loxz digital

CPI Prediction Model v1.0

By Miu Lun (Andy) Lau, Data Scientist

Table of Contents

ABSTRACT

This model utilizes Long Short Term Memory (LSTM) as a way to predict the core consumer price index (cCPI). The prediction is utilized as a way to predict the inflation and changes in the common consumer product, and ultimately a way to judge the state of the economy. The audience for this model are economists and others interested in knowing the predictions with relatively high accuracy of where the CPI is headed. The model will be used internally until the model is accurate enough to use commercially. We will maintain the model internally until the predictions are close enough to actual results to feel confident about shipping the model to clients.

I . INTRODUCTION

The Consumer Price Index (CPI) is a measure of the change in the price of a basket of goods and services consumed by households. It is commonly used as a measure of inflation, as it reflects the changing purchasing power of consumers.
LSTM networks can potentially be used for predicting changes in the CPI over time. In this case, the input to the LSTM network would be a sequence of past CPI values, and the output would be a predicted future CPI value. The network can then be trained using supervised learning, where the training data includes both the input and the known correct output.

Overall, LSTM networks can be a useful tool for predicting changes in the CPI, and can provide accurate predictions even in the presence of long term dependencies and noisy or missing data. However, it is important to note that predicting the CPI is a complex task that involves many factors, and the accuracy of any prediction model will depend on the quality of the data and the skill of the modeler.

.

II. DATA SET

The dataset comes from St Louis Fred Economic Research [4]. We can extract many different economic data from this resource either on country (U.S.) based or state based. The dataset we utilized and compiled has a total of 12 features:

- Unemployment Rate
- Personal Saving Rate
- M2 (Money Supply (M1) + saving deposits)
- Real Disposable Personal Income
- Personal Consumption Expenditures • Real Broad Effective
- Exchange Rate
- Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity
- Federal Funds Effective Rate
- Total Construction Spending
- Industrial Production: Total Index
- Core Consumer Price Index

A. Data Engineering and Wangling

We removed any samples that are missing any data since not all features are tracked that far away back. Additionally, not all features in the data are sampled and therefore we did some features accumulations so that the values are average each month. The data is normalized using min-max normalization. In additional reports, we will find additional data to help the models overall accuracy. In this first report, we use this feature set as a baseline.

B. Feature Engineering

In order to see which features are useful for forecasting core CPI, we will be using the **Granger Causality Test**. One of the key assumptions of the Granger causality test is that the time series being tested are *stationary*, meaning that their statistical properties do not change over time. If the time series are not stationary, the results of the Granger causality test may be unreliable or misleading.

To check for stationarity in the time series being tested, the Granger causality test typically involves running a stationarity test, such as the **Augmented Dickey-Fuller test,** on the time series data. If the time series are found to be non-stationary, they may need to be transformed, such as by differencing or detrending, in order to make them stationary before the Granger causality test can be performed. Both the Granger Casuality and Augmented Dickey Fuller Tests are initiated in this initial report. Subsequent reports may or may not include these tests.

III. TOOLS REQUIRED

We are using a number of software and packages used frequently by the data science community. For machine learning, we are using Keras/Tensorflow [1]. Other common used packages for data parsing and visualizations such as Numpy, Matplotlib, Pandas, and Scipy. Ipywidgets is used for demonstration and selection of targets and other parameters. *BOTO3* is used for accessing and interacting with internal our *S3* data storage. Plotly [2] is used creating interactive figures to inferences with the APIs.

IV. MODEL ALGORITHMS

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies in data. Unlike traditional RNNs, which are limited by the short-term memory of the hidden layer neurons, LSTMs are able to remember information for long periods of time, allowing them to more effectively learn from data with long-term dependencies. This makes LSTMs particularly well-suited for tasks like language translation and speech recognition, where understanding the context and meaning of words is crucial for achieving good performance.

LSTM networks can be used for univariate time series forecasting, which involves using historical data for a single variable to predict future values of that variable. In this case, the input to the LSTM network would be a sequence of past values for the variable, and the output would be a predicted future value. The network can then be trained using supervised learning, where the training data includes both the input and the known correct output. Once trained, the LSTM network can be used to make predictions on new data.

One advantage of using LSTM networks for univariate time series forecasting is that they are able to learn and model long-term dependencies in the data, which can be useful for making accurate predictions. For example, if there is a seasonal pattern in the data, such as an increase in demand for a product during the holiday season, an LSTM network can learn and incorporate this information into its predictions. Additionally, LSTM networks are able to handle noisy or missing data, making them robust to real-world data imperfections.

In addition to univariate time series forecasting, LSTM networks can also be used for multivariate time series forecasting, which involves using historical data for multiple variables to predict future values of those variables. In this case, the input to the LSTM network would be a sequence of past values for each variable, and the output would be a predicted future value for each variable. The network can then be trained using supervised learning, where the training data includes both the input and the known correct output.

Another advantage of using LSTM networks for multivariate time series forecasting is that they are able to learn and model the relationships between the different variables, which can be useful for making accurate predictions. For example, if there is a strong relationship between the demand for a product and the price of the product, an LSTM network can learn and incorporate this information into its predictions. Additionally, LSTM networks are able to handle noisy or missing data, making them robust to real-world data imperfections.

Overall, LSTM networks can be a powerful tool for both univariate and multivariate time series forecasting, and can provide accurate predictions even in the presence of long-term dependencies and noisy or missing data.

We have constructed two models using LSTM networks, one where it inputs a univariate dataset and another where it incorporates other economic metrics as inputs. For the single univariate model, it is constructed using a single layer of LSTM with **2-100 neurons.** For the multivariate model, it utilizes 2 layers of 100 neurons, and a dropout layer of 20% – The inputs of the models is set to utilize 12 data points (month) as input and returns a single datapoint.

1) Univariate Model: For the univariate model, we constructed a simple model with a single LSTM layer, no dropout regularization and no Early Stopping. The number of layer is chosen because of the complexity of the problem. The number of neurons that will be used is 100, for high dimensionality (so the model can capture the underlying trends). Most parameters are chosen through a fine-tuning and trial-and-error approach.

No dropout regularization were used due to the simple complexity nature (there is 1 layer only, and adding dropout would cause important information to be lost) of the built neural network and dataset size. Dropout regularization at optimal levels 0.1/0.2 for LSTM were trialed and significantly decreased the performance of prediction on the test set. Dropout erases important context information, especially in this problem with limited data & time steps and layer height. Furthermore, train and validation loss are carefully monitored for any potential overfitting. Similarly, a small learning rate of 0.001 is used due to size of the neural network and small dataset. The overall batch size is set as to be **100 epochs.** Batch size fine-tuning is done based on the observation of model's performance.

2) Multivariate Model: The initial model was trialed on this multivariate with similar parameters as the univariate model but we observed poor performance. The model was adjusted to run for a total of 250 epochs with default batch size of 32 and stacking an additional LSTM layer. A GridSearchCV attempt was done before to find the optimal epochs and batch size.

The model showed significant overfitting, thus regularization with EarlyStopping and Dropout were implemented. The final Dropout was set at 20% and added between the rec and Dense fully output layer. The EarlyStopping patience was set at 50. The hyper-parameters were also tuned through manual experimental runs observing the performance and outputs result.

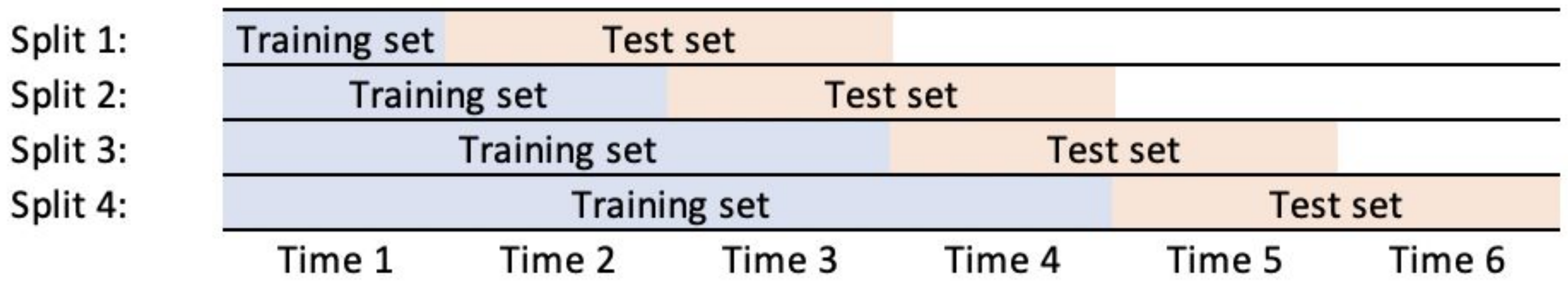

Figure 1. Example of walk forward validations

V. MODEL ASSUMPTIONS AND VALIDATIONS

We employed a training/validation ratio of around 80/20%, respectively. Since our predictions comes from a time-series predictions, we utilized a walk forward validations so that the training period (year 1-5) is used to validate the results. In a walk forward validation process, a number of periods is used as the training starting points. However, time-series predictions tends to become less accurate as it is drifted off. Therefore, as new data becomes available on a monthly basis, and the model is retrained to allow for accurate predictions. An example of walk forward validation is shown below in fig 1. The amount of time to be included in the training set is still to be determined and required additional research in the future.

A. Classifications Metrics

We utilizes two metrics to evaluate the model's forecasting power, Mean Squared Error (MSE) and Mean Model Error (MME). Additionally, we also computed the accuracy of the final predictions to understand our model's understanding.

VI. MODEL TUNING AND PERFORMANCE

We utilize the *ADAM* [3] as the optimizer and *mean square error* as the loss function. The models were run for a total of 100 epochs and batch size of 20. To prevent over-fitting, early termination was performed which monitor the validation loss. Learning rate is also automatically adjusted based on the changes due to plateau. Cross validation was employed with four different splits with each split containing a different amount of training set. Due to the amount of training required, the parameters optimizations were performed using Amazon SageMaker SDK [5, 6], which utilizes a Bayesian related search technique to discover the optimal hyper-parameters set. Sagemaker allows 3 for various selection when dealing with hyper parameters optimizations, such as *Continuous Parameters*, *IntegerParameters*, and *CategoricalParameters*. Afterward, we utilized regex to extract the metrics from AWS Cloudwatch and compute the optimal hyper-parameters.

VII. MODEL RESULT

For the univariate model, the training and validation loss were observed and shown below in Fig 2. The validation loss shows significant spike and fluctuations due to the small batch size being passed in. This shows that the data were slightly overfit.

Using the model we fitted, we tested against our data and predicted the next year of CPI. The result is shown below in Fig 3. As shown, we can observe that the univariate LSTM is able to approximate the shape of the CPI, but suffers from some under estimation in the CPI value. For November 2022, it predicts a CPI value **299.29** basis points, which means the forecasted U.S. CPI changes YOY is **5.69**%.

For the multivariate result, the training and validation loss were observed and shown in Fig 4.

Compared to the univariate model, the multivariate model experiences less spike fluctuations and can be attributed to increased batch size and early stopping which prevents over-fitting. Overall, this results in a lower mean error rate. The result of the multivariate model is shown in *fig 5*. For November 2022, the multivariate model predicts a CPI basis point of **299.46** basis points, corresponding to a CPI changes YOY of **5.75%**.

Compared to the actual YOY CPI changes presented of **7.1%**. We can conclude that both of our models (univariate, multivariate) underestimate compared to the actual YOY CPI. However, we can observe that both of our models were able to capture the shape of the CPI curve. Given this is our initial interpretation on CPI data, we have work to do, but are capturing the trend. We believe later reports will get closer to the actual number. We will continue to refine the data, and accelerate the learning curve to get as close to actual results as possible. Then when commercially available based on lower mean error rates, we can publish confidently.

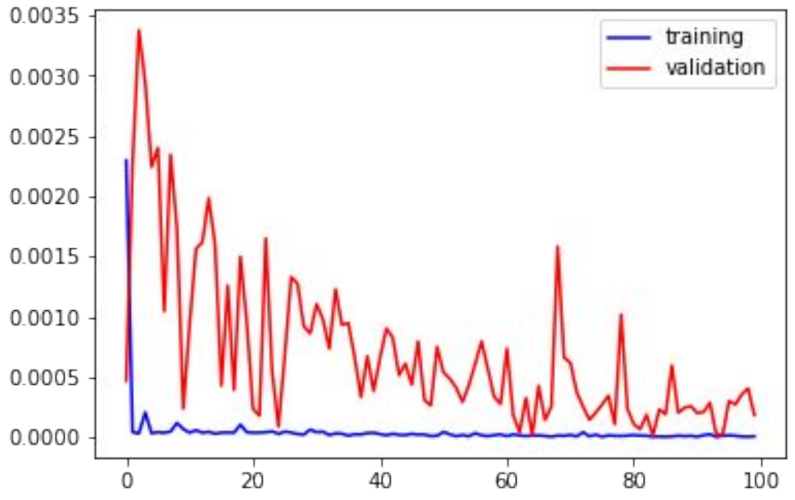

Figure 2. Univariate Training and Validation Loss

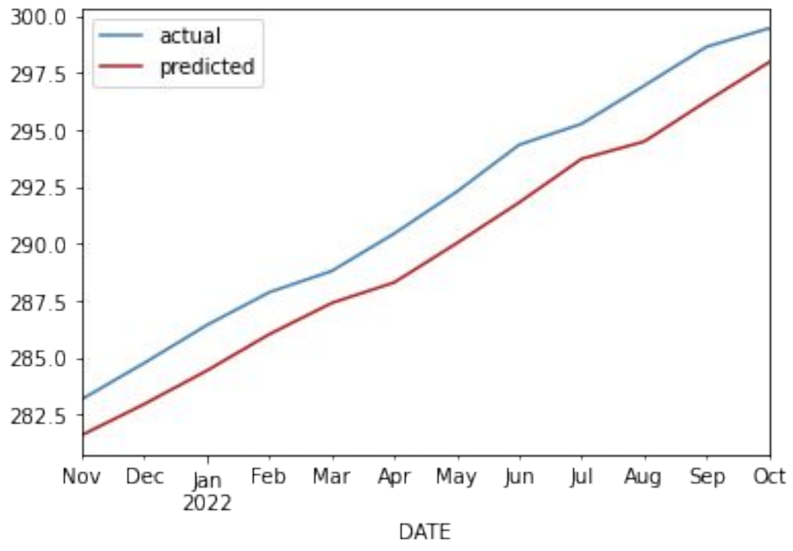

Figure 3. Univariate result for 2022 based on walk forward predictions

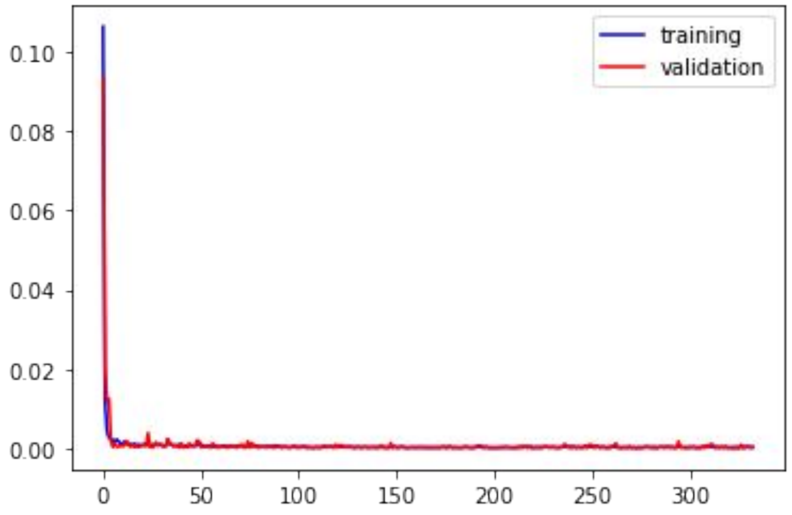

Figure 4. Multivariate Training and Validation Loss

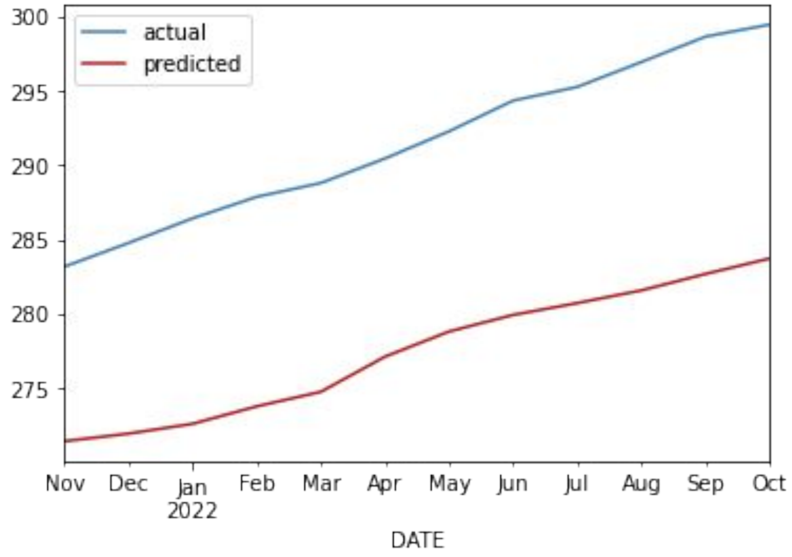

Figure 5. Multivariate Result

VIII. DATA SCIENTIST OWN CONCLUSIONS

There are many improvements still need to be made, but given the goal of this report is to demonstrate the first working product resulted from this working model for helping economic forecasts. There are still many area of improvements mentioned below in the next sections such as more labels and classes to the dataset, improvement to the architectures and many others. Additionally, it seems that the multivariate model is performing worse than univariate, but it is due to the additional features we have included. Future work includes quantifying the contributions from each feature on the prediction quality, and therefore we can include better and faster predictions.

IX. CONCLUSIONS AND FUTURE

CONSIDERATIONS Even though LSTM is very powerful in predictions for time-series problems, it can be rather difficult in training since the structure of RNN contains a large number of variables. In the future we might utilize transformer based neural networks which are currently state of the art in NLP. Transformer performs much faster than LSTM since the information is digested all at once. Moreover, it's rather difficult to perform transfer learning in LSTM networks, but it has shown since the CNN weights.

We only utilized some of the features that are available in the Fred database. In the future, including more relevant features or contributing factors might increase the model accuracy. Additionally, future versions of the model involve significant reduction in the amount of temporal period.

REFERENCES

[1] Francois Chollet et al. Keras. 2015. URL: https: //github.com/fchollet/keras.

[2] Plotly Technologies Inc. Collaborative data science. 2015. URL: https://plot.ly.

[3] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: arXiv preprint arXiv:1412.6980 (2014).

[4] Michael W McCracken and Serena Ng. “FRED MD: A monthly database for macroeconomic research”. In: Journal of Business & Economic Statistics 34.4 (2016), pp. 574–589.

[5] Valerio Perrone et al. “Amazon Sage Maker automatic model tuning: Scalable black-box optimization”. In: arXiv preprint arXiv:2012.08489 (2020).

[6] Valerio Perrone et al. “Amazon sagemaker automatic model tuning: Scalable gradient-free optimization”. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021, pp. 3463–3471.