Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 140 (2018) 383–392

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme: Cyber Physical Systems and Deep Learning, CAS 2018, 5 November – 7 November 2018, Chicago, Illinois, USA

Predicting the Future with Artificial Neural Network

Anifat Olawoyin*, Yangjuin Chen

University of Winnipeg, Winnipeg R3B2E9, CANADA

Abstract

Accurate prediction of future values of time series data is crucial for strategic decision making such as inventory management, budget planning, customer relationship management, marketing promotion, and efficient allocation of resources. However, time series prediction can be very challenging especially when there are elements of uncertainty including natural disaster, change in government policies and weather condition. In this research, four different *multilayer perceptron (MLP) artificial neural networks* have been discussed and compared with *Autoregressive Integrated Moving Average (ARIMA)* for this task. The models are evaluated using two statistical performance evaluation measures, *Root Mean Squared Error (RMSE)* and *coefficient of determination (R^2)*. The experimental result shows that a 4-layer *MLP* architecture using the *tanh* activation function in each of the hidden layer and a linear function in the output layer has the lowest prediction error and the highest coefficient of determination among the configured multilayer perceptron neural networks. In addition, comparative analysis of performance result reveals that the *multilayer perceptron neural network MLP* has a lower prediction error than the ARIMA model.

Keywords: Artificial Neural Network, ARIMA, Multilayer Perceptron, Time Series, Data Preprocessing

* Corresponding author. *E-mail address:* olawoyin-a@uwinnipeg.ca

1. Introduction

Machine learning is a domain of computational intelligence focusing on models that can iteratively learn from data to find hidden insights and patterns without being explicitly programmed. Learning methods can be supervised, semi-supervised or unsupervised. *The Artificial neural network (ANN)* is a form of the supervised machine learning model that mimics a biology nervous system. The *ANN* can detect patterns and trends that are too complex for human or other statistical models such as non-linearity in time series data to analyse. Real world applications of the *ANN* include pattern classification such as handwritten recognition, time series prediction, image compression, credit scoring for loan approval, and machine control, just to name a few.

This research designs a *Multilayer Perceptron neural network* for time series prediction and compares this with one of the traditional statistical time series prediction techniques known as the *Autoregressive Integrated Moving Average*, *ARIMA*. The study varies the number of hidden layers and investigates the best activation function for a set of data. In addition, this study explores the significance of pre-processing in time series prediction through data transformation by which a dataset having 5 attributes and 1,098,044 instances is converted to another dataset having 2 attributes and 366 instances by using aggregation, equal frequency binning and feature selection techniques.

The rest of this paper is organized as follows: Section 2 gives the background information and related works, Section 3 presents the main theoretical framework, Section 4 describes the implementation details, Section 5 is devoted to the experimental result and discussion. Finally, a short conclusion is set forth in Section 6.

2. Related Work

Artificial Neural network (ANN) has been applied to time series forecasting problems by many researchers. The study in [1] employed the Elman recurrent neural network (ERNN) with stochastic time effective functions for predicting price indices of stock markets. The ERNN can keep memory of recent events in predicting the future. The study in [2] used the Multilayer Feed Forward Neural Network (MLFFNN) and the Nonlinear Autoregressive models with the Exogenous Input (NARX) Neural Network to forecast exchange rates in a multivariate framework. Experimental findings indicated that the MLFFNN and NARX were more efficient when compared with the Generalized Autoregressive Conditional Heteroskedastic (GARCH) and Exponential Generalized Autoregressive Conditional Heteroskedastic (EGARCH).

Another advanced statistical technique for predicting future time series is the Autoregressive Integrated Moving Average (ARIMA) model, which assumes that the time series data are stationary. That is, the data are not time dependent. Thus, to use the ARIMA for time series prediction requires checking for stationarity; and a common approach to do this is to use the augmented Dickey-Fuller test (ADF) to test the presence of a unit root in a sample. Specifically, if the p-value is greater than 0.05, null hypothesis is accepted.

Besides, the hybrid techniques combining ARIMA and ANN have been shown to be successful by [3,4, 5, 6]. However, in [3] it is assumed that the linear and non-linear pattern can be separately modelled, their relationship is additive, and the residual from the linear model will contain only the non-linear pattern which may lead to performance degeneration for instance if the relationship is multiplicative. The empirical evidence from the study in [7] showed that such integrated approaches may not necessarily outperform the individual forecasting techniques. Although, the authors in [4] proposed a hybrid model to overcome the limitation of the traditional hybrid models and guarantee that the model will not be worse than using the individual ARIMA and artificial neural network, this assurance cannot be *true* in all cases. Hence, in this study we focus on the individual model comparison using the parking tickets dataset.

3.0 THEORETICAL FRAMEWORK

3.1 Artificial Neural Network (ANN)

The Artificial Neural network (ANN) is made up of a series of interconnected nodes that simulate individual neurons like a biological neural system. The ANN can be used for classification, pattern recognition and forecasting problem in situations of complex processes characterized by chaotic features such as trends and seasonality observed in parking ticket data, nonlinear and non-stationary in stock market data, chaotic features in ozone concentration measurements and weather related problems with non-linear relationships between inputs and the outputs.

The earlier ANN has only a single layer and follows a local learning rule known as the *Widrow-Hoff* or *Perceptron Learning Rule* (*PLP*) to update the weights associated with a network. A single layer neural network has no hidden layer; each input neuron has an associated weight and the output neuron uses a simple *Linear Threshold*

Unit (*LTU*) activation function. The activation function commonly used in most artificial network configurations is the *sigmoid* function because of its ability to combine linear, curvilinear and constant behaviors, as well as being smoothly differentiable.

The single perceptron output is defined by

$$\boldsymbol{t = w_0 + w_1 x_1 + w_2 x_2 \dots + w_m x_m} \quad (1)$$

where $\boldsymbol{t}$ = threshold, $\boldsymbol{w_1, w_2 \dots w_m}$ are the associated weight of the input attributes $\boldsymbol{x_1, x_2 \dots x_m}$.

The major drawbacks of a simple neural network include:

- Single neurons cannot solve complex tasks;
- It is restricted to linear calculations.
- Nonlinear features need to be generated by hand, an expensive operation.

The focus of this paper is the *Multilayer Perceptron* (*MLP*). A multi-layer perceptron is a feedforward neural network consisting of a set of inputs, one or more hidden layers and an output layer. The layers in MLP are fully connected such that neurons between adjacent layers are fully pairwise connected while neurons within a layer share no connection.

The input represents the raw data $(x_1, \dots, x_n)$ fed into the network. The raw data and the weight are fed into the hidden layer. The input to the hidden layer is thus given as

$$I = f(x) = \sum_{i=1}^{n}(x_i w_i) \quad (2)$$

The hidden layer is the processing unit where the learning occurs. The hidden layer transforms the values received from the input layer using an activation function. A commonly used activation function is the sigmoid function given as

$$\sigma = 1/(1 + e^{-x}) \quad (3)$$

Other activation functions are:

i. *tanh*(*x*)- non-linearity activation function is a scaled sigmoid function given as:

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 \quad (4)$$

tanh(*x*) can also be expressed in the form of sigmoid as $2\sigma(2x) - 1$.

ii. *Rectifier Linear unit* (*RELU*) is an activation function with a threshold of zero given as:

$$f(x) = \max(0, x) \quad (5)$$

The output of the hidden layer is given as:

$$H = f(A(I)) = f(A\big(f(x)\big) = f(A(\sum_{i=1}^{n}(x_i w_i)) \quad (6)$$

where *A* is the activation function. Assuming that the sigmoid gives

$$H = 1/(1 + e^{-\sum_{i=1}^{m} x_i w_i}) \quad (7)$$

The output layer receives the output and the associated weight of the hidden layer neurons as inputs. The output *Y* of the output layer assuming a sigmoid function is given as

$$Y = f\big(A(\sum_{j=1}^{m} h_j w_j)\big) \quad (8)$$

where h_j and w_j are the output and weight of individual neurons of the hidden layer.

The activation function of the output layer is commonly a linear function, and depending on the task, a *tanh* or a sigmoid function may be applicable.

A multilayer perceptron architecture having 2 hidden layers denoted as *2-layer multilayer perceptron* neural network is shown in Figure 2.

The main issue with a Multilayer Perceptron neural network is *weight adjustment* in the hidden layer which is necessary to reduce the error at the output layer. The weight adjustment in the hidden layer is achieved using backpropagation algorithm. The back propagation takes the sequence of training samples (time series data for this study):

$$(x_1, y_1),\ (x_2, y_2) \dots \dots ., (x_n, y_n)$$

as the input and produces a sequence of weights $(w_0, w_1, w_2 \ldots \ldots . w_n)$ starting from some initial weight w_0, usually chosen at random [4]. Generally, the backpropagation rule is given as:

$$\Delta \mathrm{w} = \mathrm{w} - w_{old} = -\eta \frac{\partial E(w)}{\partial w} = \eta\, \partial x \tag{9}$$

where w represents the weights, $E(w)$ is the cost function that measures how far the current network's output is from the desired one. $\partial E(w)/\partial w$ is the partial derivative of the cost function E that specifies the direction of the weight adjustment to reduce the error, η is the learning rate, measured as the number steps for each iteration of the weight update equation.

The weight change for the hidden layer is given as:

$$\Delta \mathrm{w} = \eta\, \partial_j x_i \tag{10}$$

where $\partial_j = o_j\,(1 - o_j)\sum w_{jz\,\partial_j}$.

The weight change for the output layer is given as:

$$\Delta \mathrm{w} = \eta\, \partial_z o_j \tag{11}$$

where $\partial_z = o_j\,(1 - o_j)(T - o_j)$, and T is the target output and o_j is the output.

The network is trained by adjusting the network weights as defined in equation 9 -11 above to minimize the output errors on a set of training data.

The training of a multilayer perceptron can be summarized as:

- Given a dataset D with $(x_1, \ldots, x_n)$ input and P patterns for the network to learn
- The network with n input units is fully connected to h nonlinear hidden layers via connection weight w_{ij} associated with each input unit.
- The hidden layer is fully connected to T output units via connection weight w_{ij} associated with each neuron in the hidden layer.
- The training is initiated with random initial weight for each neuron in the network.
- An appropriate error function $E(w_{jz})$, for instance the *Mean Square Error* (*MSE*) to minimize by the network is predetermined.
- The learning rate η is also predetermined.
- The weight associated with each neuron in the hidden layer and the output layer is updated using the equation: $\Delta \mathrm{w} = -\eta \frac{\partial E(w)}{\partial w}$ until the error function is minimized.

A momentum α is an inertia term used to diminish fluctuations of weight changes over consecutive iterations. Thus, the weight update equation becomes:

$$\Delta w = -\eta \frac{\partial E(w)}{\partial w} + \alpha\, \Delta w_{ij} \tag{12}$$

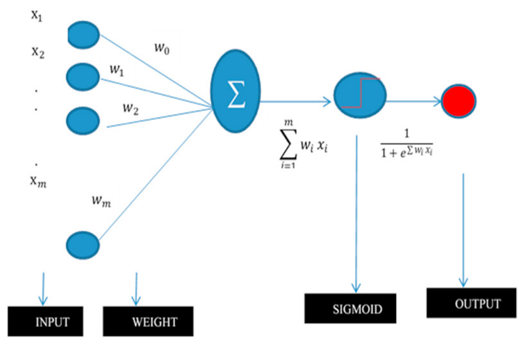

Figure 1: Single Layer Perceptron Neural Network

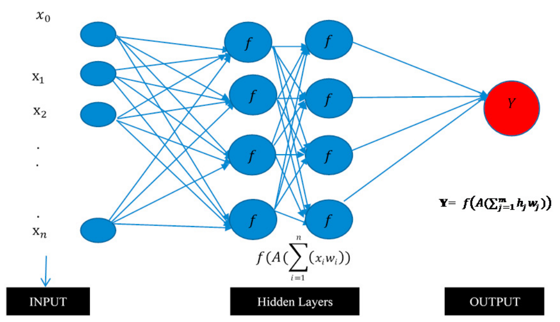
x_0
x_1
x_2
x_n
f
Y
$\mathbf{Y} = f(A(\sum_{j=1}^{m} h_j w_j))$
$f(A(\sum_{i=1}^{n}(x_i w_i))$
INPUT
Hidden Layers
OUTPUT

Figure 2: Multi-Layer Neural network (MLP)

3.2 The Auto Regressive Integrated Moving Average (ARIMA)

ARIMA is proposed by Box and Jenkins [2]. The model assumes that time series is stationary and follows the normal distribution. To achieve the notion of stationary in time series, the model subtracts an observation at time $\boldsymbol{t}$ from an observation at time $\boldsymbol{t}$ - 1. Here, the name 'ARIMA' stands for

i. Autoregressive, ***AR*** - the lag of the stationary time series data. ***AR*** is represented as ***p*** in the model.
ii. Integrated, ***I*** - a differencing transformation applied to time series to make it stationary. A stationary series is independent of observation time, represented as ***d*** in the model.
iii. Moving average, ***MA*** - the lag of the forecast errors and is represented as, ***q*** in the model.

Thus, a non -seasonal ARIMA model can be summarized as $ARIMA(p, d, q)$ where:

p is the number of autoregressive terms;

d is the number of non-seasonal differences;

q is the number of moving average terms.

$ARIMA(p, d, q)$ Forecasting equation is defined with respect to the number of differencing necessary to make the time series data stationary as follows:

Let *Y*= *original series*; *y = stationary series*; *d* = 0 (indicating *no difference*); then $y_t = Y_t$.

First difference, *d* = 1 then $y_t = Y_t - Y_{t-1}$.

Second Difference, *d* = 2 then $y_t = (Y_t - Y_{t-1}) - Y_{t-1} - Y_{t-2} = Y_t - 2Y_{t-1} + Y_{t-2}$[1].

To use ARIMA for time series prediction requires checking for stationarity, a common approach is to use the *augmented Dickey-Fuller test* (*ADF*). The ADF tests the presence of a unit root in a sample; if the *p*-value is greater than 0.05, null hypothesis is accepted. This research uses the *statsmodels* package in python to implement the ARIMA model for the dataset.

4.0 IMPLEMENTATION

4.1 Development Environment and Tools

All our experiments are performed on a 64-bit operating system. The processor is 2.4GHz Intel(R) core™i5 laptop, 8GB installed memory. Programming language is Python, and Development environment is Enthought Canopy. The used Machine learning tool is Scikit-learn [9], Keras libraries [10] with Pandas, NumPy, stats Models and Matplotlib.

4.2 Dataset

The dataset for this study is a set of parking contravention transactions updated monthly by the city of Winnipeg on open data government license available in [11]. The dataset has five attributes and over a million instances comprising of parking tickets issued between January 1st, 2010 and March 31st, 2017. For this paper, seven years' data (2010-2016) are used. The description and preview of the dataset is presented in Table 1 and table 2, respectively.

Table 1:Dataset Description

Dataset Name	Number of attributes	Number of Instances
Parking_Contravention_Citaitons.csv	5	1.09M

[1] Robert Nau Lecture notes on forecasting: Fuqua School of Business. Duke Universityhttp://people.duke.edu/~rnau/Slides_on_ARIMA_models--Robert_Nau.pdf

Table 2: Sample data

Issue Date	Ticket Number	Violation	Street	Location
12/13/2016 12:59:55 PM	70219201	01Meter Expired	Hargrave ST	(49.8884066, -97.142226)
12/13/2016 12:58:05 PM	74920668	05Overtime	Kenneth ST	(49.839005, -97.149891)
12/13/2016 12:53:09 PM	75508386	05Overtime	Girton BLVD	
12/13/2016 12:51:36 PM	73418686	01Meter Expired	Portage AVE	(49.89496, -97.136288)

In Table 2, the data types of the five attributes are as follows:
Issue Date – Timestamp;
Ticket Number – Transaction unique identifier;
Violation (offence) – Text;
Street – Text;
Location – (x, y) (coordinate).

4.3 Evaluation

The models are evaluated using the *root mean square error* (*RMSE*) and *coefficient of determination* (*R*²).

- The RMSE is the square root of mean square error, a risk metric corresponding to the expected value of the squared error loss function, defined as:

$$RMSE = \sqrt{\frac{1}{n}} \sum_{i=1}^{n} (\breve{\sigma}_t - \sigma_T)^2. \quad (13)$$

- The coefficient of determination is a measure of goodness of the model. It explains how well future samples are likely to be predicted by the model [4]. The value of R^2 can be negative or positive. A negative R^2 defines an arbitrary worse model, defined as

$$R^2 = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \text{ , where } \bar{y} = \frac{1}{n\sum_{i=0}^{n-1} y_i}. \quad (14)$$

4.4 Implementation Chart

This study pre-processes the dataset before designing the models. The output of the pre-processing serves as inputs to the two major models under consideration. The Multilayer Perceptron neural network is run first, and then followed by the ARIMA model. Comparative analysis of the results is done after the experiment. The working flow for this study is depicted in Figure 3.

5.0 EXPERIMENT AND RESULTS

5.1 Pre-processing

The pre-processing stage involves aggregation of the dataset into daily counts and weekly average is then calculated. Using a feature selection, the end-date of each week is taken as the period and the weekly average is the time series data. Thus, after the pre-processing stage the dataset has 2 attributes and 366 instances. Sample outputs of the pre-processing stage are presented in table 3.

The summary statistics for the dataset presented in table 4 shows that the minimum weekly mean between year 2010 and 2016 is 178 tickets while the maximum is 1341 tickets. The graph for the dataset presented in figure 4 shows that there is a spike in ticket numbers around January-February each year when the snow related violation tickets are issued.

5.2 ARIMA (p, d, q) Model

The assumption of the ARIMA model is that the time series is independent of time. Thus, the Augmented Dickey-Fuller (ADF) test is performed to test for stationarity in time series data. The ADF null hypothesis states that a sample data has unit root, and the data is not stationary while the alternative hypothesis states that the data is stationary. If the p-value > 0.05 the null hypothesis is accepted and if p-value < 0.05 the null hypothesis is rejected. The result of the Augmented Dickey-Fuller (ADF) presented in table 4 shows that the p-value < 0.05. Thus, the data is stationary, and the null hypotheses is rejected.

Since the time series is stationary, the value of parameter d is assumed to be zero ($d = 0$).

The significant of the pre-processing stage is observed in the result of the augmented Dickey-Fuller Test. The mean weekly ticket calculated at the pre-processing is a useful tool in transforming time series data to stationary.

Next, the log transformation is applied to the dataset for scaling and the ACF and PACF are plotted to determine the value of p and q parameters of the$ARIMA(p, d, q)$. The Autocorrelation Function (ACF) is defined as a measure of correlation between the time series with a lagged version of itself.

The Partial Autocorrelation Function (PACF) is defined as the correlation between a time series with a lagged version of itself after removing the effect already explained by previous lag. For instance, at lag 5, say X_5, the PACF is the correlation after removing the effect of x_1, x_2, x_3 and x_4.

Parameter p is defined as the point where the PACF crosses the upper confidence interval for the first time. From the result in Figure 5, $p = 1$. Parameter q is defined as the point where the PACF tail off. From the result in Figure 4, $q = 2$.

The experiment was tested with varying training percentage - 75%, 80%, 85% and 90% of the dataset and the remaining percentage (25%,20%,15%, and 10%) is used for testing. The $ARIMA(p, d, q)$ is implemented using $ARIMA(1, 0, 2)$, $ARIMA(2, 0, 2)$, $ARIMA(3, 0, 2)$ and $ARIMA(4, 0, 2)$, respectively. The result presented in table 5 shows performance degradation at 15% where RMSE increases for all models except $ARIMA(2,0,2)$; hence, the result obtained at 10% is considered to be overfitting while $ARIMA(2,0,2)$ using 20% testing (RMSE = 0.145; R^2 = 0.301) is taken as the baseline for comparison.

5.3 Multilayer Perceptron (MLP) Neural Network

Similar to the ARIMA model implementation, the MLP is tested with varying training and testing percentage ranging from 75% to 90% and the test data ranges from 10% to 25%. Four different MLP architecture were designed in this paper; a 2-layer with one neuron in the hidden layer denoted as 2H1, a 2-layer with four neurons in the hidden layer denoted as 2H4 to test the effect of increasing neuron, a 3-layer having four neurons in one hidden layer and one neuron in the other layer denoted as 3H41 and 4-layer having four neurons in the second layer, one neuron in the third layer and one neuron in the last layer denoted as 4H411.

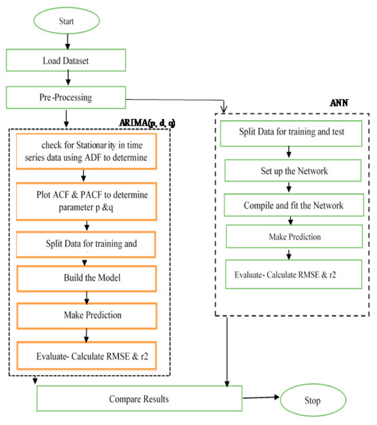
Start
Load Dataset
Pre-Processing
ARIMA(p, d, q)
check for Stationarity in time series data using ADF to determine
Plot ACF & PACF to determine parameter p &q
Split Data for training and
Build the Model
Make Prediction
Evaluate- Calculate RMSE & r2
ANN
Split Data for training and test
Set up the Network
Compile and fit the Network
Make Prediction
Evaluate- Calculate RMSE & r2
Compare Results
Stop

Figure 3: Implementation Chart

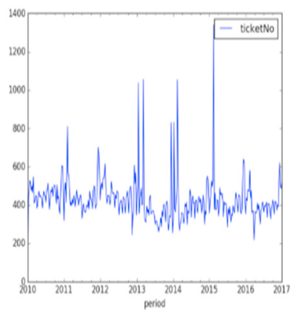

Figure 4: Dataset Trends Graph

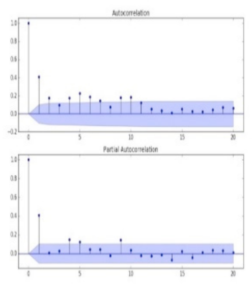

Figure 5: ACF and PACF Plot

Table 3: Pre-Processing Sample Output

Period	Weekly Mean
2010-01-03	178.33
2010-01-10	442.57
2010-01-17	483.57
2010-01-24	527.86
2010-01-31	513.43

Table 4. Pre-Processing Data Summary

Period	Weekly Mean
2010-01-03	178.33
2010-01-10	442.57
2010-01-17	483.57
2010-01-24	527.86
2010-01-31	513.43

Table 5: Augmented Dickey-Fuller (ADF) Test

ADF Test Result	
ADF Statistic:	-4.111741
p-value:	0.000926
Critical Values:	
1%	-3.449
5%	-2.870
10%	-2.571

Table 6: ARIMA (p, d, q) Results-(RMSE and R2)

Model	RMSE (Test %)				R^2 (Test %)			
	10%	15%	**20%**	25%	10%	15%	**20%**	25%
$ARIMA(1,0,2)$	0.114	0.146	**0.145**	0.144	0.28	0.286	**0.296**	0.26
$\boldsymbol{ARIMA(2,0,2)}$	**0.108**	**0.144**	**0.145**	**0.143**	**0.35**	**0.307**	**0.301**	**0.26**
$ARIMA(3,0,2)$	0.107	0.146	**0.145**	0.144	0.37	0.282	**0.300**	0.26
$ARIMA(4,0,2)$	0.108	0.148	**0.146**	0.145	0.35	0.263	**0.291**	0.25

All the models are separately trained for up to 1000 epochs using the *sigmoid* activation function and a comparison is made using the *tanh* activation function. The relationship between the *sigmoid* and *tanh* activation functions is stated in equation (6). The optimizer selected for the training is the *Stochastic Gradient Descent* (*SGD*) optimizer with a default learning rate of 0.01. The dataset is standardized using the *MixMaxScaler* function in the range (-1, 1). An attempt to use the *sigmoid* activation function in the output layer resulted in negative r^2 (-9.67); thus, a linear activation function is used for the output layer of all the architectures. The setup is presented in table 7.The loss function specified for all the models is the Mean Square Error (MSE), RMSE and R^2 are subsequently calculated for evaluation.

The result presented in Table 8 for the sigmoid activation function shows that a 2-layer with one neuron in the hidden layer has the best goodness of fit having correlation of determination R^2 of 0.61 and an error of 0.103. Adding more neuron to a layer does not improve performance as seen in the result for 2H1 and 2H4. Similarly, addition of layer to the network does not improve the prediction capability of the network. The root means square error, RMSE increases from 0.103 for 2H1 network to 0.104 for 3H41 while the coefficient of determination, R^2 increases to 0.66 for 3H41 network from 0.61 for 2H1 network.

The result from table 9 for the network designed using *tanh* activation function shows performance improvement when more layers are added to the network up to 4H411 (depicted in figure 7) where the best result is recorded. Further additions of layer beyond 4H411 add no value to the prediction capability and goodness of fit of the network.

The Comparative analysis of the result presented in table 10 and figure 6 shows that the 4H411 neural network designed with tanh activation function has the lowest error (RMSE=0.099) having an average prediction error of 57 tickets per week. A 2-layer MLP with one neuron in the hidden layer also has a better performance than ARIMA (2,0,2) having an average prediction error of 60 tickets per week.

Table 7: Multilayer Perceptron architecture

Epoch=1000, Optimizer=SGD, learning rate=0.01, loss function=MSE Standardization = MinMaxScaler; **Activation Function: Hidden Layer: Sigmoid/ tanh, output: Linear**		
Models	**No of Hidden Layer**	**No of Neuron in Hidden Layer**
2H1	1	1
2H4	1	4
3H41	2	4,1
4H411	3	4,1,1

Table 8: Sigmoid Function Evaluation Results (RMSE and R^2)

Model	RMSE (Test %)				R^2 (Test %)			
	10%	15%	**20%**	25%	10%	15%	**20%**	25%
2H1	0.085	0.104	**0.103**	0.102	0.60	0.61	**0.61**	0.65
2H4	0.105	0.115	**0.110**	0.115	0.30	0.30	**0.45**	0.30
3H41	0.081	0.103	**0.104**	0.101	0.73	0.62	**0.66**	0.67
4H411	0.102	0.128	**0.126**	0.125	-0.08	-0.19	**-0.01**	-0.09

Table 9: **Tanh Activation Function Evaluation Results (RMSE and R^2)**

Model	RMSE (Test %)				R^2 (Test %)			
	10%	15%	20%	25%	10%	15%	20%	25%
2H1	0.083	0.102	0.101	0.099	0.70	0.70	0.74	0.75
2H4	0.091	**0.107**	**0.101**	0.105	0.52	**0.69**	0.84	0.69
3H41	0.076	0.097	0.100	0.098	0.82	0.78	0.79	0.73
4H411	0.074	0.096	0.099	0.095	0.82	0.79	0.77	0.79

Table 10: **Comparative Analysis of Results (RMSE and R^2)**

Model	RMSE	R^2	RMSE (actual dataset)
MLP tan h (4H411)	**0.099**	**0.77**	**57.25**
MLP sigmoid (2H1)	0.103	0.61	59.69
ARIMA (2,0,2)	0.145	0.30	61.37

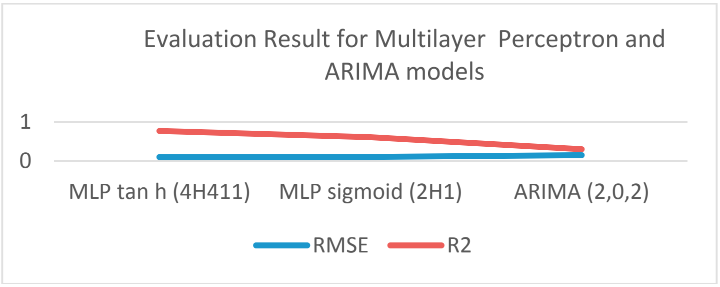

Figure 6: Comparison evaluation result (RMSE and R2)

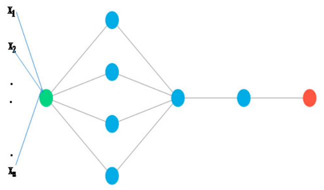

Figure 7: 4H411 MLP

6.0 CONCLUSION

The performance of the Multilayer Perceptron neural network and ARIMA models have been investigated in this research. Observations from the performance evaluation of the models revealed that the four MLP architectures designed using *tanh* activation function outperform the ARIMA model. Specifically, with the 4H411 model, they produce the best goodness of fit ($R^2 = 0.77$) and lowest prediction error (RMSE = 0.099). The effect of adding more layers on the performance of a multilayer perceptron neural network is also investigated. Using the sigmoid activation function, a 2-layer MLP having one neuron in the hidden layer has the best performance in term of prediction error (RMSE = 0.103) and the coefficient of determination ($R^2 = 0.61$) measures. The empirical evidence from this study indicates that adding more layers to a network configured using sigmoid function may not necessarily improve the predictive power of the network and may result in performance degeneration.

Like the sigmoid activation function, the *tanh* activation function also has a saturation effect, however, unlike the sigmoid, the output of the *tanh* activation function is zero-centered. Thus, adding layers to a network configured using the *tanh* activation function can improve the performance of a network as demonstrated in this study. From the result in Table 9, it can be observed that adding more layers reduces the prediction error and improves the goodness of fit of the network up to the 4-layer network (4H411).

In addition, pre-processing datasets is a necessity to some models like the ARIMA and MLP investigated in this study. The ARIMA model requires a stationary time series data. This is achieved by first aggregating the ticket transaction to daily counts and using equal weekly frequency to group the mean values and then apply the logarithm function to them. Standardization is a requirement for multilayer perceptron networks to remove bias that might be caused from wide variation in range of values of raw data during a training. From the summary of pre-processing stage in table 4, it can be observed that standardization is required since the minimum average ticket per week is 178 while the maximum is 1340. This study used the MinMaxScaler function of the Scikit-learn library to transform the dataset to a range [-1, 1].

Our experiments suggest that choosing a good activation function can significantly improve the performance of a multilayer perceptron neural network.

ACKNOWLEDGEMENTS

The first author would like to thank two anonymous referees for their helpful comments. Special thanks to Dr Sheela Ramanna and Dr. Sergio Camorlinga, University of Winnipeg, for their helpful comments at the initial stage of this work.

REFERENCES

[1] J. Wang, J. Wang, W. Fang, and H. Niu, Financial time series prediction using Elman recurrent random neural networks, Computational Intelligence and Neuroscience, vol. 2016, Article ID 4742515, 14 pages, 2016.

[2] Chaudhuri T. D. et al. Artificial Neural Network and Time Series Modeling Based Approach to Forecasting the Exchange Rate in a Multivariate Framework" Journal of Insurance and Financial Management, Vol. 1, Issue 5 (2016), pp 92-123.

[3] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159-175.

[4] Khashei, Mehdi, and Mehdi Bijari. "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting." *Applied Soft Computing* 11.2 (2011): 2664-2675.

[5] Babu, C. N., & Reddy, B. E. (2014). A moving-average filter based hybrid ARIMA–ANN model for forecasting time series data. *Applied Soft Computing*, *23*, 27-38.

[6] Khandelwal, I., Adhikari, R., & Verma, G. (2015). Time series forecasting using hybrid ARIMA and ANN models based on DWT decomposition. *Procedia Computer Science*, *48*, 173-179.

[7] Taskaya-Temizel, T., & Casey, M. C. (2005). A comparative study of autoregressive neural network hybrids. *Neural Networks*, *18*(5-6), 781-789.

[8] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature*. **323** (6088): 533–536.

[9] Scikit-learn Machine Learning in Python http://scikit-learn.org/stable/index.html

[10] Keras Deep Learning Documentation https://keras.io/

[11] City of Winnipeg Parking contravention dataset: https://data.winnipeg.ca/Parking/Parking-Contravention-Citations-/bhrt-29rb/data