[1,827]=-0.0929578931620811;
fullsyn1[1,228]=-0.0756251979469507; fullsyn1[1,828]=-0.105191951255596;
fullsyn1[1,229]=-0.027420770041785; fullsyn1[1,829]=0.253110333452762;
fullsyn1[1,230]=0.0851288157178204; fullsyn1[1,830]=-0.0049265073948597;
fullsyn1[1,231]=0.481137417201446; fullsyn1[1,831]=-0.294061383535069;
fullsyn1[1,232]=0.519071367805169; fullsyn1[1,832]=0.448883177146637;
fullsyn1[1,233]=-0.113007625210492; fullsyn1[1,833]=-0.560485306120623;
fullsyn1[1,234]=0.598085237485697; fullsyn1[1,834]=-0.419441535764936;
fullsyn1[1,235]=-0.0680119331954046; fullsyn1[1,835]=0.966708885614443;
fullsyn1[1,236]=-0.866418986938041; fullsyn1[1,836]=0.117902654101693;
fullsyn1[1,237]=-0.146855480135004; fullsyn1[1,837]=0.736782344458542;
fullsyn1[1,238]=-0.346309755732703; fullsyn1[1,838]=0.545337242062573;
fullsyn1[1,239]=0.891157024243359; fullsyn1[1,839]=-0.195027059764402;
fullsyn1[1,240]=0.486768809997316; fullsyn1[1,840]=0.582786354652478;
fullsyn1[1,241]=0.43382678473788; fullsyn1[1,841]=0.568034518189746;

fullsyn1[1,242]=0.336166598456112;
fullsyn1[1,243]=-0.877412327051682;
fullsyn1[1,244]=-0.968753655572132;
fullsyn1[1,245]=-0.588925640175504;
fullsyn1[1,246]=-0.456628563041046;
fullsyn1[1,247]=0.960676707602536;
fullsyn1[1,248]=0.181381031079452;
fullsyn1[1,249]=0.117212409893784;
fullsyn1[1,250]=0.153833176713402;
fullsyn1[1,251]=0.511901007838943;
fullsyn1[1,252]=0.0222518622483172;
fullsyn1[1,253]=0.223442621554396;
fullsyn1[1,254]=-0.140013490155416;
fullsyn1[1,255]=0.23190846662513;
fullsyn1[1,256]=-0.326847825196339;
fullsyn1[1,257]=-0.632653330673726;
fullsyn1[1,258]=-0.384825995781804;
fullsyn1[1,259]=-0.27795132761669;
fullsyn1[1,260]=-0.291798050107216;
fullsyn1[1,261]=0.559393050530121;
fullsyn1[1,262]=0.464110104676814;
fullsyn1[1,263]=0.226213966960342;
fullsyn1[1,264]=0.226712172738818;
fullsyn1[1,265]=0.259039488666309;
fullsyn1[1,266]=-1.05549847711742;
fullsyn1[1,267]=-1.0920191972959;
fullsyn1[1,268]=1.0448555781534;
fullsyn1[1,269]=-1.03271715140952;
fullsyn1[1,270]=-0.732994830362398;
fullsyn1[1,271]=-0.459582352745039;
fullsyn1[1,272]=-0.0378373522159834;
fullsyn1[1,273]=0.603559206243836;
fullsyn1[1,274]=1.15900889428492;
fullsyn1[1,275]=-0.215807209821636;
fullsyn1[1,276]=0.178054930518895;
fullsyn1[1,277]=-0.488001831679881;
fullsyn1[1,278]=0.492193619128391;
fullsyn1[1,279]=-0.403562735008525;
fullsyn1[1,280]=-0.366492845637703;
fullsyn1[1,281]=0.342108356961792;
fullsyn1[1,282]=-0.976927998354436;
fullsyn1[1,283]=0.336518384849295;
fullsyn1[1,284]=0.758683561344192;
fullsyn1[1,285]=-0.884863174399065;
fullsyn1[1,286]=0.5702333484159;
fullsyn1[1,287]=-0.50103428583188;
fullsyn1[1,288]=0.710815911116035;
fullsyn1[1,289]=-0.760872170514399;
fullsyn1[1,290]=-0.0634816254057992;
fullsyn1[1,291]=0.610809506101378;
fullsyn1[1,842]=0.346860099466181;
fullsyn1[1,843]=0.379431273820383;
fullsyn1[1,844]=1.23830390834853;
fullsyn1[1,845]=-0.0731162043718128;
fullsyn1[1,846]=-0.551836506621922;
fullsyn1[1,847]=-0.0903042397262816;
fullsyn1[1,848]=0.0645637276650622;
fullsyn1[1,849]=0.545364730894598;
fullsyn1[1,850]=0.112455799579372;
fullsyn1[1,851]=0.413659013346203;
fullsyn1[1,852]=0.294865810838153;
fullsyn1[1,853]=-0.601800485242161;
fullsyn1[1,854]=0.458406363509133;
fullsyn1[1,855]=-0.0913850013790457;
fullsyn1[1,856]=-0.315462480059251;
fullsyn1[1,857]=-0.16530687587905;
fullsyn1[1,858]=-0.545408352878769;
fullsyn1[1,859]=-0.0273612438342977;
fullsyn1[1,860]=-0.308679954119639;
fullsyn1[1,861]=-0.0663369639556461;
fullsyn1[1,862]=0.309934749672138;
fullsyn1[1,863]=-0.618179290483667;
fullsyn1[1,864]=-0.586146490403686;
fullsyn1[1,865]=-0.696600195576563;
fullsyn1[1,866]=-0.0536910579462412;
fullsyn1[1,867]=0.324308821812953;
fullsyn1[1,868]=0.493272636014713;
fullsyn1[1,869]=0.223836108883044;
fullsyn1[1,870]=-0.623925355408234;
fullsyn1[1,871]=0.24473676423261;
fullsyn1[1,872]=1.86378231812985;
fullsyn1[1,873]=-1.38968444059121;
fullsyn1[1,874]=-0.787610325916946;
fullsyn1[1,875]=0.561673573060127;
fullsyn1[1,876]=-0.444321676429917;
fullsyn1[1,877]=0.240906936115805;
fullsyn1[1,878]=0.544201387814097;
fullsyn1[1,879]=-0.0351524763012889;
fullsyn1[1,880]=-0.383310561780849;
fullsyn1[1,881]=0.124231675102733;
fullsyn1[1,882]=-0.273318910704555;
fullsyn1[1,883]=-0.228070331863952;
fullsyn1[1,884]=-0.553666823649285;
fullsyn1[1,885]=0.528775456596068;
fullsyn1[1,886]=1.19865519102357;
fullsyn1[1,887]=-1.5294811942387;
fullsyn1[1,888]=0.501696571678348;
fullsyn1[1,889]=-0.333913902672245;
fullsyn1[1,890]=1.44055995952305;
fullsyn1[1,891]=-0.239094056019427;

fullsyn1[1,292]=0.539310906243987;
fullsyn1[1,293]=0.20399566923178;
fullsyn1[1,294]=1.02211173398932;
fullsyn1[1,295]=-0.0830378496401606;
fullsyn1[1,296]=-0.470970540982058;
fullsyn1[1,297]=-1.05072358525763;
fullsyn1[1,298]=-0.797459059558382;
fullsyn1[1,299]=-0.0101946182050777;
fullsyn1[1,300]=-0.122787219808267;
fullsyn1[1,301]=0.0362194849167433;
fullsyn1[1,302]=0.138786503551414;
fullsyn1[1,303]=-0.262966579634252;
fullsyn1[1,304]=-0.340681378508285;
fullsyn1[1,305]=0.60663525537613;
fullsyn1[1,306]=-0.755797342848994;
fullsyn1[1,307]=0.144637633619179;
fullsyn1[1,308]=0.0710298060528924;
fullsyn1[1,309]=0.0338613495071111;
fullsyn1[1,310]=0.0912772567736526;
fullsyn1[1,311]=0.538842114599718;
fullsyn1[1,312]=-0.680522829368308;
fullsyn1[1,313]=1.05264316303988;
fullsyn1[1,314]=-0.444838985839811;
fullsyn1[1,315]=1.19527481226309;
fullsyn1[1,316]=0.13898413189481;
fullsyn1[1,317]=0.335275026369851;
fullsyn1[1,318]=0.591620132937547;
fullsyn1[1,319]=0.306729334204731;
fullsyn1[1,320]=1.0196601549266;
fullsyn1[1,321]=-0.986545489282926;
fullsyn1[1,322]=0.747297556014545;
fullsyn1[1,323]=-0.352802754601927;
fullsyn1[1,324]=0.710410771567932;
fullsyn1[1,325]=-0.00963666357695118;
fullsyn1[1,326]=0.168690784322116;
fullsyn1[1,327]=0.168950309239256;
fullsyn1[1,328]=-0.118594824808107;
fullsyn1[1,329]=-0.123372614660371;
fullsyn1[1,330]=-0.749167245590045;
fullsyn1[1,331]=0.0355900728848806;
fullsyn1[1,332]=-0.279990511009306;
fullsyn1[1,333]=0.497295194733249;
fullsyn1[1,334]=0.266289496130292;
fullsyn1[1,335]=0.442645742004918;
fullsyn1[1,336]=0.468948266384499;
fullsyn1[1,337]=0.394119418468843;
fullsyn1[1,338]=-0.62425555560083;
fullsyn1[1,339]=0.629055919782368;
fullsyn1[1,340]=0.225378857463202;
fullsyn1[1,341]=0.933980229040334;
fullsyn1[1,892]=0.207465164054402;
fullsyn1[1,893]=0.870302907326965;
fullsyn1[1,894]=-1.29514549709775;
fullsyn1[1,895]=-0.0771235200393209;
fullsyn1[1,896]=0.577230526443372;
fullsyn1[1,897]=0.250319151405766;
fullsyn1[1,898]=0.336462408012351;
fullsyn1[1,899]=-0.307763625857742;
fullsyn1[1,900]=0.201222572762154;
fullsyn1[1,901]=0.290529012940396;
fullsyn1[1,902]=-0.0506300982002135;
fullsyn1[1,903]=0.283688246976208;
fullsyn1[1,904]=-0.632719809586005;
fullsyn1[1,905]=0.194335438032152;
fullsyn1[1,906]=-0.463349895579935;
fullsyn1[1,907]=-0.174065223541811;
fullsyn1[1,908]=-1.07334556732339;
fullsyn1[1,909]=0.585967412129337;
fullsyn1[1,910]=0.549157176617184;
fullsyn1[1,911]=0.0823762706636178;
fullsyn1[1,912]=-0.877632378073429;
fullsyn1[1,913]=-0.684044421058033;
fullsyn1[1,914]=0.310464069329164;
fullsyn1[1,915]=-0.573709199143563;
fullsyn1[1,916]=0.100788468065629;
fullsyn1[1,917]=-0.438689007352682;
fullsyn1[1,918]=0.397382330915719;
fullsyn1[1,919]=1.52937808841958;
fullsyn1[1,920]=-1.41887264194189;
fullsyn1[1,921]=-0.0866317210870437;
fullsyn1[1,922]=0.114854444825845;
fullsyn1[1,923]=0.262612082311846;
fullsyn1[1,924]=0.460536355077326;
fullsyn1[1,925]=-0.210087886625782;
fullsyn1[1,926]=0.0429139961904712;
fullsyn1[1,927]=0.159918253283983;
fullsyn1[1,928]=0.54192275873359;
fullsyn1[1,929]=-0.33175746331484;
fullsyn1[1,930]=-0.0903574403522881;
fullsyn1[1,931]=-0.13432151626749;
fullsyn1[1,932]=-0.201217632409558;
fullsyn1[1,933]=-0.0710476284659399;
fullsyn1[1,934]=-0.194311720469981;
fullsyn1[1,935]=0.189525883209077;
fullsyn1[1,936]=0.191762501174421;
fullsyn1[1,937]=0.20226864113362;
fullsyn1[1,938]=-0.462812303266471;
fullsyn1[1,939]=0.768412666867993;
fullsyn1[1,940]=-0.470626818425587;
fullsyn1[1,941]=0.453423477019415;

fullsyn1[1,342]=0.0402562954510246;
fullsyn1[1,343]=-0.208820462012985;
fullsyn1[1,344]=0.932279014266803;
fullsyn1[1,345]=-0.0671494900960982;
fullsyn1[1,346]=-1.3440980263678;
fullsyn1[1,347]=-0.171912092700122;
fullsyn1[1,348]=-0.254947368624253;
fullsyn1[1,349]=0.746183111023311;
fullsyn1[1,350]=0.0968461963438232;
fullsyn1[1,351]=0.436569171805484;
fullsyn1[1,352]=-0.3922995080278;
fullsyn1[1,353]=0.192368210708709;
fullsyn1[1,354]=-0.387000293391088;
fullsyn1[1,355]=0.364529935611657;
fullsyn1[1,356]=-0.603671050442464;
fullsyn1[1,357]=-0.236789218362194;
fullsyn1[1,358]=-0.11250761757082;
fullsyn1[1,359]=0.38982851086161;
fullsyn1[1,360]=0.334916849008818;
fullsyn1[1,361]=-0.452851345656524;
fullsyn1[1,362]=-0.579719551920187;
fullsyn1[1,363]=0.849476152309794;
fullsyn1[1,364]=0.0333431529997211;
fullsyn1[1,365]=-0.413487679002539;
fullsyn1[1,366]=-0.229604915701671;
fullsyn1[1,367]=-0.126689345795072;
fullsyn1[1,368]=-0.937186565153949;
fullsyn1[1,369]=0.132988861161545;
fullsyn1[1,370]=-0.419148665576634;
fullsyn1[1,371]=-0.370594212758524;
fullsyn1[1,372]=-0.136488811483327;
fullsyn1[1,373]=-0.509908051374467;
fullsyn1[1,374]=-0.968442333860518;
fullsyn1[1,375]=-0.577866497741722;
fullsyn1[1,376]=-0.472366002202659;
fullsyn1[1,377]=-0.025384232552188 3;
fullsyn1[1,378]=-0.315926118509834;
fullsyn1[1,379]=0.33853558587898;
fullsyn1[1,380]=-0.063215507855829 1;
fullsyn1[1,381]=0.418461717936014;
fullsyn1[1,382]=0.525308712745306;
fullsyn1[1,383]=0.553582041341201;
fullsyn1[1,384]=-0.356757397693161;
fullsyn1[1,385]=-0.967597647344649;
fullsyn1[1,386]=1.22590264066539;
fullsyn1[1,387]=-0.168815693034824;
fullsyn1[1,388]=-0.0435320609891568;
fullsyn1[1,389]=-1.32824030849863;
fullsyn1[1,390]=0.457670664799639;
fullsyn1[1,391]=0.247075984360087;
fullsyn1[1,942]=0.81175993270889;
fullsyn1[1,943]=-0.869600768034066;
fullsyn1[1,944]=-0.380241563138751;
fullsyn1[1,945]=0.2827957573027;
fullsyn1[1,946]=-0.379498154351339;
fullsyn1[1,947]=0.0601441922211047;
fullsyn1[1,948]=-0.193874429548308;
fullsyn1[1,949]=-0.0834097203145352;
fullsyn1[1,950]=-0.0487716765986738;
fullsyn1[1,951]=-0.0285646147136071;
fullsyn1[1,952]=-0.536428107475651;
fullsyn1[1,953]=0.218311991848239;
fullsyn1[1,954]=0.468135780050682;
fullsyn1[1,955]=-0.244963681681391;
fullsyn1[1,956]=-0.239084473223955;
fullsyn1[1,957]=0.291637865737076;
fullsyn1[1,958]=0.503498317304701;
fullsyn1[1,959]=0.341539368493705;
fullsyn1[1,960]=0.778228044941527;
fullsyn1[1,961]=-0.0388863647425673;
fullsyn1[1,962]=-0.256387479593037;
fullsyn1[1,963]=0.0391075800786848;
fullsyn1[1,964]=-0.575459980328048;
fullsyn1[1,965]=0.901820776805469;
fullsyn1[1,966]=-0.376191045232453;
fullsyn1[1,967]=-1.2368468616342;
fullsyn1[1,968]=0.0483275268231796;
fullsyn1[1,969]=1.80204162462708;
fullsyn1[1,970]=-0.801841466830117;
fullsyn1[1,971]=0.239367999226468;
fullsyn1[1,972]=0.0255289898914548;
fullsyn1[1,973]=-0.0606432347508665;
fullsyn1[1,974]=0.203291358380515;
fullsyn1[1,975]=0.325946574810119;
fullsyn1[1,976]=0.454650925220485;
fullsyn1[1,977]=-0.228214820237299;
fullsyn1[1,978]=-0.0305044214032642;
fullsyn1[1,979]=0.727674275308294;
fullsyn1[1,980]=-0.299200760469699;
fullsyn1[1,981]=0.175191585337551;
fullsyn1[1,982]=-0.987777666993958;
fullsyn1[1,983]=0.511922491646903;
fullsyn1[1,984]=-0.176759823597575;
fullsyn1[1,985]=-0.0611147845278055;
fullsyn1[1,986]=0.855196373043702;
fullsyn1[1,987]=0.762331324542836;
fullsyn1[1,988]=1.79266262058352;
fullsyn1[1,989]=-1.27690792883153;
fullsyn1[1,990]=-1.37325441073612;
fullsyn1[1,991]=0.349280255395141;

fullsyn1[1,392]=1.1511149570535;
fullsyn1[1,393]=0.195947081945689;
fullsyn1[1,394]=0.270649437569565;
fullsyn1[1,395]=-0.393992072542726;
fullsyn1[1,396]=0.4166567140697;
fullsyn1[1,397]=-0.647676726557468;
fullsyn1[1,398]=-0.158566781626887;
fullsyn1[1,399]=-0.130988096818421;
fullsyn1[1,400]=-0.209841005277826;
fullsyn1[1,401]=-0.429780543500039;
fullsyn1[1,402]=0.789471637056745;
fullsyn1[1,403]=-0.445194375940189;
fullsyn1[1,404]=1.40170665049299;
fullsyn1[1,405]=0.00244403354877981;
fullsyn1[1,406]=0.840879455238027;
fullsyn1[1,407]=-0.588253380607704;
fullsyn1[1,408]=0.000717837229391387;
fullsyn1[1,409]=0.895869006548665;
fullsyn1[1,410]=-0.0837170750980399;
fullsyn1[1,411]=-0.487102389480559;
fullsyn1[1,412]=0.0236605611443546;
fullsyn1[1,413]=0.341748559100032;
fullsyn1[1,414]=-0.273313477055366;
fullsyn1[1,415]=0.552900032358592;
fullsyn1[1,416]=0.769544590352662;
fullsyn1[1,417]=-0.110527770001736;
fullsyn1[1,418]=-0.584628090246844;
fullsyn1[1,419]=0.0553359810614984;
fullsyn1[1,420]=0.189419510168947;
fullsyn1[1,421]=0.00931336292495278;
fullsyn1[1,422]=0.202317870611417;
fullsyn1[1,423]=0.493754633445339;
fullsyn1[1,424]=0.458602661630355;
fullsyn1[1,425]=0.778856052359083;
fullsyn1[1,426]=-0.0666153048145704;
fullsyn1[1,427]=0.428252332748973;
fullsyn1[1,428]=-0.0256729350471967;
fullsyn1[1,429]=0.33708741273205;
fullsyn1[1,430]=-0.809963333161693;
fullsyn1[1,431]=1.15661409673061;
fullsyn1[1,432]=-0.0785062664733464;
fullsyn1[1,433]=-0.522501266459987;
fullsyn1[1,434]=-0.15919964417923;
fullsyn1[1,435]=0.0558707537587216;
fullsyn1[1,436]=0.297449161902105;
fullsyn1[1,437]=-0.423039836957078;
fullsyn1[1,438]=-0.182975091295532;
fullsyn1[1,439]=-0.716204399600088;
fullsyn1[1,440]=0.0262853414596583;
fullsyn1[1,441]=-0.212177596681906;
fullsyn1[1,992]=0.289234872069527;
fullsyn1[1,993]=-0.582642368083366;
fullsyn1[1,994]=-0.497417055004444;
fullsyn1[1,995]=0.213138218788122;
fullsyn1[1,996]=0.0401927191673457;
fullsyn1[1,997]=0.178097649607592;
fullsyn1[1,998]=-0.175676020718077;
fullsyn1[1,999]=0.0407652151115685;
fullsyn1[1,1000]=0.30196473277552;
fullsyn1[1,1001]=0.400318589505503;
fullsyn1[1,1002]=-0.459808694965612;
fullsyn1[1,1003]=0.379840907116946;
fullsyn1[1,1004]=-0.954515089616119;
fullsyn1[1,1005]=-0.28863604675389;
fullsyn1[1,1006]=0.359546530011664;
fullsyn1[1,1007]=0.617637125132204;
fullsyn1[1,1008]=-0.731250383932516;
fullsyn1[1,1009]=-0.595140133638685;
fullsyn1[1,1010]=1.06210950381955;
fullsyn1[1,1011]=0.00854938106242228;
fullsyn1[1,1012]=-0.89737846551801;
fullsyn1[1,1013]=-0.430918054432417;
fullsyn1[1,1014]=-0.0458844095686585;
fullsyn1[1,1015]=0.796255497240072;
fullsyn1[1,1016]=0.170033475930421;
fullsyn1[1,1017]=-0.339364129798455;
fullsyn1[1,1018]=-0.16223206090199;
fullsyn1[1,1019]=0.405859519701349;
fullsyn1[1,1020]=0.825318920706393;
fullsyn1[1,1021]=0.0971487716109809;
fullsyn1[1,1022]=0.352427665688715;
fullsyn1[1,1023]=0.648075285100819;
fullsyn1[1,1024]=-0.0349572372488982;
fullsyn1[1,1025]=0.54253872441997;
fullsyn1[1,1026]=0.579869509161908;
fullsyn1[1,1027]=0.32663032653057;
fullsyn1[1,1028]=-0.47891621150846;
fullsyn1[1,1029]=0.13956110297861;
fullsyn1[1,1030]=-0.485347804990489;
fullsyn1[1,1031]=-0.22441879186961;
fullsyn1[1,1032]=-0.166444027840146;
fullsyn1[1,1033]=0.0982149014440779;
fullsyn1[1,1034]=-0.377786624125077;
fullsyn1[1,1035]=-0.721220127139489;
fullsyn1[1,1036]=0.303765532215828;
fullsyn1[1,1037]=0.746073158982318;
fullsyn1[1,1038]=0.00700961080182008;
fullsyn1[1,1039]=0.380188481680625;
fullsyn1[1,1040]=-0.210632351419244;
fullsyn1[1,1041]=-0.0569057148459355;

fullsyn1[1,442]=-0.0247706144196219;
fullsyn1[1,443]=0.121314635630453;
fullsyn1[1,444]=0.661361661942906;
fullsyn1[1,445]=0.247215138323454;
fullsyn1[1,446]=0.073684967908705;
fullsyn1[1,447]=-0.30263736818025;
fullsyn1[1,448]=0.262018476692686;
fullsyn1[1,449]=0.282619523130559;
fullsyn1[1,450]=-0.168524691024437;
fullsyn1[1,451]=0.279315007998584;
fullsyn1[1,452]=0.115912310965918;
fullsyn1[1,453]=-0.23659404525841;
fullsyn1[1,454]=0.503935395153629;
fullsyn1[1,455]=-0.248113601161977;
fullsyn1[1,456]=0.417554251532188;
fullsyn1[1,457]=-0.690062103642403;
fullsyn1[1,458]=-0.722455302089579;
fullsyn1[1,459]=0.80744345686729;
fullsyn1[1,460]=0.625840725922431;
fullsyn1[1,461]=0.445564098330145;
fullsyn1[1,462]=0.255770618286496;
fullsyn1[1,463]=0.1891407324249;
fullsyn1[1,464]=-1.28354155769137;
fullsyn1[1,465]=0.346637315419266;
fullsyn1[1,466]=0.570714890705622;
fullsyn1[1,467]=0.155403832386579;
fullsyn1[1,468]=0.140708554445588;
fullsyn1[1,469]=-0.16281029544814;
fullsyn1[1,470]=0.0436614405146688;
fullsyn1[1,471]=0.326986987220258;
fullsyn1[1,472]=-0.117981136193256;
fullsyn1[1,473]=0.0744897877161048;
fullsyn1[1,474]=0.619787883972755;
fullsyn1[1,475]=-0.508844799729194;
fullsyn1[1,476]=0.0314900940276112;
fullsyn1[1,477]=-0.822653382191105;
fullsyn1[1,478]=-0.248364295263502;
fullsyn1[1,479]=-0.634901944349359;
fullsyn1[1,480]=-0.0276855984694728;
fullsyn1[1,481]=-2.04865717668574;
fullsyn1[1,482]=0.123619157299285;
fullsyn1[1,483]=-0.495253393734028;
fullsyn1[1,484]=0.691132399593981;
fullsyn1[1,485]=-0.578587655553934;
fullsyn1[1,486]=0.236152605956609;
fullsyn1[1,487]=0.0820135540000913;
fullsyn1[1,488]=-0.922574918641167;
fullsyn1[1,489]=-0.0576643026017306;
fullsyn1[1,490]=0.912266219292183;
fullsyn1[1,491]=-0.422663412185974;
fullsyn1[1,1042]=0.990606493584795;
fullsyn1[1,1043]=0.0941078649053519;
fullsyn1[1,1044]=-0.462263103504138;
fullsyn1[1,1045]=-0.0584980949140777;
fullsyn1[1,1046]=0.151035689834585;
fullsyn1[1,1047]=-0.0942721574336563;
fullsyn1[1,1048]=-0.41438640534694;
fullsyn1[1,1049]=-0.131929272662533;
fullsyn1[1,1050]=0.0566430277375622;
fullsyn1[1,1051]=0.463181776234006;
fullsyn1[1,1052]=0.437506100012287;
fullsyn1[1,1053]=-0.524293185120914;
fullsyn1[1,1054]=0.0718587845409806;
fullsyn1[1,1055]=-0.759798445737788;
fullsyn1[1,1056]=0.270039581955605;
fullsyn1[1,1057]=-1.27552717143561;
fullsyn1[1,1058]=-0.175051339901847;
fullsyn1[1,1059]=-0.688144682476532;
fullsyn1[1,1060]=-0.0911912063489476;
fullsyn1[1,1061]=0.466799751870785;
fullsyn1[1,1062]=-0.287972742976231;
fullsyn1[1,1063]=0.366340188549672;
fullsyn1[1,1064]=-1.21946765865426;
fullsyn1[1,1065]=0.147737545249354;
fullsyn1[1,1066]=-0.934327479714211;
fullsyn1[1,1067]=0.0878409349608217;
fullsyn1[1,1068]=-0.115388600899183;
fullsyn1[1,1069]=0.201564351865831;
fullsyn1[1,1070]=-0.453432693615504;
fullsyn1[1,1071]=-0.467025460435232;
fullsyn1[1,1072]=-0.397215947686344;
fullsyn1[1,1073]=0.240766704094033;
fullsyn1[1,1074]=-0.324126276703254;
fullsyn1[1,1075]=0.471653662129966;
fullsyn1[1,1076]=0.132887338955192;
fullsyn1[1,1077]=0.464178965663924;
fullsyn1[1,1078]=-0.90238210438531;
fullsyn1[1,1079]=-0.442003385947732;
fullsyn1[1,1080]=-0.675547144012516;
fullsyn1[1,1081]=-0.999681447410683;
fullsyn1[1,1082]=-0.147503896729888;
fullsyn1[1,1083]=0.410650034139299;
fullsyn1[1,1084]=0.358273791027365;
fullsyn1[1,1085]=-0.213753588960876;
fullsyn1[1,1086]=-0.328140574942003;
fullsyn1[1,1087]=0.204597780829051;
fullsyn1[1,1088]=0.404393477594525;
fullsyn1[1,1089]=-0.405534473565403;
fullsyn1[1,1090]=0.52256527357284;
fullsyn1[1,1091]=-0.159170765591725;

fullsyn1[1,492]=0.179888315650157;
fullsyn1[1,493]=-0.0278312073928141;
fullsyn1[1,494]=0.000463523537947749;
fullsyn1[1,495]=-0.533427783135569;
fullsyn1[1,496]=0.406755838411599;
fullsyn1[1,497]=0.240654535372664;
fullsyn1[1,498]=0.0356361065843427;
fullsyn1[1,499]=-0.107603013562963;
fullsyn1[1,500]=0.0223466339050809;
fullsyn1[1,501]=-0.446814430758595;
fullsyn1[1,502]=0.0192245355217451;
fullsyn1[1,503]=-0.453638101008267;
fullsyn1[1,504]=0.443494466739245;
fullsyn1[1,505]=-0.47494385014478;
fullsyn1[1,506]=-0.511049058645007;
fullsyn1[1,507]=1.11550083041377;
fullsyn1[1,508]=0.782665132806053;
fullsyn1[1,509]=-0.422187698916641;
fullsyn1[1,510]=-0.343391650849733;
fullsyn1[1,511]=-0.203989092933759;
fullsyn1[1,512]=1.02860420350982;
fullsyn1[1,513]=-1.07776948628373;
fullsyn1[1,514]=0.816268114075289;
fullsyn1[1,515]=-0.248245917247693;
fullsyn1[1,516]=-0.530751245596231;
fullsyn1[1,517]=-0.0152965506966284;
fullsyn1[1,518]=-0.298977795651351;
fullsyn1[1,519]=0.361079764793951;
fullsyn1[1,520]=-0.687986505834569;
fullsyn1[1,521]=-0.17442491863881;
fullsyn1[1,522]=-0.106273554160255;
fullsyn1[1,523]=0.224683560974175;
fullsyn1[1,524]=0.391225521765474;
fullsyn1[1,525]=0.163137069679575;
fullsyn1[1,526]=0.861886634841125;
fullsyn1[1,527]=-0.5657296590963;
fullsyn1[1,528]=0.530764379415107;
fullsyn1[1,529]=-0.735628006136073;
fullsyn1[1,530]=1.24186203089171;
fullsyn1[1,531]=-0.263420224496292;
fullsyn1[1,532]=1.17231005260089;
fullsyn1[1,533]=0.136304688550867;
fullsyn1[1,534]=-0.719669908759905;
fullsyn1[1,535]=-0.65525988275823;
fullsyn1[1,536]=-0.467345858375093;
fullsyn1[1,537]=-0.0114573145789059;
fullsyn1[1,538]=-1.17756310843065;
fullsyn1[1,539]=-0.518277568742035;
fullsyn1[1,540]=0.271117316633584;
fullsyn1[1,541]=0.31483890724894;
fullsyn1[1,1092]=0.0693442929053741;
fullsyn1[1,1093]=-0.560630850907523;
fullsyn1[1,1094]=-0.475745002261226;
fullsyn1[1,1095]=0.215643079509408;
fullsyn1[1,1096]=0.158082266924262;
fullsyn1[1,1097]=-0.46681567612524;
fullsyn1[1,1098]=-0.19394309275226;
fullsyn1[1,1099]=-0.125742686932927;
fullsyn1[1,1100]=0.233410575356395;
fullsyn1[1,1101]=0.607284330969445;
fullsyn1[1,1102]=0.129587191585855;
fullsyn1[1,1103]=0.188985969125788;
fullsyn1[1,1104]=-0.194566332043637;
fullsyn1[1,1105]=0.14964260114374;
fullsyn1[1,1106]=0.154260035850194;
fullsyn1[1,1107]=0.311287438675611;
fullsyn1[1,1108]=0.780920815422783;
fullsyn1[1,1109]=0.457588363750139;
fullsyn1[1,1110]=0.566894834916107;
fullsyn1[1,1111]=0.668752600736067;
fullsyn1[1,1112]=-0.932715855428325;
fullsyn1[1,1113]=-0.119919616268299;
fullsyn1[1,1114]=-0.455149000015957;
fullsyn1[1,1115]=0.523846629654084;
fullsyn1[1,1116]=0.104755329607488;
fullsyn1[1,1117]=0.050745374894241;
fullsyn1[1,1118]=0.384045312038609;
fullsyn1[1,1119]=-0.374918036230216;
fullsyn1[1,1120]=-0.0724391764872728;
fullsyn1[1,1121]=0.0637451965322349;
fullsyn1[1,1122]=0.128654667921455;
fullsyn1[1,1123]=0.168128371193607;
fullsyn1[1,1124]=0.0370510889839005;
fullsyn1[1,1125]=-0.0165162337200829;
fullsyn1[1,1126]=-0.238546350300428;
fullsyn1[1,1127]=-0.288436326547835;
fullsyn1[1,1128]=-0.299035736369758;
fullsyn1[1,1129]=-0.282331840199492;
fullsyn1[1,1130]=0.0137971077877112;
fullsyn1[1,1131]=0.655459762095435;
fullsyn1[1,1132]=0.0498076062716255;
fullsyn1[1,1133]=0.428594961887805;
fullsyn1[1,1134]=0.211224127871819;
fullsyn1[1,1135]=-0.355264487100077;
fullsyn1[1,1136]=0.717032411059329;
fullsyn1[1,1137]=-0.343299383808399;
fullsyn1[1,1138]=-0.500414334036021;
fullsyn1[1,1139]=-1.06925430708881;
fullsyn1[1,1140]=0.363186049772297;
fullsyn1[1,1141]=-0.41426788269594;

fullsyn1[1,542]=-0.594294005911931;
fullsyn1[1,543]=-0.510642960675523;
fullsyn1[1,544]=-0.373381764601662;
fullsyn1[1,545]=0.454689869475106;
fullsyn1[1,546]=-0.156527234243376;
fullsyn1[1,547]=-0.652288345160077;
fullsyn1[1,548]=-0.67866420177731;
fullsyn1[1,549]=0.430406948386196;
fullsyn1[1,550]=-0.763778404781983;
fullsyn1[1,551]=-0.172454935829694;
fullsyn1[1,552]=-0.339274190937674;
fullsyn1[1,553]=-0.135340547982747;
fullsyn1[1,554]=-0.767788132799613;
fullsyn1[1,555]=1.21390782450574;
fullsyn1[1,556]=0.791175779692376;
fullsyn1[1,557]=0.643526486289664;
fullsyn1[1,558]=0.167625450737627;
fullsyn1[1,559]=-0.750109503613141;
fullsyn1[1,560]=-1.54321774655677;
fullsyn1[1,561]=-1.36656844399905;
fullsyn1[1,562]=0.813163565423093;
fullsyn1[1,563]=-0.150384722188648;
fullsyn1[1,564]=0.488787139675102;
fullsyn1[1,565]=0.0764670225487867;
fullsyn1[1,566]=-0.723938654537257;
fullsyn1[1,567]=-0.206795602243025;
fullsyn1[1,568]=-0.457285608926294;
fullsyn1[1,569]=-0.659015258200928;
fullsyn1[1,570]=0.743161411362024;
fullsyn1[1,571]=-0.746041700635426;
fullsyn1[1,572]=0.946363335019169;
fullsyn1[1,573]=0.381068974396603;
fullsyn1[1,574]=0.164441854188981;
fullsyn1[1,575]=0.0334498179839541;
fullsyn1[1,576]=0.300017595180738;
fullsyn1[1,577]=-0.845196281313786;
fullsyn1[1,578]=0.30795576529351;
fullsyn1[1,579]=0.606910526163658;
fullsyn1[1,580]=-0.992100302248848;
fullsyn1[1,581]=0.514983713836186;
fullsyn1[1,582]=-0.264030393407525;
fullsyn1[1,583]=-0.0216145622249216;
fullsyn1[1,584]=0.381494929122304;
fullsyn1[1,585]=-0.821004489818457;
fullsyn1[1,586]=0.332832477478053;
fullsyn1[1,587]=0.48767977136115;
fullsyn1[1,588]=-0.0971725138896334;
fullsyn1[1,589]=0.0703442340715185;
fullsyn1[1,590]=-0.522940464901746;
fullsyn1[1,591]=-0.192724975296236;
fullsyn1[1,1142]=0.29739857721521;
fullsyn1[1,1143]=-0.432668521730323;
fullsyn1[1,1144]=-0.367120993521371;
fullsyn1[1,1145]=-0.876498113437781;
fullsyn1[1,1146]=0.0534130202415712;
fullsyn1[1,1147]=-0.446298443371552;
fullsyn1[1,1148]=-1.13967579468641;
fullsyn1[1,1149]=-0.311769368674978;
fullsyn1[1,1150]=0.340314660372613;
fullsyn1[1,1151]=0.378357476942265;
fullsyn1[1,1152]=0.607787135608788;
fullsyn1[1,1153]=-0.156155295198395;
fullsyn1[1,1154]=1.19946320077536;
fullsyn1[1,1155]=-0.247070291636638;
fullsyn1[1,1156]=1.15621590528008;
fullsyn1[1,1157]=0.731774599415564;
fullsyn1[1,1158]=0.55572979736226;
fullsyn1[1,1159]=-0.154044432596511;
fullsyn1[1,1160]=0.159722612105018;
fullsyn1[1,1161]=-0.916648369857766;
fullsyn1[1,1162]=-0.39126618816954;
fullsyn1[1,1163]=-0.00911299352132931;
fullsyn1[1,1164]=-0.0246279683998544;
fullsyn1[1,1165]=0.0518787669378041;
fullsyn1[1,1166]=-0.211328988514197;
fullsyn1[1,1167]=0.429798008372359;
fullsyn1[1,1168]=0.00567096479424798;
fullsyn1[1,1169]=-0.368787820991757;
fullsyn1[1,1170]=-1.06472331093616;
fullsyn1[1,1171]=-0.315009885704549;
fullsyn1[1,1172]=-0.0524054052454072;
fullsyn1[1,1173]=-0.725272428327073;
fullsyn1[1,1174]=-0.796360534869749;
fullsyn1[1,1175]=-0.736190987895303;
fullsyn1[1,1176]=0.298517025887953;
fullsyn1[1,1177]=-0.810477152158122;
fullsyn1[1,1178]=-0.27222798352301;
fullsyn1[1,1179]=0.499998233607403;
fullsyn1[1,1180]=-0.202268550519428;
fullsyn1[1,1181]=0.207172184261818;
fullsyn1[1,1182]=0.0157278287755876;
fullsyn1[1,1183]=-0.828972764764711;
fullsyn1[1,1184]=-0.319024518347468;
fullsyn1[1,1185]=0.414647546903568;
fullsyn1[1,1186]=0.664474200199403;
fullsyn1[1,1187]=-0.396327513302443;
fullsyn1[1,1188]=-0.149100348634937;
fullsyn1[1,1189]=0.289946124698752;
fullsyn1[1,1190]=-0.153389557053021;
fullsyn1[1,1191]=-0.175047232676797;

```
fullsyn1[1,592]=-0.276624918186397;
fullsyn1[1,593]=0.153073235903001;
fullsyn1[1,594]=-0.121213877879976;
fullsyn1[1,595]=0.440072789898326;
fullsyn1[1,596]=0.655564812743196;
fullsyn1[1,597]=-0.321664548394233;
fullsyn1[1,598]=-0.285405865622562;
fullsyn1[1,599]=-0.359202426368263;
fullsyn1[1,600]=-0.768407215526221;
fullsyn1[1,1192]=-0.00139267114384749;
fullsyn1[1,1193]=0.166237382907284;
fullsyn1[1,1194]=0.19118577828198;
fullsyn1[1,1195]=0.367624228434971;
fullsyn1[1,1196]=0.154287677144361;
fullsyn1[1,1197]=-0.826923665968322;
fullsyn1[1,1198]=-0.306412056318159;
fullsyn1[1,1199]=0.304988689381861;
fullsyn1[1,1200]=0.496365934234298;

  // Hidden Layer 2
fullsyn1[2,1]=0.431719517860281;   fullsyn1[2,201]=-0.180468050015086;
fullsyn1[2,2]=0.356664062678777;   fullsyn1[2,202]=-0.650922852102899;
fullsyn1[2,3]=0.292975989194337;   fullsyn1[2,203]=-0.253896494217855;
fullsyn1[2,4]=0.305184852767207;   fullsyn1[2,204]=-0.0808084418092387;
fullsyn1[2,5]=-0.573100929905267;  fullsyn1[2,205]=0.232208719403048;
fullsyn1[2,6]=-0.0876601576430648; fullsyn1[2,206]=-0.205909255889899;
fullsyn1[2,7]=0.505820234995847;   fullsyn1[2,207]=0.347027396988292;
fullsyn1[2,8]=0.434671541221646;   fullsyn1[2,208]=0.192316187535363;
fullsyn1[2,9]=0.207996566942837;   fullsyn1[2,209]=-0.639975291070898;
fullsyn1[2,10]=-0.454446532728464; fullsyn1[2,210]=-0.466475091193866;
fullsyn1[2,11]=0.471544122491261;  fullsyn1[2,211]=0.396781219523745;
fullsyn1[2,12]=0.421280392248609;  fullsyn1[2,212]=-0.48753376398673;
fullsyn1[2,13]=-0.310031320860376; fullsyn1[2,213]=0.145382457420084;
fullsyn1[2,14]=0.220845773463362;  fullsyn1[2,214]=-0.456385626034454;
fullsyn1[2,15]=0.517876441989179;  fullsyn1[2,215]=0.105193898580664;
fullsyn1[2,16]=0.0910095573729809; fullsyn1[2,216]=0.20074084379662;
fullsyn1[2,17]=-0.367379836180466; fullsyn1[2,217]=-0.111317856124194;
fullsyn1[2,18]=0.509817654046228;  fullsyn1[2,218]=-0.0712623460565191;
fullsyn1[2,19]=-0.489348404956808; fullsyn1[2,219]=-0.492003687686531;
fullsyn1[2,20]=-0.200633839334812; fullsyn1[2,220]=0.0627427184572792;
fullsyn1[2,21]=-0.541503315899056; fullsyn1[2,221]=0.00375779430493521;
fullsyn1[2,22]=0.573971036920896;  fullsyn1[2,222]=0.543541620202254;
fullsyn1[2,23]=-0.138829984435277; fullsyn1[2,223]=0.374846741282235;
fullsyn1[2,24]=0.170667142481251;  fullsyn1[2,224]=0.499115330149728;
fullsyn1[2,25]=0.344991942081595;  fullsyn1[2,225]=0.0555636765075669;
fullsyn1[2,26]=0.117859954969742;  fullsyn1[2,226]=0.117734434815924;
fullsyn1[2,27]=0.129305986027945;  fullsyn1[2,227]=0.244654739491612;
fullsyn1[2,28]=0.566238820373827;  fullsyn1[2,228]=-0.420562443465491;
fullsyn1[2,29]=0.724178864228149;  fullsyn1[2,229]=-0.0206936427315012;
fullsyn1[2,30]=0.445243345249032;  fullsyn1[2,230]=-0.331879565176124;
fullsyn1[2,31]=-0.281554406110921; fullsyn1[2,231]=0.193724611382029;
fullsyn1[2,32]=-0.108539706924981; fullsyn1[2,232]=0.508892445581011;
fullsyn1[2,33]=-0.28641657195564; fullsyn1[2,233]=0.319404327746251;
fullsyn1[2,34]=0.153179350796762;  fullsyn1[2,234]=0.0207140459335272;
fullsyn1[2,35]=0.208106251671961;  fullsyn1[2,235]=-0.307491571554126;
fullsyn1[2,36]=-0.0655249242098698; fullsyn1[2,236]=-0.309059233544059;
fullsyn1[2,37]=0.581597880326693;  fullsyn1[2,237]=-0.0380365530987496;
fullsyn1[2,38]=0.256308156696832;  fullsyn1[2,238]=-0.547254139242287;
fullsyn1[2,39]=0.332602329202936;  fullsyn1[2,239]=0.545046607689045;
```

fullsyn1[2,40]=0.0171114116861305; fullsyn1[2,240]=0.121571393848502;
fullsyn1[2,41]=0.167323398212998; fullsyn1[2,241]=0.415872396825844;
fullsyn1[2,42]=0.584993795204307; fullsyn1[2,242]=-0.529972585405019;
fullsyn1[2,43]=0.112755298266301; fullsyn1[2,243]=0.140229506620505;
fullsyn1[2,44]=-0.0798547157278386; fullsyn1[2,244]=0.142079773490329;
fullsyn1[2,45]=-0.456736199705915; fullsyn1[2,245]=-0.532759297947059;
fullsyn1[2,46]=0.0347737075949148; fullsyn1[2,246]=0.408583957313793;
fullsyn1[2,47]=-0.491976438684471; fullsyn1[2,247]=-0.180255096350339;
fullsyn1[2,48]=0.208269804426291; fullsyn1[2,248]=-0.402533633244596;
fullsyn1[2,49]=-0.104016346652935; fullsyn1[2,249]=0.574505736793043;
fullsyn1[2,50]=0.0794139326709866; fullsyn1[2,250]=0.309577153459919;
fullsyn1[2,51]=0.187614678775383; fullsyn1[2,251]=-0.40326808371515;
fullsyn1[2,52]=-0.440178481642566; fullsyn1[2,252]=-0.298890474203006;
fullsyn1[2,53]=0.140800645600431; fullsyn1[2,253]=-0.296922823014991;
fullsyn1[2,54]=0.380406232455064; fullsyn1[2,254]=0.519418330298522;
fullsyn1[2,55]=0.260653484451189; fullsyn1[2,255]=-0.256437035737391;
fullsyn1[2,56]=0.253064500731167; fullsyn1[2,256]=0.00471981404965763;
fullsyn1[2,57]=-0.245662791063113; fullsyn1[2,257]=0.0567120581311341;
fullsyn1[2,58]=0.492729189948672; fullsyn1[2,258]=-0.0441448562715092;
fullsyn1[2,59]=0.325145908426639; fullsyn1[2,259]=-0.134325100822218;
fullsyn1[2,60]=-0.35710150801266; fullsyn1[2,260]=-0.0698013203633207;
fullsyn1[2,61]=-0.145618524571969; fullsyn1[2,261]=0.428453518415819;
fullsyn1[2,62]=0.508474492084555; fullsyn1[2,262]=0.131418126820442;
fullsyn1[2,63]=0.242094706426777; fullsyn1[2,263]=0.562552359111249;
fullsyn1[2,64]=0.124911090842838; fullsyn1[2,264]=-0.356399823934989;
fullsyn1[2,65]=0.356623449783275; fullsyn1[2,265]=-0.441751454111038;
fullsyn1[2,66]=0.180668788282543; fullsyn1[2,266]=-0.267315985208791;
fullsyn1[2,67]=-0.443772892483116; fullsyn1[2,267]=0.180495115858108;
fullsyn1[2,68]=-0.299983908117984; fullsyn1[2,268]=0.172609546746976;
fullsyn1[2,69]=0.178852752438369; fullsyn1[2,269]=0.350474200295218;
fullsyn1[2,70]=0.149544489142156; fullsyn1[2,270]=-0.583426605634849;
fullsyn1[2,71]=0.19506823893817; fullsyn1[2,271]=-0.373759677391435;
fullsyn1[2,72]=0.202244006346514; fullsyn1[2,272]=0.627438077923198;
fullsyn1[2,73]=0.220955128561266; fullsyn1[2,273]=0.524248589220049;
fullsyn1[2,74]=0.539865713855923; fullsyn1[2,274]=-0.0878160088089187;
fullsyn1[2,75]=-0.318702297118952; fullsyn1[2,275]=-0.365416961748693;
fullsyn1[2,76]=-0.167326376509732; fullsyn1[2,276]=-0.0577999968118891;
fullsyn1[2,77]=0.287344635152516; fullsyn1[2,277]=0.513054000916136;
fullsyn1[2,78]=-0.526082756517879; fullsyn1[2,278]=0.337010440022134;
fullsyn1[2,79]=-0.46901217084715; fullsyn1[2,279]=-0.476306946680414;
fullsyn1[2,80]=0.394719513610615; fullsyn1[2,280]=-0.0892364211199636;
fullsyn1[2,81]=0.106958516561523; fullsyn1[2,281]=0.629286537342629;
fullsyn1[2,82]=0.447541185334701; fullsyn1[2,282]=-0.135452926475837;
fullsyn1[2,83]=0.0802173073024129; fullsyn1[2,283]=-0.374669770026309;
fullsyn1[2,84]=0.410478236479913; fullsyn1[2,284]=-0.580233270951462;
fullsyn1[2,85]=0.453030812846475; fullsyn1[2,285]=0.385607006193306;
fullsyn1[2,86]=-0.328253983426336; fullsyn1[2,286]=-0.560775303667701;
fullsyn1[2,87]=0.0995040487836004; fullsyn1[2,287]=0.529966565705813;
fullsyn1[2,88]=0.08453885646416; fullsyn1[2,288]=0.310866011480814;
fullsyn1[2,89]=-0.00507995119799792; fullsyn1[2,289]=0.0219357262352666;

fullsyn1[2,90]=-0.27931695689809;
fullsyn1[2,91]=0.220827288584835;
fullsyn1[2,92]=-0.0190018280659225;
fullsyn1[2,93]=0.223293883977954;
fullsyn1[2,94]=-0.144330425881593;
fullsyn1[2,95]=-0.0429425284060869;
fullsyn1[2,96]=0.25567683679305;
fullsyn1[2,97]=0.240214313099191;
fullsyn1[2,98]=0.347246425250581;
fullsyn1[2,99]=0.26131240967345;
fullsyn1[2,100]=-0.381391151789847;
fullsyn1[2,101]=-0.224882939971166;
fullsyn1[2,102]=0.202468593094309;
fullsyn1[2,103]=0.310655493296926;
fullsyn1[2,104]=0.00133954970995613;
fullsyn1[2,105]=-0.43009342660587;
fullsyn1[2,106]=-0.309994303542022;
fullsyn1[2,107]=-0.0327616149914788;
fullsyn1[2,108]=0.514880033771089;
fullsyn1[2,109]=-0.383459562744442;
fullsyn1[2,110]=-0.174936657915999;
fullsyn1[2,111]=-0.0957688376336949;
fullsyn1[2,112]=0.356739374930951;
fullsyn1[2,113]=-0.52143912587097;
fullsyn1[2,114]=0.175801277508878;
fullsyn1[2,115]=-0.244421073377395;
fullsyn1[2,116]=-0.498876833494534;
fullsyn1[2,117]=-0.200925005125644;
fullsyn1[2,118]=-0.0130444691860793;
fullsyn1[2,119]=0.459856148237256;
fullsyn1[2,120]=-0.193356640474804;
fullsyn1[2,121]=-0.167069189154978;
fullsyn1[2,122]=0.134693533729081;
fullsyn1[2,123]=0.156986334009692;
fullsyn1[2,124]=0.276207665268213;
fullsyn1[2,125]=-0.462684152996995;
fullsyn1[2,126]=0.0625262943201742;
fullsyn1[2,127]=0.0883106468993218;
fullsyn1[2,128]=-0.280849322889985;
fullsyn1[2,129]=0.376458300021159;
fullsyn1[2,130]=-0.45176476639591;
fullsyn1[2,131]=-0.27489201102914;
fullsyn1[2,132]=0.231790705638893;
fullsyn1[2,133]=-0.432386294364431;
fullsyn1[2,134]=0.075968355179239;
fullsyn1[2,135]=-0.263310516753238;
fullsyn1[2,136]=-0.0125925502400166;
fullsyn1[2,137]=-0.0511892849727696;
fullsyn1[2,138]=-0.263281053045976;
fullsyn1[2,139]=0.00232917643492137;
fullsyn1[2,290]=-0.0192934485405487;
fullsyn1[2,291]=-0.0485468362689162;
fullsyn1[2,292]=0.540332266871563;
fullsyn1[2,293]=-0.617102858368198;
fullsyn1[2,294]=0.387199840047941;
fullsyn1[2,295]=-0.0441945701003513;
fullsyn1[2,296]=0.107050641832651;
fullsyn1[2,297]=0.0905442320806394;
fullsyn1[2,298]=-0.354001354831379;
fullsyn1[2,299]=-0.153221049081396;
fullsyn1[2,300]=-0.421291735785366;
fullsyn1[2,301]=-0.558666803939057;
fullsyn1[2,302]=-0.404703819573809;
fullsyn1[2,303]=0.218995318691582;
fullsyn1[2,304]=-0.40753834533625;
fullsyn1[2,305]=0.35060772522211;
fullsyn1[2,306]=-0.324025048175227;
fullsyn1[2,307]=-0.23515938762016;
fullsyn1[2,308]=0.0613678626990196;
fullsyn1[2,309]=-0.418408582220504;
fullsyn1[2,310]=-0.450948457525653;
fullsyn1[2,311]=0.180415111273805;
fullsyn1[2,312]=0.303729971115002;
fullsyn1[2,313]=-0.236274966567145;
fullsyn1[2,314]=-0.602637369780981;
fullsyn1[2,315]=-0.242164453054874;
fullsyn1[2,316]=0.0491332351766317;
fullsyn1[2,317]=0.103085930827932;
fullsyn1[2,318]=-0.428551888691093;
fullsyn1[2,319]=0.491689479082796;
fullsyn1[2,320]=0.203395386817524;
fullsyn1[2,321]=-0.234091875870822;
fullsyn1[2,322]=-0.183782561144348;
fullsyn1[2,323]=0.283115666209302;
fullsyn1[2,324]=0.487411325085914;
fullsyn1[2,325]=0.0524183006812727;
fullsyn1[2,326]=0.169593708115662;
fullsyn1[2,327]=0.279030296570308;
fullsyn1[2,328]=0.0422677036993756;
fullsyn1[2,329]=0.729770785281019;
fullsyn1[2,330]=-0.512976252387145;
fullsyn1[2,331]=-0.162993044585072;
fullsyn1[2,332]=-0.10869149134928;
fullsyn1[2,333]=0.0350376961312865;
fullsyn1[2,334]=-0.312353110944816;
fullsyn1[2,335]=0.373038004923151;
fullsyn1[2,336]=-0.314574880487883;
fullsyn1[2,337]=0.367704325360817;
fullsyn1[2,338]=-0.24088188588432;
fullsyn1[2,339]=-0.25411229756816;

fullsyn1[2,140]=-0.109546865317694;
fullsyn1[2,141]=0.393837958564615;
fullsyn1[2,142]=0.33754793046977;
fullsyn1[2,143]=-0.524036469790182;
fullsyn1[2,144]=-0.385252131524286;
fullsyn1[2,145]=0.47074509604045;
fullsyn1[2,146]=0.271191373056022;
fullsyn1[2,147]=-0.0967087219386041;
fullsyn1[2,148]=-0.374047739537605;
fullsyn1[2,149]=0.509188291154835;
fullsyn1[2,150]=0.523666323530091;
fullsyn1[2,151]=0.199487812586381;
fullsyn1[2,152]=-0.121372879554762;
fullsyn1[2,153]=-0.067178697173929;
fullsyn1[2,154]=0.151824498921536;
fullsyn1[2,155]=0.600899759581339;
fullsyn1[2,156]=-0.338814646797441;
fullsyn1[2,157]=-0.209296362881082;
fullsyn1[2,158]=0.364227064635989;
fullsyn1[2,159]=0.44455455597344;
fullsyn1[2,160]=-0.115100164121947;
fullsyn1[2,161]=0.214602637135003;
fullsyn1[2,162]=-0.200138584667265;
fullsyn1[2,163]=0.410593237943324;
fullsyn1[2,164]=0.481936455281758;
fullsyn1[2,165]=-0.105169770396396;
fullsyn1[2,166]=0.324423617823847;
fullsyn1[2,167]=-0.467783172453923;
fullsyn1[2,168]=0.0330298917640596;
fullsyn1[2,169]=0.079610706052749;
fullsyn1[2,170]=0.389285803766173;
fullsyn1[2,171]=-0.286894241803096;
fullsyn1[2,172]=0.0869772283370661;
fullsyn1[2,173]=-0.271226675871084;
fullsyn1[2,174]=-0.144431646361165;
fullsyn1[2,175]=-0.271155793089995;
fullsyn1[2,176]=0.31028543653703;
fullsyn1[2,177]=-0.380990017754359;
fullsyn1[2,178]=-0.286325164747621;
fullsyn1[2,179]=0.0170660568803646;
fullsyn1[2,180]=-0.531388077458939;
fullsyn1[2,181]=0.0130482895051666;
fullsyn1[2,182]=-0.0719245432782057;
fullsyn1[2,183]=0.205912592515044;
fullsyn1[2,184]=0.0209540174887149;
fullsyn1[2,185]=-0.245451052871958;
fullsyn1[2,186]=-0.151544809748505;
fullsyn1[2,187]=-0.41562861902516;
fullsyn1[2,188]=0.314059593462082;
fullsyn1[2,189]=0.142573064986407;
fullsyn1[2,340]=-0.147396617612685;
fullsyn1[2,341]=-0.350235089079727;
fullsyn1[2,342]=0.149820931666365;
fullsyn1[2,343]=-0.431260774765777;
fullsyn1[2,344]=0.453843321350139;
fullsyn1[2,345]=0.17456769658264;
fullsyn1[2,346]=0.226695532619679;
fullsyn1[2,347]=-0.205502744096704;
fullsyn1[2,348]=-0.149980053877716;
fullsyn1[2,349]=0.0110765135769983;
fullsyn1[2,350]=-0.0353622173070988;
fullsyn1[2,351]=0.285467456613318;
fullsyn1[2,352]=-0.209845890630313;
fullsyn1[2,353]=-0.617886398807025;
fullsyn1[2,354]=0.0592657919320289;
fullsyn1[2,355]=-0.202053315950203;
fullsyn1[2,356]=-0.522218629154535;
fullsyn1[2,357]=-0.225941467369846;
fullsyn1[2,358]=0.0216940483677402;
fullsyn1[2,359]=0.00227608390877984;
fullsyn1[2,360]=0.030145789365726;
fullsyn1[2,361]=0.164187071441866;
fullsyn1[2,362]=-0.493800267030848;
fullsyn1[2,363]=-0.662040024134258;
fullsyn1[2,364]=-0.037832622409198;
fullsyn1[2,365]=0.0747788700723645;
fullsyn1[2,366]=-0.355061521530791;
fullsyn1[2,367]=-0.560195150628329;
fullsyn1[2,368]=-0.369304337184421;
fullsyn1[2,369]=-0.0176727098434577;
fullsyn1[2,370]=-0.575281913981111;
fullsyn1[2,371]=0.0724151890368101;
fullsyn1[2,372]=-0.0012501506009315;
fullsyn1[2,373]=-0.0427026932110658;
fullsyn1[2,374]=0.55786049264254;
fullsyn1[2,375]=-0.541702491932443;
fullsyn1[2,376]=-0.20275967122873;
fullsyn1[2,377]=-0.260864900412847;
fullsyn1[2,378]=0.360503976344338;
fullsyn1[2,379]=0.0346954524517217;
fullsyn1[2,380]=0.456139814773668;
fullsyn1[2,381]=-0.503680921021553;
fullsyn1[2,382]=-0.272162835074147;
fullsyn1[2,383]=0.53063291573095;
fullsyn1[2,384]=-0.144749073167128;
fullsyn1[2,385]=-0.302851252947279;
fullsyn1[2,386]=0.239422570125331;
fullsyn1[2,387]=-0.288098749906504;
fullsyn1[2,388]=0.601356952188928;
fullsyn1[2,389]=-0.243190764608142;

```
fullsyn1[2,190]=0.406798938030249;
fullsyn1[2,191]=-0.22733532350497;
fullsyn1[2,192]=-0.272519362858214;
fullsyn1[2,193]=-0.0981111601091294;
fullsyn1[2,194]=0.0248892270446076;
fullsyn1[2,195]=-0.205663293312768;
fullsyn1[2,196]=0.166274930330489;
fullsyn1[2,197]=-0.428674806460798;
fullsyn1[2,198]=0.0874089615820524;
fullsyn1[2,199]=0.0588339768539012;
fullsyn1[2,200]=0.601708001179738;
fullsyn1[2,390]=0.4823138378945;
fullsyn1[2,391]=-0.445842474255292;
fullsyn1[2,392]=-0.0780234412945238;
fullsyn1[2,393]=0.0608266109034226;
fullsyn1[2,394]=-0.0886108329338584;
fullsyn1[2,395]=0.227629352039011;
fullsyn1[2,396]=0.542902094318184;
fullsyn1[2,397]=-0.0962886785231981;
fullsyn1[2,398]=-0.641743057965777;
fullsyn1[2,399]=0.623652862300777;
fullsyn1[2,400]=-0.197915677788439;

  // Output Layer
fullsyn1[3,1]=0.43799127421934;fullsyn1[3,2]=0.244482890235801;
fullsyn1[3,3]=-0.280221118211213;
fullsyn1[3,4]=-0.18437826612804;fullsyn1[3,5]=0.580549450721184;
fullsyn1[3,6]=0.561912980047603;fullsyn1[3,7]=0.55094657175095;
fullsyn1[3,8]=-0.666835597958821;

  // Biases
bias[1,1]=-0.0272251951140398;
bias[1,2]=0.196756105341875;
bias[1,3]=0.64165302598821;
bias[1,4]=-0.179032267144605;
bias[1,5]=-0.406844348464442;
bias[1,6]=-0.373320405024269;
bias[1,7]=-0.284882576331039;
bias[1,8]=0.409384533419348;
bias[1,9]=-0.0143138963840587;
bias[1,10]=-0.458946905370941;
bias[1,11]=-0.465589995393675;
bias[1,12]=0.506750641421591;
bias[1,13]=-0.256846820994878;
bias[1,14]=-0.185698321359546;
bias[1,15]=0.00531048024929647;
bias[1,16]=0.237344383268763;
bias[1,17]=-0.36870039974538;
bias[1,18]=-0.27456946161616;
bias[1,19]=-0.297964683765259;
bias[1,20]=0.230374209261039;
bias[1,21]=-0.135617251518436;
bias[1,22]=-0.713763867182415;
bias[1,23]=0.00931315978673793;
bias[1,24]=-0.0351543738387615;
bias[1,25]=-0.230472917260572;
bias[1,26]=-0.223102603443474;
bias[1,27]=-0.463692005677045;
bias[1,28]=0.433731969315942;
bias[1,29]=0.729856547710838;
bias[1,30]=-0.506154385487535;
bias[1,31]=0.198809616331712;
bias[1,32]=0.123086275763156;
bias[1,33]=-0.819086493889508;
bias[1,34]=-0.0112671186522424;
bias[1,35]=0.30501757270444;
bias[1,36]=-0.413696466717676;
bias[1,37]=-0.450875111927179;
bias[1,38]=0.646264478842674;
bias[1,39]=-0.164582381455313;
bias[1,40]=-0.390808195859175;
bias[1,41]=0.139228246046815;
bias[1,42]=0.843225495308673;
bias[1,43]=0.135737305404231;
bias[1,44]=-0.505103038319927;
bias[1,45]=0.00152410888154581;
bias[1,46]=0.208287978513986;
bias[1,47]=-0.579997161362147;
bias[1,48]=-0.14481940533577;
bias[1,49]=0.181774848285751;
bias[1,50]=0.271237702662799;

bias[2,1]=-0.218680613610214;
bias[2,2]=0.357806415925815;
bias[2,3]=-0.416856857237482;
bias[2,4]=-0.360403319743084;
bias[2,5]=-0.122860363413092;
bias[2,6]=-0.077006729626786 2;
bias[2,7]=0.27079489727561;
bias[2,8]=0.454439692093879;
```

```
bias[3,1]=0.0649636077683965;

  // Normalize denormalize vectors
ampin[1]=0.359999999999999;       offin[1]=-1.635;
ampin[2]=0.257142857142857;       offin[2]=-1.15714285714285;
ampin[3]=0.00590163934426229;     offin[3]=-0.9;
ampin[4]=0.00335820895522388;     offin[4]=-0.9;
ampin[5]=0.00271903323262839;     offin[5]=-0.9;
ampin[6]=0.00233766233766233;     offin[6]=-0.9;
ampin[7]=0.00262390670553935;     offin[7]=-0.902623906705539;
ampin[8]=0.00262008733624454;     offin[8]=-0.9;
ampin[9]=0.00244565217391304;     offin[9]=-0.9;
ampin[10]=0.00413793103448275;    offin[10]=-0.904137931034482;
ampin[11]=87.9966815570946;       offin[11]=-88.8966815570946;
ampin[12]=76.5409090909094;       offin[12]=-75.6409090909094;
ampin[13]=36.9054057681246;       offin[13]=-36.9712191511802;
ampin[14]=38.864344661993;        offin[14]=-38.6877939412663;
ampin[15]=36.4860899241875;       offin[15]=-36.6110197859602;
ampin[16]=36.3851620945626;       offin[16]=-36.145838287988;
ampin[17]=32.362664658824;        offin[17]=-32.6126123984888;
ampin[18]=35.7079174732325;       offin[18]=-35.4003614477056;
ampin[19]=23.1943916194993;       offin[19]=-23.6284977613048;
ampin[20]=32.1554900281214;       offin[20]=-31.6248714241577;
ampin[21]=18.8194947950079;       offin[21]=-19.3414772473544;
ampin[22]=25.4292884211934;       offin[22]=-24.786801483483;
ampin[23]=15.1601190714691;       offin[23]=-15.8523957549074;
ampin[24]=17.3876920982857;       offin[24]=-16.6637708687261;

ampdes=0.9; offdes=0;

  // Input parameter calculations
                    level0=Low[Lowest(NULL, 0, MODE_LOW, 376, 0)];
                    level8=High[Highest(NULL, 0, MODE_HIGH, 376, 0)];
                    level1=level0+(level8-level0)/8;
                    level2=level0+(level8-level0)*2/8;
                    level3=level0+(level8-level0)*3/8;
                    level4=level0+(level8-level0)*4/8;
                    level5=level0+(level8-level0)*5/8;
                    level6=level0+(level8-level0)*6/8;
                    level7=level0+(level8-level0)*7/8;
                    if (iClose(NULL,0,0)<level1)
                      clocation=1;
                      else if (iClose(NULL,0,0)<level2)
                           clocation=2;
                                else if (iClose(NULL,0,0)<level3)
                                     clocation=3;
                                   else if (iClose(NULL,0,0)<level4)
                                                clocation=4;
                                                  else if (iClose(NULL,0,0)<level5)
```

```
                                                clocation=5;
                                               else if
(iClose(NULL,0,0)<level6)
                                                    clocation=6;
                                                  else if
(iClose(NULL,0,0)<level7)

clocation=7;
                                                        else
clocation=8;

                L1OCount = 0; L1HCount = 0; L1LCount = 0; L1CCount = 0; L1Count
= 0;
                L2OCount = 0; L2HCount = 0; L2LCount = 0; L2CCount = 0; L2Count
= 0;
                L3OCount = 0; L3HCount = 0; L3LCount = 0; L3CCount = 0; L3Count
= 0;
                L4OCount = 0; L4HCount = 0; L4LCount = 0; L4CCount = 0; L4Count
= 0;
                L5OCount = 0; L5HCount = 0; L5LCount = 0; L5CCount = 0; L5Count
= 0;
                L6OCount = 0; L6HCount = 0; L6LCount = 0; L6CCount = 0; L6Count
= 0;
                L7OCount = 0; L7HCount = 0; L7LCount = 0; L7CCount = 0; L7Count
= 0;
                L8OCount = 0; L8HCount = 0; L8LCount = 0; L8CCount = 0; L8Count
= 0;
                for (i=0;i<=376;i++)
                  {
                        if (iOpen(NULL,0,i)>level0 &&  iOpen(NULL,0,i)<level1)
L1OCount++;
                        if (iHigh(NULL,0,i)>level0 &&  iHigh(NULL,0,i)<level1)
L1HCount++;
                        if (iLow(NULL,0,i)>level0 &&  iLow(NULL,0,i)<level1)
L1LCount++;
                        if (iClose(NULL,0,i)>level0 &&  iClose(NULL,0,i)<level1)
L1CCount++;
                        if (iOpen(NULL,0,i)>level1 &&  iOpen(NULL,0,i)<level2)
L2OCount++;
                        if (iHigh(NULL,0,i)>level1 &&  iHigh(NULL,0,i)<level2)
L2HCount++;
                        if (iLow(NULL,0,i)>level1 &&  iLow(NULL,0,i)<level2)
L2LCount++;
                        if (iClose(NULL,0,i)>level1 &&  iClose(NULL,0,i)<level2)
L2CCount++;
                        if (iOpen(NULL,0,i)>level2 &&  iOpen(NULL,0,i)<level3)
L3OCount++;
                        if (iHigh(NULL,0,i)>level2 &&  iHigh(NULL,0,i)<level3)
L3HCount++;
```

```
                    if (iLow(NULL,0,i)>level2 &&  iLow(NULL,0,i)<level3)
L3LCount++;
                    if (iClose(NULL,0,i)>level2 &&  iClose(NULL,0,i)<level3)
L3CCount++;
                    if (iOpen(NULL,0,i)>level3 &&  iOpen(NULL,0,i)<level4)
L4OCount++;
                    if (iHigh(NULL,0,i)>level3 &&  iHigh(NULL,0,i)<level4)
L4HCount++;
                    if (iLow(NULL,0,i)>level3 &&  iLow(NULL,0,i)<level4)
L4LCount++;
                    if (iClose(NULL,0,i)>level3 &&  iClose(NULL,0,i)<level4)
L4CCount++;
                    if (iOpen(NULL,0,i)>level4 &&  iOpen(NULL,0,i)<level5)
L5OCount++;
                    if (iHigh(NULL,0,i)>level4 &&  iHigh(NULL,0,i)<level5)
L5HCount++;
                    if (iLow(NULL,0,i)>level4 &&  iLow(NULL,0,i)<level5)
L5LCount++;
                    if (iClose(NULL,0,i)>level4 &&  iClose(NULL,0,i)<level5)
L5CCount++;
                    if (iOpen(NULL,0,i)>level5 &&  iOpen(NULL,0,i)<level6)
L6OCount++;
                    if (iHigh(NULL,0,i)>level5 &&  iHigh(NULL,0,i)<level6)
L6HCount++;
                    if (iLow(NULL,0,i)>level5 &&  iLow(NULL,0,i)<level6)
L6LCount++;
                    if (iClose(NULL,0,i)>level5 &&  iClose(NULL,0,i)<level6)
L6CCount++;
                    if (iOpen(NULL,0,i)>level6 &&  iOpen(NULL,0,i)<level7)
L7OCount++;
                    if (iHigh(NULL,0,i)>level6 &&  iHigh(NULL,0,i)<level7)
L7HCount++;
                    if (iLow(NULL,0,i)>level6 &&  iLow(NULL,0,i)<level7)
L7LCount++;
                    if (iClose(NULL,0,i)>level6 &&  iClose(NULL,0,i)<level7)
L7CCount++;
                    if (iOpen(NULL,0,i)>level7 &&  iOpen(NULL,0,i)<level8)
L8OCount++;
                    if (iHigh(NULL,0,i)>level7 &&  iHigh(NULL,0,i)<level8)
L8HCount++;
                    if (iLow(NULL,0,i)>level7 &&  iLow(NULL,0,i)<level8)
L8LCount++;
                    if (iClose(NULL,0,i)>level7 &&  iClose(NULL,0,i)<level8)
L8CCount++;
                  }
                L1Count = L1OCount + L1HCount + L1LCount + L1Count;
                L2Count = L2OCount + L2HCount + L2LCount + L2Count;
                L3Count = L3OCount + L3HCount + L3LCount + L3Count;
                L4Count = L4OCount + L4HCount + L4LCount + L4Count;
                L5Count = L5OCount + L5HCount + L5LCount + L5Count;
```

```
                L6Count = L6OCount + L6HCount + L6LCount + L6Count;
                L7Count = L7OCount + L7HCount + L7LCount + L7Count;
                L8Count = L8OCount + L8HCount + L8LCount + L8Count;
                cevir = MathMod(TimeCurrent()-86400,86400*7)/86400;
        gun = MathFloor(cevir);
        saat = cevir - gun;
        gun--;
        if (gun<=0) gun = gun +7;

              //Inputs
                inp5plus[1]=gun+saat;
                inp5plus[2]=clocation;
                inp5plus[3]=L1Count;
                inp5plus[4]=L2Count;
                inp5plus[5]=L3Count;
                inp5plus[6]=L4Count;
                inp5plus[7]=L5Count;
                inp5plus[8]=L6Count;
                inp5plus[9]=L7Count;
                inp5plus[10]=L8Count;
                inp5plus[11]=iHigh(NULL,0,0)/iClose(NULL,0,0);
                inp5plus[12]=iLow(NULL,0,0)/iClose(NULL,0,0);
                inp5plus[13]=iHigh(NULL,0,1)/iClose(NULL,0,0);
                inp5plus[14]=iLow(NULL,0,1)/iClose(NULL,0,0);
                inp5plus[15]=High[Highest(NULL, 0, MODE_HIGH, 2,
1)]/iClose(NULL,0,0);
                inp5plus[16]=Low[Lowest(NULL, 0, MODE_LOW, 2,
1)]/iClose(NULL,0,0);
                inp5plus[17]=High[Highest(NULL, 0, MODE_HIGH, 13,
1)]/iClose(NULL,0,0);
                inp5plus[18]=Low[Lowest(NULL, 0, MODE_LOW, 13,
1)]/iClose(NULL,0,0);
                inp5plus[19]=High[Highest(NULL, 0, MODE_HIGH, 34,
1)]/iClose(NULL,0,0);
                inp5plus[20]=Low[Lowest(NULL, 0, MODE_LOW, 34,
1)]/iClose(NULL,0,0);
                inp5plus[21]=High[Highest(NULL, 0, MODE_HIGH, 89,
1)]/iClose(NULL,0,0);
                inp5plus[22]=Low[Lowest(NULL, 0, MODE_LOW, 89,
1)]/iClose(NULL,0,0);
                inp5plus[23]=High[Highest(NULL, 0, MODE_HIGH, 233,
1)]/iClose(NULL,0,0);
                inp5plus[24]=Low[Lowest(NULL, 0, MODE_LOW, 233,
1)]/iClose(NULL,0,0);

      // Layer 1 calculations

                for(i=1;i<=24;i++)
                 {
                  inp5plus[i]=ampin[i]*inp5plus[i]+offin[i];
```

```
                    }

                for(j=1;j<=50;j++) tophid1[j]=0;
                k=0;
                for(j=1;j<=50;j++)
                  {
                   for(i=1;i<=24;i++)
                              {
                               k++;
                               tophid1[j]=tophid1[j]+inp5plus[i]*fullsyn1[1,k];
                                        }
  }

                for(j=1;j<=50;j++) hid1_o[j]=tanh(tophid1[j]+bias[1,j]);

    // Layer 2 calculations
for(j=1;j<=8;j++) tophid2[j]=0;
                k=0;
                for(j=1;j<=8;j++)
                  {
                   for(i=1;i<=50;i++)
                              {
                               k++;
                               tophid2[j]=tophid2[j]+hid1_o[i]*fullsyn1[2,k];
                                        }
  }

                for(j=1;j<=8;j++) hid2_o[j]=tanh(tophid2[j]+bias[2,j]);

    // Output Layer calculations
        tophido_1=0;
                for(i=1;i<=8;i++) tophido_1=tophido_1+hid2_o[i]*fullsyn1[3,i];
                hido_o=tanh(tophido_1+bias[3,1]);
                hido_o=(hido_o-offdes)/ampdes;

    // Trading criteria
htresh=hido_o;
Print("output: ",htresh);
                activepositioncount=0;
                toplong=0;
                topshort=0;
                takeprofit=hedefkar/100000;
                stoploss=zarardur/100000;
                takeprofits=hedefkars/100000;
                stoplosss=zarardurs/100000;
                longesikdown = longesikup - diff;
                shortesikup = shortesikdown + diffs;

     OrderSelect(OrdersTotal()-1, SELECT_BY_POS, MODE_TRADES);
```

```
if ( ThisBarTrade != Bars )
                        {

                        totalo = OrdersTotal();
                        if (totalo==0)
   {
                          activepositioncount=0;
                                toplong=0;
                                topshort=0;
                                }
  for(j=totalo-1;j>=0;j--)
   {
          OrderSelect(j, SELECT_BY_POS);
          type   = OrderType();

    if (OrderMagicNumber()>40000)
     {
      switch(type)
      {

       case OP_BUY      : toplong++; activepositioncount++;
         break;

       case OP_SELL     : topshort++; activepositioncount++;
         break;
       }

     }
   }

                        if (activepositioncount==0)
                         {
                          thistime=0;
                                OrderSelect(OrdersHistoryTotal()-1,
SELECT_BY_POS, MODE_HISTORY);
                                if (OrderMagicNumber()==40002)
                                        beklen = bekles;
                                else
                                        beklen = bekle;
                                if (OrderProfit()<0 && TimeCurrent()-
OrderCloseTime()<beklen)
                                        yapma = 1;
                                else
                                        yapma = 0;
                                if (htresh>longesikup/100 && yapma == 0)
          {
                                          ThisBarTrade = Bars;
                                                OrderSelect(OrdersTotal()-1,
SELECT_BY_POS, MODE_TRADES);
```

```
        ticket=OrderSend(Symbol(),OP_BUY,Lot,Ask,3,0,0,"x-Buy",40001,0,Blue);
                                                    tk = ticket;

OrderModify(ticket,OrderOpenPrice(),0.96*Bid,1.035*Ask,0,Blue);

                if (ticket<0)
                 {
                  Print("OrderSend failed with error #",GetLastError());
                                                        return(0);
                  }
                 }
            else if (htresh<shortesikdown/100 && yapma == 0)
                                    {
                                     ThisBarTrade = Bars;
                                                OrderSelect(OrdersTotal()-1,
SELECT_BY_POS, MODE_TRADES);
                ticket=OrderSend(Symbol(),OP_SELL,Lot,Bid,3,0,0,"x-Short",40002,0,Red);
                                                tk = ticket;

OrderModify(ticket,OrderOpenPrice(),1.04*Ask,0.965*Bid,0,Red);

                if (ticket<0)
                 {
                  Print("OrderSend failed with error #",GetLastError());
                                                        return(0);
                 }
                                        }
                                      }
        if (activepositioncount==1)
                                      {
                                       priceoffset=iClose(NULL,0,0)-OrderOpenPrice();
                                       OrderSelect(OrdersTotal()-1, SELECT_BY_POS,
MODE_TRADES);

                            if (OrderMagicNumber() == 40001)
                                        {
                                                takeprofit = (tpbaz+(hedefkar-
tpbaz)*(TimeCurrent()-OrderOpenTime())/(60*60*zamanasim))/100000;
                                                mantiksal1 = Ask >
takeprofit+OrderOpenPrice();
                                                mantiksal2 = Bid < OrderOpenPrice()-
stoploss;
                                                mantiksal3 = TimeCurrent()-
OrderOpenTime()>60*60*zamanasim && OrderProfit()<0;
                                                mantiksal4 = TimeCurrent()-
OrderOpenTime()>60*60*zamanasim && OrderProfit()>0 && htresh<longesikup/100;
                                                mantiksal5 = htresh<longesikdown/100;
```

```
                                                    if (mantiksal1 || mantiksal2 || mantiksal3 ||
mantiksal4 || mantiksal5)
                                                     {
                                                totalo = OrdersTotal();
                for(j=totalo-1;j>=0;j--)
                {
                 OrderSelect(j, SELECT_BY_POS);
                 type   = OrderType();
                 result = false;
                 if (OrderMagicNumber()>40000)
                                                                {
                   switch(type)
                   {
                    //Close opened long positions
                    case OP_BUY       : result = OrderClose( OrderTicket(), OrderLots(),
MarketInfo(OrderSymbol(), MODE_BID), 5, MediumVioletRed );
                    break;

                    //Close opened short positions
                    case OP_SELL      :
                    break;

                    //Close pending orders
                    case OP_BUYLIMIT  :
                    case OP_BUYSTOP   :
                    case OP_SELLLIMIT :
                    case OP_SELLSTOP  : result = OrderDelete( OrderTicket() );
                  }

                 if(result == false)
                  {
                   Alert("Order " , OrderTicket() , " failed to close. Error:" , GetLastError() );
                   Sleep(3000);
                   }
                 }
                }
                                                  return(0);

                          }


                        }
                                        OrderSelect(OrdersTotal()-1, SELECT_BY_POS,
MODE_TRADES);
                         if ((OrderMagicNumber() == 40002) )
                                                  {
                                                    takeprofit = (tpbazs+(hedefkar-
tpbazs)*(TimeCurrent()-OrderOpenTime())/(60*60*zamanasim))/100000;
                                                    mantiksal1 = Bid < OrderOpenPrice()-
takeprofits;
```

```
                                                mantiksal2 = Ask >
OrderOpenPrice()+stoplosss;
                                                mantiksal3 = TimeCurrent()-
OrderOpenTime()>60*60*zamanasims && OrderProfit()<0;
                                                mantiksal4 = TimeCurrent()-
OrderOpenTime()>60*60*zamanasims && OrderProfit()>0 && htresh>shortesikdown/100;
                                                mantiksal5 = htresh>shortesikup/100;

                                                if (mantiksal1 || mantiksal2 || mantiksal3 ||
mantiksal4 || mantiksal5)
                                                 {
                                totalo = OrdersTotal();
                  for(j=totalo-1;j>=0;j--)
                  {
                   OrderSelect(j, SELECT_BY_POS);
                   type   = OrderType();
                   result = false;
                   if (OrderMagicNumber()>40000)
                                                                {
                     switch(type)
                     {
                      //Close opened long positions
                      case OP_BUY       :
                      break;

                      //Close opened short positions
                      case OP_SELL      : result = OrderClose( OrderTicket(), OrderLots(),
MarketInfo(OrderSymbol(), MODE_ASK), 5, Red );
                      break;

                      //Close pending orders
                      case OP_BUYLIMIT  :
                      case OP_BUYSTOP   :
                      case OP_SELLLIMIT :
                      case OP_SELLSTOP  : result = OrderDelete( OrderTicket() );
                     }

                    if(result == false)
                     {
                      Alert("Order " , OrderTicket() , " failed to close. Error:" , GetLastError() );
                      Sleep(3000);
                     }
                    }
                   }
                 }
                 return(0);

                                }

                                        }
```

```
      }                                 // ThisBarTrade

  return;                               // Exit start()
 }


double tanh(double x)
 {
  return((MathExp(2*x)-1)/(MathExp(2*x)+1));
  //return(sonuc);
  }
```