

MACHINE LEARNING:  
A POTENTIAL FORECASTING TOOL

By

JASDEEP SINGH BANGA

Bachelor of Science (Hons.) in Agriculture  
Punjab agricultural University  
Punjab, India  
2004

Master of Business Administration  
Punjab Agricultural University  
Punjab, India  
2006

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
DOCTOR OF PHILOSOPHY  
December, 2017

MACHINE LEARNING:  
A POTENTIAL FORECASTING TOOL

Dissertation Approved:

Dr. B. Wade Brorsen

---

Dissertation Adviser

Dr. Kim Anderson

---

Dr. Eric A. DeVuyst

---

Dr. Tim Krehbiel

---

Name: JASDEEP SINGH BANGA

Date of Degree: DECEMBER, 2017

Title of Study: MACHINE LEARNING: A POTENTIAL FORECASTING TOOL

Major Field: AGRICULTURAL ECONOMICS

Abstract: Technical analysis involves predicting asset price movements from analysis of historical prices. Many studies have been conducted to determine the profitability of technical analysis. A composite prediction is considered here by using the buy and sell signals from technical indicators as inputs. Both machine learning methods like neural networks and statistical methods like logistic regression are used to get composite forecasts. Signals from trend-following and mean-reversal technical indicators are used in addition to variance of prices as inputs. Variance is added to help technical indicators switch between trend-following and mean-reversal systems. Five commodities from agricultural, livestock and foreign exchange futures markets are selected to test the hypothesis of profitability of technical indicators. Special care is taken to avoid data snooping error.

None of the individual indicators or machine learning models generate significant profit in single day forecasts. In twenty-day forecasts, only random forest and pipeline models are profitable. Neural networks and statistical models both failed to deliver here. The out of sample failure of the neural networks is partly due to the relatively large number of parameters. Managed futures, however also did poorly in the out of sample period so the results could also be due to picking a time period where technical analysis did poorly. Individual indicators did occasionally show significant profits. Random forests and decision tree find variance as the most important input. Future research should consider alternative time periods, commodities, systems, and machine learning algorithms. If a scale neutral variable for variance could be developed, it should be used so that the models could be trained on data from multiple commodities to provide more training data.

## TABLE OF CONTENTS

Chapter	Page
I. I. MACHINE LEARNING: A POTENTIAL FORECASTING TOOL	
Introduction.....	1
Technical Indicators.....	5
Prediction Models.....	9
Data and Methods.....	16
Results.....	20
Conclusions.....	31
REFERENCES.....	34
APPENDICES.....	39

## LIST OF TABLES

Table	Page
1. Selected Technical Indicators and Their Formulas.....	10
2. Summary Statistics for the Indicators .....	17
3. Summary of Model Performance in Copper .....	25
4. Summary of Model Performance for Corn .....	26
5. Summary of Model Performance for Feeder Cattle.....	27
6. Summary of Model Performance for Japanese Yen .....	29
7. Summary of Model Performance for Eurodollar .....	32
8. Summary of Significance of Profitability of Technical Indicators .....	33
9. Summary of Variable Importance Using Random Forest.....	34
10. Summary of Variable Importance Using Decision Tree.....	34

## **Introduction**

Technical analysis involves predicting asset price movements from analysis of historical price movements. Beja and Goldman (1980) argue that the trends exploited by technical analysis are due to markets frictions that cause markets to be slow to adjust in the absence of technical trading. The trend-following systems ride along on the actions of informed traders and work best when the market is unstable. Reversal systems like oscillators should work well when the market is stable. One concern is that the actions of trend-following technical traders can cause phenomena unrelated to economic fundamentals.

Brorsen and Irwin (1988) report that among a survey of 32 large commodities fund managers, only two were not using objective technical analysis. Oberlechner (2001) surveyed foreign exchange traders and find that a majority of the foreign exchange traders use some sort of technical analysis. Allen and Taylor (1992) found that 90 percent of traders in London use technical analysis as a primary or secondary source of information. Park and Irwin (2007) found trading strategies based on technical analysis were profitable in the futures markets until at least the early 1990s. As more money was devoted to trading based on technical analysis, its profitability dropped. A large number of trend-following technical traders may create market bubbles. Improved technical trading systems that could optimally switch back and forth between trend-following and reversal systems could increase trader's profits as well as potentially reduce instability created by trend followers.

The efficient markets hypothesis says that the current price reflects all available information about the commodity (Malkiel, 1989). In the absence of technical traders, markets have proven to be slow to adjust due to market frictions such as risk averse traders and behavioral anomalies such as loss aversion. Technical analysts recognize the trends arising from slow adjustments and exploit them. Sometimes, even if the trend is random but many investors follow it then the subsequent prediction becomes self-fulfilling, and sometimes creates a bubble. Boyd and Brorsen (1991) find a strong

relationship between market volatility and technical trading profits. This relationship could be useful to traders in determining whether to use a trend-following or a reversal system.

Various technical trading rules have been used in past research. Lukac, Brorsen, and Irwin (1988) use 14 trading systems approximating the full “universe” of trading systems. They find that technical trading systems produced statistically significant net returns, as compared to the buy-and-hold benchmark strategy over 1978-1985. Park and Irwin (2005) use 9,385 trading rules from 15 trading systems to study the profitability of technical analysis and find that technical trading strategies have not been profitable in the U.S. markets after correction for the costs and data snooping biases over 1985-2004. Various other studies like Ulrich (2009), Szakmary et al. (2010), Roberts (2005), Sullivan et al. (2003), Olson (2004), and Neely (2003) find evidence both in favor and against profitability of technical analysis. Roberts (2005) finds that technical rules were capable of generating significant out-of-sample profits in only 2 of 24 futures markets studied. Park and Irwin (2007) find that out of 95 modern studies, 56 find technical trading strategies being profitable, 20 studies obtaining non-profitable results and 19 studies having mixed results. They have expressed concerns about data snooping or publication bias in these studies.

Pruitt et al. (1992) use a combination system of cumulative volume, relative strength, and moving-averages to document profitability of a technical strategy over a buy-and-hold strategy in stock markets over 1986-1990. Irwin et al. (1997) compare ARIMA models to performance of technical trading system in soybean futures markets and find channel systems generate statistically significant mean returns in their out of sample period. Allen and Karjalainen (1999) use a genetic algorithm to learn technical trading rules and find that trading rules do not earn consistent excess returns over a buy-and-hold strategy after considering transaction costs in the out-of-sample test periods of S&P 500 index. Hamm and Brorsen (2000) develop a neural network trading model for agricultural commodities using lagged prices as inputs to determine the profitability of trading using signals from neural networks and find that neural networks did not produce significant profits. Ou and Wang (2009) use a logit model,

neural networks, classification tree based models among ten data mining techniques for prediction of stock markets index movement and find that they have accuracy in forecasting stock price movements.

Most studies of technical trading strategies exhibit one or many flaws like no statistical tests of return, no out-of-sample verification, data snooping problems are not given proper attention, and significance of economic profit after transaction costs are not considered. Park and Irwin (2007, p. 817) put forward three conditions for technical trading strategies that have to be satisfied for meaningful inference: “(1) markets and trading systems should be comprehensively represented in original study such that they can be considered broadly representative of the actually use technical systems, (2) testing procedure must be carefully documented, so they can be ‘written in stone’ at the point in time the study is published, and (3) the publication date of the original work should be sufficiently far in the past that a follow-up study can have a reasonable sample size.” This study takes into consideration all three conditions of Park and Irwin (2007).

One major difference between the present study and past studies is that, this study uses long-short trading signals as inputs instead of technical indicators themselves, and also an additional input representing the variance of prices is used. The potential of using long-short trading signals is that the model trained on the signals from one commodity can be extended to other commodities. The idea behind inclusion of variance of change in prices is that it should facilitate the switch between trend-following and mean-reversal trading systems depending on market conditions. In addition to determining the profitability of trading rules, random forests and decision trees can rank various trading rules according to their importance in trend recognition. The other main contribution of this study is comparison of the performance of random forests, decision tree, ensemble methods (Gaussian naive Bayesian, random forests, support vector machine, linear regression, and decision tree classifier) and single classifier models (Neural networks (NN), and logistic regression (LR)) in predicting the commodity futures market’s direction using technical indicators. In addition, random restarts are used



to avoid the local minima problem of neural networks. This study also evaluates the individual technical indicators.

## **Technical Indicators**

Technical indicators provide buy-sell trading recommendations. Even though charting is also a major type of technical analysis, this study only uses mathematically-derived trading rules. Mathematically-derived technical indicators can be divided into two main groups: trend-following (lagging indicators) and trend-reversal (leading indicators). Trend-following indicators are designed to follow price movements and work best when markets have large price movements. Some popular trend-following indicators include dual moving-average crossover, moving average convergence divergence (MACD), and price channels.

Trend-reversal or leading indicators measure the momentum in the markets. They are designed to lead the price movements and identify reversal of the trend. They represent price momentum over a fixed past period and all prior price action before that period is ignored. This study uses both trend-following and mean-reversal technical indicators such as moving average indicators, relative strength index (RSI), stochastic oscillator, commodity channel index (CCI), price channels, and variance.

This study uses the dual moving average crossover system to generate buy and sell signals. Moving-averages are trend-following techniques. Purcell and Koontz (1999, p. 175) say that “the idea is that in an upward- or downward-trending market, the shorter moving average tends to move faster and ‘leads’ the longer average. When the market turns, the shorter average turns more quickly and crosses the longer and slower-moving average. It is this crossover action that generates the buy and sell signals” (table 1). This study uses three types of dual moving averages namely (5, 10), (5, 20) and (10, 50). In the dual moving average (5, 10) and (5, 20) the 5 day moving average is considered the short moving average (SMA) while 10 and 20 day moving average is considered the long moving average

(LMA). Dual moving average (10, 50) uses 10-day moving average as SMA and 50-day moving average as LMA.

Relative strength index (RSI) is a popular momentum indicator and an oscillator. Like many other momentum oscillators, RSI works best when prices move sideways within a range. RSI is effective in both upward-and downward-trending markets. Purcell and Koontz (1999, p. 191). Schwager (1984) consider RSI as important in bringing discipline to a hedging program. Usually a 14-day look-back period is used for RSI calculation. RSI fluctuates between 0 and 100. RSI at zero means prices moved lower for all of the 14-day period and average gain equals zero. RSI at 100 means prices moved up all 14-days. If the RSI drops below 30 then it represents an oversold market and gives a buy signal. If RSI moves to 70 or higher, it is signaling a correction towards the downside will occur and it is a sell signal. This study uses both 14-day RSI and 9-day RSI (table 1).

Another oscillator used is the stochastic oscillator. It is a momentum indicator and compares the closing price to the range of prices over a certain period of time. Thus it is used to forecast reversal in the commodity markets when it has reached oversold or overbought levels. It can range between zero and 100. Usually a 14-day stochastic oscillator is used. The oversold threshold for a 14-day stochastic oscillator is considered to be at 20, and overbought threshold is represented by 80. Stochastic oscillator values below 20 indicate that the security is trading near its bottom level and thus generates a buy signal. Value above 80 indicates that the security is trading near its top level and thus provides a sell signal (table 1).

Another oscillator used is the commodity channel index (CCI). CCI is first developed to identify cyclical turns in commodities markets but is now used for equities and currencies too. CCI when used along with other oscillators can be helpful in estimating direction of price movement. CCI usually fluctuates between -100 to +100 but values can go beyond this range. If readings on CCI move above +100, it generates a buy signal and if it moves below -100, it is a sell signal. This study uses the

20-day period to calculate the CCI (CCI 20-day) and uses the typical price (obtained by the combined average of high, low and closing price of the commodity on a given day) as well as mean deviation of typical price. It is calculated by dividing the difference between “typical price (TP)” and 20-day SMA of TP with the mean deviation of the TP. TP is calculated by taking the average of high, low and closing price of the day. This study uses 20-day period to calculate the CCI. A constant (0.015) is added to ensure that approximately 70 to 80 percent of the CCI values would fall between -100 and +100. It measures the current price level relative to the average price level over a look-back period and CCI readings are higher when prices are above their average and they represent a strong trend. While CCI readings below -100 signal weakness in prices and thus give a sell signal (table 1).

A price channel is a trend-following system that consists of two lines representing support and resistance (table 1). For a 20-day channel, the support line is the 20-day low and the resistance line is the 20-day high. Price channels are used to represent trend direction for any security. They are used to identify the start of an uptrend or downtrend. In the case of a 20-day price channel, a buy signal is generated if the last day’s closing price is higher than the maximum of the previous 20 days (excluding last day) and vice versa. While in the case of 50-day price channel, a buy signal is generated if the last day’s closing price is higher than the maximum of the previous 50 days. Price channel does not include the most recent period. For example a 50-day price channel for August 11 would be based on the 50-day high and 50-day low ending the day before, August 10. This is done as a channel is not possible if the most recent period was used. Table 2 provides the summary statistics for the indicators.

Boyd and Brorsen (1992) use simulated technical trading profits to study correlation of price statistics and technical returns. They find trend-following systems are more profitable when price volatility is high. Another important variable for the present study is standard deviation (CV) of changes in close price. Yao et al. (2000) has indicated the importance of having a measure of volatility as an input for the formulation of the neural network for forecasting. Volatility is an indication of an

impending (or in process) major move. The idea behind it is to include a variable that can help switching between trend-following and mean-reversal systems.

**Table 1. Selected Technical Indicators and Their Formulas**

Name of Indicators	Buy Signal	Sell Signal	Formulas
Dual moving average crossover*	SMA>LMA	SMA<LMA	$x = \frac{C_t + C_{t-1} + C_{t-2} + \dots + C_{t-10}}{10}$
Relative strength index (RSI)**	RSI>70	RSI<30	$RSI = 100 - 100 / (1 + (\frac{\sum_{i=1}^{n-1} U_{t-i}}{n} / \frac{\sum_{i=1}^{n-1} D_{t-i}}{n}))$
Commodity channel index (CCI)*	CCI<-100	CCI>100	$CCI = \frac{M_t - MA_t}{0.015 * DM_t}$
Stochastic oscillator %k**			$\%k = \frac{C_t - L_{14}}{H_{14} - L_{14}} * 100$
Stochastic oscillator %D**	%D > 80	%D < 20	$\%D = \frac{\sum_{i=0}^{n-1} \%k_{t-1}}{n}$
20-day price channel	$C_{n-1} > MX$	$C_{n-1} < MX$	$MX = Max(C_{n-2} : C_{n-21})$
CV			$CV = (SD_t - \mu_s) / SD_s$

Note: \* denotes trend-following system, \*\* denotes mean-reversal system, SMA is small moving average (5 days), LMA is larger moving average (10 days),  $C_t$  is the closing price,  $L_t$  is the low price,  $H_t$  the high of the day,  $U_t$  is the upward price change,  $D_t$  is the downward price change,  $M_t = (H_t + L_t + C_t) / 3$ ,  $MA_t = (\sum_{i=1}^{20} M_{t-i+1}) / n$ ,  $D_t = (\sum_{i=1}^n M_{t-i+1} - MA_t)$ ,  $SD_t = \sqrt{1/14 (\sum_{t=1}^{14} (C_t - \mu)^2)}$ ,  $\mu$  is the mean of 14-days,  $SD_s$  is the standard deviation of training data set,  $\mu_s$  is the mean of training data set (Purcell and Koontz, 1999).

**Table 2. Summary Statistics for the Indicators**

Commodity	%k	%D (Three Day Average of %k)	CCI 20- day	Dual Moving Average (10,5)	Dual Moving Average (20,5)	Dual Moving Average (50,10)	RSI (14- day)	RSI (9-day)
<b>Copper</b>								
Max	100	100	362.25	1484.09	1477.81	1467.98	99.61	100
Min	0	0	-666.67	996.79	996.91	997.41	0.16	0
Mean	52.26	52.27	8.58	1152.62	1152.51	1152.19	51.29	51.37
SD	37.28	34.5	111.93	117.53	117.5	117.38	17.21	20.78
<b>Japanese Yen</b>								
Max	100	100	494.14	1000.55	1000.54	1000.52	100	100
Min	0	0	-666.67	999.6	999.61	999.61	0	0
Mean	46.6	46.6	-5.95	1000.01	1000.01	1000.01	48.87	48.58
SD	37.35	34.66	112.65	0.17	0.17	0.17	17.94	21.52
<b>Feeder cattle</b>								
Max	100	100	442.68	1127.64	1126.62	1124.79	100	100
Min	0	0	-666.67	983.16	983.55	984.26	0	0
Mean	53.31	53.31	6.68	1041.28	1041.26	1041.19	51.42	51.32
SD	37.44	34.81	110.63	28.76	28.75	28.72	17.35	21.72
<b>EuroDollar</b>								
Max	100	100	566.76	1030.95	1030.91	1030.85	100	100
Min	0	0	-666.67	999.97	1000.31	1000.8	0	0
Mean	56.53	56.53	18.16	1021.6	1021.58	1021.53	53.91	53.75
SD	38.9	36.32	114.15	6.71	6.72	6.77	21.37	25.8
<b>Corn</b>								
Max	100	100	350.64	1287.95	1285.49	1275.31	95.68	100
Min	0	0	-666.67	425.95	426.94	438.95	0	0
Mean	47.58	47.57	-7.09	871.79	871.99	872.6	48.7	48.58
SD	37.35	34.57	111.5	188.91	188.64	187.8	17.11	20.84

## Prediction Models

Several classification techniques have been used to predict the direction of financial markets e.g. logistic regression (Ohlson, 1980; Pantalone and Platt, 1987; Dimitras et al., 1996, Brownstone, 1996), multiple discriminant approaches (Altman et al., 1977, Ou and Wang, 2009), support vector machines (SVM) (Huang et al., 2006; Kim, 2003; Lee, 2009), k-nearest neighbors (Subha and Nambi, 2012),

decision trees (Wu, Lin, and Lin, 2006), neural networks (Kim and Chun, 1998), and ensemble models (Chun and Park, 2005; Lunga and Marwala, 2006 ; Patel et al., 2015).

Lukac et al. (1988) use twelve technical systems for trading commodities and find four trading systems produced significant net returns and significant risk-adjusted returns. Various statistical and machine learning models like logistic regression, decision trees, random forests, artificial neural networks, etc. are used for predicting accuracy of trading indicators. These models use their capabilities to recognize pattern and trend of prices and use this knowledge to predict the direction of trade using technical indicators. Direction of trade in buy-and-hold strategy essentially becomes a dichotomous classification problem where class labels can take values of 1 or -1, with 1 representing a buy signal and -1 representing a sell signal.

Logistic regression is one of most commonly used modeling techniques of data classification and is used to estimate the probability of arbitrary response based on one or more predictor variables. Logistic regression uses a binary output value instead of a numeric value and uses the logistic distribution function as the link function

$$prob(y = 1) = \frac{e^{(b_0 + b_1 x)}}{(1 + e^{(b_0 + b_1 x)})}$$
$$y \in [-1, 1]$$

where,  $y$  is a measure of the actual direction of prices (1 if prices went up and -1 if prices went down),  $x$  is a vector of independent variables,  $b_0$  is the bias or intercept term and  $b_1$  is the coefficient for the  $x$ ,  $D$  is the trading signal where  $D = 1$  (buy signal) if prob of success (price increase)  $> 0.5$ ,  $D = -1$  (sell signal) if prob of success (price increase)  $< 0.5$ .

Decision trees are an important machine learning model. A decision tree algorithm splits the data set according to a criterion that maximizes the separation of the data, resulting in a tree-like

structure (Breiman et al. 1984). Gini impurity is one of the most commonly used criteria to split each step in building the tree and is used to minimize misclassification. Gini impurity is computed as:

$$Gini(E) = 1 - \sum_{j=1}^c p_j^2$$

where, Gini impurity for a set of items with  $c$  classes, and  $j \in \{1, 2, 3, \dots, c\}$  and  $p_j$  is fraction of items labeled with class  $j$  in the set. The major advantage of using decision trees is that they are easy to express as rules while the major disadvantage is that continuous variables are implicitly discretized by the splitting process, losing information along the way (Dreiseitl and Ohno-Machado, 2002).

Random forests are another highly used machine learning technique for classification due to their ability to model complex interactions among predictor variables. They have very high classification accuracy and are considered robust with respect to noise. They can also be used for determining variable importance. Random forest grows many classification trees, and each tree gives a classification. The random forests prediction is the classification that receives the most votes across all trees.

In random forest, each tree is grown as 1)  $N$  number of bootstrap samples of size  $N$  are drawn at random with replacement from  $N$  observations (for this study the number of observations in the training data set of each commodity is used as  $N$ ). This bootstrapping procedure leads to better model performance because the combination of multiple trees, decreases the variance of the predictions, without increasing the bias. The process called “feature bagging” is used for candidate split in the learning process. Under this process if there are  $M$  input variables, a number  $m < M$  is specified at each node,  $m$  variables (held constant during the growing forest) are selected at random out of the  $M$  and the best split on these  $m$  variables is used to split the node (for this study  $m = 4$  and  $M = 11$ ). This feature helps to avoid the correlation due to the presence of very strong predictor variables and helps to avoid overfitting of the training set. Each tree is grown to the largest possible extent without pruning.

All the above models are used independently, but forecasting research has long found that composite forecasts outperform individual forecasts (Brandt and Bessler 1981). Lately new ensemble learning algorithms provide tools to combine machine learning models and use them together as a single model for classification purposes. An ensemble is a set of classifiers that learn a target function, and their individual predictions are combined to classify new examples. Ensemble learning can improve the performance of one or a number of models, and can be extremely useful when dealing with large and complex data sets (Dietterich 2000). The idea of ensemble methodology is to weigh several individual classifiers, and build a predictive model by integrating multiple models. In the simple majority voting ensemble model that is used here, every model makes a voting (prediction) for each instance of testing and the final prediction receives the maximum votes (Iam and Suen, 1997). In simple majority ensemble model, an equal weight of  $1/k$  to each classifier where  $k$  is the number of classifiers in an ensemble. The main advantage of the ensemble model is that the different classifiers are unlikely to make same mistake. In fact, as long as every error is made by a minority of the classifiers, you will achieve optimal classification. In particular, ensemble models tend to reduce the variance of the classifiers, and thus can be very useful for reducing the overfitting of the data. Various studies (Maslov and Gertner, 2006; Rodriguez et al., 2006; and Zhang and Zhang, 2008) have shown that the ensemble can outperform individual predictors in many cases.

Neural networks are considered universal approximators due to their non-linear approximation capabilities. Due to their flexible nature, they have potential to combine signals from various technical indicators and recognize patterns. This flexible nature also results in overfitting the data and thus can result in poor out of sample results. Hamm and Brorsen (2000) use closing prices as inputs for trading using neural networks and conclude that it does not work. Neural networks require a large amount of data to be estimated precisely. Daily futures data provides only a few thousand observations, so pre-filtering via technical indicators might be helpful. In the present study, neural networks are trained using the signals from technical indicators as described in appendix 18.



## **Data and Methods**

Commodity futures have been of renewed interest due to the need of diversification in periods of high volatility and potential equity-like benefits of commodity indexes (BIS, 2006). Commodity futures have potential to generate higher returns of a security on a risk adjustment basis (alpha generation) through long-short dynamic trading as well as their role of risk diversifiers (Chong, and Miffre, 2010). These among several other features like deep and liquid exchange-traded futures contracts make futures markets more attractive for active trading strategies than stock markets. It is interesting to consider various ways to improve profitability of quantitative trading rules for commodity trading. Many studies like Stevenson and Bear (1970), Lukac et al. (1988), Kidd and Brorsen (2004), and Sweeney (1986) find technical trading to be useful for commodity and foreign exchange markets. A total of five futures prices are selected based on these previous studies, continuity of contract, agricultural importance and volume of trade. Data consisted of one grain (Corn, C), one currency (JapaneseYen, JY), one interest rate (EuroDollar, ED), one metal (Copper, HG), and one livestock (feeder cattle, FC) futures markets. The data used for trading is Corn(C) March 1969 futures contract to December 2016, JapaneseYen (JY) March 1977 to December 2016, EuroDollar (ED) March 1982 to December 2016, copper (HG) October 1959 to December 2016, and feeder cattle data from March 1974 to December 2016. The time periods were determined by data availability. The data is divided into training, validation and test data. Training data comprise 70% of the whole data set while validation data comprise 20% and test data set comprises 10%. Depending on the commodity, training data usually represented start of the contract to 2003, while validation data set represents the time period of 2004-2012, and the test data set is 2012-2017. For the continuity of the trading signal, rollovers are used. The 20<sup>th</sup> day of the penultimate contract month is used as the rollover date. Continuous contracts are created by adding the change to the contract price of the old contract month from the previous day. Continuous contracts are commonly used in simulating technical trading as the technical signals in Table 1 depend upon changes in prices rather than price

levels. If the 20th is not a trading day then the last trading day before it is considered for calculations. This is done to avoid distortions caused by high volatility during the final contract month and would keep liquidity costs low by trading in a high volume contract. Closing prices are used for calculating changes in prices..

Special care is taken to meet all three requirements laid down by Park and Irwin (2007) for replication of technical trading strategies that have to be satisfied for meaningful inference. A pre-analysis plan was prepared before starting work on the data and is given in appendix 16. Secondly data snooping error in neural networks is avoided by using three sets of data (training, validation and testing) while for other models only training and testing data sets are used.

The dependent variable is also bivariate, and consists of buy and sell signals. It is also scale neutral and represents the direction of the futures markets. All the signals for the technical indicators are calculated based on the pseudo price series. This is done to maintain continuity of contract roll overs and scale neutral inputs. This price series is formed using changes in the closing prices of the futures contract. Initial price level is assumed to be 1000 and subsequent prices are calculated by adding the change in closing price to initial price level

$$P_1 = 1000$$

$$P_2 = P_1 + C_2$$

$$P_N = P_{N-1} + C_N$$

where,  $P_1$  is the first pseudo price, and  $C_N$  is the change in closing price. A raw variable is calculated by taking the difference of natural log of pseudo price series

$$R_i = \log P_i - \log P_{i+20}$$

where,  $R_i$  is the raw variable,  $P_i$  is the pseudo price series. Dependent variable is the signals from the raw variable which takes the form of -1 or +1

$$D_i = \begin{cases} -1 \\ +1 \end{cases}$$

where,  $D_i$  is the dependent variable, and  $R_i$  is the raw variable,  $D_i = -1$  if  $R_i > 0$  and  $D_i = 1$  otherwise.

Comparison of the profit and loss based on prediction using logistic regression, random forests, voting ensemble model, pipeline model and neural networks is done. Voting ensemble model is built using logistic regression, random forests, Gaussian Naïve Bayes, decision tree, and support vector classification models. Random forest model is initially built with '20 trees' and a batch of '10 trees' is added until the optimum is reached with lowest 'Gini' criteria.

A random number generator is used to pick 20 seeds initially and then the final seed for neural networks is selected based on the profit of predictions. A validation data set is used for computing the best random number for use in neural networks. This is done to ensure the purity of out of sample (test data set) and also as neural networks suffer from the problem of local minima, best seed is selected based on the highest revenue generated using the validation data set. The selected model is then used for final neural network using the testing data set. This is done so as to avoid local optima and reach global optima in neural networks. Three types of neural network models are built:

- a) Single hidden layer neural network with five and 17 neurons using "limited-memory BFGS (l-bfgs)" algorithm and "Softmax" activation function, and Single layer with 17 neurons and "tanh" activation function.
- b) Three hidden layer neural network with five and 17 neurons using "l-bfgs" algorithm and "Softmax" activation function.

- c) Three hidden layer neural network with five and 17 neurons using “l-bfgs” algorithm and “tanh” activation function.

L-bfgs is an optimization algorithm in the family of quasi-newton methods that approximates Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm using a limited amount of computer memory. The difference between l-bfgs and bfgs is that l-bfgs uses only a few vectors that represent the approximation implicitly rather than  $n \times n$  as in bfgs. While “Softmax” (normalized exponential function) is a generalization of the logistic function that transforms a  $K$  -dimensional vector  $z$  of arbitrary real values to a  $k$  -dimensional vector  $\sigma(z)$  of real values in the range  $[0, 1]$  that adds up to 1. The function is given by

$$\sigma(z)_j = \frac{e^z_j}{\sum_{k=1}^K e^z_k} \quad \text{for } j = 1, \dots, K.$$

Also, tanh is the hyperbolic tangent function where output values range from  $(-1, 1)$ . Thus strongly negative inputs to the tanh will map to negative outputs in the neural network. Also, only zero-values inputs are mapped to near-zero outputs. These properties helps the neural network to train regularly.

More details about the neural networks (ANNs) have been discussed in appendix 13. Also ANNs having more than one hidden layer is considered deep learning by some authors (Erhan et al., 2010). No study to knowledge has compared the profitability of technical indicators in the commodity and foreign exchange futures options markets using voting ensemble model and compared them with other highly use statistical and<sup>1</sup> machine learning methods.

The inputs used are the bivariate buy-sell signals from the technical indicators while previous studies have directly used the technical indicators directly (Sullivan, Timmermann, and White, 1999;

Chang and Osler, 1999; Neely, 2002; Lukac, Brorsen, and Irwin, 1988; Slezak, 2003). Most studies scale the data into a range of [-1.0, 1.0] with the goal of independently normalizing each feature component to a specific range, but doing so would result in a commodity specific model. This study uses the technical signals (-1 or 1), this way the information in the technical indicator can be as a classification problem and the trained model can potentially be extended to other commodities (if the variance term was excluded). The idea behind this is that the model should be able to recognize the complex trend and pattern of various indicators, theoretically this pattern should be same for all the commodities. Thus theoretically, a model formed on one commodity could be used to make predictions in any other commodity. In addition to the signals, a commodity specific input representing the variance of the prices has also been used.

The training data is used to search for optimal parameters and these parameters are employed to evaluate the out of sample performance of the model. Gross profit is calculated on the out of sample data set. Number of trades is calculated based on the change in buy or sell signal from the previous signal. Commission cost is assumed to be \$5 per single turn (CME group (2016, 4 February)). Net profit is calculated by considering cost of trading and revenue generated from the trades. A short trading position is taken initially and the account is settled at the end of the trading day

$$S_i = O_i - O_{i+1}$$

$$T_i = O_i - O_{i+20}$$

where,  $S_i$  is the single day revenue,  $T_i$  is the twenty day revenue, and  $O_i$  is the opening price of the 21st day. Opening price of the next trading day is used to calculate profit to avoid liquidity bias. Transaction cost is calculated by multiplying no. of trades with cost of trading per unit and size of the contract. Total profit is then calculated by subtracting the total cost from total revenue

*Total Cost = no. of trades \* cost per unit \* size of contract*

$$Total\ profit\ for\ single\ day = \sum_{i=1}^n S_i - cost$$

$$Total\ profit\ for\ twenty\ day = \sum_{i=1}^n T_i - cost$$

Two types of forecasts, a) single day forecast, and b) 20 day forecast, are used to calculate costs per trade and profit per contract. With a 20-day forecast, the position is held for 20 days without regard to later price movements. Evaluation is done based on these two types.

This study is hypothesizing that signals from a pool of trend-following and mean-reversal technical indicators in addition to a variable helping to switch between them when paired with modern day statistical and machine learning tools have the potential to generate profit in commodity futures markets.

Both individual and joint hypothesis tests are performed. The first hypothesis is to determine if the technical indicators and quantitative methods generate significant profit for individual commodities and the second hypothesis tests determine if any individual indicator or quantitative model generates significant profit in all the five commodities.

Hypothesis test will be performed by combining the t-values of all the five commodities for every individual indicator as well as each model

$$\begin{aligned} \left[ \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5} \right] x &= Ax, \\ Ax &\overset{d}{\rightarrow} \left( 0, \frac{1}{5} \right), \\ Net\ t - value &= \frac{Avg.t-value}{\frac{1}{\sqrt{5}}}, \\ Avg.\ t - value &= \frac{1}{5} \sum_{i=1}^5 t_i \end{aligned}$$

where,  $t_i$  is t-value for each commodity. Plugging numbers into the above formulas, gives the critical t-value as 0.87.

## Results

Trading is simulated on five commodities. Table 3 describes the profit and number of trades for copper. None of the individual indicators or models generated significant profit for single day profit. For 20-day trading forecast, moving average (10, 50) and stochastic indicator generate significant profit. Number of trades increase by more than two times when neural networks are used, this increases the transaction costs and thus reduces the profitability of the technical trading systems using neural networks. Number of trades is highest for the neural network with three hidden layers and each layer with 17 neurons and ‘tanh’ activation function and 20 day prediction. Future research may want to consider reinforcement learning (Deng et al. 2017) as a way of imposing a penalty for the number of trades.

Table 4 gives the summary of model performance for corn. Only RSI (9 days), CCI 20 day, and stochastic indicator have significant profit for single day forecast. Number of trades is highest for stochastic indicator for both single day and twenty day forecast. For twenty day forecast but only moving average (10, 50) generates significant profit. Stochastic indicator has the highest number of trades for 20-day forecasts.

Table 5 presents the summary of model performance for feeder cattle. In feeder cattle, neither any individual indicator nor any statistical or machine learning model is significantly profitable for both single as well as twenty day forecasts. For single as well as 20 day forecasts, neural networks with three hidden layers and 17 neurons with ‘l-bfgs’ algorithm and ‘tanh’ activation function has the highest number of trades, while the pipeline model has the lowest number of trades.

Table 6 presents a summary of forecasting performance on Japanese Yen. Most of the statistical and machine learning models are profitable in Japanese Yen, but only RSI (9 days) and CCI 20 days individual indicators are profitable and only CCI 20 day generates significant profit for single day forecasts. None of the other commodities have this high of a success rate for statistical and machine learning models. Another particular point is that these models are profitable not only in single day forecast but also in twenty day forecasts. In the case of 20 day forecasts, six of the seven neural networks generate significant profit along with random forest and moving average (5, 20). Neural networks with three hidden layers and 17 neurons with ‘l-bfgs’ algorithm and ‘tanh’ activation function has the highest number of trades, while moving average indicator (5, 20) has the lowest number of the lowest number of trades with single day forecast. For twenty day forecast neural networks with three hidden layers and 17 neurons with ‘l-bfgs’ algorithm and ‘tanh’ activation function is having highest number of trades, while moving average indicator (10, 50) has the lowest number of trades.

Table 7 presents the summary of model performance for EuroDollar. As seen in Japanese Yen, neural networks performed better in EuroDollar. Out of the five neural networks, three are profitable but none of the statistical and machine learning methods are significantly profitable. While most of the individual indicators do not generate profit in Japanese Yen, their performance is better in EuroDollar. In fact, individual indicators performed best in EuroDollar as compared to the other four futures markets. For single day forecast RSI 9 days and stochastic indicator generate significant profit. For twenty day forecast moving average (5, 20), 20 day channel, and 50 day channel generate significant profit. Neural networks with three hidden layer and 17 neurons with ‘l-bfgs’ algorithm and ‘tanh’ activation function has the highest number of trades for both single day and 20-day forecast.

Table 8 summarizes the results. No single technical indicator is significantly profitable in both single day and 20 day projections. RSI (9 days), CCI (20 days), and stochastic oscillator do generate significant profit in single day projections. Trend-following indicators like moving average (5, 20) and moving average (10, 50) generated significant profit in 20 day forecast. RSI (14-days), 20 day channel, 50 day channel, and machine learning methods like neural networks do not produce any significant



result. Statistical methods like logistic regression, voting ensemble models, and random forests do not generate any significant result in single day projections and logistic regression, pipeline model, and voting ensemble models are loss making in 20 day projections.

Table 3. Summary of Model Performance in Copper

Model	Single Day Profit (cents/lb.)	Mean profit (Single Day)	Standard Deviation	t-value for Single Day	No. of Trades (Single Day)	20 Day Profit (cents/lb.)	Mean Profit (20-day)	Standard Deviation	t-value for 20-day	No. of Trades (20 Day)	Cost of Trades for Single Day Forecast (\$)	Profit for Single Day (\$ /contract)	Net Profit for Single Day (\$ /contract)	Cost of Trades for 20-day Forecast (\$)	Profit for 20-days (\$/contract)	Net Profit for 20-days (\$ /contract)
Moving average (5,10)	-10.7	-0.01	0.98	-0.35	110	-92.58	-0.1	4.19	-0.71	110	550	-2673.75	-3223.75	550	-23145.63	-23695.63
Moving average (5,20)	-23.66	-0.02	0.98	-0.77	60	149.63	0.15	4.19	1.15	60	300	-5915	-6215	300	37407.5	37107.5
Moving average (10,50)	23.6	0.02	0.98	0.77	28	661.52	0.68	4.14	5.13	27	140	5899.75	5759.75	135	165380	165245
RSI (14-day)	5.86	0.01	0.98	0.19	25	-545.17	-0.56	4.16	-4.21	25	125	1465	1340	125	-136292	-136417
RSI (9-day)	20.54	0.02	0.98	0.67	47	-313.47	-0.32	4.18	-2.41	46	235	5134	4899	230	-78367.25	-78597.25
CCI 20-day	22.7	0.02	0.98	0.74	43	-141.32	-0.15	4.19	-1.08	43	215	5675.25	5460.25	215	-35328.75	-35543.75
20-day channel	-28.48	-0.03	0.98	-0.93	92	-580.72	-0.6	4.15	-4.49	92	460	-7121	-7581	460	-145179.75	-145639.75
50-day channel	-28.46	-0.03	0.98	-0.93	24	-608.59	-0.63	4.15	-4.71	24	120	-7115.5	-7235.5	120	-152146.75	-152266.75
Stochastic indicator	22.53	0.02	0.98	0.73	3	583.62	0.6	4.15	4.51	3	15	5633	5618	15	145904.88	145889.88
Logistic regression	-29.28	-0.03	0.98	-0.95	33	-643.72	-0.66	4.28	-4.83	34	165	-7319.25	-7484.25	170	-160929.63	-161099.63
Random forest	-27.75	-0.03	0.98	-0.9	37	-640.02	-0.66	4.28	-4.8	37	185	-6938.5	-7123.5	185	-160004.88	-160189.88
Pipeline model	-27.93	-0.03	0.98	-0.91	1	-676.3	-0.7	4.27	-5.08	1	5	-6983	-6988	5	-169075.38	-169080.38
Voting ensemble model	-28.41	-0.03	0.98	-0.93	242	-473.18	-0.49	4.3	-3.53	238	1210	-7101.5	-8311.5	1190	-118295.75	-119485.75
Neural network (single layer with 5 neurons)	1.97	0	0.98	0.06	129	-74.07	-0.08	4.33	-0.55	116	645	493.25	-151.75	580	-18518.38	-19098.38
Neural network (three layers with 5 neurons)	-27.62	-0.03	0.98	-0.9	125	-656.36	-0.68	4.27	-4.93	126	625	-6906	-7531	630	-164089.13	-164719.13
Neural network (three layers, 5 neurons, tanh)	-11.97	-0.01	0.98	-0.39	83	-521.02	-0.54	4.29	-3.9	144	415	-2991.75	-3406.75	720	-130256.13	-130976.13
Neural network (single layer with 17 neurons)	36.74	0.04	0.98	1.2	155	-550.75	-0.57	4.29	-4.12	159	775	9184.25	8409.25	795	-137687.63	-138482.63
Neural network (single layer, 17 neurons, tanh)	-21.35	-0.02	0.98	-0.7	204	-53	-0.05	4.33	-0.39	157	1020	-5336.5	-6356.5	785	-13248.88	-14033.88
Neural network (three layers with 17 neurons)	4.49	0	0.98	0.15	243	-244.1	-0.25	4.32	-1.81	250	1215	1122.5	-92.5	1250	-61024.88	-62274.88
Neural network (three layers, 17 neurons, tanh)	-90.35	-0.09	0.98	-2.94	255	-368.04	-0.38	4.31	-2.74	255	1275	-22587.5	-23862.5	1275	-92009.88	-93284.88

Table 4. Summary of Model Performance for Corn

Model	Single Day Profit (cents/lb.)	Mean profit (Single Day)	Standard Deviation	t-value for Single Day	No. of Trades (Single Day)	20 Day Profit (cents/lb.)	Mean Profit (20-day)	Standard Deviation	t-value for 20-day	No. of Trades (20-Day)	Cost of Trades for Single Day Forecast (\$)	Profit for Single Day Forecast (\$ /contract)	Net Profit for Single Day Forecast (\$ /contract)	Cost of Trades for 20-day Forecast (\$)	Profit for 20-day (\$/contract)	Net Profit for 20-day Forecast (\$ /contract)
Moving average (5,10)	-693	-0.58	8.02	-2.49	145	-1416.75	-1.2	39.42	-1.05	141	725	-34650	-35375	705	-70837.5	-71542.5
Moving average (5,20)	-350.5	-0.29	8.04	-1.26	75	352.25	0.3	39.41	0.26	73	375	-17525	-17900	365	17612.5	17247.5
Moving average (10,50)	-140.5	-0.12	8.04	-0.5	35	3972.75	3.36	39.29	2.94	34	175	-7025	-7200	170	198637.5	198467.5
RSI (14-day)	418	0.35	8.03	1.5	22	45.75	0.04	39.44	0.03	22	110	20900	20790	110	2287.5	2177.5
RSI (9-day)	560.5	0.47	8.03	2.01	48	1348.75	1.14	39.42	1	48	240	28025	27785	240	67437.5	67197.5
CCI 20-day	598.5	0.5	8.03	2.15	48	-3174.25	-2.69	39.35	-2.35	47	240	29925	29685	235	-158712.5	-158947.5
20-day channel	-231.5	-0.19	8.04	-0.83	114	-10.75	-0.01	39.44	-0.01	110	570	-11575	-12145	550	-537.5	-1087.5
50-day channel	-265.5	-0.22	8.04	-0.95	66	-1699.25	-1.44	39.42	-1.25	62	330	-13275	-13605	310	-84962.5	-85272.5
Stochastic indicator	1269.5	1.06	7.97	4.6	478	1182.75	1	39.41	0.87	468	2390	63475	61085	2340	59137.5	56797.5
Logistic regression	54	0.05	8.04	0.19	88	-1448.25	-1.23	40.66	-1.04	84	440	2700	2260	420	-72412.5	-72832.5
Random forest	274.5	0.23	8.04	0.99	82	929.75	0.79	40.67	0.66	79	410	13725	13315	395	46487.5	46092.5
Pipeline model	224	0.19	8.03	0.8	114	204.25	0.17	40.68	0.15	111	570	11200	10630	555	10212.5	9657.5
Voting ensemble model	262	0.22	8.03	0.94	154	-1519.75	-1.29	40.66	-1.09	151	770	13100	12330	755	-75987.5	-76742.5
Neural network (single layer with 5 neurons)	192.5	0.16	8.04	0.69	161	-2090.25	-1.77	40.65	-1.5	135	805	9625	8820	675	-104512.5	-105187.5
Neural network (three layers with 5 neurons)	39	0.03	8.04	0.14	181	-973.75	-0.82	40.68	-0.7	161	905	1950	1045	805	-48687.5	-49492.5
Neural network (three layers, 5 neurons, tanh)	-61	-0.05	8.04	-0.22	180	-1228.25	-1.04	40.67	-0.88	177	900	-3050	-3950	885	-61412.5	-62297.5
Neural network (single layer with 17 neurons)	-88	-0.07	8.04	-0.32	261	-211.25	-0.18	40.68	-0.15	213	1305	-4400	-5705	1065	-10562.5	-11627.5
Neural network (single layer, 17 neurons, tanh)	279	0.23	8.04	1	229	1674.25	1.42	40.66	1.2	221	1145	13950	12805	1105	83712.5	82607.5
Neural network (three layers with 17 neurons)	-316	-0.26	8.04	-1.13	291	-2579.75	-2.18	40.63	-1.85	282	1455	-15800	-17255	1410	-128987.5	-130397.5
Neural network (three layers, 17 neurons, tanh)	-114	-0.1	8.04	-0.41	327	-2777.25	-2.35	40.62	-1.99	380	1635	-5700	-7335	1900	-138862.5	-140762.5

Table 5. Summary of Model Performance for Feeder Cattle

Model	Single Day Profit (cents/lb.)	Mean profit (Single Day n)	Standard Deviation	t-value for Single Day	No. of Trades (Single Day)	20 Day Profit (cents/lb.)	Mean Profit (20-day)	Standard Deviation	t-value for 20-day	No. of Trades (20-Day)	Cost of Trades for Single Day Forecast (\$)	Profit for Single Day Forecast (\$ /contract)	Net Profit for Single Day Forecast (\$ /contract)	Cost of Trades for 20-day Forecast (\$)	Profit for 20-day (\$/contract)	Net Profit for 20-day Forecast (\$ /contract)
Moving average (5,10)	-205.6	-0.02	1.83	-0.35	97	-76.01	-0.08	8.61	-0.29	96	485	-1028	-1513	480	-380.03	-860.03
Moving average (5,20)	-104.91	-0.01	1.84	-0.18	54	-200.48	-0.22	8.61	-0.77	53	270	-524.55	-794.55	265	-1002.38	-1267.38
Moving average (10,50)	6.41	0	1.85	0.01	28	-221.94	-0.24	8.61	-0.85	27	140	32.05	-107.95	135	-1109.68	-1244.68
RSI (14-day)	54.46	0.01	1.85	0.09	21	150.54	0.16	8.61	0.58	20	105	272.3	167.3	100	752.68	652.68
RSI (9-day)	125.74	0.01	1.84	0.21	47	189.42	0.21	8.61	0.73	47	235	628.7	393.7	235	947.08	712.08
CCI 20-day	149.45	0.02	1.84	0.25	29	68.76	0.07	8.61	0.26	28	145	747.25	602.25	140	343.78	203.78
20-day channel	-119.22	-0.01	1.84	-0.2	116	-729.16	-0.79	8.58	-2.8	116	580	-596.1	-1176.1	580	-3645.78	-4225.78
50-day channel	-72.88	-0.01	1.84	-0.12	82	-1064.09	-1.16	8.54	-4.11	82	410	-364.4	-774.4	410	-5320.45	-5730.45
Stochastic indicator	94.76	0.01	1.84	0.16	49	367.78	0.4	8.6	1.41	49	245	473.8	228.8	245	1838.88	1593.88
Logistic regression	-41.93	0	1.85	-0.07	15	-696.99	-0.76	8.82	-2.61	15	75	-209.65	-284.65	75	-3484.95	-3559.95
Random forest	-64.59	-0.01	1.84	-0.11	43	-74.58	-0.08	8.85	-0.28	43	215	-322.95	-537.95	215	-372.9	-587.9
Pipeline model	-33.14	0	1.85	-0.06	1	-645.68	-0.7	8.82	-2.41	1	5	-165.7	-170.7	5	-3228.4	-3233.4
Voting ensemble model	-38.1	0	1.85	-0.06	129	-215.89	-0.23	8.85	-0.8	129	645	-190.5	-835.5	645	-1079.45	-1724.45
Neural network (single layer with 5 neurons)	-15.24	0	1.85	-0.03	77	-110.46	-0.12	8.85	-0.41	77	385	-76.2	-461.2	385	-552.3	-937.3
Neural network (three layers with 5 neurons)	-78.53	-0.01	1.84	-0.13	129	169.85	0.18	8.85	0.63	125	645	-392.65	-1037.65	625	849.25	224.25
Neural network (three layers, 5 neurons, tanh)	-23.8	0	1.85	-0.04	117	-52.57	-0.06	8.85	-0.2	116	585	-119	-704	580	-262.85	-842.85
Neural network (single layer with 17 neurons)	-0.19	0	1.85	0	151	309.92	0.34	8.85	1.16	147	755	-0.95	-755.95	735	1549.6	814.6
Neural network (single layer, 17 neurons, tanh)	-5.56	0	1.85	-0.01	179	211.18	0.23	8.85	0.79	177	895	-27.8	-922.8	885	1055.9	170.9
Neural network (three layers with 17 neurons)	-16.72	0	1.85	-0.03	197	635.56	0.69	8.83	2.38	196	985	-83.6	-1068.6	980	3177.8	2197.8
Neural network (three layers, 17 neurons, tanh)	12.47	0	1.85	0.02	246	226.85	0.25	8.85	0.85	246	1230	62.35	-1167.65	1230	1134.25	-95.75

Table 6. Summary of Model Performance of Japanese Yen

Model	Single Day Profit (cents/ ¥)	Mean profit (Single Day)	Standard Deviation	t-value for Single Day Forecast	No. of Trades (Single Day)	Twenty Day Profit (cents/ ¥)	Mean Profit (20-day)	Standard Deviation	t-value for 20-day Forecast	No. of Trades	Cost of Trades (Single Day)	Profit Single Day (\$/ contract)	Net Profit for Single Day (\$/ contract)	Cost of Trades (20-day)	Profit 20-day (\$/ contract)	Net Profit 20-day (\$/ contract)
Moving average (5,10)	-0.49	-0.0005	0.01	-2.63	104	0.92	0.0010	0.03	1.22	103	520	-61712.50	-62232.50	515	114706.25	114191.25
Moving average (5,20)	-0.25	-0.0003	0.01	-1.34	2	1.78	0.0020	0.03	2.38	61	9	-31500.00	-31508.92	305	223081.25	222776.25
Moving average (10,50)	-0.12	-0.0001	0.01	-0.66	28	-1.49	-0.0017	0.03	-1.99	27	140	-15537.50	-15677.50	135	-186243.75	-186378.75
RSI (14-day)	-0.21	-0.0002	0.01	-1.10	27	-3.37	-0.0038	0.02	-4.63	27	135	-25912.50	-26047.50	135	-421531.25	-421666.25
RSI (9-day)	0.26	0.0003	0.01	1.36	47	-3.66	-0.0041	0.02	-4.94	46	235	32050.00	31815.00	230	-457993.75	-458223.75
CCI 20-day	0.33	0.0004	0.01	1.78	37	-0.85	-0.0009	0.03	-1.13	35	185	41100.00	40915.00	175	-105856.25	-106031.25
20-day channel	0	0.0000	0.01	0.00	86	-0.44	-0.0005	0.03	-0.59	84	430	-12.50	-442.50	420	-55418.75	-55838.75
50-day channel	-0.12	-0.0001	0.01	-0.64	32	-1.57	-0.0018	0.03	-2.10	32	160	-14975.00	-15135.00	160	-196643.75	-196803.75
Stochastic indicator	-0.11	-0.0001	0.01	-0.57	51	-2.12	-0.0024	0.03	-2.83	50	255	-13400.00	-13655.00	250	-264906.25	-265156.25
Logistic regression	0.06	0.0001	0.01	0.30	103	1.00	0.0011	0.03	1.31	101	515	6987.50	6472.50	505	125356.25	124851.25
Random forest	0.21	0.0002	0.01	1.13	69	1.84	0.0021	0.03	2.41	67	345	25625.00	25280.00	335	229687.50	229352.50
Pipeline model	0	0.0000	0.01	0.01	87	0.52	0.0006	0.03	0.68	85	435	125.00	-310.00	425	65393.75	64968.75
Voting ensemble model	-0.03	0.0000	0.01	-0.16	153	-0.62	-0.0007	0.03	-0.81	151	765	-3650.00	-4415.00	755	-77900.00	-78655.00
Neural network (single layer with 5 neurons)	0.03	0.0000	0.01	0.17	119	-0.53	-0.0006	0.03	-0.70	143	595	3962.50	3367.50	715	-66712.50	-67427.50
Neural network (three layers with 5 neurons)	0.3	0.0003	0.01	1.57	143	2.65	0.0030	0.03	3.49	156	715	36912.50	36197.50	780	331356.25	330576.25
Neural network (three layers, 5 neurons, tanh)	0.09	0.0001	0.01	0.47	171	1.93	0.0022	0.03	2.53	145	855	11136.25	10281.25	725	241356.25	240631.25
Neural network (single layer with 17 neurons)	0.15	0.0002	0.01	0.80	161	2.83	0.0032	0.03	3.73	167	805	18912.50	18107.50	835	354106.25	353271.25
Neural network (single layer, 17 neurons, tanh)	0.11	0.0001	0.01	0.61	149	1.95	0.0022	0.03	2.55	190	745	14248.75	13503.75	950	243175.00	242225.00
Neural network (three layers with 17 neurons)	0.03	0.0000	0.01	0.14	211	1.30	0.0015	0.03	1.70	210	1055	3286.25	2231.25	1050	162581.25	161531.25
Neural network (three layers, 17 neurons, tanh)	0.09	0.0001	0.01	0.48	265	1.70	0.0019	0.03	2.23	247	1325	11361.25	10036.25	1235	212406.25	211171.25

Most of the validation sample is from year 2008 to 2012. In this period neural networks worked very well and produced good profit (Appendix 1-10). Since the validation data set is also a kind of out of sample data set, it is interesting to find that when these neural networks are tested on the test sample, many neural networks were not profitable. Negative returns of commodity trading advisors for the period of 2011- 2016 is also observed by Barclay's CTA index (Appendix 12). Neural networks are supposed to switch between the trend-following and mean-reversal system as they have high potential for pattern recognition. But, they were not successful.

Table 9 presents the ranking of variables performed by random forests on the basis of the impact on predictor variable. Variance is ranked top among all the variables studied and has the most impact in all the commodities. RSI (14-days) and RSI (9 days), Moving average (10, 50), and CCI 20 days are the other main variables important for prediction. 20 day channel and 50 day channel are always ranked lowest impact on the predictor variable.

Table 10 presents the ranking of variables performed by decision tree. As ranked by random forest, decision tree also ranked variance as the most important variable in having an impact on prediction. Also RSI (14-days) and RSI (9 days), Moving average (10, 50), and CCI 20 days are the other main variables important for prediction but one difference is that the importance percentage increased for RSI and other variables but decreased for variance. Decision tree also ranked 20 day channel and 50 day channel as the variables having lowest impact on the output variable.

Table 7. Summary of Model Performance for EuroDollar

Model	Single Day Profit (cents/EUR)	Mean profit (Single Day)	Standard Deviation	t-value for Single Day	No. of Trades	20-Day Profit (Cents/EUR)	Mean Profit (20-day)	Standard Deviation	t-value for 20-day	No. of Trades	Cost of Trades (Single Day)	Profit Single Day (\$/contract)	Net Profit Single Day (\$/Contract)	Cost of Trades (20-day)	Profit 20-day (\$/contract)	Net Profit 20-day (\$/contract)
Moving average (5,10)	-0.46	-0.0006	0.01	-1.21	85	1.71	0.0022	0.05	1.22	81	425	-57187.50	-57612.50	405	214062.50	213657.50
Moving average (5,20)	-0.37	-0.0005	0.01	-0.99	46	3.35	0.0044	0.05	2.40	45	230	-46562.50	-46792.50	225	419062.50	418837.50
Moving average (10,50)	0.08	0.0001	0.01	0.20	22	1.33	0.0017	0.05	0.95	22	110	9687.50	9577.50	110	165937.50	165827.50
RSI (14-day)	-0.09	-0.0001	0.01	-0.24	33	-6.84	-0.0089	0.05	-4.95	33	165	-11562.50	-11727.50	165	-854687.50	-854852.50
RSI (9-day)	0.79	0.0010	0.01	2.09	41	-6.39	-0.0083	0.05	-4.62	41	205	98437.50	98232.50	205	-799062.50	-799267.50
CCI 20-day	0.35	0.0004	0.01	0.94	33	-3.85	-0.0050	0.05	-2.76	32	165	44062.50	43897.50	160	-480937.50	-481097.50
20-day channel	0.43	0.0005	0.01	1.13	108	4.98	0.0065	0.05	3.58	108	540	53437.50	52897.50	540	622187.50	621647.50
50-day channel	0.61	0.0008	0.01	1.62	90	4.68	0.0061	0.05	3.36	90	450	76562.50	76112.50	450	585312.50	584862.50
Stochastic indicator	0.89	0.0011	0.01	3.98	137	-1.05	-0.0014	0.04	-1.05	133	685	111562.50	110877.50	665	-131250.00	-131915.00
Logistic regression	-0.26	-0.0003	0.01	-0.70	1	-4.75	-0.0062	0.05	-3.34	1	5	-32812.50	-32817.50	5	-593125.00	-593130.00
Random forest	0.06	0.0001	0.01	0.17	19	-0.78	-0.0010	0.05	-0.55	15	95	7812.50	7717.50	75	-98112.50	-98187.50
Pipeline model	-0.26	-0.0003	0.01	-0.70	1	-4.75	-0.0062	0.05	-3.34	1	5	-32812.50	-32817.50	5	-593125.00	-593130.00
Voting ensemble model	-0.23	-0.0003	0.01	-0.61	125	-5.61	-0.0073	0.05	-3.96	119	625	-29050.00	-29675.00	595	-701237.50	-701832.50
Neural network (single layer with 5 neurons)	0.03	0.0000	0.01	0.09	118	-0.93	-0.0012	0.05	-0.65	132	590	4062.50	3472.50	660	-116862.50	-117522.50
Neural network (three layers with 5 neurons)	0.32	0.0004	0.01	0.84	142	-4.21	-0.0055	0.05	-2.96	191	710	39687.50	38977.50	955	-526125.00	-527080.00
Neural network (three layers, 5 neurons, tanh)	-0.61	-0.0008	0.01	-1.61	134	0.39	0.0005	0.05	0.27	128	670	-75925.00	-76595.00	640	48750.00	48110.00
Neural network (single layer with 17 neurons)	-0.08	-0.0001	0.01	-0.22	120	0.52	0.0007	0.05	0.36	117	600	-10300.00	-10900.00	585	65000.00	64415.00
Neural network (single layer, 17 neurons, tanh)	0.43	0.0005	0.01	1.15	107	-4.23	-0.0055	0.05	-2.98	139	535	54062.50	53527.50	695	-529362.50	-530057.50
Neural network (three layers with 17 neurons)	0.51	0.0007	0.01	1.36	148	-1.89	-0.0025	0.05	-1.32	132	740	64062.50	63322.50	660	-236237.50	-236897.50
Neural network (three layers, 17 neurons, tanh)	0.20	0.0003	0.13	0.05	166	-1.19	-0.0016	0.05	-0.84	164	830	25312.50	24482.50	820	-149362.50	-150182.50

**Table 8. Summary of Significance of Profitability of Technical Indicators**

Model	Net t-value for Single Day Forecast	Net t-value for 20-Day Forecast
Moving average (5,10)	-1.41	0.08
Moving average (5,20)	-0.91	1.08*
Moving average (10,50)	-0.04	1.24*
RSI (14-day)	0.09	-2.64
RSI (9-day)	1.27*	-2.05
CCI 20-day	1.17*	-1.41
20-day channel	-0.17	-0.86
50-day channel	-0.2	-1.76
Stochastic indicator	1.78*	0.58
Logistic regression	-0.25	-2.1
Random forest	0.25	-0.51
Pipeline model	-0.17	-2
Voting ensemble model	-0.16	-2.04
Neural network (single layer with 5 neurons)	0.2	-0.76
Neural network (three layers with 5 neurons)	0.3	-0.89
Neural network (three layers, 5 neurons, tanh)	-0.36	-0.43
Neural network (single layer with 17 neurons)	0.29	0.19
Neural network (single layer, 17 neurons, tanh)	0.41	0.23
Neural network (three layers with 17 neurons)	0.1	-0.18
Neural network (three layers, 17 neurons, tanh)	-0.56	-0.5

Note:  $\left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right] x = Ax, Ax \xrightarrow{d} \left(0, \frac{1}{5}\right), Net\ t - value = \frac{Avg.t-value}{\frac{1}{\sqrt{5}}},$

$Avg.\ t - value = \frac{1}{5} \sum_{i=1}^5 t_i$  where,  $t_i$  is mean t-value for each commodity. Critical t-value is calculated as 0.87.



Table 9. Summary of Variable Importance Using Random Forest

Variable/ Commodity	Corn	Copper	Japanese Yen	EuroDollar	Feeder Cattle
Moving average (5, 10)	1.00%	0.91%	1.08%	0.87%	1.15%
Moving average (5,20)	0.88%	0.23%	0.94%	0.70%	0.92%
Moving average (10,50)	1.17%	0.38%	1.26%	1.07%	1.10%
RSI (14-day)	1.17%	1.24%	1.76%	1.30%	1.08%
RSI (9-day)	1.14%	0.90%	1.02%	0.88%	1.26%
Stochastic indicator	0.87%	0.71%	0.96%	0.72%	1.13%
CCI 20-day	1.05%	1.00%	1.36%	0.38%	1.15%
20-day channel	0.33%	0.21%	0.35%	0.20%	0.40%
50-day channel	0.25%	0.38%	0.18%	0.09%	0.25%
Variance	92.14%	94.04%	91.11%	93.79%	91.56%

Table 10. Summary of Variable Importance Using Decision Tree

Variable/ Commodity	Corn	Copper	Japanese Yen	EuroDollar	Feeder Cattle
Moving average (5, 10)	0.67%	0.89%	0.80%	0.84%	0.78%
Moving average (5,20)	0.59%	0.13%	0.75%	0.55%	0.66%
Moving average (10,50)	0.99%	0.19%	0.96%	1.00%	0.72%
RSI (14-day)	0.94%	1.02%	1.34%	1.12%	0.68%
RSI (9-day)	0.94%	1.11%	0.93%	0.94%	1.05%
Stochastic indicator	0.56%	0.58%	0.70%	0.59%	0.60%
CCI 20-day	0.49%	0.77%	1.22%	0.29%	0.48%
20-day channel	0.25%	0.12%	0.21%	0.14%	0.25%
50-day channel	0.25%	0.22%	0.08%	0.11%	0.16%
Variance	94.33%	94.99%	93.00%	94.41%	94.64%

## **Conclusion**

Numerous studies have determined profitability of technical indicators in commodity markets. This study uses individual indicators alone as well as statistical and machine learning methods with signals from technical indicators as inputs. Models like the voting ensemble model and pipeline models are studied for the first time in commodity markets. These models are compared with most commonly used models like logistic regression, random forests, decision trees, and neural networks. Three types of neural networks (in total seven neural networks) are studied and special care has been taken to prevent data snooping error. Among all the individual technical indicators, RSI 9 days, stochastic indicator, and CCI 20 days generate significant profit for single day forecast. Trend-following systems like moving average (5, 20) and moving average (10, 50) generate significant profit for twenty day forecast (long term forecast). Statistical and machine learning methods are theoretically better as they are supposed to recognize the patterns, but practically they fail to live up to their potential and work only during certain time periods. None of the statistical and machine learning methods made significant profit in all the cases, even though they are profitable in a very few cases but never always.

None of the individual indicators or models generated significant profit in single day forecast for corn. In twenty day forecasts, only random forests and pipeline models are profitable. Japanese Yen is an interesting case where six of the seven neural networks studied generated significant profit for 20 day forecast. Neural networks should be explored more in foreign currency markets. EuroDollar did not give a similar result for neural networks. Feeder cattle is an interesting case, it is the only market where 20 day forecast of neural networks is better than single day forecast. For single day forecasts, all technical indicators, machine learning and statistical models failed to generate significant profit. For 20 day forecasts neural network with 3 hidden layers of 17 neurons each and “Softmax” activation function generated significant profit. Copper is the only

precious metal in this study. Here, twenty day forecasts are not profitable for any quantitative model, be it statistical or machine learning. Only stochastic oscillator and moving average (10, 50) generated significant profit and that too only in 20 day forecasts.

In general oscillators did better than trend-following systems in short term forecasting (single day forecast) but with long term forecasts (twenty day forecast) trend-following indicators had better success than mean-reversal indicators. Technical indicators should switch between bull and bear markets. One new indicator, namely the variance of change in closing prices is added to the other technical indicators, so as to help the switch, but it did not generate profit in all time periods. Performance of neural networks depend on the time period, they can be highly profitable for one period and completely fail in another. Neural networks generated profit in validation data set but the same neural networks worked poorly in testing data, even though both these data sets are out of sample. For neural networks one problem is likely to be lack of training data, given the relatively large number of parameters. One possibility is to use intraday data as the remedy. Another possibility is to pool data across commodities. As the present research clearly shows the value of composite forecasting, combining the forecasts from all 20 neural network models might have led to better forecasts. Future research should look into these possibilities. Future research should consider using reinforcement learning to estimate the parameters of all models. Future research should also include a new type of technical indicator to help the system switch between trend-following and mean-reversal technical trading systems. Variance is the only variable that is not unit-less. Variance for most of these commodities is higher in the training period than in the out-of-sample period. Future research needs to develop a scale neutral volatility measure so that it can estimate a model across a set of commodities.

## REFERENCES

- Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2), 245-271.
- Altman, E. I., Haldeman, R. G., & Narayanan, P. (1977). ZETATM analysis: A new model to identify bankruptcy risk of corporations. *Journal of Banking & Finance*, 1(1), 29-54.
- Barclay's CTA index . Retrieved June 22, 2017, from <https://www.barclayhedge.com/research/indices/cta/sub/cta.html>
- Beja, A., & Goldman, M. B. (1980). On the dynamic behavior of prices in disequilibrium. *The Journal of Finance*, 35(2), 235-248
- Boyd, M. S., & Brorsen, B. W. (1991). Factors related to futures markets disequilibrium. *Canadian Journal of Agricultural Economics*, 39(4), 769-778.
- Brandt, JA, & Bessler, D.A. (1981). Composite forecasting: an application with US hog prices. *American Journal of Agricultural Economics* 63(1), 135-140.
- Breiman, L., Friedman, J. H., & Olshen, R. A., Stone, C. J., (1984) Classification and regression trees. Wadsworth, Belmont, California.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.
- Chang, J., Jung, Y., Yeon, K., Jun, J., Shin, D., & Kim, H. (1996). Technical indicators and analysis methods. Seoul: Jinritamgu Publishing.
- Chang, P. K., & Osler, C. L. (1999). Methodical madness: Technical analysis and the irrationality of exchange-rate forecasts. *The Economic Journal*, 109(458), 636-661.
- Chong, J., & Miffre, J. (2010). Conditional correlation and volatility in commodity futures and traditional asset markets. *The Journal of Alternative Investments*, 12(13), 061-075.
- Chun, S. H., & Park, Y. J. (2005). Dynamic adaptive ensemble case-based reasoning: application to stock markets prediction. *Expert Systems with Applications*, 28(3), 435-443.
- CME group, (2016, 4 February). The big picture: A cost comparison of futures and ETFs. Retrieved June 22, 2017, from <http://www.cmegroup.com/trading/equity-index/report-a-cost-comparison-of-futures-and-etfs.html>.
- De Groot, C., & Würtz, D. (1991). Analysis of univariate time series with connectionist nets: A case study of two classical examples. *Neurocomputing*, 3(4), 177-192.
- Deng, Y., Bao, F., Kong, Y., Ren, Z. & Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems* 28(3), 653-664.

- Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Springer Berlin Heidelberg.
- Dimitras, A. I., Zanakis, S. H., & Zopounidis, C. (1996). A survey of business failures with an emphasis on prediction methods and industrial applications. *European Journal of Operational Research*, 90(3), 487-513.
- Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5), 352-359.
- Erhan, D., Courville, A., & Bengio, Y. (2010). Understanding representations learned in deep architectures. Department d'Informatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep, 1355.
- Hamm, L., & Wade Brorsen, B. W., (2000). Trading futures markets based on signals from a neural network. *Applied Economics Letters*, 7(2), 137-140.
- Huang, C. L., & Wang, C. J. (2006). A GA-based feature selection and parameters optimization for support vector machines. *Expert Systems with applications*, 31(2), 231-240.
- Kang, S., (1991). An Investigation of the Use of Feedforward Neural Networks for Forecasting. Ph.D. Thesis, Kent State University.
- Kidd, W. V., & Brorsen, B. W. (2004). Why have the returns to technical analysis decreased? *Journal of Economics and Business*, 56(3), 159-176.
- Kim, K. J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1), 307-319.
- Kim, S. H., & Chun, S. H. (1998). Graded forecasting using an array of bipolar predictions: application of probabilistic neural networks to a stock markets index. *International Journal of Forecasting*, 14(3), 323-337.
- Kuan, C. M., & Liu, T. (1995). Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics*, 10(4), 347-364.
- Lam, L., & Suen, S. Y. (1997). Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(5), 553-568.
- Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8), 10896-10904.
- Lukac, L. P., Brorsen, B. W., & Irwin, S. H. (1988). A test of futures markets disequilibrium using twelve different technical trading systems. *Applied Economics*, 20(5), 623-639.
- Lunga, D., & Marwala, T. (2006). Time series analysis using fractal theory and online ensemble classifiers. *AI 2006: Advances in Artificial Intelligence*, 312-321.
- Maslov, I. V., & Gertner, I. (2006). Multi-sensor fusion: an evolutionary algorithm approach. *Information Fusion*, 7(3), 304-330.

- Malkiel, B. G. (1989). Efficient market hypothesis. *The New Palgrave: Finance*. Norton, New York, 127-134.
- Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37), 870-877.
- Neely, C. J. (2002). The temporal pattern of trading rule returns and exchange rate intervention: intervention does not generate technical trading profits. *Journal of International Economics*, 58(1), 211-232.
- Neely, C. J., & Weller, P. A. (2003). Intraday technical trading in the foreign exchange markets. *Journal of International Money and Finance*, 22(2), 223-237.
- Oberlechner, T. (2001). Importance of technical and fundamental analysis in the European foreign exchange markets. *International Journal of Finance & Economics*, 6(1), 81-93.
- Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 109-131.
- Olson, D. (2004). Have trading rule profits in the currency markets declined over time? *Journal of Banking & Finance*, 28(1), 85-105.
- Ou, P., & Wang, H. (2009). Prediction of stock markets index movement by ten data mining techniques. *Modern Applied Science*, 3(12), 28.
- Pantalone, C. C., & Platt, M. B. (1987). Predicting commercial bank failure since deregulation. *New England Economic Review*, (July/August 1987b), 37-47.
- Park, C. H., & Irwin, S. H. (2005). The profitability of technical trading rules in US futures markets: A data snooping free test. *AgMAS Research Report*, 2005-04.
- Park, C. H., & Irwin, S. H. (2007). What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4), 786-826.
- Park, C. H., & Irwin, S. H. (2010). A reality check on technical trading rule profits in the US futures markets. *Journal of Futures Markets*, 30(7), 633-659.
- Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock markets index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4), 2162-2172.
- Patuwo, E., Hu, M. Y., & Hung, M. S. (1993). Two-group classification using neural networks. *Decision Sciences*, 24(4), 825-845.
- Pruitt, S. W., Tse, K. M., & White, R. E. (1992). The CRISMA trading system: The next five years. *The Journal of Portfolio Management*, 18(3), 22-25.
- Purcell, W.D. and S.R. Koontz. 1999. *Agricultural futures and options: Principles and strategies*, 2nd edition, Upper Saddle River, NJ, Prentice Hall.
- Roberts, M. C. (2005). Technical analysis and genetic programming: Constructing and testing a commodity portfolio. *Journal of Futures Markets*, 25(7), 643-660.

- Rodriguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10), 1619-1630.
- Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms* (No. VG-1196-G-8). Cornell Aeronautical Lab Inc Buffalo NY.
- Schwager, J. D. (1984). A complete guide to the futures markets: Fundamental analysis, technical analysis, trading, spreads, and options. New York, John Wiley & Sons.
- Shah, S., Brorsen, B. W., & Anderson, K. B. (2009, April). Liquidity costs in futures options markets. In Proceedings of the NCCC-134 Conference on Applied Commodity Price Analysis, Forecasting, and Markets Risk Management, St. Louis, MO.
- Sharda, R., & Patil, R. (1990, June). Neural networks as forecasting experts: an empirical test. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. 2, pp. 491-494). IEEE.
- Slezak, S. L. (2003). On the impossibility of weak-form efficient markets. *Journal of Financial and Quantitative Analysis*, 38(03), 523-554.
- Stevenson, R. A., & Bear, R. M. (1970). Commodity futures: Trends or random walks? *The Journal of Finance*, 25(1), 65-81.
- Subha, M. V., & Nambi, S. T. (2012). Classification of stock index movement using k-nearest neighbors (k-NN) algorithm. *Wseas Transactions on Information Science and Applications*, 9, 261-270.
- Sullivan, R., Timmermann, A., & White, H. (1999). Data-snooping, technical trading rule performance, and the bootstrap. *The Journal of Finance*, 54(5), 1647-1691.
- Sullivan, R., Timmermann, A., & White, H. (2003). Forecast evaluation with shared data sets. *International Journal of Forecasting*, 19(2), 217-227.
- Sweeney, R. J. (1988). Some new filter rule tests: Methods and results. *Journal of Financial and Quantitative Analysis*, 23(03), 285-300.
- Szakmary, A. C., Shen, Q., & Sharma, S. C. (2010). Trend-following trading strategies in commodity futures: A re-examination. *Journal of Banking & Finance*, 34(2), 409-426.
- Tang, Z., de Almeida, C., & Fishwick, P. A. (1991). Time series forecasting using neural networks vs. Box-Jenkins methodology. *Simulation*, 57(5), 303-310.
- Taylor, M. P., & Allen, H. (1992). The use of technical analysis in the foreign exchange markets. *Journal of International Money and Finance*, 11(3), 304-314.
- Weigend A., Huberman B.A. and Rummelhart D.E. (1992), Predicting sunspots and exchange rates with connectionist networks, in: Nonlinear Modeling and Forecasting, eds. M. Casdagli and S. Eubank, *SFI Studies in the Sciences of Complexity, Vol. XII* (Addison-Wesley, Reading, MA), pp. 397-434.

Wu, M. C., Lin, S. Y., & Lin, C. H. (2006). An effective application of decision tree to stock trading. *Expert Systems with Applications*, 31(2), 270-274.

Yao, J., & Tan, C. L. (2000). A case study on using neural networks to perform technical forecasting of Forex. *Neurocomputing*, 34(1), 79-98.

Zhang, C. X., & Zhang, J. S. (2008). A local boosting algorithm for solving classification problems. *Computational Statistics & Data Analysis*, 52(4), 1928-1941.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35-62.



## APPENDICES

### Appendix 1. Profit from Random Numbers for Single Day Forecast in Copper

Sr. No.	Random Numbers	5 Neurons				17 Neurons		
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	498.35	436.25	588.95	400.95	566.65	531.05	206.25
2	78747	176.35	175.95	608.65	717.25	651.15	518.05	470.95
3	50408	610.05	542.15	377.05	404.75	442.85	585.85	607.95
4	93875	175.95	632.75	271.35	671.85	276.85	453.85	372.45
5	44434	168.55	471.25	496.65	629.25	571.55	458.25	460.95
6	40150	489.35	764.15	767.45	388.05	361.35	377.85	735.55
7	25600	69.75	179.15	103.85	548.75	568.15	420.15	486.95
8	30450	1018.05	-67.35	430.45	545.75	958.65	328.45	287.05
9	15476	614.15	164.95	348.35	409.75	623.85	380.95	534.75
10	59372	398.85	180.15	311.75	633.35	360.75	593.25	-41.65
11	4988	426.65	437.95	-193.45	491.55	541.15	-14.65	309.45
12	96774	1140.25	385.15	799.35	663.65	673.15	672.55	476.05
13	81503	469.75	374.95	164.75	754.55	757.55	682.55	255.05
14	24885	214.05	506.95	317.15	831.95	200.95	331.65	649.15
15	82716	577.35	361.65	496.75	462.15	587.05	377.65	78.85
16	27177	544.35	327.55	318.55	519.55	384.55	421.75	306.45
17	96878	233.75	929.05	112.15	528.75	384.95	522.35	665.25
18	14024	717.45	188.95	387.45	526.95	642.35	608.65	318.35
19	74705	386.65	238.25	353.95	353.65	627.15	502.25	453.15
20	6769	218.15	-239.85	500.95	300.85	597.75	674.25	471.45
	Mean	457.39	349.5	378.11	539.17	538.92	471.34	405.22

Appendix 2. Profit from Random Numbers for 20-day Forecast in Copper

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	4408.8	1994.8	5900.3	5407.3	3479.1	517	615.8
2	78747	3139.1	-68.9	1229.4	3113.2	4082.1	3937	3692.3
3	50408	2334.3	4731	5290	5440.1	2703.7	3060	3707.3
4	93875	2152	5239	-2156	5069.8	4266.7	4094.1	1097.2
5	44434	1994.4	3119.9	7086	4353.1	4699.3	2369	2857.8
6	40150	3075	7364.1	4629.5	2865.4	1343	3167.3	5124
7	25600	772.5	1346.4	901.4	5311	2984.5	1711.5	3340.7
8	30450	5522.6	-280.6	4412.2	5537.1	6143.7	3292.2	830.5
9	15476	8054.2	2841.3	2970.5	4624.7	6636.4	2844.6	4651.4
10	59372	3135.4	3682.2	2629	6272	3636.1	5429.5	364.1
11	4988	771.2	3912.8	1035	5498.6	2711.4	225	654
12	96774	4385.9	2626.9	3499.3	5853	6074.1	4635	4069.7
13	81503	7595.3	1177.2	1454	5336.8	6246.7	5202.9	2895.1
14	24885	4121.9	4180.2	4640.6	5801.3	1753.5	2118.6	3777.3
15	82716	944.8	2122.9	5719.8	5440.9	5348.3	3191	2967
16	27177	6130.1	4938.7	2923.2	6064.5	4928.9	4492.7	1495.9
17	96878	4830.8	3986.1	2537.5	2858.8	4523.6	70.2	1994.1
18	14024	4499.4	2459.5	2737.9	4643.5	5414.4	4739.4	1385.8
19	74705	3694.4	2406.7	2882	6217.3	4215.2	2889	1812.5
20	6769	4226.2	-2248.2	2916.2	2417.6	4687.6	5139	2063.1
	Mean	3789.42	2776.6	3161.89	4906.3	4293.92	3156.25	2469.78

Appendix 3. Profit from Random Numbers for Single Day Forecast in Japanese Yen

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	0.7571	0.9145	0.3909	1.1497	0.5467	0.8137	0.7079
2	78747	1.0711	0.8044	0.5452	1.0709	1.0047	0.8395	0.4861
3	50408	0.9919	0.995	1.2795	0.7601	0.8677	0.8243	1.0891
4	93875	1.0591	0.5723	0.8205	0.7457	0.7421	0.9223	0.9997
5	44434	0.2495	1.0721	1.0385	0.7497	0.8475	0.9121	0.1305
6	40150	0.5456	1.2763	0.8913	0.6775	0.8563	0.9753	0.7023
7	25600	1.6641	0.8245	1.2897	1.0275	0.8567	0.6822	0.6021
8	30450	1.1375	0.8837	1.0533	0.9961	1.2121	0.7633	0.5743
9	15476	1.5931	1.3857	1.4693	0.8191	0.8679	0.6635	0.5163
10	59372	1.4065	0.9297	0.7083	1.1289	0.7477	0.9119	0.8065
11	4988	1.3149	0.3099	1.0405	1.0873	0.9518	0.8959	0.7631
12	96774	1.1733	0.9619	1.3707	0.7293	0.8895	0.8809	0.6523
13	81503	1.4119	0.9373	0.7543	0.9983	1.2749	0.7929	1.0637
14	24885	0.5918	0.5401	1.1413	0.7581	0.9907	0.8367	0.6213
15	82716	1.2719	0.7826	1.3381	1.2997	0.9779	0.6781	0.7391
16	27177	1.7763	0.5445	1.0453	0.6661	0.7861	0.4805	0.9006
17	96878	0.6623	1.1661	1.7701	0.6533	0.6603	0.2927	0.7319
18	14024	1.1553	1.1277	1.3425	0.6349	0.8949	0.3417	0.7195
19	74705	0.8126	0.8931	0.5133	0.6263	0.7241	0.6245	1.2611
20	6769	0.6947	0.4199	1.5881	1.1365	0.8192	0.6089	0.4857
	Mean	1.067025	0.867065	1.069535	0.88575	0.87594	0.73705	0.727655

Appendix 4. Profit from Random Numbers for 20-day Forecast in Japanese Yen

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	0.71	(0.57)	(2.04)	2.67	0.35	(1.17)	(0.08)
2	78747	1.07	0.80	0.55	1.07	1.00	0.84	0.49
3	50408	(0.46)	(1.19)	(1.15)	(0.91)	(0.14)	2.23	2.23
4	93875	0.83	0.18	(0.96)	(0.97)	0.11	0.04	1.89
5	44434	(0.75)	0.57	(1.62)	0.42	0.08	(0.52)	2.08
6	40150	(2.16)	(1.13)	1.17	0.83	(1.03)	1.50	(0.91)
7	25600	0.23	(2.58)	(1.30)	0.59	1.35	0.03	0.39
8	30450	(2.30)	(1.25)	(1.77)	1.45	0.16	1.37	0.42
9	15476	(0.07)	(0.26)	(0.06)	(0.84)	(0.59)	0.11	1.24
10	59372	(0.59)	1.08	(1.68)	1.96	1.12	(0.01)	1.50
11	4988	(1.49)	(1.69)	(3.77)	1.63	(1.03)	(0.31)	0.87
12	96774	(2.06)	2.48	(0.92)	(0.23)	0.86	0.93	(0.30)
13	81503	(0.55)	(0.09)	2.17	(0.79)	0.54	(1.45)	1.62
14	24885	(2.29)	(1.25)	(2.10)	0.24	1.80	(0.09)	(0.94)
15	82716	(1.30)	(2.17)	2.02	0.17	(1.43)	(0.67)	0.29
16	27177	(0.29)	(1.04)	(1.39)	(0.89)	0.83	0.10	0.74
17	96878	(0.90)	(1.27)	0.93	0.39	0.51	1.18	0.91
18	14024	0.84	(1.05)	(0.14)	0.32	(0.62)	(1.05)	(0.05)
19	74705	0.66	(1.50)	(3.01)	(1.15)	(0.32)	(0.46)	1.33
20	6769	(2.42)	(1.37)	(0.38)	(0.55)	(0.08)	0.68	1.87
	Mean	(0.66)	(0.66)	(0.77)	0.27	0.17	0.16	0.78

Appendix 5. Profit from Random Numbers for Single Day Forecast in Feeder Cattle

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	44.02	75.49	244.07	74.47	94.24	124.53	52.85
2	78747	128.18	129.45	108.49	156.20	111.73	117.18	75.80
3	50408	(53.47)	46.17	312.84	74.15	94.50	89.46	56.76
4	93875	173.80	130.15	107.13	89.64	86.65	121.05	92.77
5	44434	116.86	59.62	91.10	130.91	99.08	108.61	101.47
6	40150	77.47	46.60	65.25	143.20	57.19	104.80	132.74
7	25600	(39.48)	104.71	69.78	136.01	154.00	130.49	48.44
8	30450	161.83	159.62	77.78	104.83	57.86	57.33	48.93
9	15476	196.67	79.76	194.45	81.71	76.78	66.63	74.59
10	59372	118.92	24.78	94.65	116.39	60.51	121.68	131.11
11	4988	75.18	145.11	133.12	53.15	101.27	87.08	55.64
12	96774	209.14	148.50	61.59	88.37	105.80	73.10	85.47
13	81503	66.66	140.99	51.15	126.36	78.70	48.27	41.01
14	24885	249.26	3.13	215.77	95.14	89.15	89.37	50.58
15	82716	144.51	215.40	24.31	143.65	88.84	70.19	40.93
16	27177	102.55	147.84	240.16	122.55	77.50	98.35	74.31
17	96878	137.65	136.72	209.65	58.95	75.22	105.32	42.26
18	14024	118.88	73.13	167.55	112.82	125.08	86.51	60.35
19	74705	77.23	118.23	21.49	122.90	143.22	98.52	49.91
20	6769	118.35	81.10	171.39	95.06	91.00	96.73	71.54
	Mean	111.21	103.33	133.09	106.32	93.42	94.76	69.37

Appendix 6. Profit from Random Numbers for 20-day Forecast in Feeder Cattle

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	1176.17	1228.78	1017.26	1149.44	1040.69	884.35	403.25
2	78747	384.02	1347.35	1528.64	1095.95	1023.6	945.78	549.44
3	50408	94.81	1288.79	1490.02	621.99	917.71	847.78	642.21
4	93875	1024.68	1317.26	887.03	1137.43	753.43	886.97	1144.79
5	44434	887.51	846.1	1079.96	1047.77	1012.09	921.56	688.15
6	40150	1086.64	938.14	232.34	1001.29	880.81	588.1	980.86
7	25600	675.6	34.76	1113.15	789.02	699.71	1091.5	692.82
8	30450	981.55	1082.64	33.36	637.76	1008.72	910.46	554.09
9	15476	1070.44	488.2	1036.73	354.01	665.64	773.19	510.61
10	59372	1075.82	613.04	735.7	1055.14	899.88	790.17	987.93
11	4988	990.35	1296.74	1053.35	798.81	956.08	736.71	508.71
12	96774	1069.58	849.63	622.02	757.5	1239.59	596.3	603.26
13	81503	897.89	860.62	843.09	944.17	1043.64	605.26	722.84
14	24885	1364.88	1215.29	1134.21	722.88	951.76	994.46	884.18
15	82716	1127.72	932.61	908.85	1118.56	1371.37	759.24	718.93
16	27177	1070.53	1340.92	1645.99	1098.45	939.53	890.18	339.56
17	96878	1731.16	1160.18	1130.89	838.32	807.29	783.41	534.34
18	14024	577.61	798.01	1218.79	901.6	976.69	703.23	540.38
19	74705	889.99	465.46	294.56	1132.91	910.85	881.08	653.91
20	6769	1368.05	1061.73	1588.5	915.74	884.55	849.92	738.52
	Mean	977.25	958.3125	979.722	905.937	949.1815	821.983	669.939

Appendix 7. Profit from Random Numbers for Single Day Forecast in EuroDollar

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	(5.99)	(6.09)	(5.61)	(3.30)	(1.71)	(1.86)	(2.63)
2	78747	(6.69)	(3.58)	(6.82)	(4.56)	(1.90)	(3.73)	(3.13)
3	50408	(8.99)	(4.54)	(4.31)	(3.22)	(2.39)	(1.66)	(1.68)
4	93875	(5.32)	(6.87)	(7.67)	(4.50)	(3.59)	(2.17)	(1.79)
5	44434	(7.76)	(4.30)	(4.96)	(2.91)	(4.32)	(2.49)	(0.07)
6	40150	(6.07)	(7.38)	(5.94)	(2.99)	(4.23)	(0.84)	(1.72)
7	25600	(7.99)	(6.21)	(6.70)	(3.48)	(3.04)	(0.79)	(1.22)
8	30450	(6.68)	(5.61)	(5.09)	(5.41)	(3.09)	(2.08)	(0.61)
9	15476	(1.29)	(6.05)	(3.52)	(4.22)	(3.72)	(1.45)	(3.01)
10	59372	(7.55)	(5.54)	(5.48)	(3.89)	(2.41)	(1.95)	(2.81)
11	4988	(5.59)	(7.54)	(6.49)	(3.42)	(4.39)	(4.15)	(1.31)
12	96774	(8.00)	(3.59)	(6.01)	(3.17)	(3.41)	(3.13)	(2.16)
13	81503	(5.87)	(5.68)	(1.50)	(3.22)	(2.43)	(2.16)	(2.40)
14	24885	(3.90)	(3.69)	(7.26)	(3.51)	(3.56)	(2.42)	(2.96)
15	82716	(8.15)	(4.74)	(1.37)	(5.40)	(4.34)	(1.38)	(2.63)
16	27177	(3.88)	(2.73)	(5.49)	(4.17)	(5.94)	(4.60)	(2.64)
17	96878	(6.04)	(6.09)	(6.59)	(5.07)	(3.94)	(4.33)	(2.61)
18	14024	(7.31)	(6.50)	(4.56)	(4.87)	(4.69)	(1.05)	(2.03)
19	74705	(7.91)	(4.73)	(7.64)	(3.57)	(2.46)	(2.52)	(1.09)
20	6769	(6.33)	(4.33)	(6.94)	(2.43)	(4.52)	(1.91)	(0.46)
	Mean	(6.37)	(5.29)	(5.50)	(3.86)	(3.50)	(2.33)	(1.95)

Appendix 8. Profit from Random Numbers for 20-day Forecast in EuroDollar

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1	66109	(58.71)	(87.28)	(45.73)	(58.71)	(39.93)	(37.81)	(36.05)
2	78747	(84.39)	(62.52)	(86.72)	(35.96)	(40.40)	(38.75)	(27.78)
3	50408	(71.02)	(64.34)	(72.42)	(50.51)	(58.41)	(40.69)	(32.46)
4	93875	(62.72)	(34.95)	(73.71)	(41.17)	(51.74)	(40.02)	(34.76)
5	44434	(79.92)	(53.32)	(80.53)	(44.40)	(40.50)	(32.79)	(15.71)
6	40150	(68.70)	(71.45)	(92.35)	(33.99)	(18.97)	(39.47)	(28.36)
7	25600	(85.88)	(65.37)	(86.41)	(56.72)	(29.32)	(39.03)	(30.10)
8	30450	(71.05)	(78.00)	(67.65)	(58.06)	(22.36)	(28.93)	(22.44)
9	15476	(48.56)	(75.61)	(58.07)	(57.51)	(47.38)	(23.87)	(27.72)
10	59372	(73.24)	(69.61)	(79.99)	(37.96)	(39.34)	(34.62)	(32.57)
11	4988	(59.98)	(79.30)	(54.24)	(53.05)	(48.78)	(37.74)	(30.10)
12	96774	(60.22)	(35.79)	(77.49)	(42.01)	(53.01)	(36.53)	(24.49)
13	81503	(75.49)	(85.22)	(51.98)	(56.13)	(52.64)	(45.51)	(38.88)
14	24885	(74.15)	(82.10)	(88.31)	(39.84)	(28.16)	(34.24)	(28.40)
15	82716	(80.44)	(51.54)	(60.62)	(50.95)	(49.44)	(25.36)	(36.67)
16	27177	(27.07)	(67.09)	(53.02)	(39.95)	(34.66)	(46.36)	(25.67)
17	96878	(84.51)	(80.67)	(94.72)	(61.49)	(37.06)	(23.51)	(38.45)
18	14024	(73.20)	(61.93)	(70.93)	(47.09)	(48.58)	(24.76)	(27.94)
19	74705	(81.41)	(66.62)	(75.93)	(49.45)	(36.19)	(43.60)	(32.31)
20	6769	(80.83)	(66.55)	(80.73)	(49.80)	(51.47)	(31.59)	(35.03)
	Mean	(70.07)	(66.96)	(72.58)	(48.24)	(41.42)	(35.26)	(30.29)



Appendix 9. Profit from Random Numbers for Single Day Forecast in Corn

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1.00	66109	223.00	363.00	350.00	514.50	218.00	(144.00)	243.00
2.00	78747	187.00	(2.00)	215.00	856.00	152.50	478.50	694.00
3.00	50408	(239.00)	557.50	184.50	367.00	537.50	431.50	(76.50)
4.00	93875	818.50	644.00	667.50	190.50	172.00	694.00	(26.50)
5.00	44434	308.50	804.50	269.00	520.50	(29.00)	439.00	(183.00)
6.00	40150	145.50	(3.50)	(196.00)	238.00	596.50	356.00	(448.00)
7.00	25600	(61.50)	211.00	627.00	364.00	641.00	445.00	63.50
8.00	30450	278.50	437.00	92.50	210.50	375.50	385.50	469.50
9.00	15476	415.50	357.50	170.00	516.50	788.00	204.00	386.50
10.00	59372	137.00	149.50	141.50	407.00	509.00	176.00	50.50
11.00	4988	388.00	78.00	(179.50)	30.00	153.00	498.00	222.50
12.00	96774	116.00	255.00	187.00	188.00	236.50	176.50	397.50
13.00	81503	(95.50)	450.00	205.50	(30.00)	431.00	46.50	460.00
14.00	24885	586.50	141.00	95.00	458.50	464.50	696.50	237.50
15.00	82716	81.00	410.50	298.50	434.50	205.00	26.50	117.00
16.00	27177	367.00	547.00	437.00	529.50	381.50	79.50	225.00
17.00	96878	(66.00)	209.50	450.50	286.00	742.50	323.00	511.50
18.00	14024	494.00	207.00	(35.00)	360.00	137.00	512.50	46.50
19.00	74705	(41.00)	273.50	357.50	383.00	404.00	450.00	126.00
20.00	6769	(74.00)	213.00	319.50	557.50	607.00	287.00	636.00
	Mean	198.45	315.15	232.85	369.08	386.15	328.08	207.63

Appendix 10. Profit from Random Numbers for 20-day Forecast in Corn

Sr. No.	Random Numbers	5 Neurons			17 Neurons			
		MLP Only	MLP with Three Hidden Layers	MLP with "tanh" Function (Three Layers)	MLP only	MLP with "tanh" Function (Single Layer)	MLP with "Logistic" Function	MLP with "tanh" Function (Three Layers)
1.00	66109	(1340.75)	2496.25	3130.75	4859.25	1686.75	1010.75	3958.75
2.00	78747	5201.25	608.25	4545.75	6235.25	5060.25	4331.25	2611.75
3.00	50408	(2221.75)	3985.75	7093.75	5801.25	4986.25	1570.25	3123.25
4.00	93875	3526.25	5372.25	7942.75	(1229.75)	3876.25	5827.25	(961.25)
5.00	44434	2235.25	3545.75	1625.25	3391.75	343.75	3953.25	2708.75
6.00	40150	5070.75	(2729.75)	(931.75)	2819.75	2447.25	3172.75	3856.25
7.00	25600	(2012.25)	3929.25	5527.25	4382.75	5827.25	4946.25	1484.25
8.00	30450	5992.75	6218.25	7007.75	2277.25	4729.25	1684.75	(348.75)
9.00	15476	5075.75	3316.25	4451.25	3421.25	7766.25	4253.75	(164.75)
10.00	59372	(2936.25)	6881.25	3535.75	3449.25	3482.25	1186.75	2306.25
11.00	4988	4233.75	202.75	(4124.25)	2843.25	(1250.75)	2384.75	3028.25
12.00	96774	(5477.25)	5009.75	(5812.25)	2605.25	3598.75	2566.25	6253.25
13.00	81503	(2758.25)	7784.75	1463.25	4722.75	6984.25	2428.25	4220.75
14.00	24885	5575.25	(2246.25)	(4824.25)	2287.25	2955.25	4444.75	(2353.75)
15.00	82716	(824.75)	5035.75	(2696.75)	4381.75	843.25	1273.25	1929.75
16.00	27177	(3010.75)	(1163.25)	3884.25	5699.75	(245.25)	3773.75	5030.75
17.00	96878	3607.75	(1690.75)	7267.75	(1055.25)	2287.25	3022.75	3674.75
18.00	14024	5662.25	2223.75	805.25	2971.25	(2547.25)	3759.25	2437.75
19.00	74705	(1997.75)	6945.25	5594.75	(2819.25)	4789.75	2677.25	2510.75
20.00	6769	(2223.25)	(4190.75)	7442.25	3685.75	3312.25	(914.25)	1976.50
	Mean	1068.90	2576.73	2646.43	3036.53	3046.65	2867.65	2364.16

### Appendix 11. t-values for Profit/ Loss of Corn

Model	Mean Single Day Profit	Standard Deviation	t-value for Single Day Forecast	Mean 20-day Profit	Standard Deviation	t-value for 20-day Forecast
Moving average (5,10)	-0.58	8.019	-2.49472	-1.2	39.42	-1.04581
Moving average (5,20)	-0.29	8.0354	-1.25919	0.3	39.411	0.260081
Moving average (10,50)	-0.12	8.039	-0.50453	3.36	39.29	2.942282
RSI (14-day)	0.35	8.033	1.502131	0.04	39.44	0.033754
RSI (9-day)	0.47	8.027	2.015727	1.14	39.42	0.995611
CCI 20-day	0.5	8.028	2.152118	-2.69	39.35	-2.34732
20-day channel	-0.19	8.038	-0.8314	-0.01	39.44	-0.00793
50-day channel	-0.22	8.0376	-0.95356	-1.44	39.416	-1.25447
Stochastic indicator	1.06	7.97	4.598157	1	39.41	0.873296
Logistic regression	0.05	8.04	0.193886	-1.23	40.66	-1.03646
Random forest	0.23	8.0374	0.985907	0.79	40.67	0.665223
Pipeline model	0.19	8.03	0.805271	0.17	40.68	0.146102
Voting ensemble model	0.22	8.03	0.941879	-1.29	40.66	-1.08763
Neural network (single layer with 5 neurons)	0.16	8.039	0.691255	-1.77	40.6476	-1.49637
Neural network (three layers with 5 neurons)	0.03	8.04	0.140029	-0.82	40.6778	-0.69657
Neural network (three layers, 5 neurons, tanh)	-0.05	8.0405	-0.21901	-1.04	40.672	-0.87875
Neural network (single layer with 17 neurons)	-0.07	8.04	-0.31596	-0.18	40.68	-0.15111
Neural network (single layer, 17 neurons, tanh)	0.23	8.0373	1.002082	1.42	40.6614	1.198155
Neural network (three layers with 17 neurons)	-0.26	8.0363	-1.13512	-2.18	40.6273	-1.84771
Neural network (three layers, 17 neurons, tanh)	-0.1	8.04	-0.40932	-2.35	40.618	-1.98963

## Appendix 12. t-values for Profit/ Loss of Copper

Model	Mean Single Day Profit	Standard Deviation	t-value for Single Day Forecast	Mean 20-day Profit	Standard Deviation	t-value for 20-day Forecast
Moving average (5,10)	-0.01	0.9759	-0.3485	-0.1	4.1946	-0.7087
Moving average (5,20)	-0.02	0.9757	-0.7711	0.15	4.1928	1.1459
Moving average (10,50)	0.02	0.9757	0.7691	0.68	4.1397	5.1308
RSI (14-day)	0.01	0.9759	0.1909	-0.56	4.157	-4.2108
RSI (9-day)	0.02	0.9757	0.6693	-0.32	4.1832	-2.406
CCI 20-day	0.02	0.9757	0.7398	-0.15	4.1932	-1.0821
20-day channel	-0.03	0.9756	-0.9284	-0.6	4.1527	-4.4901
50-day channel	-0.03	0.9756	-0.9277	-0.63	4.1484	-4.7104
Stochastic indicator	0.02	0.9757	0.7343	0.6	4.152	4.5132
Logistic regression	-0.03	0.9755	-0.9543	-0.66	4.2771	-4.8324
Random forest	-0.03	0.9756	-0.9046	-0.66	4.2776	-4.804
Pipeline model	-0.03	0.9755	-0.9105	-0.7	4.271	-5.0842
Voting ensemble model	-0.03	0.9755	-0.9259	-0.49	4.3007	-3.5327
Neural network (single layer with 5 neurons)	0	0.976	0.0643	-0.08	4.3277	-0.5496
Neural network (three layers with 5 neurons)	-0.03	0.9756	-0.9004	-0.68	4.2749	-4.9298
Neural network (three layers, 5 neurons, tanh)	-0.01	0.9759	-0.3899	-0.54	4.2948	-3.8952
Neural network (single layer with 17 neurons)	0.04	0.9752	1.1979	-0.57	4.2908	-4.1213
Neural network (single layer, 17 neurons, tanh)	-0.02	0.9757	-0.6957	-0.05	4.328	-0.3932
Neural network (three layers with 17 neurons)	0	0.9759	0.1463	-0.25	4.321	-1.8138
Neural network (three layers, 17 neurons, tanh)	-0.09	0.9759	-2.9439	-0.38	4.311	-2.7411

Appendix 13. t-values for Profit/ Loss of EuroDollar

Model	Mean Single Day Profit	Standard Deviation	t-value for Single Day Forecast	Mean 20-day Profit	Standard Deviation	t-value for 20-day Forecast
Moving average (5,10)	-0.0006	0.0135	-1.2105	0.0022	0.0505	1.2221
Moving average (5,20)	-0.0005	0.0135	-0.9853	0.0044	0.0504	2.3990
Moving average (10,50)	0.0001	0.0135	0.2049	0.0017	0.0506	0.9470
RSI (14-day)	-0.0001	0.0135	-0.2445	-0.0089	0.0498	-4.9518
RSI (9-day)	0.0010	0.0134	2.0873	-0.0083	0.0499	-4.6206
CCI 20-day	0.0004	0.0134	0.9371	-0.0050	0.0503	-2.7567
20-day channel	0.0005	0.0135	1.1309	0.0065	0.0502	3.5781
50-day channel	0.0008	0.0135	1.6218	0.0061	0.0502	3.3628
Stochastic indicator	0.0011	0.0080	3.9757	-0.0014	0.0359	-1.0548
Logistic regression	-0.0003	0.0134	-0.6978	-0.0062	0.0512	-3.3413
Random forest	0.0001	0.0134	0.1662	-0.0010	0.0515	-0.5496
Pipeline model	-0.0003	0.0134	-0.6978	-0.0062	0.0512	-3.3413
Voting ensemble model	-0.0003	0.0135	-0.6145	-0.0073	0.0511	-3.9620
Neural network (single layer with 5 neurons)	0.0000	0.0134	0.0864	-0.0012	0.0515	-0.6546
Neural network (three layers with 5 neurons)	0.0004	0.0134	0.8441	-0.0055	0.0512	-2.9645
Neural network (three layers, 5 neurons, tanh)	-0.0008	0.0134	-1.6148	0.0005	0.0515	0.2731
Neural network (single layer with 17 neurons)	-0.0001	0.0134	-0.2191	0.0007	0.0515	0.3641
Neural network (single layer, 17 neurons, tanh)	0.0005	0.0134	1.1498	-0.0055	0.0512	-2.9827
Neural network (three layers with 17 neurons)	0.0007	0.0135	1.3564	-0.0025	0.0515	-1.3233
Neural network (three layers, 17 neurons, tanh)	0.0003	0.1347	0.0535	-0.0016	0.0515	-0.8367

Appendix14. t-values for Profit/ Loss of Feeder Cattle

Model	Mean Single Day Profit	Standard Deviation	t-value for Single Day Forecast	Mean 20- day Profit	Standard Deviation	t-value for 20-day Forecast
Moving average (5,10)	-0.0106	1.84	-0.1755	-0.2181	8.61	-0.7680
Moving average (5,20)	0.0006	1.85	0.0107	-0.2415	8.61	-0.8503
Moving average (10,50)	0.0055	1.85	0.0910	0.1638	8.61	0.5766
RSI (14-day)	0.0127	1.84	0.2105	0.2061	8.61	0.7256
RSI (9-day)	0.0150	1.84	0.2504	0.0748	8.61	0.2633
CCI 20-day	-0.0120	1.84	-0.1995	-0.7934	8.58	-2.8043
20-day channel	-0.0073	1.84	-0.1218	-1.1579	8.54	-4.1125
50-day channel	0.0095	1.84	0.1584	0.4002	8.60	1.4100
Stochastic indicator	-0.0042	1.85	-0.0700	-0.7584	8.82	-2.6071
Logistic regression	-0.0065	1.84	-0.1079	-0.0812	8.85	-0.2779
Random forest	-0.0033	1.85	-0.0553	-0.7026	8.82	-2.4137
Pipeline model	-0.0038	1.85	-0.0636	-0.2349	8.85	-0.8048
Voting ensemble model	-0.0015	1.85	-0.0254	-0.1202	8.85	-0.4117
Neural network (single layer with 5 neurons)	-0.0079	1.84	-0.1312	0.1848	8.85	0.6331
Neural network (three layers with 5 neurons)	-0.0024	1.85	-0.0397	-0.0572	8.85	-0.1959
Neural network (three layers, 5 neurons, tanh)	0.0000	1.85	-0.0003	0.3372	8.85	1.1557
Neural network (single layer with 17 neurons)	-0.0006	1.85	-0.0093	0.2298	8.85	0.7872
Neural network (single layer, 17 neurons, tanh)	-0.0017	1.85	-0.0279	0.6916	8.83	2.3756
Neural network (three layers with 17 neurons)	0.0013	1.85	0.0208	0.2468	8.85	0.8457
Neural network (three layers, 17 neurons, tanh)	-0.0106	1.84	-0.1755	-0.2181	8.61	-0.7680

Appendix 15. t-values for Profit/ Loss of Japanese Yen

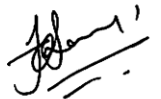
Model	Mean Single Day Profit	Standard Deviation	t-value for Single Day Forecast	Mean 20-day Profit	Standard Deviation	t-value for 20-day Forecast
Moving average (5,10)	-0.0005	0.0062	-2.6325	0.001	0.0251	1.2214
Moving average (5,20)	-0.0003	0.0062	-1.3398	0.002	0.025	2.3809
Moving average (10,50)	-0.0001	0.0062	-0.6604	-0.0017	0.0251	-1.9859
RSI (14-day)	-0.0002	0.0062	-1.1018	-0.0038	0.0244	-4.6267
RSI (9-day)	0.0003	0.0062	1.3632	-0.0041	0.0248	-4.9388
CCI 20-day	0.0004	0.0061	1.7752	-0.0009	0.0251	-1.127
20-day channel	0	0.0062	-0.0005	-0.0005	0.0251	-0.5897
50-day channel	-0.0001	0.0062	-0.6364	-0.0018	0.0251	-2.0973
Stochastic indicator	-0.0001	0.0062	-0.5695	-0.0024	0.025	-2.8311
Logistic regression	0.0001	0.0062	0.297	0.0011	0.0256	1.3087
Random forest	0.0002	0.006	1.1295	0.0021	0.0255	2.4073
Pipeline model	0	0.0062	0.0053	0.0006	0.0256	0.6822
Voting ensemble model	0	0.0062	-0.1551	-0.0007	0.0256	-0.8133
Neural network (single layer with 5 neurons)	0	0.0062	0.1684	-0.0006	0.0256	-0.6965
Neural network (three layers with 5 neurons)	0.0003	0.0062	1.5708	0.003	0.0254	3.4866
Neural network (three layers, 5 neurons, tanh)	0.0001	0.0062	0.4733	0.0022	0.0255	2.5257
Neural network (single layer with 17 neurons)	0.0002	0.0062	0.804	0.0032	0.0254	3.7259
Neural network (single layer, 17 neurons, tanh)	0.0001	0.0062	0.6057	0.0022	0.0255	2.5487
Neural network (three layers with 17 neurons)	0	0.0062	0.1397	0.0015	0.0255	1.704
Neural network (three layers, 17 neurons, tanh)	0.0001	0.0062	0.4831	0.0019	0.0255	2.2262

Appendix 16.

Pre analysis plan

1. Use technical indicators for predicting direction of trade.  
List: Dual Moving Average indicators(5,10), (5, 20) and (10, 50) days, RSI (14, 9) days, CCI 20 days, 20 day channel, 50 day Channel and Stochastic indicator, Variance (CV).
2. Compare profit and loss based on prediction using Logistic Regression, Random Forest, voting ensemble model, pipeline model and neural networks.
3. Voting Ensemble model uses: Logistic regression, Random forest, Gaussian Naive Bayes, Decision tree, Support vector classification.
4. Random forest will be with 20 trees, using "gini" and out of bag error as criteria for selection of estimators. Start with '20 trees' and keep on adding trees in the batch of '10 trees' each till optimum is reached.
5. Random forest and decision trees will be used to find the mostly used indicators for future studies.
6. Random number generator will be used to pick 20 seeds initially and then final seed will be selected based on the accuracy and profit of predictions. Validation data set will be used for these computations. This will be computed for neural network only.
7. Neural network will be of three types:
  - a) Single hidden layer with one hidden layer and 5 and 17 neurons.
  - b) Three hidden layer with 5 and 17 neurons each.
  - c) Repeat the step "a" and "b" with pretraining. Pretraining will be done using RBMs having 2 hidden layers of 5 neurons each. If pretraining will give same results as in step (b) then step (c) will not be done.
8. Neural networks will use "l-fgs (limited-memory BFGS)" algorithm and "Logistic" activation function. Step "c" will use "tanh" activation function instead of "Logistic" activation function.
9. New model will be formed for individual commodities.  
List:
  - a) CME EuroDollar futures (ED)
  - b) CME Japanese Yen JPY futures (JY)
  - c) CME Copper Futures (HG)
  - d) CME Feeder Cattle Futures (FC)
  - e) CME CBOT Corn Futures (C)
10. Regression tree alone will also be formed for comparison purposes.
11. Profit will also be calculated using individual indicators.
12. Validation data set will be used to find random seed only.
13. Evaluation will be done using the costs per trade and Profit will be calculated per contract. Two types of forecasts will be used to calculate profit: a) single day prediction and b) 20 day prediction.

Mark Brown  
2-9-2017

  
2/9/17



Appendix 17. Barclay's CTA Index

Year	CTA Index	Year	CTA Index	Year	CTA Index
1980	63.69%	1993	10.37%	2006	3.54%
1981	23.90%	1994	-0.65%	2007	7.64%
1982	16.68%	1995	13.64%	2008	14.09%
1983	23.75%	1996	9.12%	2009	-0.10%
1984	8.74%	1997	10.89%	2010	7.05%
1985	25.50%	1998	7.01%	2011	-3.09%
1986	3.82%	1999	-1.19%	2012	-1.70%
1987	57.27%	2000	7.86%	2013	-1.42%
1988	21.76%	2001	0.84%	2014	7.61%
1989	1.80%	2002	12.36%	2015	-1.50%
1990	21.02%	2003	8.69%	2016	-1.23%
1991	3.73%	2004	3.30%	2017	-0.67% <sup>†</sup>
1992	-0.91%	2005	1.71%		

Appendix 18. Neural Networks (Artificial neural networks, ANNs)

ANNs with the  $k$  output nodes can be used to forecast multi-step ahead points directly using all the useful past observations as inputs. ANNs are considered as the universal function approximators hence, they can capture nonlinear relationships in a better way. In addition to these characteristics ANNs have more properties like ANN learning methods are quite robust to noise in the training data, long training times are acceptable for ANNs, and they use the black box approach which may or may not be acceptable to all humans (Mitchell, 1997). Weigend et al. (1992) find the ANN model to be better than random walk model.

ANN can be constructed using many ways including feedforward and recurrent networks. Most studies have used the straightforward Multilayer perceptron (MLP) for forecasting (Kang, 1991; Sharda and Patil, 1990). A MLP is an feedforward ANN model that maps sets of input data onto a set of appropriate outputs (Rosenblatt 1961). This study uses MLP neural networks.

An ANN is typically composed of layers of nodes. MLP neural networks have all the input nodes in input layer, hidden layer is distributed into one or more hidden layers between input and output nodes, while the output layer consists of the output nodes.

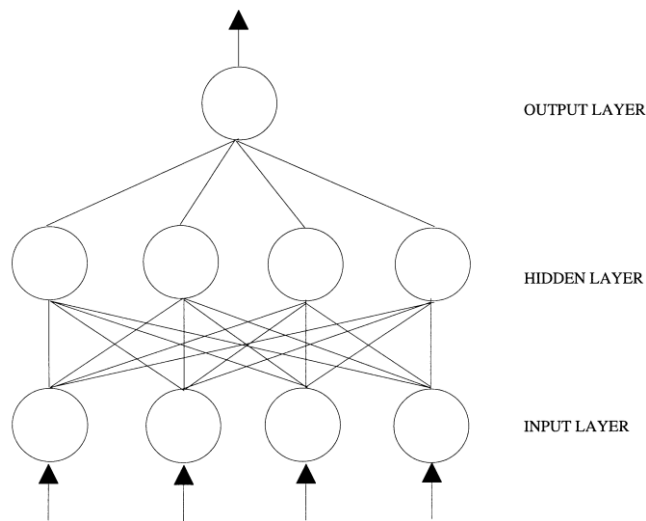


Figure: An illustrative example of MLP neural network.

There are many critical parameters that effect the performance of an ANN. One of them is determining the architecture of the ANN. Number of hidden layers, number of hidden neurons , number of output neurons, transfer (activation) function for hidden and ouput layer, training algorithm, Evaluating criteria, number of training iterations and learning rate and momentum are very crucial for the architecture of the ANN. Several researchers have tried to address these issues, but there is no consensus on method of determination of these parameters.

Kang (1991) use 1 hidden layer with variable hidden neurons, 1 output neuron, sigmoid transfer function for both hidden and ouput layer, generalized reduced gradient algorthim and MSE, mean algebraic percent error (MAPE) and MAD as evaluating criteria with simulated and real time series data. Schoneburg(1990) use the daily stock price data for forecasting daily stock prices with 10 input neurons, 2 hidden layers, 1 output neuron, sigmoid and sine, sigmoid transfer function for hidden and output layer respectively, backpropagation (BP) algorithm and MAPE as the evaluating criteria. Weigend et al. (1992) 12 input neurons. 1 hidden layer with 8 neurons, 1 output neuron, sigmoid and tanh as activation function, linear function for ouput layer, BP algorithm, average relative variance (ARV) as evaluation criteria in his forecasting for sunspots daily exchange rate. Kuan and Liu, 1995 in their work on daily exchange rates use the Newton training algorithm with sigmoid activation function and linear transfer function with root mean sqare error (RMSE) as evaluation criterion.

One hidden layer is considered sufficient to approximate any complex nonlinear function with desired accuracy (Cybenko, 1989) , but one hidden layer results in long training time and bad network generalization as it requires a very large number of hidden nodes. Also ANNs having more than one hidden layer is considered deep learning by some authors (Erhan et al., 2010).

Number of hidden nodes is another crucial aspect of making ANN. Networks with too few hidden nodes (neurons) may not be able to train and model data while it is preferred to have fewer hidden nodes so as to have lower overfitting and better generalization. There is no universal rule for selecting the number of hidden nodes, it is mostly done by trial and error methods. Generally the number of hidden nodes depend on the number of input nodes. Many researchers have use different rules for number of hidden nodes. Lippman (1987) use “ $2n+1$ ”, while Kang (1991) uses “ $n/2$ ” hidden nodes where  $n$  is the number of input nodes. Tang et al. (1993) and De Groot and Wurtz (1991) set the number of hidden nodes to be equal to number of input nodes.

Number of input nodes is considered to be the most critical decision variable for a time series forecasting problem as it contains the important information about the linear/nonlinear autocorrelation structure in the data. Zhang et al. (1998), prefer the use of theoretical research to determine the number of input nodes for nonlinear time series analysis. Many others have adopted some intuitive or empirical ideas for selecting the number of input nodes. Tang et al. (1991) use the four input nodes for the quarterly data while Sharda and Patil (1992) use 12 input nodes for the monthly data. For the purpose of this paper we have relied on theoretical research to decide the number of input nodes. This study uses five and 17 neurons with one and three hidden layers.

Transfer functions also called activation function determines the relationship between inputs and output nodes of the neural network. They are known to introduce nonlinearity in the neural networks. Generally speaking any differentiable continuous function can be used as a transfer function. But for the purpose of time series data bounded, monotonically increasing and differentiable functions like sigmoid (logistic) , hyperbolic tangent (tanh), sine or cosine function are use. We can use one activation function for all nodes or we can use different activation functions for different nodes. Mostly, networks use the same activation function for the nodes in the same layer. Majority of the reserachers simply use the logistic activation function for all the hidden and

output nodes but few researchers like De Groot and Wurtz (1991) use tanh activation function but there is no consensus about this. This study uses the sigmoid function which is special case of softmax where the number of classes equals to two.

Training algorithm also known as the optimization method is a nonlinear minimization problem which works by giving arc weights to parameters to minimize the total or mean squared errors between the actual and desired output levels. There are many training algorithms like backpropagation, Levenberg-Marquardt, quasi-Newton (like BFGS and L-BFGS) available to the researchers. None of these algorithms guarantee the global optimal solution to the problem so the emphasis is mainly on finding the “best” local optima for the solution if global solution is not available. Patuwo et al. (1993). Apart from being a widely available algorithm in optimization software, it does not require the learning parameters such as learning rate and momentum which are required in backpropagation methods. This study uses the L-BFGS algorithm (as explained earlier in the main text).

Evaluation criteria also known as the cost function or objective function is another important issue in the neural network architecture. Objective functions such as SSE and MSE or others that can be described as error are used.

This study uses the L2 penalty (regularization term) parameter of 0.0001. As this study uses the “constant” (0.5) base learning rate for weight updates and as it stands “constant” keeps the learning rate constant throughout training. Maximum iterations ( $i$ , epochs) is 20,000. Also tolerance for optimization criteria is 0.00001 for this study. This means that when the loss at iteration  $i+1$  differs less than 0.00001 from that at iteration  $i$ , convergence is considered to be reached and the algorithm exits.

Yao et al. (2000) suggested using a measure of volatility as input for the formulation of the neural network for forecasting. The ANN should be self-adapting to different situations, for the purpose of the volatility of the current measurement should be incorporated to the model. A sudden change in the volatility is an indication of an impending major move. It can signal the beginning of a trend, an end or a reversal of a trend, or possibly even a price crash or a switch from trend to reversal.

Appendix 19.

Python code:

```

"""Utilities for the neural network modules
"""
from numpy import genfromtxt
import gzip, cPickle
from glob import glob
import numpy as np
from itertools import chain
import pandas as pd
import baseMultilayerPerceptron
class MultilayerPerceptronClassifier as MLP
csvFile = "C:\\Users\\jasdeep\\Desktop\\datacopper.csv"
csvFileY="C:\\Users\\jasdeep\\Desktop\\datacopper1.csv"
my_data = genfromtxt(csvFile, delimiter=',', skip_header=1)
my_data1 = genfromtxt(csvFileY, delimiter=',', skip_header=1)
# Data and labels are read
train_set_x = my_data[:7140]
valid_set_x = my_data[7171:9191]
test_set_x = my_data[9221:10210]
train_set_y = my_data1[:7140]
valid_set_y = my_data1[7171:9191]
test_set_y = my_data1[9221:10210]
# Divided dataset into 3 parts. 70%,20%,10%
train_set = train_set_x, train_set_y
valid_set = valid_set_x, valid_set_y
test_set = test_set_x, test_set_y
#random_state = 0
n_hidden=50
import numpy as np
from sklearn.utils.fixes import expit as logistic_sigmoid

def identity(X):
    return X
def logistic(X):
    return logistic_sigmoid(X, out=X)
def tanh(X):
    return np.tanh(X, out=X)
def relu(X):
    np.clip(X, 0, np.finfo(X.dtype).max, out=X)
    return X
def softmax(X):
    tmp = X - X.max(axis=1)[:, np.newaxis]
    np.exp(tmp, out=X)
    X /= X.sum(axis=1)[:, np.newaxis]
    return X
ACTIVATIONS = {'identity': identity, 'tanh': tanh, 'logistic': logistic,
                'relu': relu, 'softmax': softmax}
def logistic_derivative(Z):
    return Z * (1 - Z)
def tanh_derivative(Z):
    return 1 - (Z ** 2)
def relu_derivative(Z):
    return (Z > 0).astype(Z.dtype)
DERIVATIVES = {'tanh': tanh_derivative, 'logistic': logistic_derivative,
                'relu': relu_derivative, 'identity': lambda x: 1}
def squared_loss(y_true, y_pred):
    return ((y_true - y_pred) ** 2).sum() / (2 * y_true.shape[0])

def log_loss(y_true, y_prob):
    y_prob = np.clip(y_prob, 1e-10, 1 - 1e-10)
    return -np.sum(y_true * np.log(y_prob) +
                  (1 - y_true) * np.log(1 - y_prob)) / y_prob.shape[0]

```

```

LOSS_FUNCTIONS = {'squared_loss': squared_loss, 'log_loss': log_loss}
def binary_KL_divergence(p, p_hat):

    p_hat = np.clip(p_hat, 1e-10, 1 - 1e-10)
    return (p * np.log(p / p_hat)) + ((1 - p) * np.log((1 - p) / (1 - p_hat)))
from __future__ import print_function
print(__doc__)
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.datasets import load_digits
from sklearn.neural_network import BernoulliRBM
from sklearn.pipeline import Pipeline
from sklearn import linear_model, datasets, metrics
random_state = 66109
print("Random_state:", random_state)
mlp = MultilayerPerceptronClassifier(hidden_layer_sizes=(5,), activation="logistic",
    algorithm='l-bfgs', alpha=0.00001,
    batch_size=10, learning_rate="constant",
    learning_rate_init=0.5, power_t=0.5, max_iter=20000,
    shuffle=False, random_state=random_state, tol=1e-5,
    verbose=False, warm_start=False)

mlp.fit(train_set_x, train_set_y)
#score_with_mlp_only = mlp.score(test_set_x, test_set_y)
x = [mlp.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
#print ("mlp only prediction: ",x2 [:50])
#print ("shape of x2: ",x2.shape)
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach\\Copper\\random_number_check_data.xlsx")
#df.head()
#print (df)
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
#rev=result.values[:,3]
#clospr=result.values[:,4]
rev = 0
#for index,val in np.ndenumerate(prediction[:-1]):
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
    #print("index: "+str(index)+" val: "+str(val)+" rev: "+str(rev))
#rev = df['rev'].sum()
#print(rev,df['rev'][:-1].sum())
print("Revenue from mlp 20 day,single layer, 5 neurons:", rev)
mlp2 = MultilayerPerceptronClassifier(hidden_layer_sizes=(5,5,5,), activation="logistic",
    algorithm='l-bfgs', alpha=0.00001,
    batch_size=10, learning_rate="constant",
    learning_rate_init=0.25, power_t=0.5, max_iter=20000,
    shuffle=False, random_state=random_state, tol=1e-5,
    verbose=False, warm_start=False)

mlp2.fit(train_set_x, train_set_y)

```



```

#score_without_pretraining = mlp2.score(test_set_x, test_set_y)
x = [mlp2.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach_data\\Copper\\random_number_check_data.xlsx")
#df.head()
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
rev = 0
#for index,val in np.ndenumerate(prediction[:-1]):
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
print("Revenue from 20 day, three layers, 5 neurons :", rev)

# Cross-validate multi-layer perceptron with rbm pre-training
rbms = [BernoulliRBM(batch_size=10, n_components=n_hidden, random_state=random_state,
                    learning_rate=0.5, n_iter=1000),
        BernoulliRBM(n_components=n_hidden, random_state=random_state, batch_size=10,
                    learning_rate=0.5, n_iter=1000)]

mlp3 = MultilayerPerceptronClassifier(hidden_layer_sizes=(5,5,5), activation="tanh",
                                     algorithm='l-bfgs', alpha=0.00001,
                                     batch_size=10, learning_rate="constant",
                                     learning_rate_init=0.25, power_t=0.5, max_iter=20000,
                                     shuffle=False, random_state=random_state, tol=1e-5,
                                     verbose=False, warm_start=rbms)

mlp3.fit(train_set_x, train_set_y)
x = [mlp3.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach_data\\Copper\\random_number_check_data.xlsx")
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
rev = 0
#for index,val in np.ndenumerate(prediction[:-1]):
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
print("Revenue from tanh, 20 day, three layres, 5 neurons:", rev)
mlp = MultilayerPerceptronClassifier(hidden_layer_sizes=(17,), activation="logistic",
                                     algorithm='l-bfgs', alpha=0.00001,
                                     batch_size=10, learning_rate="constant",

```

```

learning_rate_init=0.5, power_t=0.5, max_iter=20000,
shuffle=False, random_state=random_state, tol=1e-5,
verbose=False, warm_start=False)

mlp.fit(train_set_x, train_set_y)
x = [mlp.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach_data\\Copper\\random_number_check_data.xlsx")
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
rev = 0
#for index,val in np.ndenumerate(prediction[:-1]):
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
print("Revenue from mlp, 20 day, 17 neurons, 1 layer:", rev)
mlp = MultilayerPerceptronClassifier(hidden_layer_sizes=(17,), activation="tanh",
    algorithm='l-bfgs', alpha=0.00001,
    batch_size=10, learning_rate="constant",
    learning_rate_init=0.5, power_t=0.5, max_iter=20000,
    shuffle=False, random_state=random_state, tol=1e-5,
    verbose=False, warm_start=False)

mlp.fit(train_set_x, train_set_y)
x = [mlp.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
import pandas as pd
import numpy as np
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach_data\\Copper\\random_number_check_data.xlsx")
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
rev = 0
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
print("Revenue from tanh, 20 day, 17 neurons, 1 layer:", rev)
mlp2 = MultilayerPerceptronClassifier(hidden_layer_sizes=(17,17,17), activation="logistic",
    algorithm='l-bfgs', alpha=0.00001,
    batch_size=10, learning_rate="constant",
    learning_rate_init=0.25, power_t=0.5, max_iter=20000,
    shuffle=False, random_state=random_state, tol=1e-5,
    verbose=False, warm_start=False)
mlp2.fit(train_set_x, train_set_y)

```

```

x = [mlp2.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
import pandas as pd
import numpy as np
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach_data\\Copper\\random_number_check_data.xlsx")
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
rev = 0
#for index,val in np.ndenumerate(prediction[:-1]):
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
print("Revenue from without_pretraining, 20 day, 17 neurons, 3 layers:", rev)
n_hidden =5
rbms = [BernoulliRBM(batch_size=10, n_components=n_hidden, random_state=random_state,
                    learning_rate=0.5, n_iter=1000),
        BernoulliRBM(n_components=n_hidden, random_state=random_state,batch_size=10,
                    learning_rate=0.5, n_iter=1000)]
mlp2 = MultilayerPerceptronClassifier(hidden_layer_sizes=(17,17,17,)), activation="tanh",
    algorithm='l-bfgs', alpha=0.00001,
    batch_size=10, learning_rate="constant",
    learning_rate_init=0.25, power_t=0.5, max_iter=20000,
    shuffle=False, random_state=random_state, tol=1e-5,
    verbose=False, warm_start=False)
mlp2.fit(train_set_x, train_set_y)
x = [mlp2.predict(valid_set_x)];
a = np.array(x)[np.newaxis]
x2= a.T
x3 = len(x2)
x4 = np.reshape(x2, (x3,-1))
df = pd.read_excel("C:\\Users\\jasdeep\\Desktop\\reserach_data\\Copper\\random_number_check_data.xlsx")
df1 = pd.DataFrame(df)
list6 = pd.DataFrame(x4)
list7 = pd.DataFrame(data = list6).reset_index()
list7.columns = ['Second_Index', 'prediction']
del list7['Second_Index']
result = pd.concat([list7, df1], axis=1)
prediction=result.values[:,0]
orgdep=result.values[:,1]
opnpr=result.values[:,2]
rev = 0
#for index,val in np.ndenumerate(prediction[:-1]):
for index,val in np.ndenumerate(prediction[:-21]):
    # For some reason, index is a tuple, so to get the integer index, it is index[0]
    rev += val * (opnpr[index] - opnpr[index[0]+20])
    df['rev'][index] = val * (opnpr[index] - opnpr[index[0]+20])
print("Revenue from tanh, 20 day, 17 neurons, 3 layers:", rev)

```

VITA

JASDEEP SINGH BANGA

Candidate for the Degree of

Doctor of Philosophy

Thesis: MACHINE LEARNING: A POTENTIAL FORECASTING TOOL

Major Field: Agricultural Economics

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy/Education in Agricultural Economics at Oklahoma State University, Stillwater, Oklahoma in December, 2017.

Completed the requirements for the Master of Quantitative Financial Economics at Oklahoma State University, Stillwater, Oklahoma in December, 2017.

Completed the requirements for the Master of Business Administration at Punjab Agricultural University, Ludhiana, Punjab, India in May, 2006.

Completed the requirements for the Bachelor of Science (Agriculture) at Punjab Agricultural University, Ludhiana, Punjab, India in May, 2004.