An Auto Regressive Deep Learning Model for Sales Tax Forecasting from Multiple Short Time Series

1st Elham Khorasani Buxton
Department of Computer Science
University of Illinois at Springfield
esahe2@uis.edu

2nd Kenneth A. Kriz
Department of Public Administration
University of Illinois at Springfield
kkriz4@uis.edu

3rd Matthew Cremeens
Department of Computer Science
University of Illinois at Springfield
mcrem2@uis.edu

4th Kim Jay
Department of Computer Science
University of Illinois at Springfield
jkim754@uis.edu

***Abstract*—This study explores the application of deep learning to forecasting the state of Illinois sale tax receipts in ten categories: general merchandise, food, drinking and eating, apparel, furniture, building and hardware, automotive and filling stations, drugs and retail, agriculture and all others, and manufacturers. The state of Illinois has used traditional techniques of economic and tax receipt forecasting in order to project the amount of resources that it will have in order to finance its activities and debts. Such techniques are mostly linear and lack the ability to model more complex non-linear or long term dependencies. Recently, deep learning models have shown promising results in time series forecasting. In this study, we use two types of neural networks (a simple Multi-Layer Perceptron and a Long Short Term Memory network to forecast the state of Illinois sale tax receipts and compare the performance of both models against the more traditional autoregressive integrated moving Average model. Unfortunately, only limited tax receipt data is publicly made available by the state of Illinois which makes it particularly challenging to train a robust neural network model without overfitting. To address this data limitation, we propose to use a global model with an embedding layer for all ten tax categories. The empirical results show that the global Multi-Layer Perceptron model has the best performance in one step forecasting of Illinois sale tax receipts followed by the global Long Short Term memory model. On average, both neural network models outperformed the traditional Integraded Moving Average model.**

I. Introduction

The state of Illinois faces numerous financial difficulties. At the end of fiscal year 2017 (June 30), the state reported that its liabilities (the amount that it owes individuals or organizations outside of the state itself) totaled $161.2 billion more than its assets. To put this amount in perspective, this equates to a debt of over $12,500 borne by each person in the state. Out of the "larger" states (ones with population greater than 10,000,000), the state with the next largest debt burden (California) has $4,250 in per capita debt. Interest on state debt accounts for 3 percent of all revenues, and this amount does not take into account that the state is not making its required payments for pension benefits, essentially incurring more debt each year to finance its activities (all figures are from [1]). With the state holding so much debt, each fiscal decision that the state makes is magnified in importance. Those decisions are based on data on past, current, and most importantly future revenues and expenditures. The state has used traditional techniques of economic and revenue forecasting in order to project the amount of resources that the state will have in order to finance its activities and debts.

Numerous techniques for forecasting the economy and revenues in the medium-to-long-term have been developed. These include: qualitative techniques where forecasts of future values made by multiple sources are combined into a consensus forecast; naive quantitative forecasts where only past values of a variable are used to develop a forecast; parameterized quantitative forecasts using past values of a variable along with the current and past values of other variables; full systems models where a relatively complex model featuring multiple linkages between variables is estimated. Most of the models described above have been linear. Models such as Autoregressive Integrated Moving Average (ARIMA) have been used frequently in econometrics for forecasting future behavior based on past data [2]. However, they have been plagued by issues such as structural breaks in the data, excessive noise in past observations and instability of the models over time [3]. Over the last two decades more complex, non-linear machine learning models have proven themselves to have a superior predictive performance in several forecasting competitions [4, 5, 6]. These models typically convert a one-step or multiple-step forecasting problem (where past data is extrapolated to forecast one period or many periods ahead) to a supervised learning problem where each observation consists of the values of a variable in the previous time steps and the value of that variable in the next time step. A machine learning algorithm then learns a mapping between the previous time steps and the next time step. Such mapping can be used to forecast the value of a variable in a future time given its values in the past.

More recently, studies have shown that deep artificial neural networks outperform other types of machine learning algorithms in financial time series forecasting [7, 8, 9]. The advantages of using deep neural networks for forecasting problems

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00221

are their robustness to noise, ability to capture non-linear relationship between input and output variables and inherently supporting multivariate input and multi-step forecasts.

In this study we use a simple Multi-Layer Perceptron (MLP) and a recurrent neural network (in particular, Long Short Term Memory Network (LSTM)), for forecasting the state of Illinois sale tax receipts in different sale categories from the past tax receipt data. We compare the prediction performance of the neural network models with the more traditional ARIMA model.

Unfortunately, the state of Illinois has made only limited tax receipt data publicly available for each sales tax category (quarterly tax receipt is provided since 1999 for a total of 77 data points for each series). The lack of sufficient data makes it particularly challenging to train a robust neural network model without overfitting [10]. For this reason, we use the idea of transfer learning [11] and instead of training a separate neural network model per tax category, we train a single global model using the combined time series data from *all* tax categories. This allows the model to learn shared dependencies between time series.At the same time, we add an additional feature to represent the tax category in order to learn dependencies specific to each individual time series. This additional feature is passed through an embedding layer to reduce its dimensions.

II. Background

A. ARIMA model

ARIMA is a general linear model that combines *autoregression* with *differencing* and *moving average*. Autoregression (AR) means that the time lagged values of a variable are used as predictors to forecast the value of that variable in future.

A time series is said to be stationary if its statistical properties (such as mean, variance, and autocorrelation) are constant over time. Time series with trend and/or seasonality components are not stationary. Trend exists if there is a long term increase or decrease in the data. Seasonality exists when data is affected by seasonal factors such as the time of the year or the day of the week and exhibits repeated patterns within any fixed period. The Integrating step in ARIMA model computes the difference between consecutive observations to remove possible trends. The Moving Average (MA) step in ARIMA uses the residual errors in the previous time steps as predictors. The ARIMA model is characterized by three hyper-parameters (p, q, d) and is written as follows:

$$y_t = c + \sum_{i=1}^{p} \phi_i * y_{t-i} + \sum_{i=0}^{q} \theta_i * \epsilon_{t-i}$$

where c is a constant, y_t is the series obtained after differencing d times where d is the difference order, p is the order of autoregressive part (the number of lagged observations used in autoregression), q is the order of moving average (the number of previous error terms used in the regression) and ϵ_t is Gaussian white noise.

If a time series has a seasonal component, then either the seasonality is removed before applying the ARIMA model or a seasonal ARIMA model (SARIMA) is used which directly incorporate both seasonal and nonseasonal factors into a multiplicative model [12].

B. MLP model

A Multi-Layer Perceptron (aka feed forward neural network) consists of at least three layers of neurons where the neurons in each layer are connected to all of the neurons in the next layer. The first layer is the input layer and has the same number of neurons as the number of features in the dataset. The input neurons are identity functions forwarding the input signals to the next hidden layer. A MLP might have one or more hidden layers. Each neuron in a hidden layer takes the weighted sum of the outputs of the neurons in the previous layer and combines them using a non-linear activation function, such as (sigmoid, hyperbolic tangent, Relu, etc.). The outputs of the neurons in the final layer form the output of the neural network model. Formally, the output of a hidden neuron in a simple MLP model can be written as:

$$h^{(l+1)} = f(W^{(l)} h^{(l)} + b^{(l)})$$

where $h^{(l+1)}$ and $h^{(l)}$ are vectors denoting the outputs of the neurons in layers $l+1$ and l, respectively. $W^{(l)}$ is a matrix representing the weights of the links connecting the neurons in layer l to the neurons in layer $l+1$, and $b^{(l)}$ (called bias) is a vector of constant values associated with the neurons in layer l. Training a MLP involves using a gradient descent algorithm to learn the optimum values for the weight matrix and the bias vector in each layer.

C. LSTM model

LSTM is a type of recurrent neural network (RNN). RNNs are capable of modeling sequential data by including a feedback loop. In an RNN, the output of a neuron at time step t not only depends on its input at time step t but also its previous output at time step $t-1$. The vanilla RNN suffers from instability due to the vanishing gradient problem when the input sequence is long [13]. One of the most widely used models proposed to address this instability is the Long Short Term Memory (LSTM) model [14].

LSTM keeps a cell state consisting of three gates which control removing, adding and filtering the information in the cell state. A *forget gate* takes $\mathbf{h}_{t-1}$ (the output of the neuron at the previous time step) and x_t (the input at the current time step) and outputs a number between zero and one which is then multiplied by the old cell state and essentially decides which information in the cell state should pass through and which information should be forgotten.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

where σ is the sigmoid function. An *update gate* decides which new information should be stored in the cell state. This has two parts. First a sigmoid is applied to $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ to decide which value in the cell state should be updated. A hyperbolic tangent is also applied to create a vector $\tilde{\mathbf{c}}_t$ of new candidate values to be added to the cell state.

$$\mathbf{i}_t = \sigma(\mathbf{W}_i.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$$

Second, the cell state is updated by applying the forget gate to the old cell state and adding the new information from the update gate (that is, $\mathbf{i}_t * \tilde{\mathbf{c}}_t$)

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{c}}_t$$

Finally, an *output gate*, $\mathbf{o}_t$ decides what should be the output of the neuron $\mathbf{h}_t$, base on the previous output h_{t-1}, new input x_t, and the updated cell state, c_t:

$$\mathbf{o}_t = \sigma(\mathbf{W}_o.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t * tanh(\mathbf{c}_t)$$

III. Illinois sale tax Receipt Data

We used data from the Illinois Department of Revenue on sales tax receipts by Standard Industrial Classification (SIC) code [16]. This data is available quarterly going back to 1999 in ten tax categories, including general merchandise, food, drinking & eating, apparel, furniture, building & hardware, automotive & filling stations, drugs & retail, agriculture & all others, and manufacturers. There are a total of 77 time steps in each category.Using data at a finer level than most revenue forecasts (which consider only the total revenue received by the state) offers an ability to see changes more readily and capture trends. Tax receipt series for all categories are plotted in figure 1

IV. Data Preparation

Although, in principle, neural networks can capture seasonality and trend variations in time series data, they require a lot more data to do so. Hence, in practice, it often works better if time series data is transformed using seasonal and/or trend adjustments [17].Transformations such as logarithm can help stabalise the variance of a time series and differencing can help stabalising the mean of a time series, eliminating or reducing trend and seasonality [12]. A difference series is built from an original series by taking the difference of the lagged observations. If the series has a seasonal effect, the difference series is built by taking the difference between an observation and the previous observation from the same season.

Looking at figure 1, the tax receipt series for all categories show strong annual seasonal effects (that is, the pattern is repeated every four quarters). Before feeding the data to the deep learning model we take the logarithm of tax receipt values and make seasonal adjustments by taking the difference between each value and the value of the same quarter in previous year. That is,

$$y_t = log(r_t) - log(r_{t-4})$$

where r_t is the raw tax receipt at time step t and y_t is the transformed tax receipt. Once data is transformed, it was split into train and test sets. The last 12 quarters of each series was used for testing and the rest was used for training. One step forecasting is used to forecast the tax receipt data for one quarter ahead and the actual, rather than forecasted values, are then used for the next prediction in the forecasting horizon.

A *walk-forward validation* method (also called *rolling cross validation*) [18, 19] was used to evaluate each model on the test data. A walk-forward model is periodically retrained using all the data currently available. This means that in each fold the training data is expanded to include an additional data point from the test set. In the first fold, a model is trained on the original training data and is used to forecast the tax receipt for the next quarter. The forecast is validated by the first data point in the test set. In the second fold, the training data is expanded to include the first data point in the original test data and the model is retrained on this expanded set. The model is used to forecast the tax receipt for the next quarter and validated on the second data point in the original test set. This process continues until the original test set is exhausted. The forecasting error is then averaged over all the folds. This is explained in figure 2.

The training data in each fold is standardized to have zero mean and unit standard deviation. The Mean Absolute Percentage Error (MAPE) is used to measure the forecasting error on the test data in each fold:

$$MAPE = |\frac{\hat{r}_t - r_t}{r_t}|$$

where $\hat{r}_t$ is the forecasted tax receipt at time t (after inverting the scaling, seasonal differencing, and log transformations) and r_t is the actual raw tax receipt at time t.

V. Model Architecture

We propose to train a global model for all sales tax categories rather than a separate model per category. Since the tax receipt data is very limited for each time series (77 quarters per tax category), it is challenging to train even a shallow neural network without overfitting. As the time series data for different sales tax categories are related, it makes sense to train a global model for all categories. To do so, we add an additional feature indicating the sales tax category to which each tax receipt data belongs. This gives the network more data and allows it to learn the shared patterns between categories as well as the characteristics of each individual category. The tax category is a discrete feature with ten levels. Hence, a one-hot-encoding should be used to avoid imposing an artificial ordering on the categories.Using one-hot-encoding, the tax category for each observation is represented as a binary vector of size 10 with only one nonzero entry corresponding to that tax category. This representation is sparse and adds ten dimensions to the feature space. Hence, we add an embedding layer on top of the one-hot-encoded tax category to reduce the dimension of this feature, map it to a continuous space and capture the relationship between categories.

The embedding layer is essentially a set of linear neurons on top of the one hot encoded input vector where each neuron

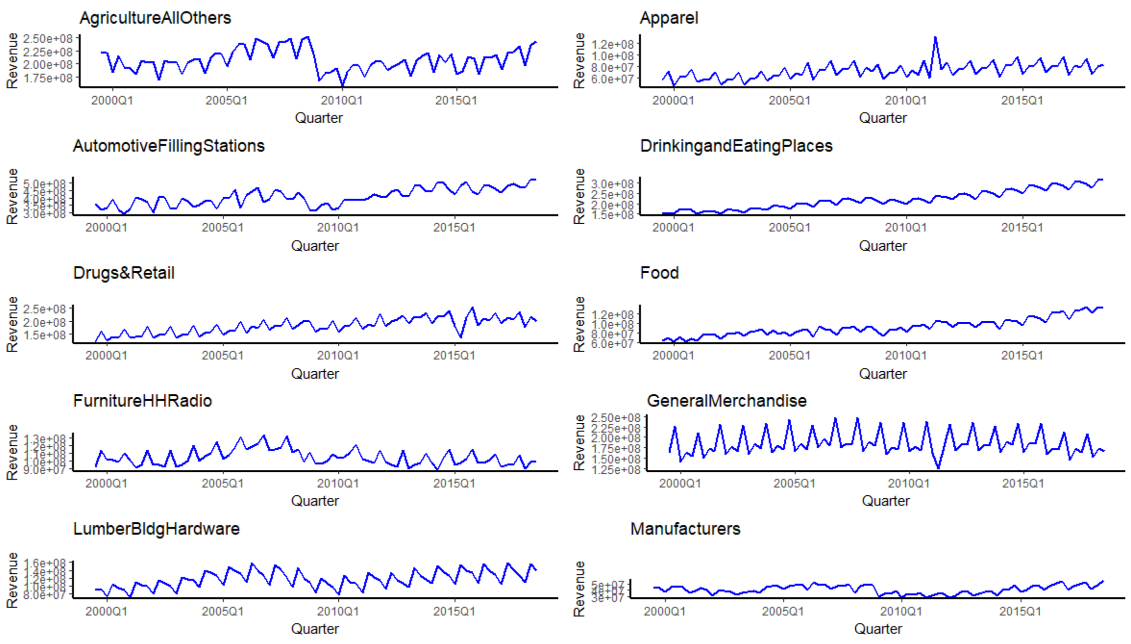

Fig. 1. Illinois State tax receipt series

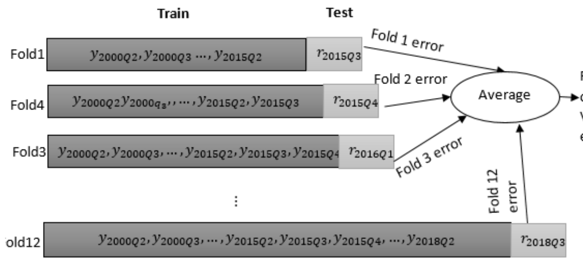
Train
Test
Fold1
Fold4
Fold3
old12
$y_{2000Q2}, y_{2000Q3} \cdots, y_{2015Q2}$
r_{2015Q3}
$y_{2000Q2} y_{2000q_3}, , \cdots, y_{2015Q2}, y_{2015Q3}$
r_{2015Q4}
$y_{2000Q2}, y_{2000Q3}, \cdots, y_{2015Q2}, y_{2015Q3}, y_{2015Q4}$
r_{2016Q1}
$y_{2000Q2}, y_{2000Q3}, \cdots, y_{2015Q2}, y_{2015Q3}, y_{2015Q4}, \cdots, y_{2018Q2}$
r_{2018Q3}
Fold 1 error
Fold 2 error
Fold 3 error
Fold 12 error
Average
Rolling cross Validation error

Fig. 2. One step rolling cross validation for Illinois sale tax receipt data. The training set is expanded in each fold by one quarter, and the forecasting is done for the quarter ahead

simply outputs the weighted sum of the one-hot-encoded vector. Since there is only one nonzero entry in the one-hot-encoded vector for each observation, an embedding neuron essentially outputs the weight for the category corresponding to the nonzero entry. Hence, the embeddings are simply the weights of the linear layer and can be learned the same way as the other parameters of the neural network [20].

If **c** is a ten-dimensional one-hot encoded binary vector with value one at position i (category i) and zero values everywhere else and $\mathbf{w_j}$ is the weight vector connecting the one-hot-encoded input layer to neuron j in the embedding layer, then the output of neuron j in the embedding layer e_j is formulated as:

$$e_j = \mathbf{w}_j \mathbf{c}^T = w_{ij}$$

where w_{ij} is the weight connecting neuron i with the non-zero entry in the one hot encoded input layer to neuron j in the embedding layer. This means that the embedding layer simply forwards the weights connected to it from the non-zero entry in the one-hot-encoded layer.

We start with a simple global MLP model for forecasting the sale tax receipts. This model has a dense hidden layer which takes the lagged values of tax receipt and the embeddings of the tax category as input. We used domain knowledge and random search to tune the hyper-parameters including the number of lagged values, the number of embeddings of the tax category, the number of neurons in the hidden layer and its dropout rate and activation function. We found that a time window of 8 lagged values, three embeddings for the tax category, 50 neurons in the hidden layer with RELU activation function and 0.2 dropout rate gives the best out-of-sample performance for our simple MLP model. The architecture of this model is shown in figure 3. The second model we tried was a global LSTM model as shown in figure 4. The picture shows the LSTM cells unrolled in 16 time steps. Similar to the MLP architecture, this means that overlapping time delayed windows are created for each time series and the forecasted tax receipt at time step t depends on the previous lagged time steps. However, unlike the MLP model where the input is flattened to remove the notion of sequence, LSTM cells preserve the sequence between the time steps in the input. The output of LSTM cells and the embeddings of the tax categories are then connected to a dense layer which outputs the forecasted tax receipt. A random search was used to tune model hyper parameters and the number of embeddings=2, number of LSTM cells=20, dropout rate=0.2, and recurrent dropout rate=0.1, were found to yield the best out-of-sample performance for this model architecture.

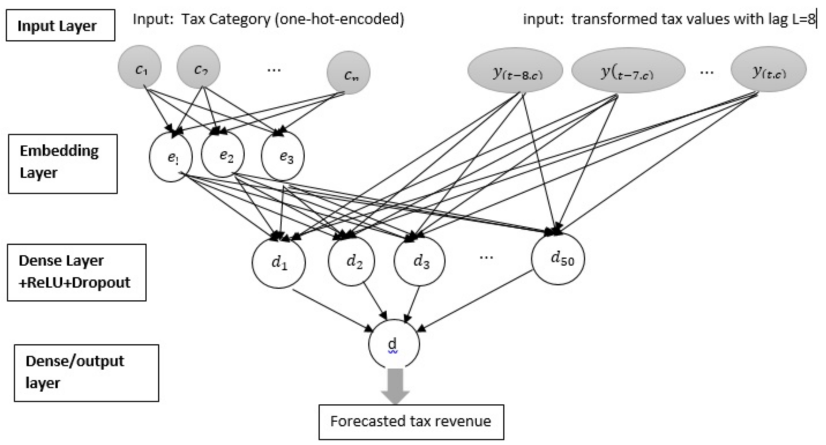

Fig. 3. MLP architecture for tax receipt forecasting

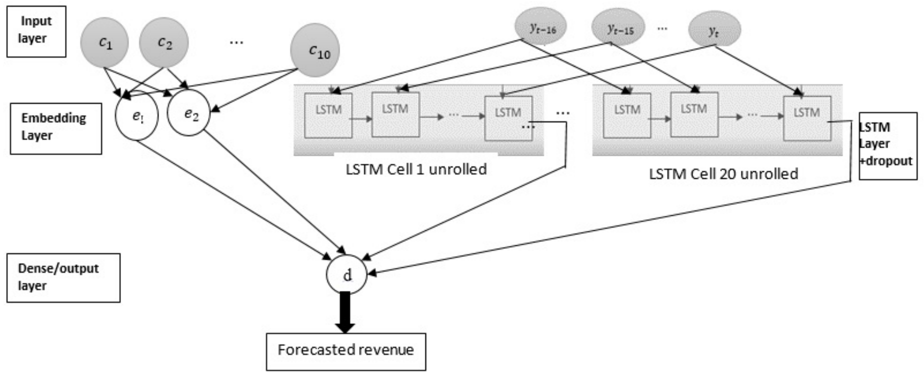

Fig. 4. LSTM architecture for tax receipt forecasting

VI. Experiment and Results

Keras was used with Tensorflow backend to implement the global MLP and the global LSTM models described above. The batchsize in both models were set to the size of the training window for each category in rolling cross validation. This make sense for both models as we want the weights to be updated after observing all data points from each category. In addition, the hidden state in each LSTM cell was reset for each tax category.

The MLP model was trained for 500 epochs and the LSTM model was trained for 3000 epochs in each fold. The rolling cross validation for each model is repeated 50 times to get a more robust estimate of the out-of-sample performance of each model and offset model sensitivity to random initialization of weights. The mean of MAPE over all runs for both neural network models are shown in table I. ARIMA was used as a benchmark to evaluate the out-of-sample performance of each neural network model.

Table I shows that on average, the global MLP model performs the best followed by the LSTM model. Both neural network models outperform the ARIMA model on average. This could be due to the fact that the neural network models can capture non-linear dependencies not captured by the ARIMA model. The global MLP model has a lower MAPE compared to the other two models for all but one category (drugs and retail) for which ARIMA performed better. The global LSTM model performed better than ARIMA on all but three categories (automotive & filling stations, drugs & retail, and manufacturer).

The variations of the MAPE across different runs for both neural network models are shown in table II. As can be seen in this table, the rolling cross validation MAPE of the global MLP model is more stable and has less variation across different runs. The MLP model is also more stable and much faster to train than the LSTM model as it does not need to maintain a hidden state and there are no dependencies between the outputs.

VII. Conclusion and Future Work

This study explored the application of deep learning to one step forecasting of Illinois sale tax receipts on ten different sales tax categories. An auto regressive shallow MLP model and a LSTM model were trained on Illinois sale tax receipt data and their out-of-sample performances were compared against each other and the traditional ARIMA model. To address the data limitation, the time series for all tax categories were combined and an additional feature was added to the data to show the tax category of each data point. This feature is embedded and added as input to both neural network models.

The empirical results showed that the MLP model trained on the combined data had the lowest average rolling cross validation MAPE followed by the LSTM model. The MLP model is also more stable and faster to train than the LSTM model. Both neural network models outperformed ARIMA on average.

As a future work of this study the authors are planning to explore sequence to sequence models such as transformers [21] or attention-based encoder-decoder models [22] for more medium to long-term forecasting of the tax receipt data (e.g., forecasts made for at least a period of four quarters or more). This will help the sate of Illinois to do longer-term planing and budgeting.

TABLE I
COMPARISON OF THE ROLLING CROSS VALIDATION MAPE OF THE MLP AND LSTM MODELS VERSUS THE ARIMA MODEL FOR FORECASTING ILLINOIS SALE TAX RECEIPTS

	Agriculture Others	**Apparel**	**Automotive Filling Stations**	**Drinking Eating Places**	**Drugs Retail**
MLP	3.07	2.79	2.54	0.5	11.48
LSTM	4.16	4.16	4.44	1.47	10.69
ARIMA	5.51	7.39	3.34	3.87	9.87
	Food	**Furniture HHRadio**	**General Merchandise**	**Lumber Bldg Hardware**	**Manufac.**
MLP	3.36	2.72	2.89	3.07	3.77
LSTM	3.91	3.04	5.34	4.0	6.80
ARIMA	8.03	5.06	9.50	6.85	3.91

	Mean across all categories
MLP	3.63
LSTM	4.80
ARIMA	6.33

TABLE II
THE VARIATIONS OF MAPE FOR THE GLOBAL MLP AND LSTM MODELS ACROSS 50 RUNS

		Agriculture AllOthers	**Apparel**	**Automotive Filling Stations**	**Drinking Eating Places**	**Drugs Retail**
MLP	std	427E-4	290E-4	274E-4	102E-4	895E-4
	min	2.95	2.71	2.49	0.57	11.31
	25%	3.04	2.77	2.53	0.58	11.43
	75%	3.10	2.81	2.56	0.59	11.53
	max	3.20	2.84	2.60	0.61	11.70
LSTM	std	586E-3	787E-3	960E-3	217E-3	115E-2
	min	3.09	2.97	3.09	1.02	8.83
	25%	3.75	3.49	3.69	1.32	9.80
	75%	4.64	4.72	4.88	1.56	11.25
	max	5.17	5.69	6.89	2.13	13.31
		Food	**Furniture HHRadio**	**General Merchandise**	**Lumber Bldg Hardware**	**Manufac.**
MLP	std	269E-4	272E-4	335E-4	268E-4	488E-4
	min	3.29	2.67	2.80	3.02	3.67
	25%	3.34	2.70	2.86	3.05	3.73
	75%	3.38	2.74	2.90	3.08	3.80
	max	3.41	2.78	2.98	3.13	3.9
LSTM	std	542E-3	498E-3	825E-3	419E-3	419E-3
	min	2.88	2.11	3.41	3.05	5.55
	25%	3.60	2.67	4.90	3.79	6.31
	75%	4.32	3.36	5.89	4.29	7.23
	max	5.16	4.06	6.60	4.72	8.21

REFERENCES

[1] S. A. Mendoza, “Illinois comprehensive annual financial report for the fiscal year ended june 30, 2017,” 2017.

[2] T. Mills, *Time Series Techniques for Economists*. Cambridge University Press, 1991. [Online]. Available: https://books.google.com/books?id=cNe3xrFg3PcC

[3] K. Kriz, “Long-term forecasting models,” in *Handbook of local government fiscal health*. Jones & Bartlett Publishers, 2012, pp. 125–155.

[4] G. Bontempi, S. Ben Taieb, and Y.-A. Le Borgne, *Machine Learning Strategies for Time Series Forecasting*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 62–77. [Online]. Available: https://doi.org/10.1007/978-3-642-36318-4_3

[5] S. Mullainathan and J. Spiess, “Machine learning: An applied econometric approach,” *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 87–106, May 2017. [Online]. Available: http://www.aeaweb.org/articles?id=10.1257/jep.31.2.87

[6] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, “An empirical comparison of machine learning models for time series forecasting,” *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010. [Online]. Available: https://doi.org/10.1080/07474938.2010.481556

[7] “Deep learning with long short-term memory networks for financial market predictions,” *European Journal of Operational Research*, vol. 270, no. 2, pp. 654 – 669, 2018.

[8] W. Bao, J. Yue, and Y. Rao, “A deep learning framework for financial time series using stacked autoencoders and long-short term memory,” *PLOS ONE*, vol. 12, no. 7, pp. 1–24, 07 2017.

[9] S. Siami-Namini and A. S. Namin, “Forecasting economics and financial time series: ARIMA vs. LSTM,” *CoRR*, vol. abs/1803.06386, 2018. [Online]. Available: http://arxiv.org/abs/1803.06386

[10] F. Chollet, *Deep Learning with Python*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2017.

[11] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 2010, pp. 242–264.

[12] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[13] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[15] C. Olah. (2015) Understanding lstm networks. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[16] (2018) Standard industrial classification (sic) code reporting. [Online]. Available: https://www.revenue.state.il.us/app/kob/index.jsp

[17] G. Zhang and M. Qi, “Neural network forecasting for seasonal and trend time series,” *European Journal of Operational Research*, vol. 160, no. 2, pp. 501 – 514, 2005, decision Support Systems in the Internet Age.

[18] I. Kaastra and M. Boyd, “Designing a neural network for forecasting financial and economic time series,” *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.

[19] M. Y. Hu, G. Zhang, C. X. Jiang, and B. E. Patuwo, “A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting,” *Decision Sciences*, vol. 30, no. 1, pp. 197–216, 1999.

[20] C. Guo and F. Berkhahn, “Entity embeddings of categorical variables,” *arXiv preprint arXiv:1604.06737*, 2016.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[22] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, “A dual-stage attention-based recurrent neural network for time series prediction,” *arXiv preprint arXiv:1704.02971*, 2017.