Final Project
IE 7860
Alex Neuman and Joel Woznicki

Intro

The goal of this project is to create a model that can predict movement for a portfolio of globally traded currencies. In this case, our model choice is an important piece of determining the success of the project. We will benchmark our performance against a naïve approach, using a simple statistic such as the prior time period's price as our predicted value.

Foreign exchange markets average over $5 trillion per day (Bank for International Settlements, 2016). As the most liquid market in global finance, it facilitates trade between countries and investment from foreign entities.

An important aspect in considering our approach is the efficient market hypothesis. Whether or not foreign exchange markets behave under the efficient market hypothesis is up for debate (LeBaron, 1996), but the advances in computing suggest gains can be made in this space (Jingtao Yao, 2000).

Approach

Time series data is well researched from a forecasting perspective, and model choice will greatly impact our results. Within each model, architecture and hyperparameter optimization can further improve results. Along with the prices per time period, other variables are present to help explain increased or decreased price. In order to understand the data, each variable is plotted over time. A distribution of the values for each variable is plotted as well. To understand the time-series dependencies, autocorrelation and partial autocorrelation are plotted.

Our initial approach included multiple technical and functional indicators, but those results proved poor and have been discarded. The following analyses focus solely on predicting price based on prior price values.

Prices vary based on multiple factors that can be assigned to three groups:

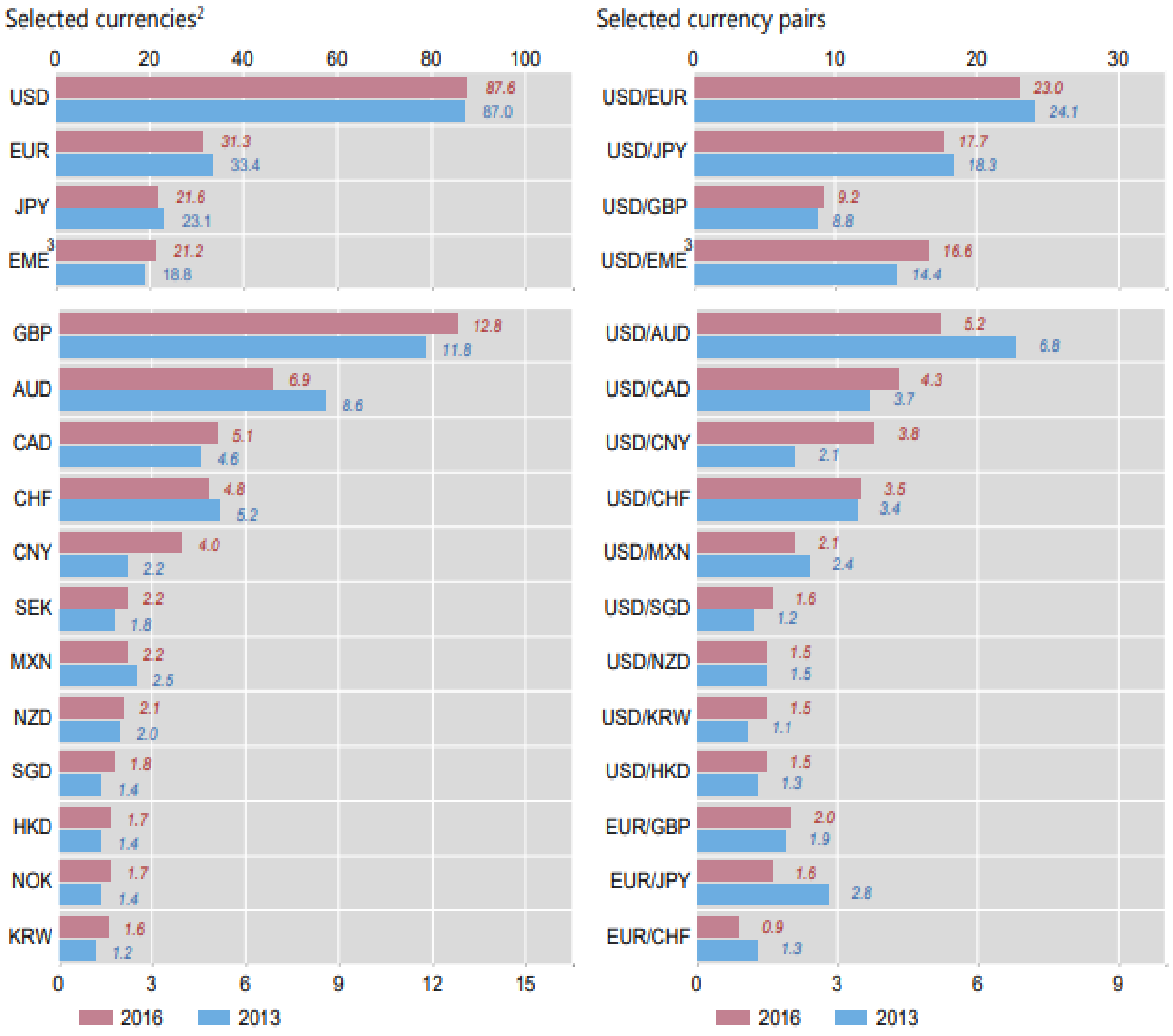
1) Economic indicators within a country
2) Political impact on currency price manipulation
3) Expectation of future prices based on those political and economic factors

Our research found that including these features masks the true underlying function from our model. Attempts at including additional features fails to include everything needed to make an accurate prediction.

Currency prices are calculated with respect to the US Dollar. The top traded currencies with respect to the US Dollar are Euro (EUR), Japanese Yen (JPY), Great British Pound (GBP), Australian Dollars (AUS), and Canadian Dollar (CAD). We choose these top five as our targeted currencies. We also model Swiss Francs (CHF), a relatively stable currency in the foreign exchange markets. To increase the degree of generalization outside of the top traded currencies, we model South African Rand (ZAR) as an emerging market currency. Lessons learned here can be applied to lower volume currencies. The difficulty of our problem is shown especially with South African Rand.

Foreign exchange market turnover by currency and currency pairs

Net-net basis,[1] daily averages in April, in per cent Graph 1



[1] Adjusted for local and cross-border inter-dealer double-counting. [2] As two currencies are involved in each transaction, the sum of shares in individual currencies will total 200%. [3] Emerging market currencies.

Source: BIS Triennial Central Bank Survey. For additional data by currency and currency pairs, see Tables 2 and 3 on pages 10 and 11.

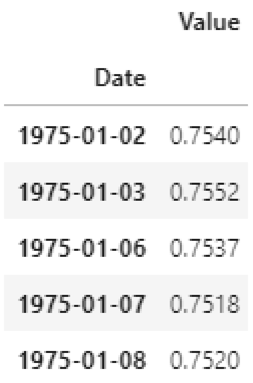
Data

All currency data is sourced from the Federal Reserve, and is pulled using the Quandl API. The Quandl API allows for easy access to the data in python and other sources. Via the API, we update our data source at each run to ensure the most up to date information is used in our models.

Data represents currencies at a given date, representing noon buying rates in New York for cable transfers payable in the listed currencies (Board of Govenors of the Federal Reserve, 2019). While currency markets never close, our data represents only trading days available from US brokers. This is representative of the data needed to generate a trading strategy.

Data begins on January 1, 1975 for all currencies. Data is up through April 28, 2019.

Here is a sample for Australian Dollars:



	Value
Date	
1975-01-02	0.7540
1975-01-03	0.7552
1975-01-06	0.7537
1975-01-07	0.7518
1975-01-08	0.7520

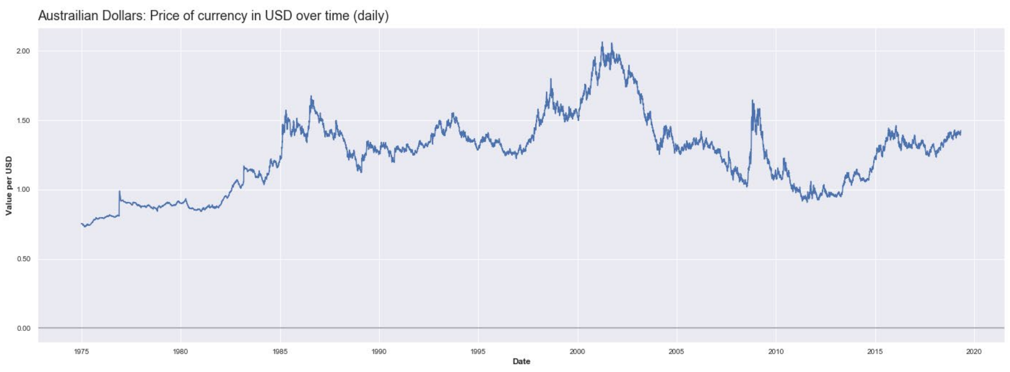
Analysis

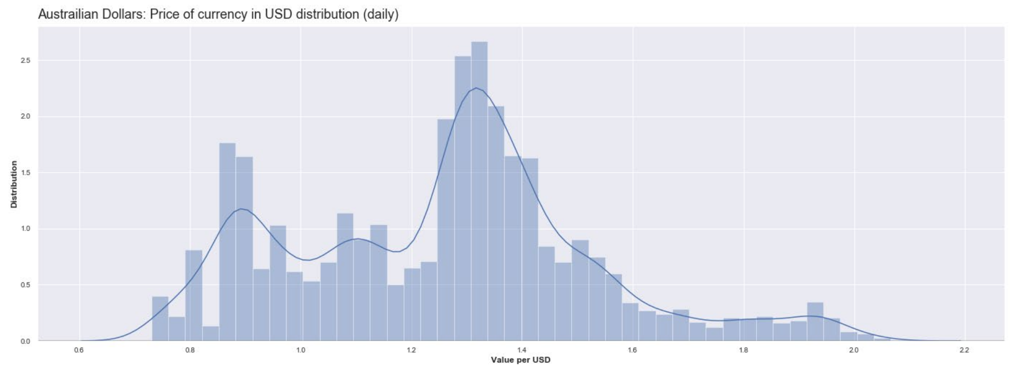
What follows is the exploratory data analysis on price for each of our six currencies, with plots of the value of the currency, the distribution of the value, the autocorrelation and partial autocorrelation, and the seasonal decomposition.

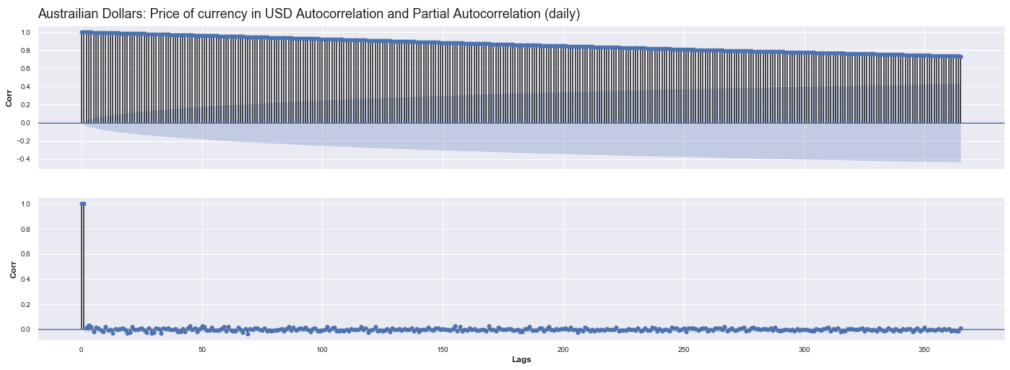
In general, the currencies exhibit growth during periods of recession in the United States. The partial autocorrelation shows that the prior day is the biggest predictor of the next day value, highlighting the value of the naïve model. The seasonal decomposition shows nothing of consequence.

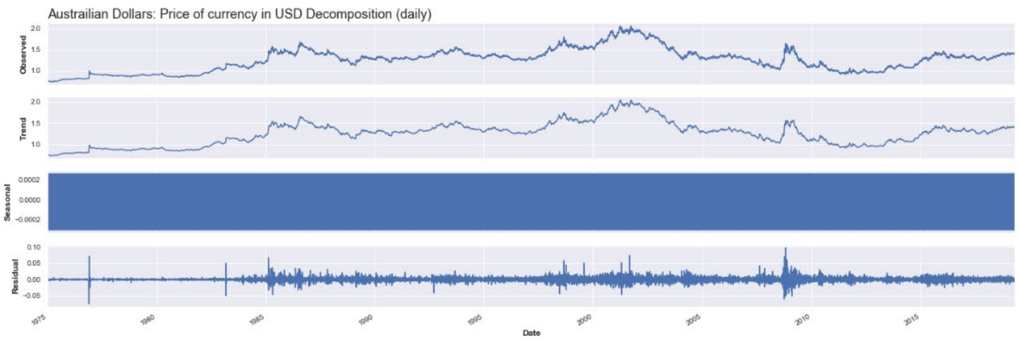
1) **Australian Dollars**

Australian dollars began at 0.75 USD, peaked between 2000 and 2005 at above 2.0, and jumped during the 2008 United States recession. It currently sits below 1.5 USD.



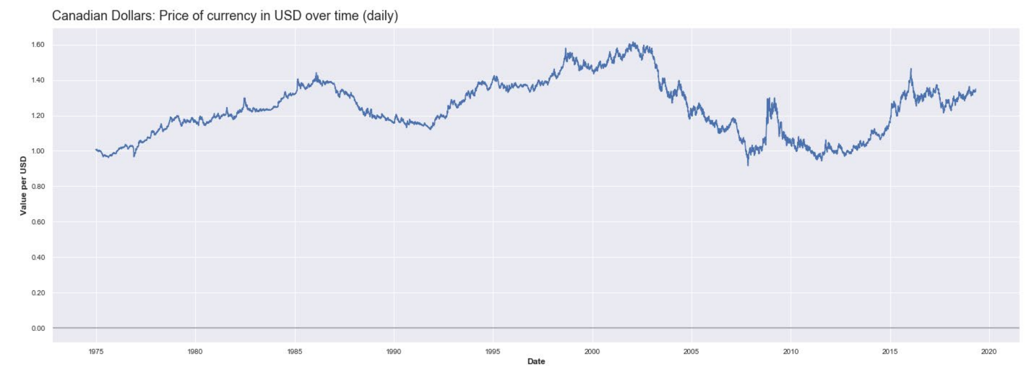


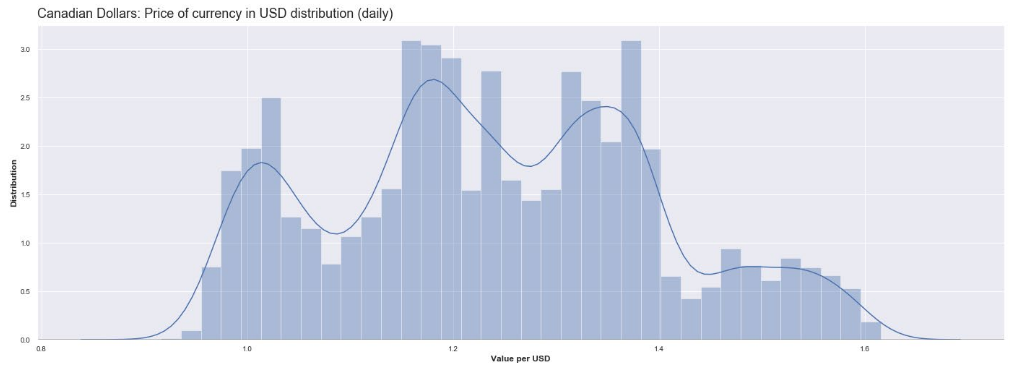


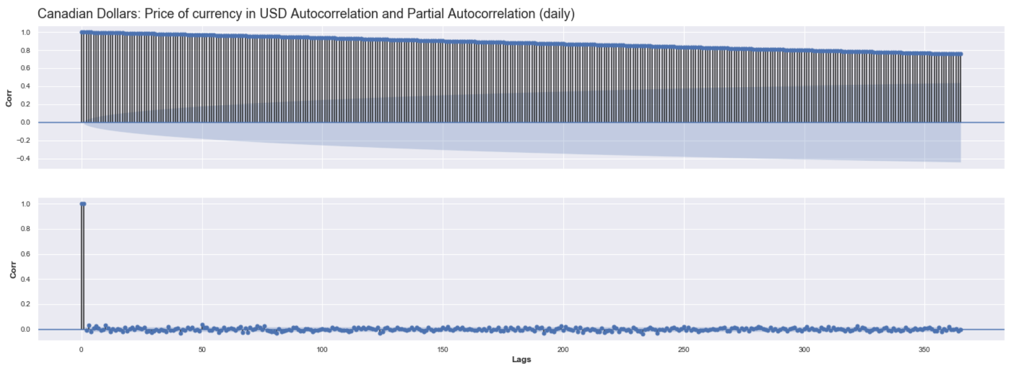


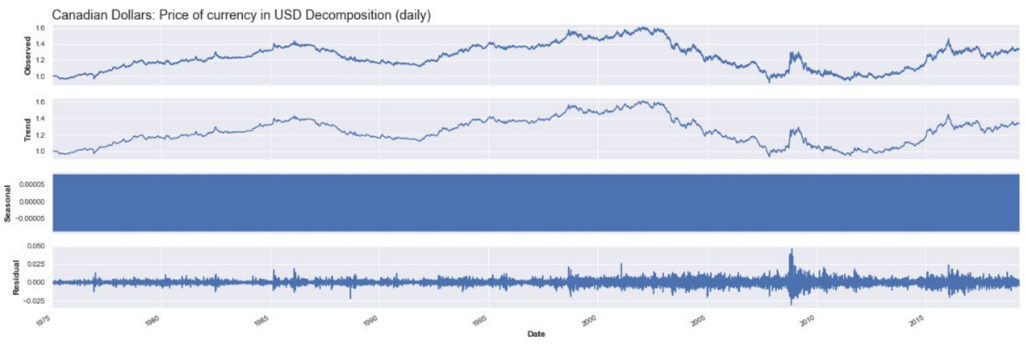
2) Canadian Dollars

Canadian Dollars began at 1.0 USD. Like Australian dollars, it peaked between 2000 and 2005 at above 1.6 and jumped during the 2009 recession. It currently sits at 1.3 USD.



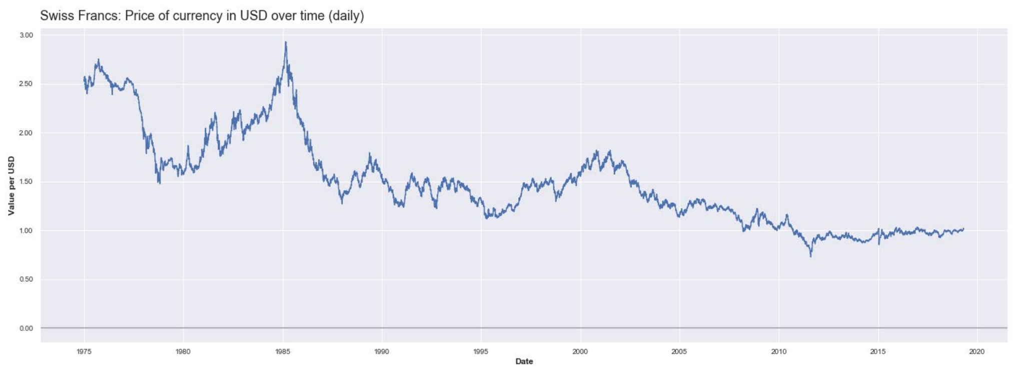


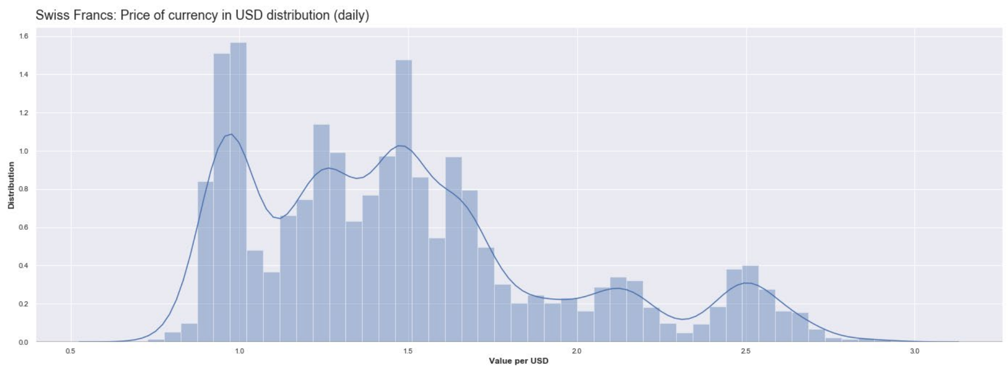


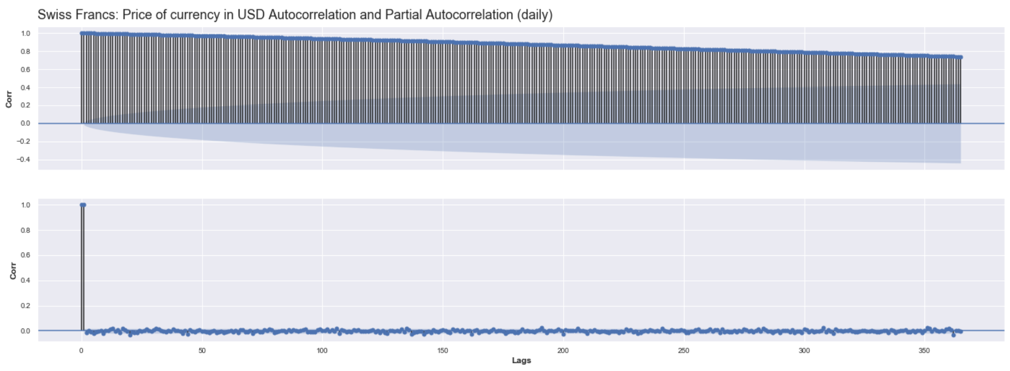


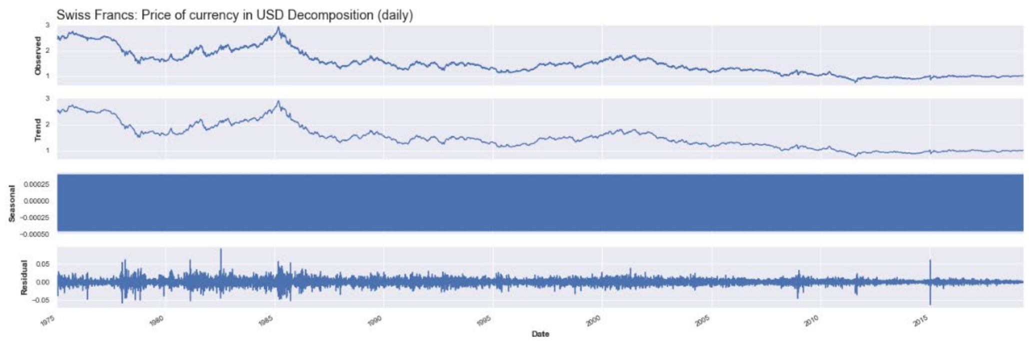
3) Swiss Franc

Swiss Francs began at 2.5 USD. It peaked in 1985 at almost 3.0 USD. There was no jump during the recession, and it currently sits at almost 1.0 USD. The recent consistency should be noted.



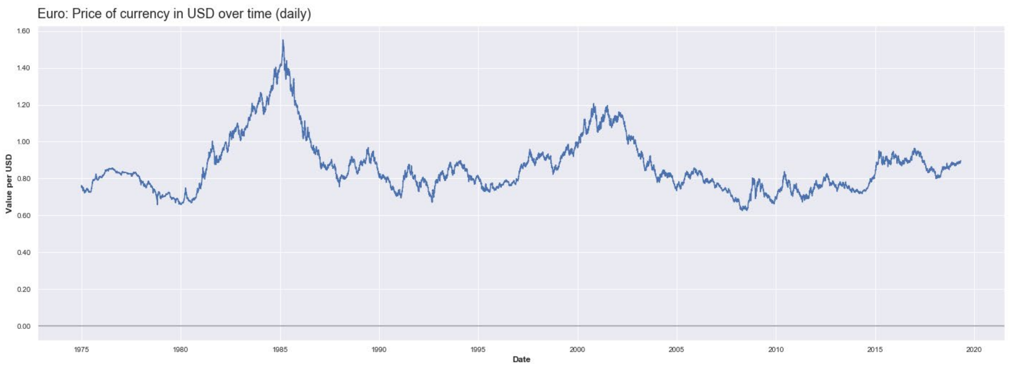


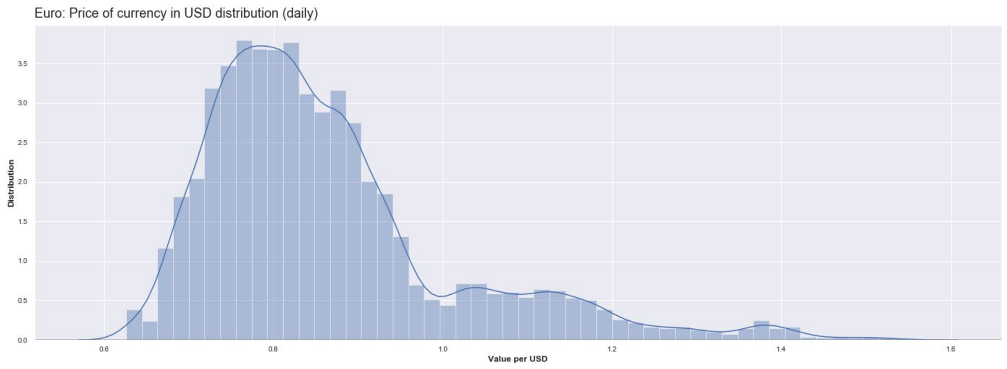


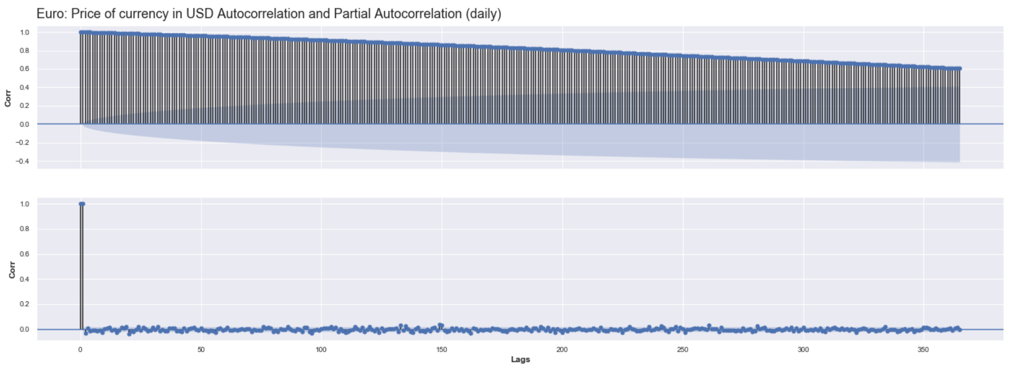


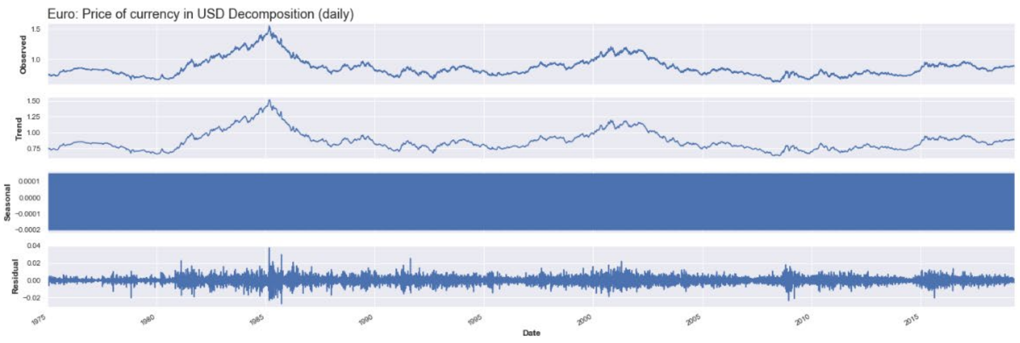
4) Euros

Euro began at 0.79 USD. It peaked in 1985 at 1.6 USD. It jumped in 2008, and again in 2010 and 2015. It is currently at 0.9 USD.



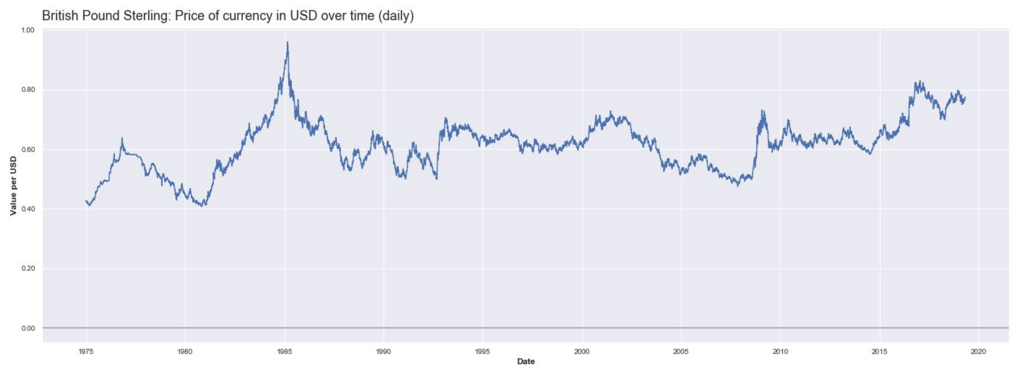


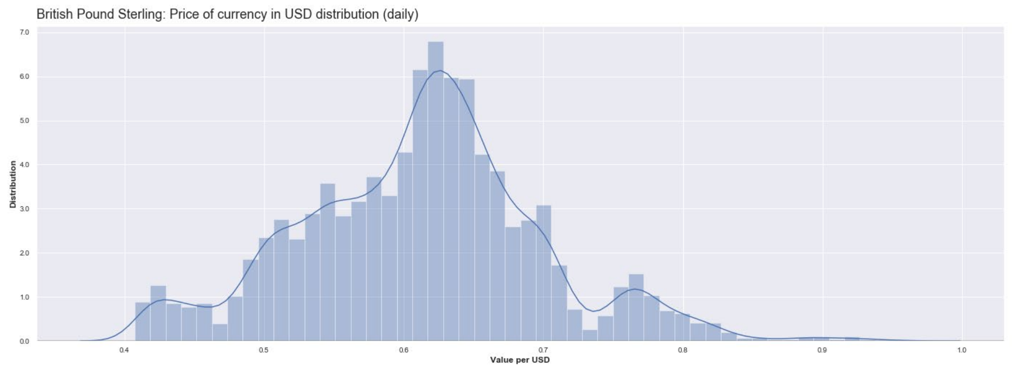


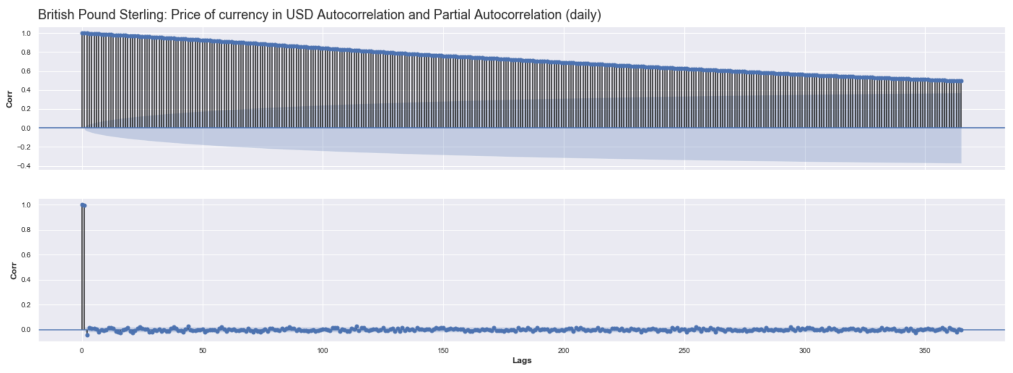


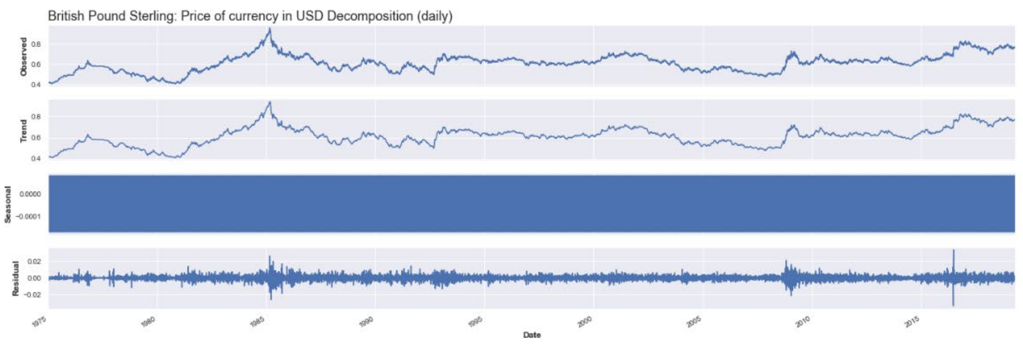
5) Great British Pounds

Great British Pounds started at 0.4 USD. It peaked in 1985 at almost 1.0 USD. It jumped in 2008 and followed a similar trend to the Euro. It is currently just below 0.8 USD.



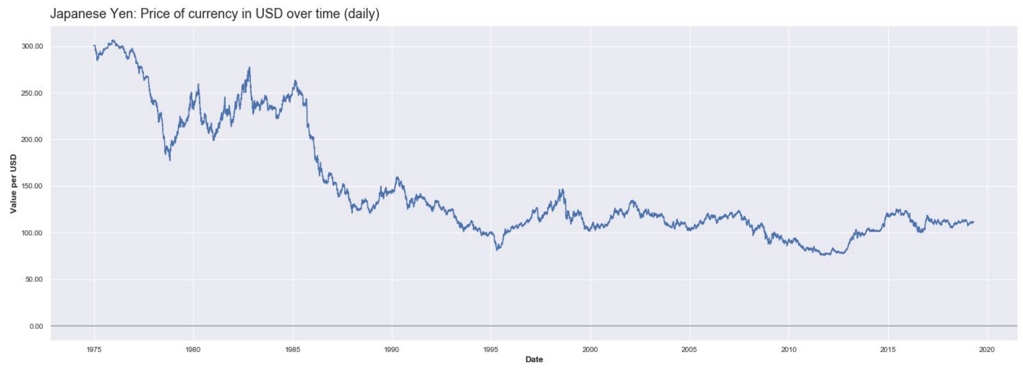


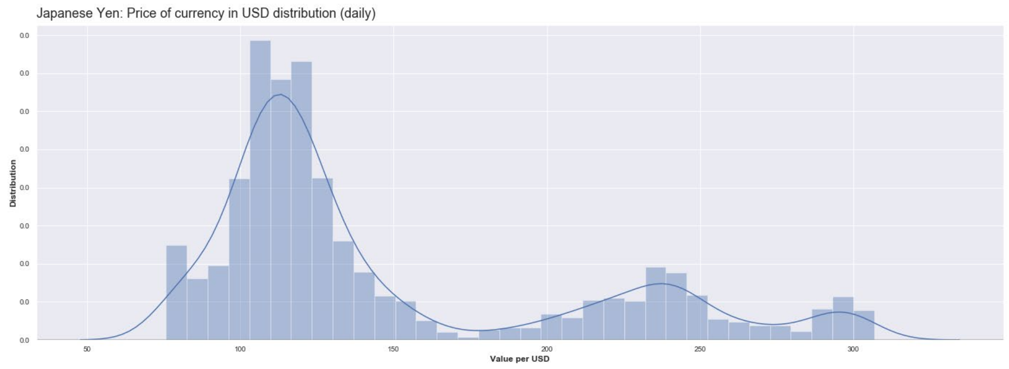


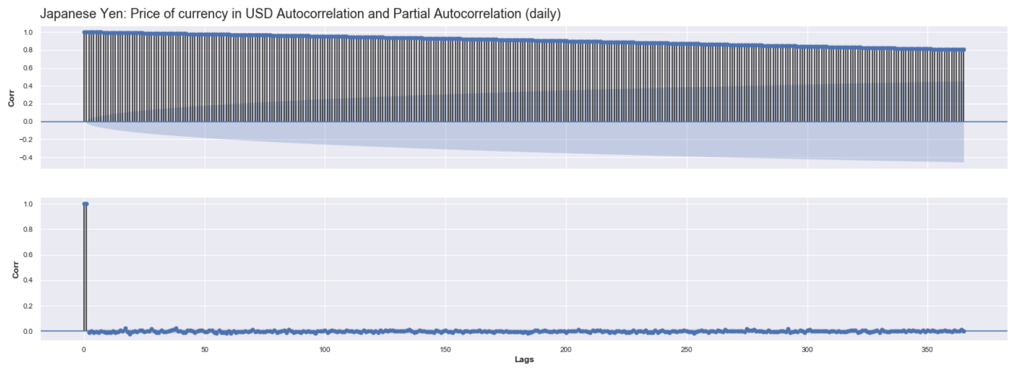


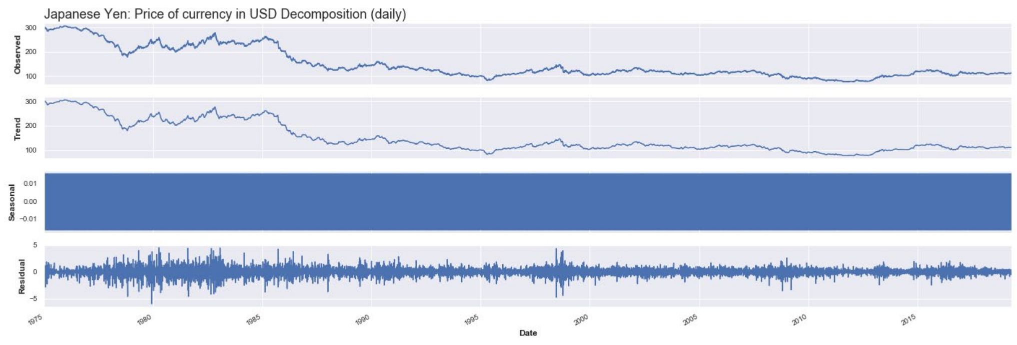
6) Japanese Yen

Japanese Yen began at 300, and experienced early volatility. It peaked shortly after our data began in 1975 at 300, but currently sits around 110. Using recent data here would especially help.



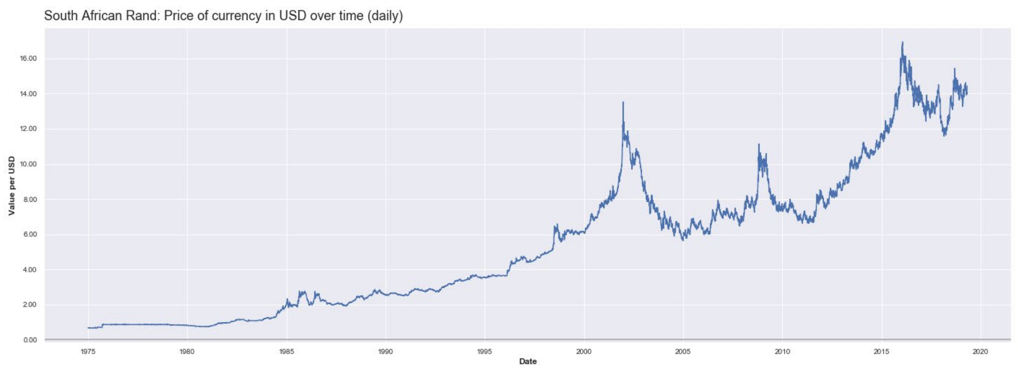


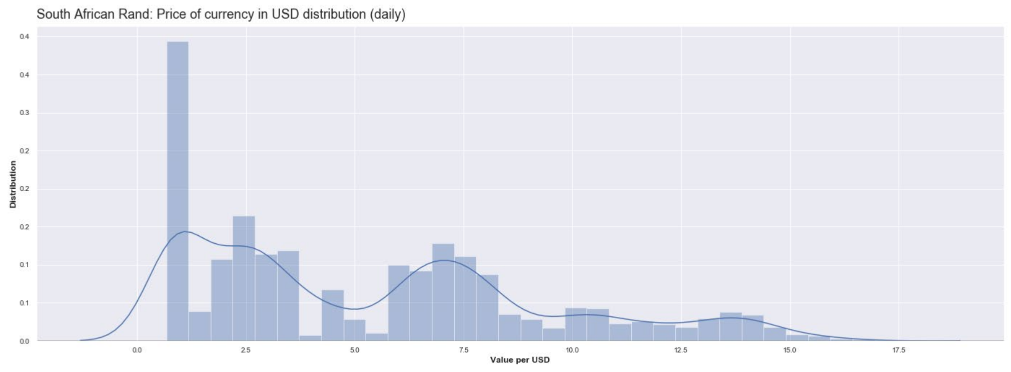


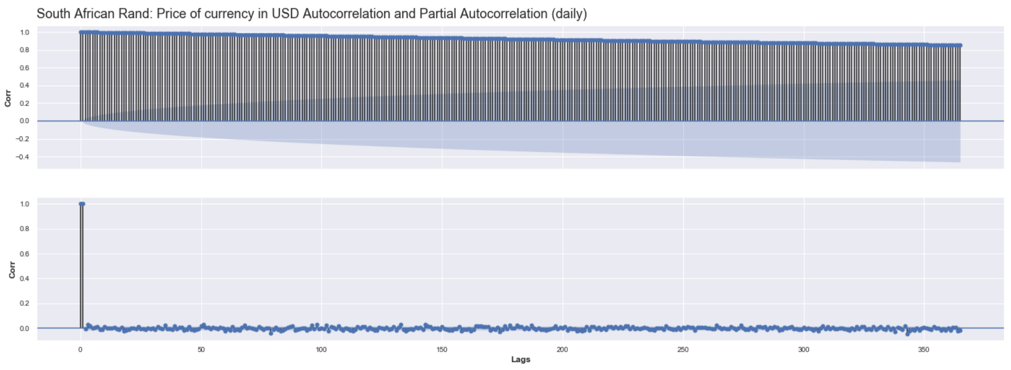


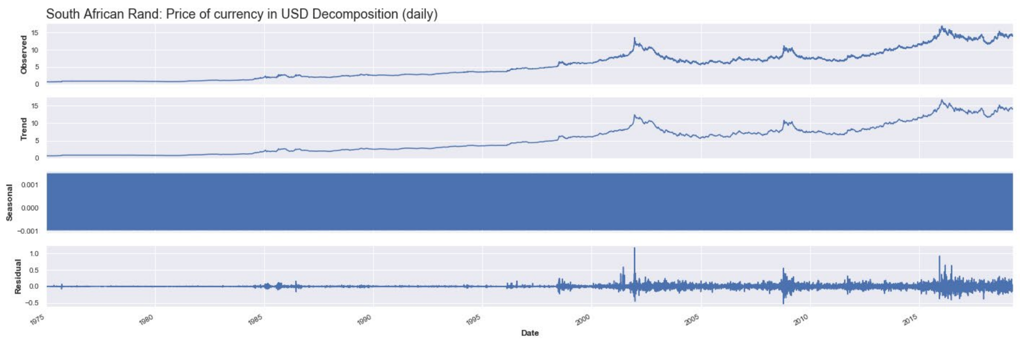
7) South African Rand

South African Rand began at 1 USD, and quickly rose with significant jumps between 2000 and 2005, 2008, and a rise post 2010 through 2018. It currently sits at 14 USD. Recent data would help here also.





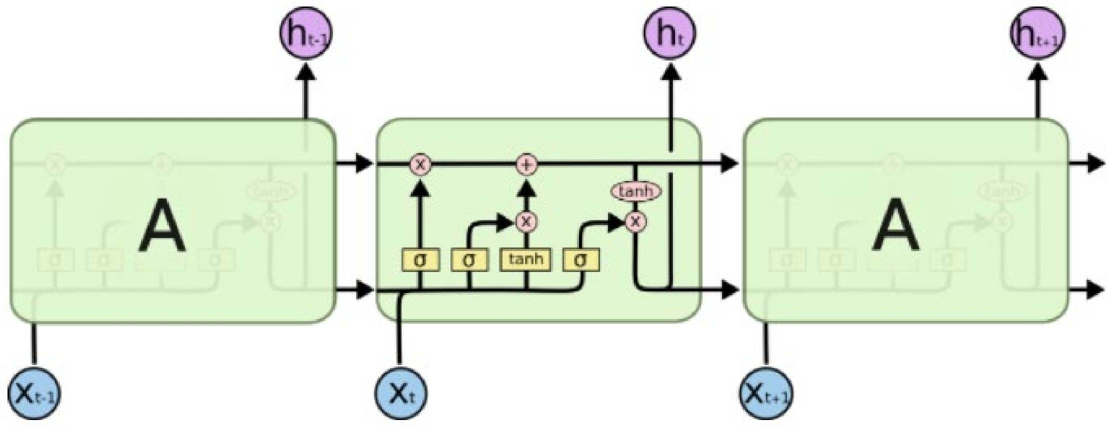




Model

We choose an LSTM based on strong results in practice (Sepp Hochreiter, 1997). The LSTM architecture stores a cell state, and feeds that along with a hidden state representing our output back into the network for recurrence. The cell state and hidden state are updated with 3 gates: the input gate, output gate, and forget gate. These gates represent linear interactions to update the cell state and hidden state. The strength of LSTMs comes from the elimination of the vanishing gradient problem experienced in other recurrent networks. Typical neural networks suffer from vanishing gradients as a result of the non-linear propagation of the error through the network. By only including linear interactions via the gates, this vanishing gradient is eliminated.

Below is a visual representation of the LSTM architecture detailing the flow of information from the cell state and hidden state, through the gates, and to the output.



To forecast future values, a prediction is made as follows:

For each time step *t* Letting i_t represent the input gate and f_t represent the forget gate, the hidden state h_t and cell state c_t are calculated using the following equations. In each of these, weight tensors are given by W, features by x_t, and biases by b.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$h_t = o_t tanh(c_t)$$

We customize our LSTM by adding gradient clipping to avoid exploding gradients. Our hidden state is fed through a feed-forward network, given as a multi-layer or single layer perceptron with a linear activation.

Data Preparation

To prepare the data, we follow these steps:

1) Drop any rows where the target is missing.
2) Impute the data with the previous days value for numeric data, “other” for categorical, or False for Boolean.
3) One-hot encode categorical data.
4) Scale the input and output data using the standard scaler:

$$\frac{x - \mu}{\sigma}$$

5) Split the data into training, validation, and testing using a 60%, 20%, 20% out of time split.
6) Reshape the data for LSTMs, where our new data is in three dimensions of batch size, sequence length, number of features/targets.

Batch size is used to reduce training time, but also results in a boost in accuracy. The sequence length is identified above by looking at the partial autocorrelation. It should be noted that any one-hot encodings and scaling is done first on the training data, and the same transformations are applied to the validation and test sets.

The out of time split suffers from issues related to market cycles, as seen below. We might improve the model by using more recent data or employing K-Fold cross validation.

Results

Benchmarking

Prior to constructing any benchmark models each of the seven currency pairs were tested for stationarity, random walk correlation, and heteroscedasticity. The difference between observation *t* and *(t-1)* was taken (i.e. the daily returns) for each observation in order to make each time series mean reverting. Mean reversion for the daily returns was confirmed utilizing the Augmented Dickey-Fuller Test via the *daftest* function in MATLAB for which every series passed at the 95% confidence level. A variance ratio test was then applied via the *vratiotest* function in MATLAB to confirm that none of the daily return series exhibited random walk behavior for which every series passed at the 95% confidence level. Finally, a Leybourne-McCabe test was conducted via the *lmctest* function in MATLAB in order to ensure each series was in fact stationary for which every currency pair failed at the 95% confidence level. By observing the behavior of each time series, it is easy to see that while they all are mean reverting, the variance is not standard throughout the observed time period thus failing the other criteria necessary for confirming stationarity. Based on the lack of stationarity present in the data, a generalized autoregressive conditional heteroskedastic (GARCH) model was selected as the parameterized model for which to benchmark.

A GARCH(*P,Q)* model is utilized when "variance clustering" is presented in the data which is very clearly the case in the seven exchange rates presented here. GARCH models control for which the variance at time *t* isn't random but appears to be correlated with the variance at (*t-1)..(t-2)…*by modeling the error as:

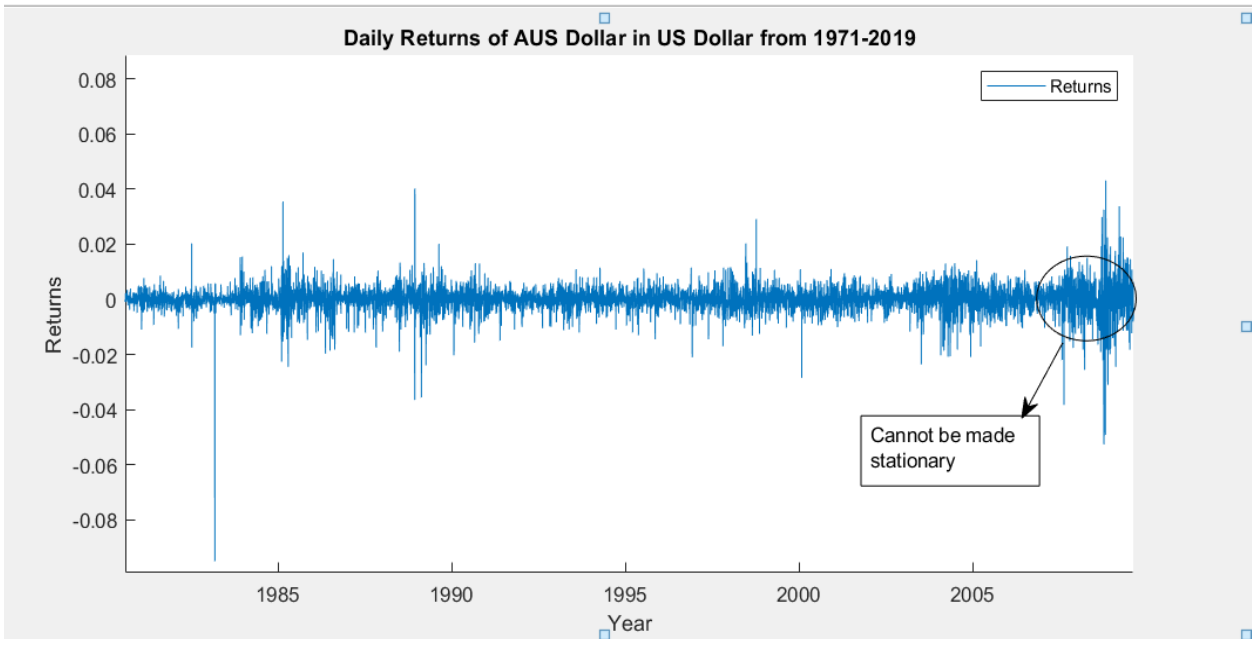
$$\varepsilon_t = K + \Upsilon_1 \sigma^2_{t-1} + \ldots \Upsilon \sigma^2_{t-P} + \alpha_1 \varepsilon^2_{t-1} + \alpha \varepsilon^2_{t-Q}$$

Where P = the number of lagged variances in the model &

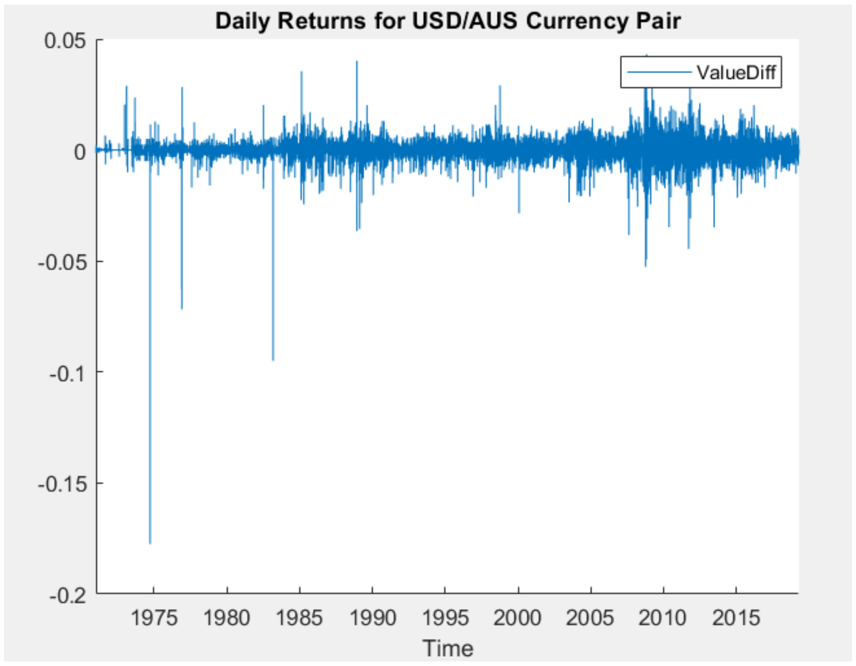
Q = the number of lagged errors in the model

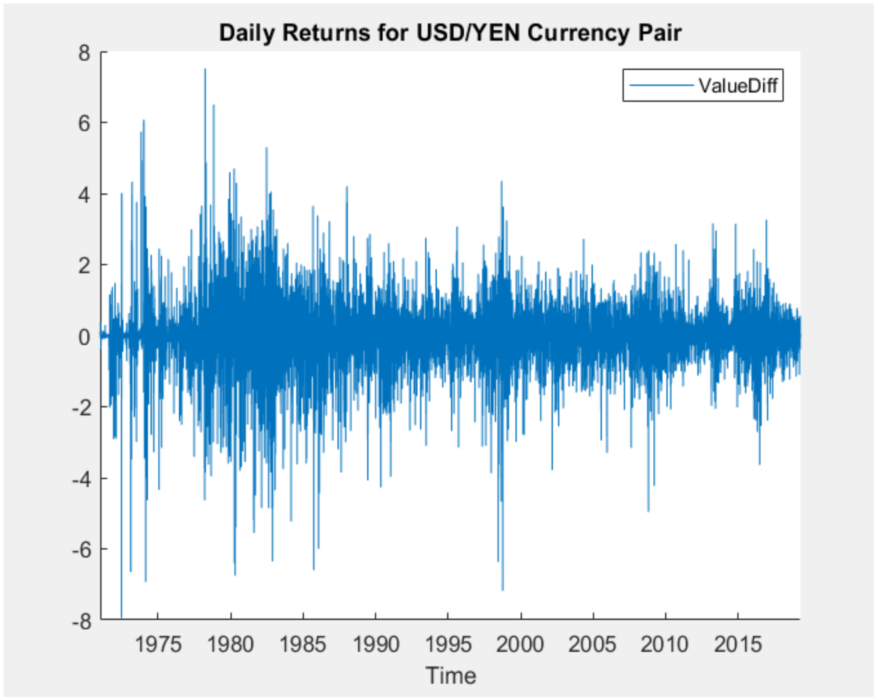
Based on the partial autocorrelation observed in the currency pairs, a standard GARCH(1,1) model (that is every observation is a function of both the error and variance of the observation directly before it) was employed utilizing MATLAB's *garch* function. It is worth noting that, unlike other predictive models, GARCH models attempt to maximize the likelihood that a given prediction matches the distribution of the training data as opposed to minimizing a specific cost parameter (like MSE). For financial data, in which the autocorrelation between observations is incredibly strong and difficult to model via other features, this methodology can lead to better performance as maximum likelihood estimators are less likely to overfit the training data presented. Ten GARCH models for each currency pair (70 in total) were trained utilizing the first 80% of the observations available and tested on the final 20%. The results are reviewed below.

The GARCH(1,1) model for the USD/AUS, USD/CAN, USD/EURO, USD/POUND, and USD/FRANC daily returns performed quite well with average MSE well below 0.0001 on the test set. The GARCH(1,1) model performance was less impressive when predicting the daily returns for USD/RAND and USD/YEN with an average MSE of 0.000276 and 0.0010 respectively. The predicted monthly return accuracy across all seven currency pairs was also lower with none of the GARCH(1,1) models reaching an MSE below 0.0001. Given that the USD/YEN daily returns, USD/RAND daily returns, and the collective monthly returns are all inherently more volatile and, given that the GARCH model was standardized across each series with the P and Q values set to 1, it is understandable that these models would perform worse than the more stable return series'. The more complex LSTM network will ideally outperform the GARCH(1,1) model for these datasets in particular thereby proving the need for nonlinear time series architectures when dealing with more volatile target features.

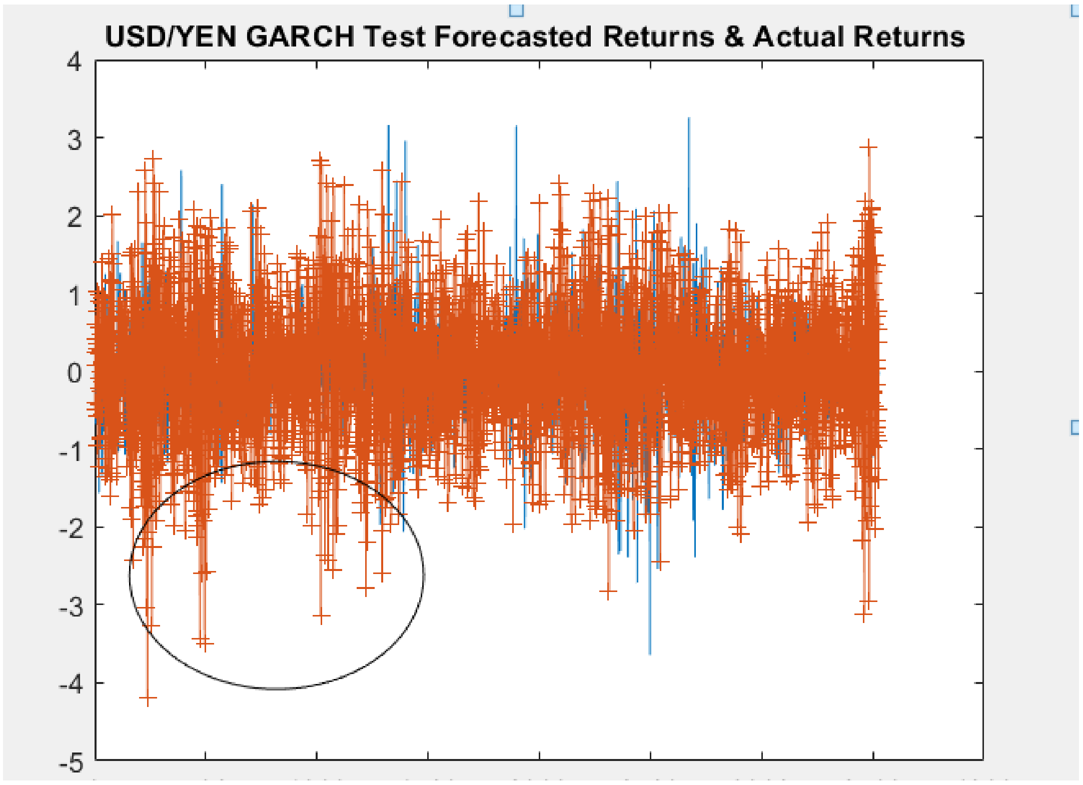


Daily Returns of the USD/AUS currency pair from 1971-Apri 12,2019. Note the presence of "variance clusters" requiring a GARCH model to accurately predict.





Daily Returns for both the USD/AUS and USD/YEN currency pairs. Note how much more volatile the USD/YEN returns are.



GARCH(1,1) predictions and actuals for the USD/YEN currency pair. There are periods in which the model obviously overestimates the actual data's volatility which is indicative of a less stable time series.

USD/AUS Daily Returns MSE	USD/CAN Daily Returns MSE	USD/EURO Daily Returns MSE	USD/FRANC Daily Returns MSE	USD/POUND Daily Returns MSE	USD/RAND Daily Returns MSE	USD/YEN Daily Returns MSE
8.01E-05	8.01E-05	1.56E-04	2.03E-04	1.54E-04	2.12E-03	3.50E-03
1.07E-05	1.07E-05	1.59E-04	2.48E-05	4.84E-04	2.60E-03	6.63E-03
3.41E-07	3.41E-07	7.98E-05	2.37E-04	3.76E-04	2.24E-03	9.46E-04
1.73E-04	1.73E-04	3.77E-05	1.75E-04	2.12E-04	3.42E-03	1.62E-02
3.83E-06	3.83E-06	4.39E-05	2.91E-05	4.41E-05	4.37E-03	8.54E-03
7.07E-06	7.07E-06	1.13E-04	1.70E-04	5.92E-04	2.88E-03	2.01E-02
1.45E-05	1.45E-05	4.94E-05	1.10E-04	1.94E-04	2.73E-03	3.78E-02
3.18E-05	3.18E-05	1.02E-05	9.63E-05	1.32E-03	2.66E-03	8.18E-05
5.63E-07	5.63E-07	1.20E-04	7.05E-05	5.91E-05	2.18E-03	2.32E-03
1.38E-04	1.38E-04	3.25E-04	1.31E-05	3.47E-04	2.35E-03	1.35E-02
4.60E-05	4.60E-05	1.09E-04	1.13E-04	3.78E-04	2.76E-03	1.10E-02

MSE for each of the ten daily return GARCH(1,1) forecasting models.

USD/AUS Monthly Returns MSE	USD/CAN Monthly Returns MSE	USD/EURO Monthly Returns MSE	USD/FRANC Monthly Returns MSE	USD/POUND Monthly Returns MSE	USD/RAND Monthly Returns MSE	USD/YEN Monthly Returns MSE
0.000507	0.005052	0.0004877	0.002251	0.00051735	0.0826948	0.052203
0.001544	0.007712	0.0030476	0.0108208	0.00892247	0.0898333	0.099662
0.002098	0.002952	0.0021721	0.0008694	0.01106982	0.0998859	0.50101
0.001286	0.003632	0.0016472	0.0123437	0.01058953	0.0845227	0.147766
0.002001	0.005129	0.0183642	7.34E-05	0.00414557	0.0867681	0.301634
0.000312	0.005132	0.0008766	0.0034065	0.00049145	0.0743053	0.431472
0.001871	0.001511	0.0022733	0.003373	0.00216955	0.1301817	0.264326
0.004074	0.001757	0.0094361	0.009283	0.00843445	0.0991771	0.289675
0.005173	0.002486	0.0122503	0.0085699	0.00494324	0.137285	0.453961
0.000191	0.006444	0.0034423	0.0035561	0.00553124	0.0690231	0.32377
1.91E-03	4.18E-03	5.40E-03	5.45E-03	5.68E-03	9.54E-02	2.87E-01

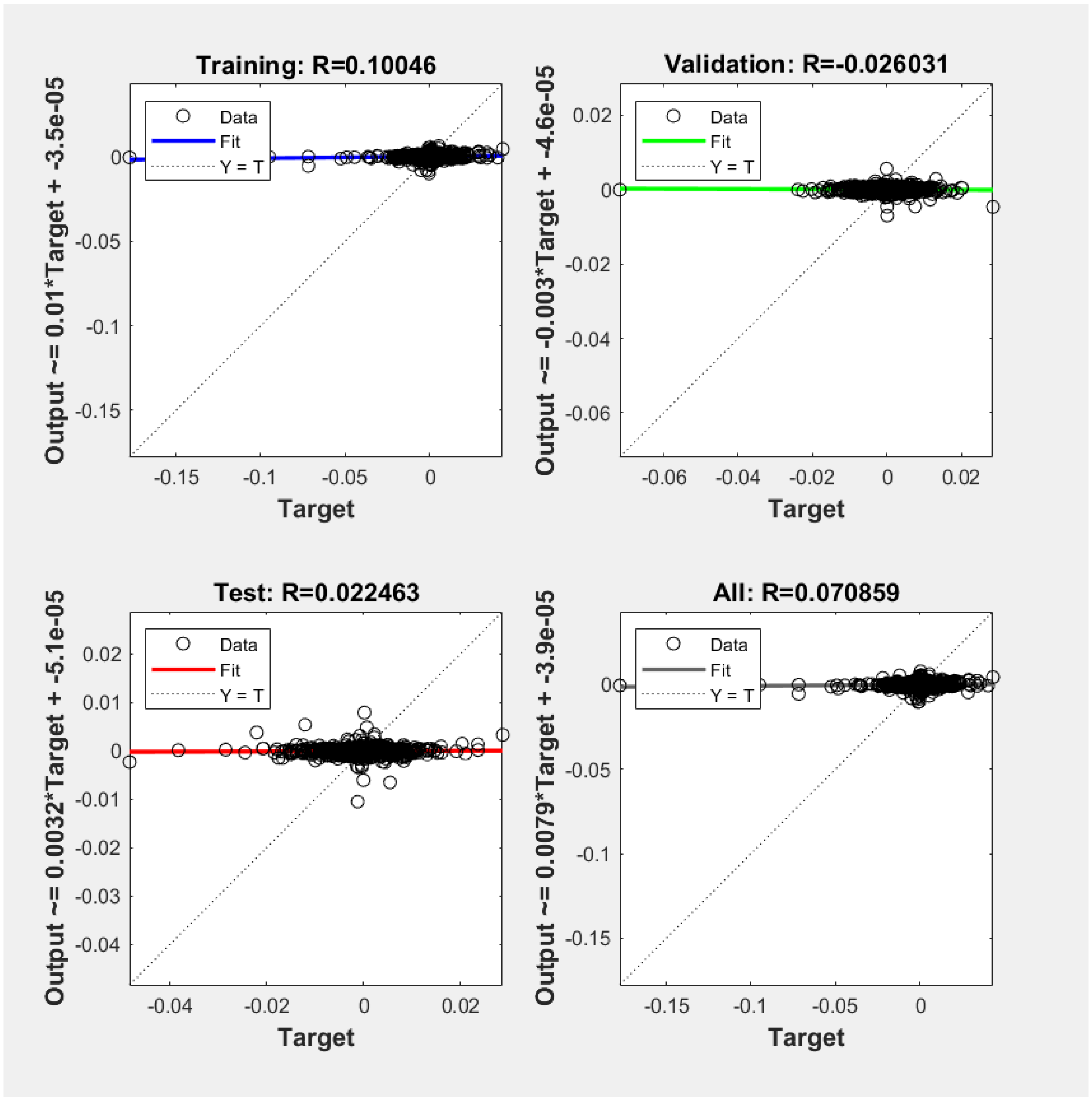
MSE for each of the ten monthly return GARCH(1,1) forecasting models.

Focused Time Delayed Neural Network

Attempts to build a nonlinear autoregressive model (NARX) with multiple exogenous features ultimately bared dismal results. While several of these models produced mean squared error rates below 0.001, the R-values on the test observations varied sporadically and never consistently reached above 10%. This is likely due to the target domain (daily & monthly returns) being measured in fractions of a penny; the mean error is a function of the distance between the prediction and the observed target and thus, when measuring said error in such a small domain, the MSE can be deceiving. The R-Value measures how much of the deviation between each target observation is explained by a given set of input features and, no matter how many were added to the NARX architecture, the R-Value never reached a reliable level of significance. This would lead one to conclude that the variance observed in each currency pair is completely random however, based on the previously mentioned variance ratio test, autocorrelation is clearly observable within each series. This was a considerable hurdle that took a significant amount of time and effort to overcome.

Consulting the work of Gers, Eck, and Schmidhuber proved to be quite beneficial in addressing the observed inconsistencies present in the benchmark NARX models as they found similar findings when utilizing recursive neural networks for short-term financial predictions. According to Gers et al.'s publication "Applying LSTM to Time Series Predictable through Time-Window Approaches", "Time series benchmark problems found in the literature are often conceptually simpler than many tasks already solved by LSTM. They often do not require RNNs at all, because all relevant information about the next event is conveyed by a few recent events contained within a small time window." The paper goes on to make the argument that, for financial data that is highly autocorrelated, recurrent neural networks often discard more recent, relevant lagged values and favor more distant, less relevant lagged values. This can be overcame by increasing the number of training epochs but, as mentioned above, recurrent neural networks are often unnecessary for simple, heavily autocorrelated observations. Based on these findings a focused time delayed network was constructed for each currency return utilizing MATLAB's *timedelaynet* package. Focused time delayed networks are rather novel because they have no intrinsic memory built explicitly into the model at all; rather they simply present lagged observations for a given target as independent inputs. Bayesian optimization was utilized in order to identify the optimal hidden layer size, input delay, and learning rate for each series modeled. The findings for this exercise are discussed below.

Gers, Eck, and Schmidhuber's findings proved to be correct as for both the daily and monthly returns the time delayed network achieve an R-Value well above 0.99 for every currency pair modeled. Of interesting note is the optimal hidden layer size and input delay produced by the Bayesian optimization algorithm; for the less volatile currency pairs the optimal hidden layer size and input delay were both set to 1 indicating that any model more complex was prone to overfitting whereas, for the more volatile currency pairs and monthly returns respectively, hidden layer sizes and input delays closer to 10 was much more common. However, average MSE for the time delayed neural networks was up anywhere between 40% to 1400% when compared to the GARCH(1,1) models proving that, for highly autocorrelated features, even the simplest nonlinear models have a propensity to overfit and thus achieve worse results when compared to more traditional, parameterized time series models.



R-Values for typical NARX model. A test R-Value of 2% was deemed insufficient for model benchmarking.

USD/AUS Daily Returns MSE	USD/CAN Daily Returns MSE	USD/EURO Daily Returns MSE	USD/FRANC Daily Returns MSE	USD/POUND Daily Returns MSE	USD/RAND Daily Returns MSE	USD/YEN Daily Returns MSE
2.90E-05	3.33E-05	0.0006685	0.0002103	0.00015179	0.003145	1.42960
USD/AUS Daily Returns R-value	USD/CAN Daily Returns R-value	USD/EURO Daily Returns R-value	USD/FRANC Daily Returns R-value	USD/POUND Daily Returns R-value	USD/RAND Daily Returns R-value	USD/YEN Daily Returns R-value
0.99985	0.99974	0.99919	0.99986	0.99987	0.99977	0.99992

MSE & R-Value for of the ten Time Lagged Network daily return forecasts

USD/EURO Monthly Returns MSE	USD/FRANC Monthly Returns MSE	USD/POUND Monthly Returns MSE	USD/RAND Monthly Returns MSE	USD/YEN Monthly Returns MSE
0.041201	0.0095402	0.0062322	0.15767	193.33

USD/AUS Monthly Returns R-value	USD/CAN Monthly Returns R-value	USD/EURO Monthly Returns R-value	USD/FRANC Monthly Returns R-value	USD/POUN D Monthly Returns R-value	USD/RAND Monthly Returns R-value	USD/YEN Monthly Returns R-value
0.99307	0.98501	0.97712	0.99292	0.98542	0.99033	0.99348

MSE & R-Value for of the ten Time Lagged Network monthly return forecasts

```
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 85.2015 seconds.
Total objective function evaluation time: 74.6378

Best observed feasible point:
    hiddenLayerSize        lr        inputdelays
    _______________    _________    ___________

           1           0.0012598        10

Observed objective function value = 0.00015179
Estimated objective function value = -0.0053459
Function evaluation time = 6.0322

Best estimated feasible point (according to models):
    hiddenLayerSize        lr        inputdelays
    _______________    _________    ___________

           1           0.0010371         4

Estimated objective function value = -0.0053459
Estimated function evaluation time = 3.2265
```

```
__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 29.1168 seconds.
Total objective function evaluation time: 10.3623

Best observed feasible point:
    hiddenLayerSize        lr        inputdelays
    _______________    _________    ___________

          20           0.0010959         9

Observed objective function value = 193.3263
Estimated objective function value = 512.9864
Function evaluation time = 0.41544

Best estimated feasible point (according to models):
    hiddenLayerSize        lr        inputdelays
    _______________    _________    ___________

          20           0.0010959         9

Estimated objective function value = 512.9864
Estimated function evaluation time = 0.41142
```

Optimal Hyperparameters for the daily USD/AUS & USD/YEN Focused Time Delay Networks. Note the increased complexity of the USD/YEN as observed by its larger hidden layer size and input delays.

Technical Indicators

Day traders historically have utilized a set of technical indictors to detect trends in a given financial asset's historical movement. These indicators fall into two overarching categories: mean reverting trends and momentum trends. As their labels imply, these metrics attempt to identify patterns in an underlying asset and allow financial analysts to tease out whether said asset is going to increase or decrease in future subsequent periods. In order to allow the LSTM model to learn every possible pattern in the seven currency pairs presented, some of the most common mean reverting and trend identifying metrics were included as input features with the explicit goal of improving each LSTM's overall prediction accuracy. A brief overview of these technical indicators is presented below.

10-Day Simple Moving Average

As the name implies, the simple moving average for a given observation is merely the average of the previous ten observations. This measures how much deviation there is between a particular financial asset's price at time *t* and its historical price over the previous ten days.

12-Day and 26-Day Exponential Moving Average

Exponential moving average (EMA) or decay is similar to the simple moving average however the key distinction is that the EMA weighs more recent observations more heavily than past observations. This indicator is a key measurement for other technical indicators as historically it has performed well when utilized on live financial data. The formulas for both the 12-day EMA and 26-day EMA are given below.

EMA_{12}=$Price_{12}$*(smoothing coefficient/1+12)+…EMA_{1})

EMA_{26}=$Price_{26}$*(smoothing coefficient/1+26)+…EMA_{1})

Upper & Lower Bollinger Bands

Bollinger bands are a straightforward volatility metric indicating whether a given asset's deviation from its mean is increasing or decreasing. The calculation is as follows:

Upper Band = 10-Day Moving Avg+$2\sigma^2$

Lower Band = 10-Day Moving Avg-$2\sigma^2$

As the 10-day moving average begins to fluctuate both the upper and lower Bollinger bands expand and as the 10-day moving average begins to stabilize the upper and lower Bollinger bands contract.

Moving Average Convergence Divergence (MACD)

The MACD is a very straightforward indicator that simply subtracts the 26-day EMA from the 12-day EMA. When this metric increases, it indicates that an asset's recent price that is trending below the historic price and when the MACD decreases, it indicates that an asset's recent price is trending above the historic price.

10-Day Relative Strength Index (RSI)

The RSI is an important trend indicator as it is when of the few traditional financial metrics that measures historic returns rather than price. It is calculated by taking the average 10-day gains and dividing it by the average 10-day loses allowing financial analysts to identify the total returns of a given asset over a 10-day time span.

LSTM

For an initial run, we create the LSTM with a single layer and 10 nodes in the hidden state. We look back 365-time steps in the cell state. The hidden state is fed to a simple feed forward network with no non-linear activation and only one layer. To begin, we run a simple auto-regressive model with no exogenous features, only including the prior prices as features in our model, for reasons described above. The baseline is calculated as a naïve model by simply using the energy usage from the prior day to predict the next day.

Training involves optimization using ADAM with a learning rate of 0.001. We train for 2000 epochs. Mean Absolute Error is chosen as our loss. This high number of epochs is allowed as a result of high computing power and contributes to the low loss. However, this low loss is a result of higher weights put on the most recent time period.

The overall prediction performs well on the training data and does seem to generalize well to the validation or test sets. Some currencies are more difficult to predict, as seen below. Training takes place rapidly at first followed by a very gradual decrease in loss. Increasing the number of epochs may increase accuracy, although based on the chart we expect decreasing returns with respect to training time.

To try and beat the naïve model, we attempted to add epochs, and simplify the model. This is where we found it is better to decrease complexity and model only using the prior prices as features. We also removed any extra layers from the LSTM and kept the feed forward network simple at 10 hidden nodes.

As a final attempt to beat the naïve model in an environment that we could use for portfolio optimization, we attempt to model the data out 1 month. Details on that follow initial model evaluation.

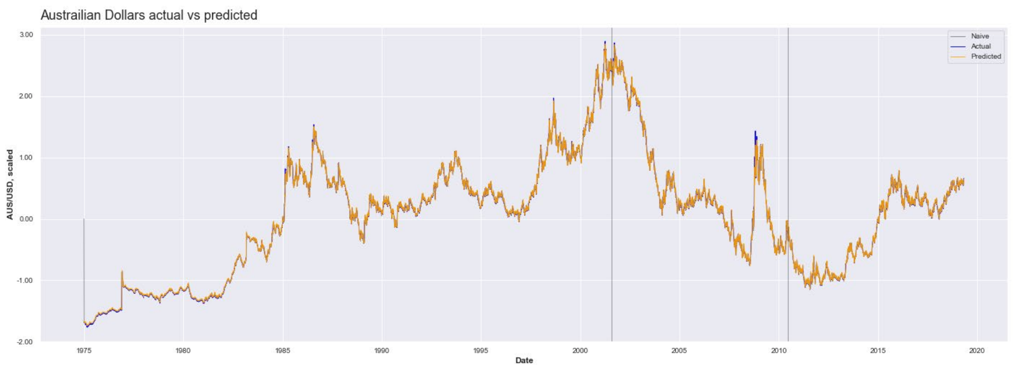
For each of the currencies, we evaluate the model on the following pages.

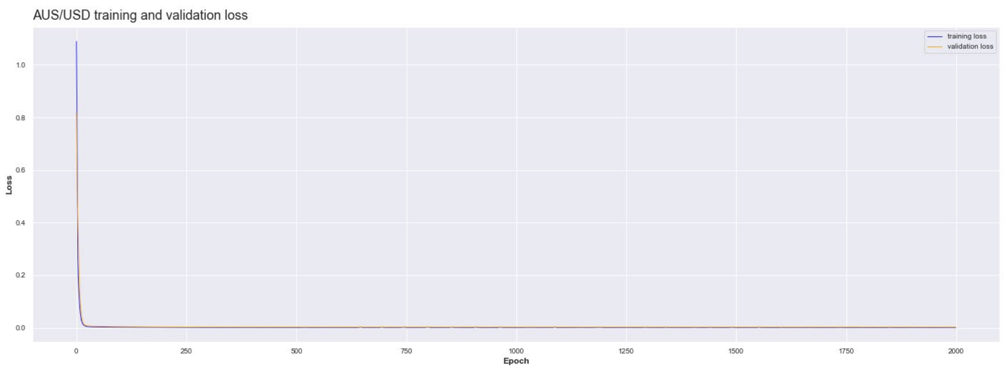
1) Australian Dollars

Test loss remains just above the baseline loss (0.0008 vs 0.0007), while remaining well below the validation and training loss. As shown in the actual vs predicted plot, validation loss shows failure in periods of extreme growth. Other than that, no periods show any pattern in the residual.

Training happens very quickly and levels off for the remaining epochs. The validation loss is still improving at the end of the run, indicating we should decrease the learning rate further or decrease the number of epochs.

```
Predicting AUS, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 10:03:19.646785
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 1.0895072714654062, Validation Loss: 0.8218388083494372
Epoch 250, Train Loss: 0.01655512549401739, Validation Loss: 0.019597325645219793
Epoch 500, Train Loss: 0.00888765476391019, Validation Loss: 0.011154850263140904
Epoch 750, Train Loss: 0.006322952305871814, Validation Loss: 0.008261274719950687
Epoch 1000, Train Loss: 0.0050260085378644286, Validation Loss: 0.006785608779776225
Epoch 1250, Train Loss: 0.004239781712139341, Validation Loss: 0.005898610268703694
Epoch 1500, Train Loss: 0.0037121254262967953, Validation Loss: 0.005304280791499495
Epoch 1750, Train Loss: 0.00332738478578337, Validation Loss: 0.004879244616146144
Epoch 1999, Train Loss: 0.0030361201367477405, Validation Loss: 0.004564247329812336
Training completed at 2019-04-28 10:39:29.353335, taking: 2169.70655 seconds. Test error: 0.00080162158216505 (vs baseline: 0.0007333749672397971)
```



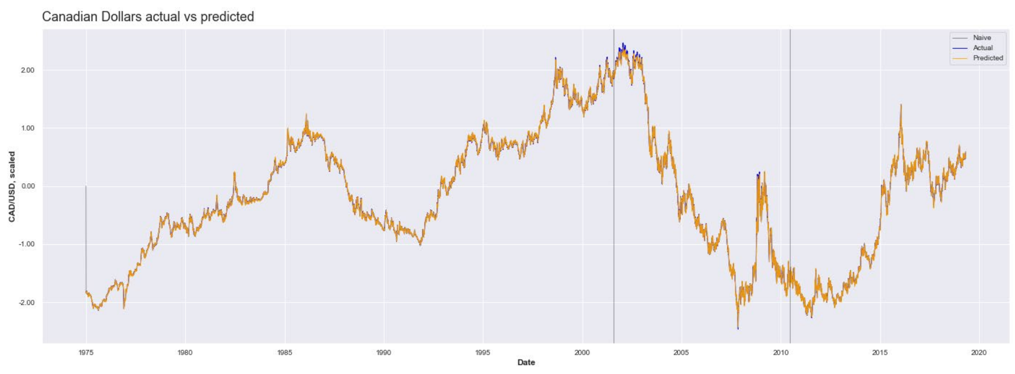


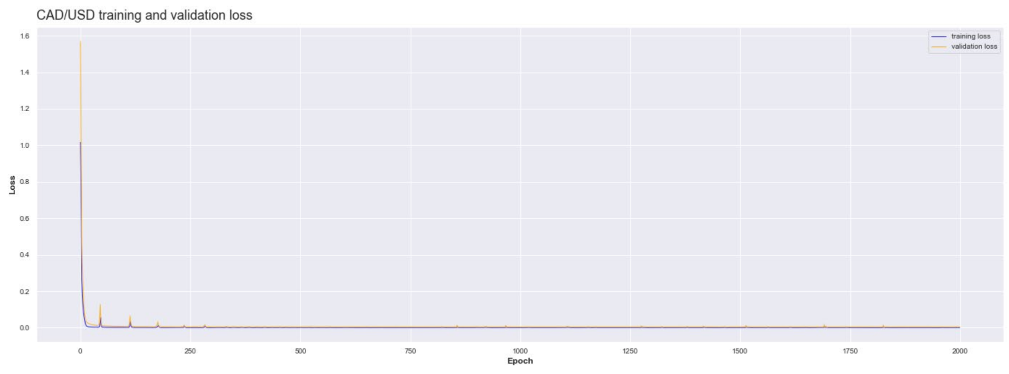
2) Canadian Dollars

Test loss remains just above the baseline loss (0.0019 vs 0.0018), while remaining well below the validation and training loss. As with Australian dollars, and as shown in the actual vs predicted plot, validation loss shows failure in periods of extreme growth. Other than that, no periods show any pattern in the residual.

Training happens very quickly and levels off for the remaining epochs. The validation loss is still improving at the end of the run, indicating we should decrease the learning rate further or decrease the number of epochs.

```
Predicting CAD, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 10:39:34.624201
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 1.016331946182961, Validation Loss: 1.5694306755645409
Epoch 250, Train Loss: 0.016147611745420547, Validation Loss: 0.03087857402126501
Epoch 500, Train Loss: 0.008680367501866862, Validation Loss: 0.01753166244608304
Epoch 750, Train Loss: 0.006128249459652753, Validation Loss: 0.012964245728711216
Epoch 1000, Train Loss: 0.004833248883038282, Validation Loss: 0.010685505582868822
Epoch 1250, Train Loss: 0.004052705398567487, Validation Loss: 0.009300703978755473
Epoch 1500, Train Loss: 0.0035304050667035783, Validation Loss: 0.008381170816193169
Epoch 1750, Train Loss: 0.003155823550558032, Validation Loss: 0.007727111637271208
Epoch 1999, Train Loss: 0.0028723374603571946, Validation Loss: 0.007225425336199501
Training completed at 2019-04-28 11:15:46.529198, taking: 2171.904997 seconds. Test error: 0.001861455636875083 (vs baseline: 0.0018063904717564583)
```



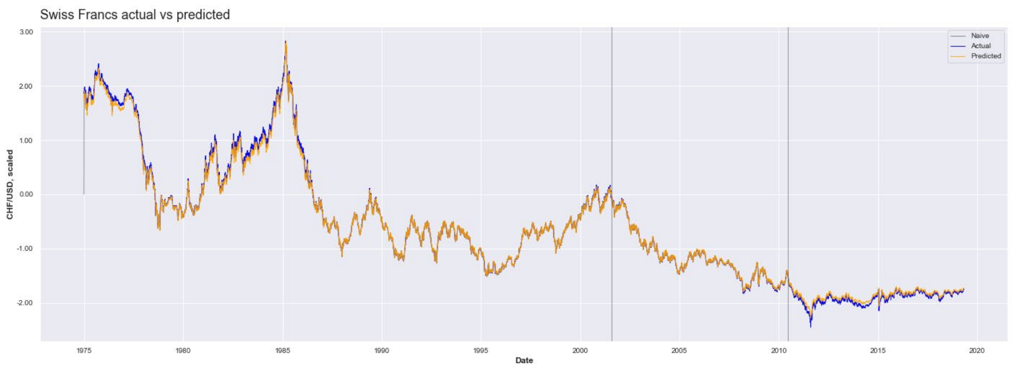


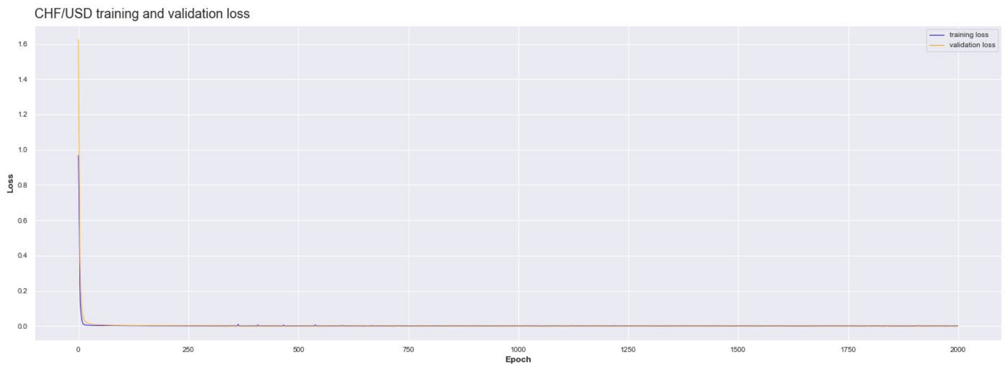
3) Swiss Franc

Test loss remains just above the baseline loss (0.0024 vs 0.0022), while remaining well below the validation and training loss. As with the other currencies, and as shown in the actual vs predicted plot, validation loss shows failure in periods of extreme growth. Test loss shows that the predicted values are too high vs the actual, highlighting the need for more recent data in training. After 1990 would be appropriate.

Training happens very quickly and levels off for the remaining epochs. The validation loss is still improving at the end of the run, indicating we should decrease the learning rate further or decrease the number of epochs.

```
Predicting CHF, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 14:19:05.929963
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 0.9671268462080438, Validation Loss: 1.6231175139546394
Epoch 250, Train Loss: 0.01441373852345126, Validation Loss: 0.025275310025539708
Epoch 500, Train Loss: 0.008026247552478038, Validation Loss: 0.013398586781830193
Epoch 750, Train Loss: 0.005825302826475539, Validation Loss: 0.009429394207725499
Epoch 1000, Train Loss: 0.004691235882472077, Validation Loss: 0.007400605599856475
Epoch 1250, Train Loss: 0.0040008323791858305, Validation Loss: 0.006155430400190882
Epoch 1500, Train Loss: 0.003533242262082792, Validation Loss: 0.005357459233284384
Epoch 1750, Train Loss: 0.0031947983624913143, Validation Loss: 0.004768934571554613
Epoch 1999, Train Loss: 0.002940944575471117, Validation Loss: 0.004326896800593305
Training completed at 2019-04-28 14:55:26.478689, taking: 2180.548726 seconds. Test error: 0.0024201391659314847 (vs baseline: 0.000222513685002923)
```



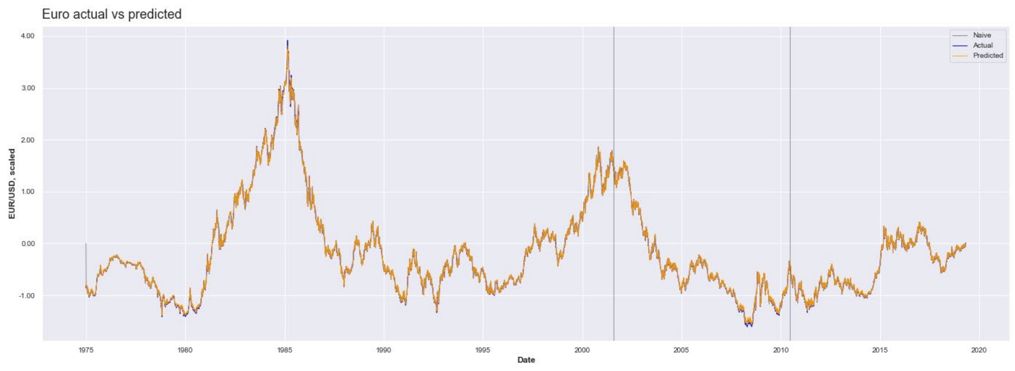


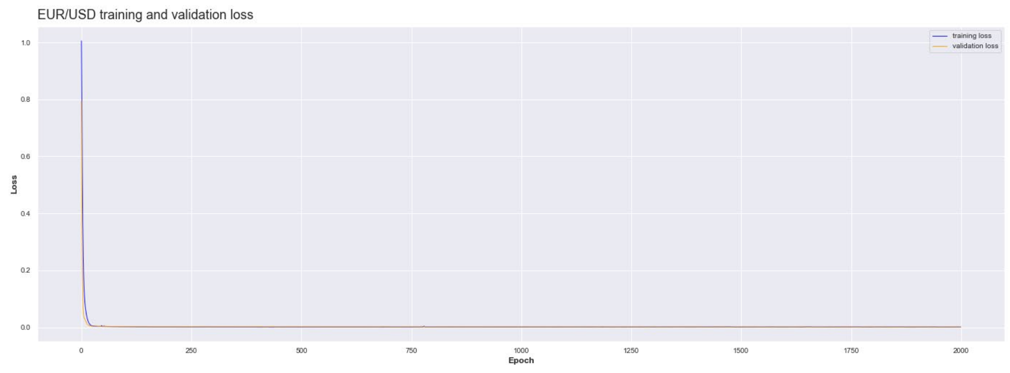
4) Euro

Test loss remains just above the baseline loss (0.00080 vs 0.00075), while remaining well below the validation and training loss. No periods show any pattern in the residual.

Training happens very quickly and levels off for the remaining epochs. The validation loss is still improving at the end of the run, indicating we should decrease the learning rate further or decrease the number of epochs.

```
Predicting EUR, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 14:55:31.725643
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 1.0054136367585018, Validation Loss: 0.7948299360771974
Epoch 250, Train Loss: 0.017343198302769033, Validation Loss: 0.009490894237708169
Epoch 500, Train Loss: 0.009414715095541962, Validation Loss: 0.005662001168359825
Epoch 750, Train Loss: 0.006726428075301557, Validation Loss: 0.004319252726327236
Epoch 1000, Train Loss: 0.005391305166281724, Validation Loss: 0.00359675032073506
Epoch 1250, Train Loss: 0.004578533328046335, Validation Loss: 0.003168729610373545
Epoch 1500, Train Loss: 0.004030684340139276, Validation Loss: 0.00287477524022292
Epoch 1750, Train Loss: 0.003637320079405004, Validation Loss: 0.0026527210119683727
Epoch 1999, Train Loss: 0.003340924542470226, Validation Loss: 0.002479847566442509
Training completed at 2019-04-28 15:31:52.985981, taking: 2181.260338 seconds. Test error: 0.0008016711823681059 (vs baseline: 0.000758805894292891)
```



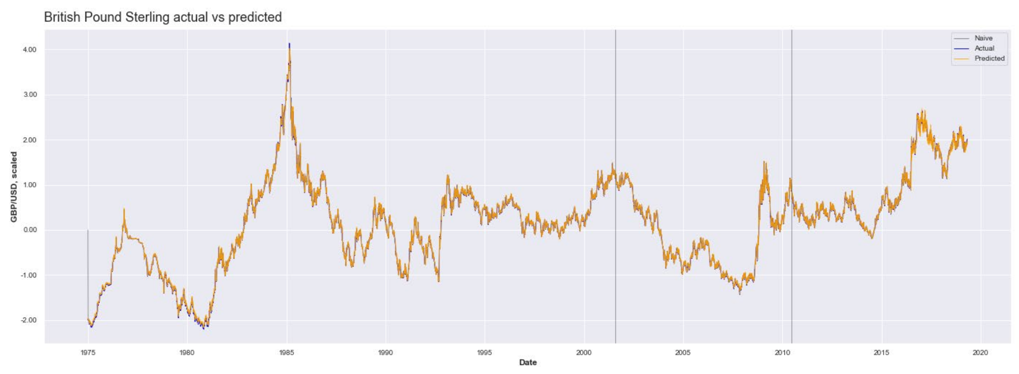


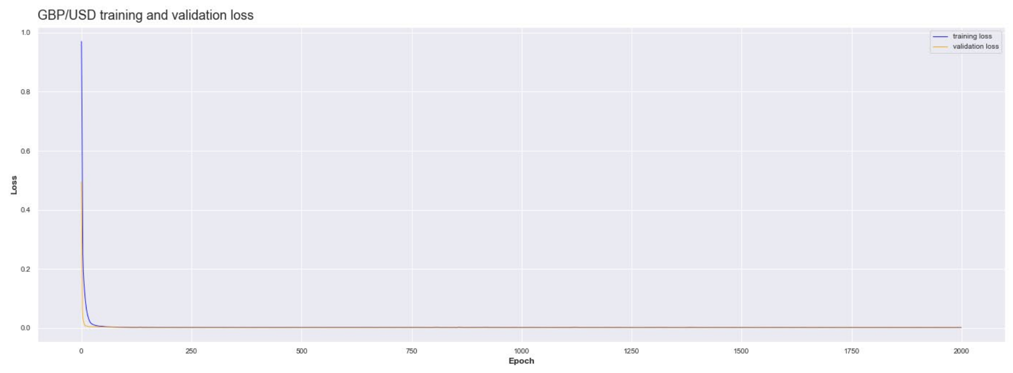
5) Great British Pounds

Test loss remains just above the baseline loss (0.0027 vs 0.0020), staying in line with the validation loss while remaining well below the training loss. No periods show any pattern in the residual.

Training happens very quickly and levels off for the remaining epochs. The validation loss is still improving at the end of the run, indicating we should decrease the learning rate further or decrease the number of epochs.

```
Predicting GBP, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 13:06:04.124263
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 0.970524599916249, Validation Loss: 0.4950203150510788
Epoch 250, Train Loss: 0.018089026467805837, Validation Loss: 0.006663130684184231
Epoch 500, Train Loss: 0.01009911628940312, Validation Loss: 0.0043926930640868335
Epoch 750, Train Loss: 0.00743005318801044, Validation Loss: 0.003644948925635217
Epoch 1000, Train Loss: 0.006093302793542257, Validation Loss: 0.0032751495514196784
Epoch 1250, Train Loss: 0.005280443557582312, Validation Loss: 0.003052290544517146
Epoch 1500, Train Loss: 0.004732174053872804, Validation Loss: 0.002902987859540512
Epoch 1750, Train Loss: 0.004334380959704092, Validation Loss: 0.002795949961300971
Epoch 1999, Train Loss: 0.004034635434757599, Validation Loss: 0.002717846107484042
Training completed at 2019-04-28 13:42:26.594636, taking: 2182.470373 seconds. Test error: 0.0027400521761996467 (vs baseline: 0.002023068256676197)
```



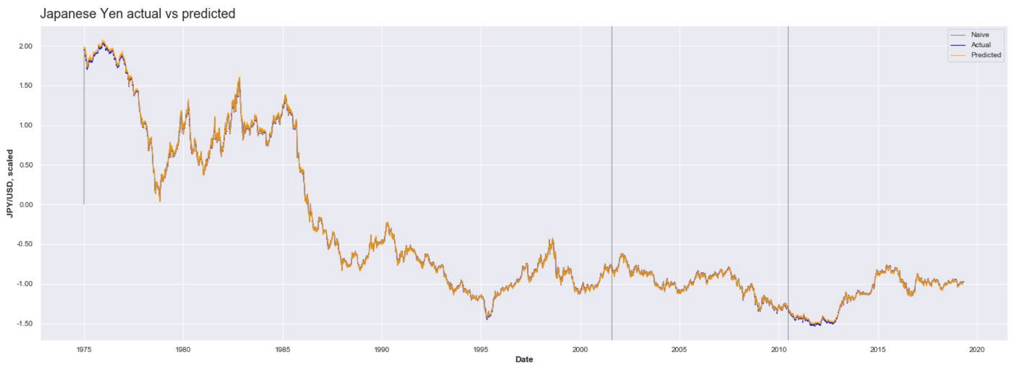


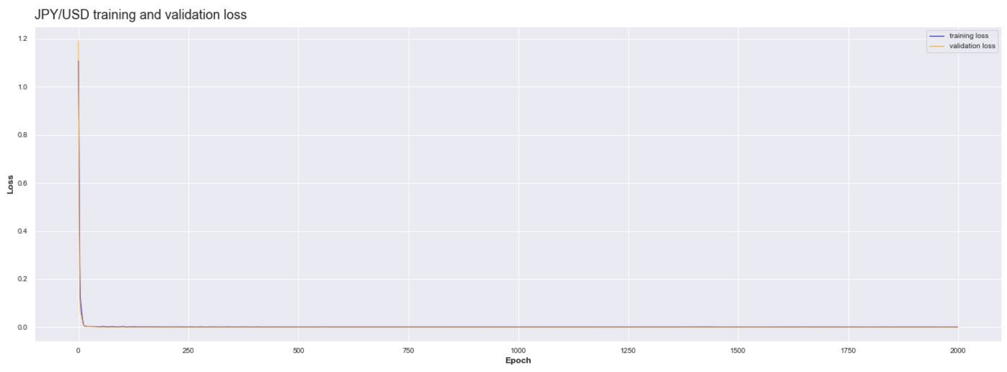
6) Japanese Yen

Test loss remains well above the baseline loss (0.00014 vs 0.000088), while remaining just below the validation and training loss. The test prediction is above the actual, highlighting the need for more recent data.

Training happens very quickly and levels off for the remaining epochs. The validation loss is still improving at the end of the run, indicating we should decrease the learning rate further or decrease the number of epochs.

```
Predicting JPY, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 11:15:51.829831
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 1.1079794197993458, Validation Loss: 1.1912962529394362
Epoch 250, Train Loss: 0.014494443807298571, Validation Loss: 0.011916393102186689
Epoch 500, Train Loss: 0.007509819038925857, Validation Loss: 0.006059714815853146
Epoch 750, Train Loss: 0.0051475369991407565, Validation Loss: 0.004090658690430278
Epoch 1000, Train Loss: 0.003959916359733246, Validation Loss: 0.0031034797386894747
Epoch 1250, Train Loss: 0.0032420554400275493, Validation Loss: 0.002512828399832587
Epoch 1500, Train Loss: 0.0027638203167906967, Validation Loss: 0.002118508220793841
Epoch 1750, Train Loss: 0.0024212641227703917, Validation Loss: 0.0018369166934365426
Epoch 1999, Train Loss: 0.0021637898209828565, Validation Loss: 0.0016257855497699184
Training completed at 2019-04-28 13:05:58.845381, taking: 6607.01555 seconds. Test error: 0.0001418299887316405 (vs baseline: 8.801816875347868e-05)
```



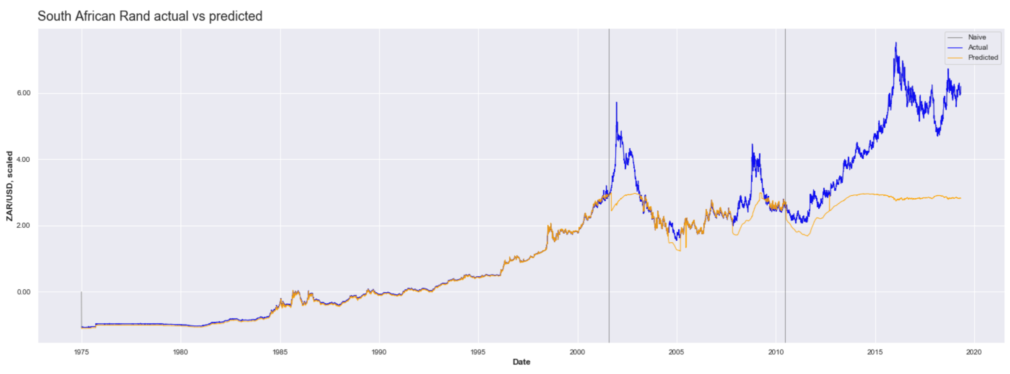


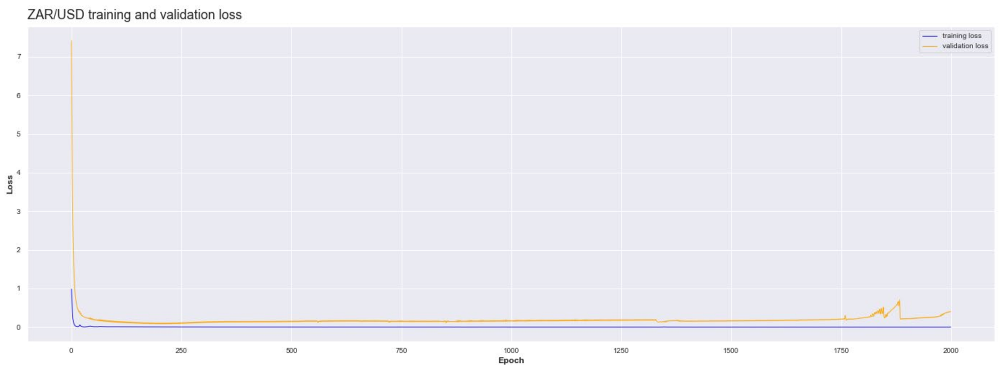
7) South African Rand

South African Rand is our worst model. It overfits and does not generalize well. More recent data may help here, but it also appears to memorize the simple rise in the training data and not learn the underlying function.

Training happens very quickly and levels off for the remaining epochs. Validation loss is increasing at the end of the run, highlighting how we overfit the data.

```
Predicting ZAR, daily


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 365 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 2000 epochs - start time: 2019-04-28 13:42:31.832584
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 0.9841609414016722, Validation Loss: 7.412496937645806
Epoch 250, Train Loss: 0.018773370579422605, Validation Loss: 0.2573138707017072
Epoch 500, Train Loss: 0.01068401445942608, Validation Loss: 0.1960654010602285
Epoch 750, Train Loss: 0.007703759131036293, Validation Loss: 0.182139840228398
Epoch 1000, Train Loss: 0.006123908036954011, Validation Loss: 0.17457367351054606
Epoch 1250, Train Loss: 0.005130230589767989, Validation Loss: 0.17407830654367065
Epoch 1500, Train Loss: 0.004439140218884069, Validation Loss: 0.17243212584950304
Epoch 1750, Train Loss: 0.003930031880671505, Validation Loss: 0.17318886698421473
Epoch 1999, Train Loss: 0.0035388330009456224, Validation Loss: 0.18825505185726474
Training completed at 2019-04-28 14:19:00.599109, taking: 2188.766525 seconds. Test error: 4.794680006802082 (vs baseline: 0.004028945695608854)
```





Monthly LSTM Results

In an additional attempt to beat the naïve model, and to enable a trading strategy, we model next month's price. Modelling next month's price gives us a long enough time horizon to make a significant return, while creating a strategy that can be executed manually. However, we reduce the accuracy of our model.

Improving this accuracy is the focus of our next steps, which we will then iterate a monthly trading strategy on using portfolio optimization theory.

To perform this task, we take the price on the first day of each month. We use the prior 12 months to predict, and train over 500 epochs. All other parameters remain the same as above:

- Test size = 20%
- Hidden size = 10 nodes
- 1-layer LSTM
- Mean Absolute Error Loss
- Adam Optimizer
- Learning rate = 0.001

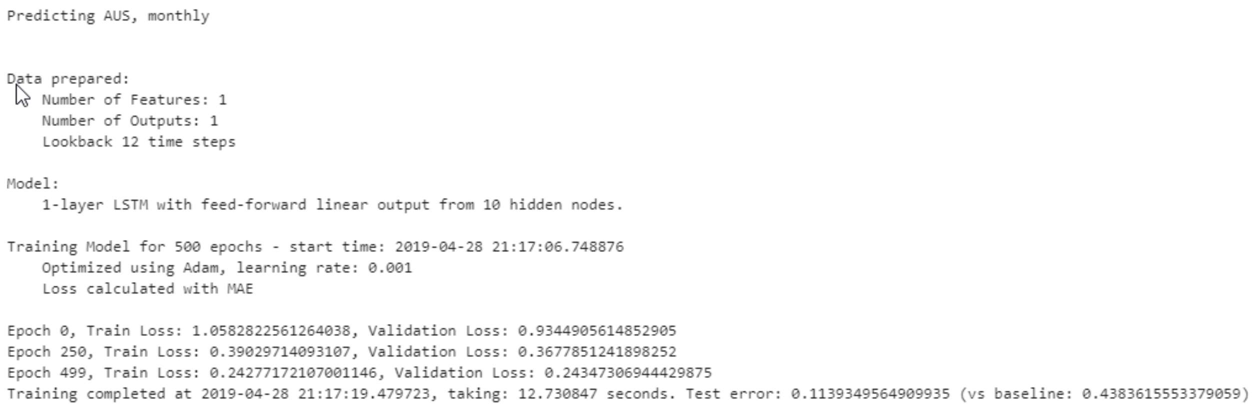
1) Australian Dollars, Monthly

On a monthly basis, Australian dollars beat the naïve model. The training and validation loss are similar, but well above the test loss.

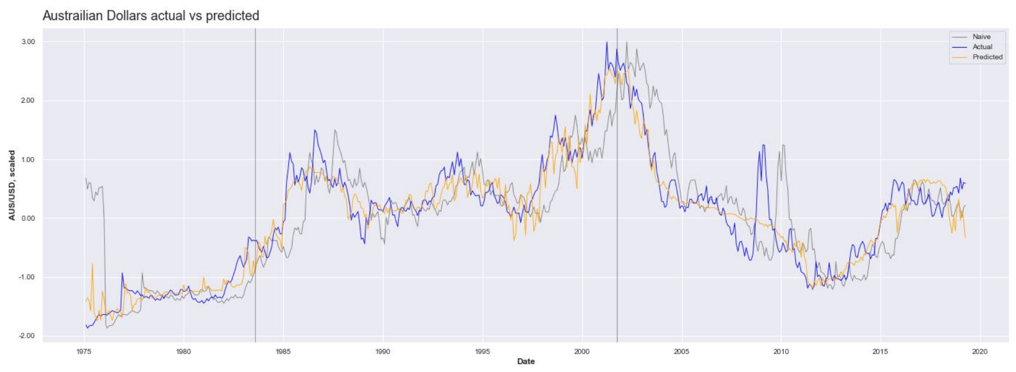
- Train Loss = 0.24, Val Loss = 0.24
- Test Loss = 0.11
- Baseline = 0.44

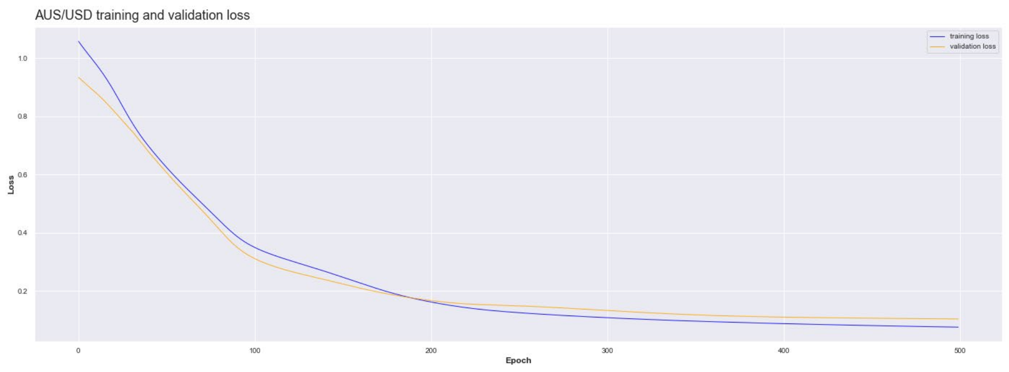
While imperfect, our model does pick up some of the nuance of jumps in the price of Australian Dollars with respect to US Dollars.

The training plot shows a gradual decrease in loss over the number of epochs and is decreasing through the end. Expanding the number of epochs will increase accuracy.



```
Predicting AUS, monthly


Data prepared:
    Number of Features: 1
    Number of Outputs: 1
    Lookback 12 time steps

Model:
    1-layer LSTM with feed-forward linear output from 10 hidden nodes.

Training Model for 500 epochs - start time: 2019-04-28 21:17:06.748876
    Optimized using Adam, learning rate: 0.001
    Loss calculated with MAE

Epoch 0, Train Loss: 1.0582822561264038, Validation Loss: 0.9344905614852905
Epoch 250, Train Loss: 0.39029714093107, Validation Loss: 0.3677851241898252
Epoch 499, Train Loss: 0.24277172107001146, Validation Loss: 0.24347306944429875
Training completed at 2019-04-28 21:17:19.479723, taking: 12.730847 seconds. Test error: 0.1139349564909935 (vs baseline: 0.4383615553379059)
```





2) Canadian Dollars, Monthly

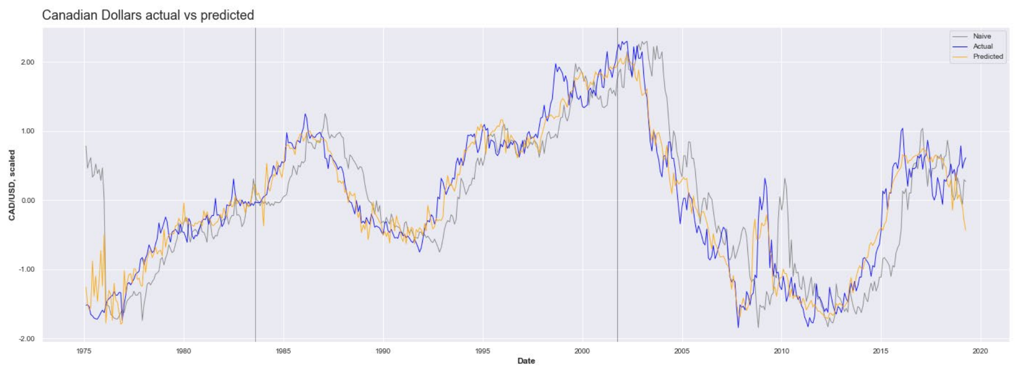
Canadian dollars beat the naïve model. The training and validation loss are similar, but well above the test loss.

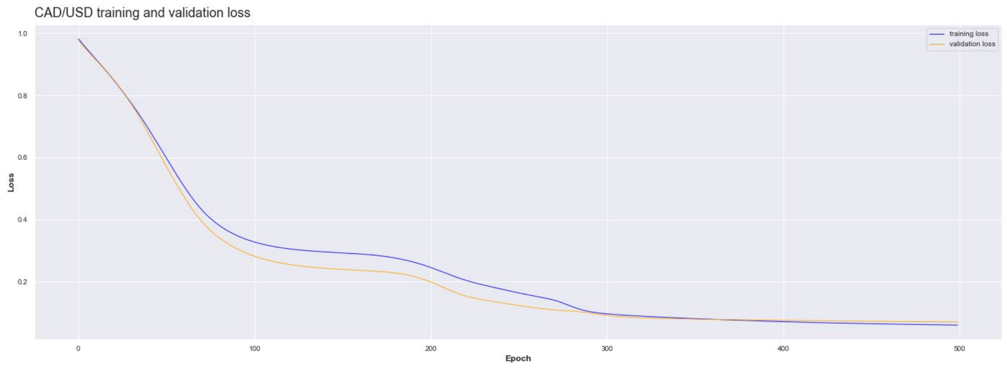
- Train Loss = 0.24, Val Loss = 0.22
- Test Loss = 0.08
- Baseline = 0.41

While imperfect, our model does pick up some of the nuance of jumps in the price.

The training plot shows a gradual decrease in loss over the number of epochs and is decreasing through the end. Expanding the number of epochs will increase accuracy.

```
Epoch 0, Train Loss: 0.982248326142629, Validation Loss: 0.976020097732544
Epoch 250, Train Loss: 0.40286983717801883, Validation Loss: 0.36441444350310054
Epoch 499, Train Loss: 0.24444253457834325, Validation Loss: 0.22420657202601432
Training completed at 2019-04-28 21:17:34.578147, taking: 13.110794 seconds. Test error: 0.08004167675971985 (vs baseline: 0.4081822633743286)
```



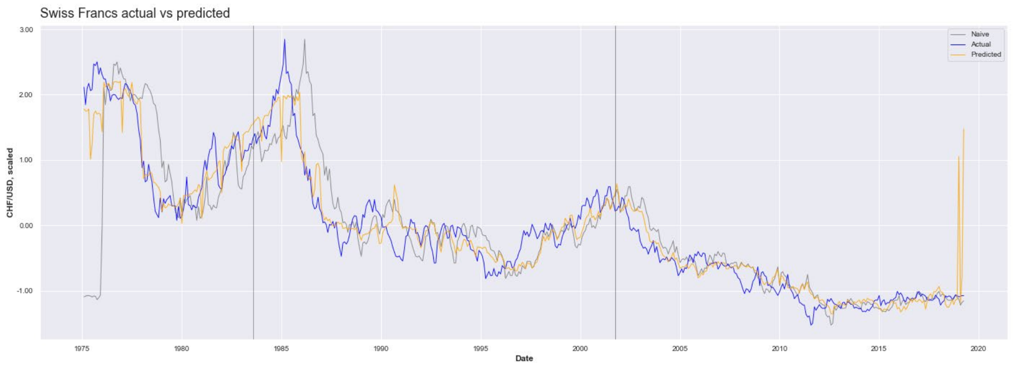


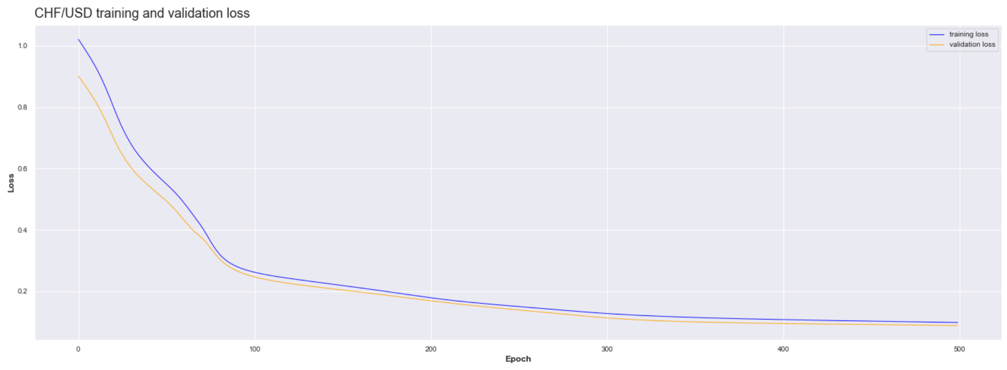
3) Swiss Franc, Monthly

Swiss Franc beats the naïve model. It shows some strange predictions near the end of the test set, requiring further exploration. The training and validation loss gradually decreases through each epoch, showing that more epochs will improve our prediction.

- Train Loss = 0.23, Val Loss = 0.21
- Test Loss = 0.08
- Baseline = 0.37

```
Epoch 0, Train Loss: 1.0207653045654297, Validation Loss: 0.9008913040161133
Epoch 250, Train Loss: 0.3489297027211107, Validation Loss: 0.3174342202594081
Epoch 499, Train Loss: 0.23257190070052944, Validation Loss: 0.21044793090224265
Training completed at 2019-04-28 21:18:34.813430, taking: 13.199588 seconds. Test error: 0.0803111344575882 (vs baseline: 0.36555740237236023)
```



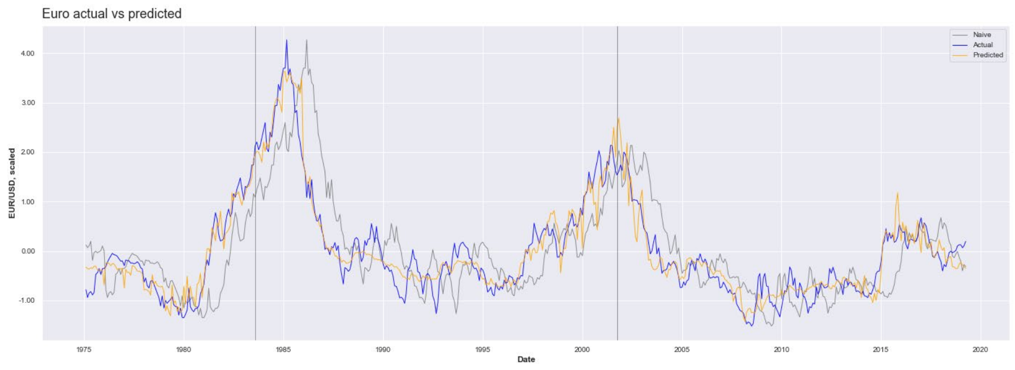


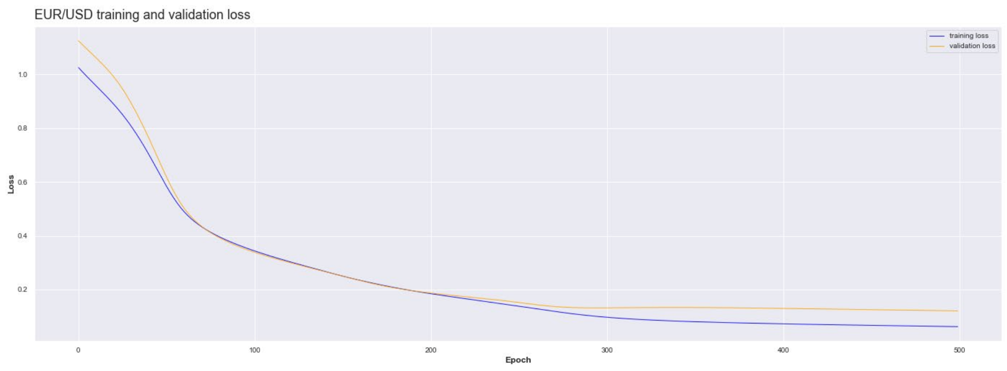
4) Euro, Monthly

Euro beats the naïve model and shows strong predictive power through the swings in price. The training plot shows a gradual decline through the epochs and is still decreasing near the end.

- Train Loss = 0.24, Val Loss = 0.27
- Test Loss = 0.15
- Baseline = 0.42

```
Epoch 0, Train Loss: 1.0250035325686138, Validation Loss: 1.1243358850479126
Epoch 250, Train Loss: 0.38693838097540983, Validation Loss: 0.4058036627285034
Epoch 499, Train Loss: 0.23543201071023942, Validation Loss: 0.2687499915212393
Training completed at 2019-04-28 21:18:49.903876, taking: 12.90335 seconds. Test error: 0.14595335721969604 (vs baseline: 0.424247682094574)
```





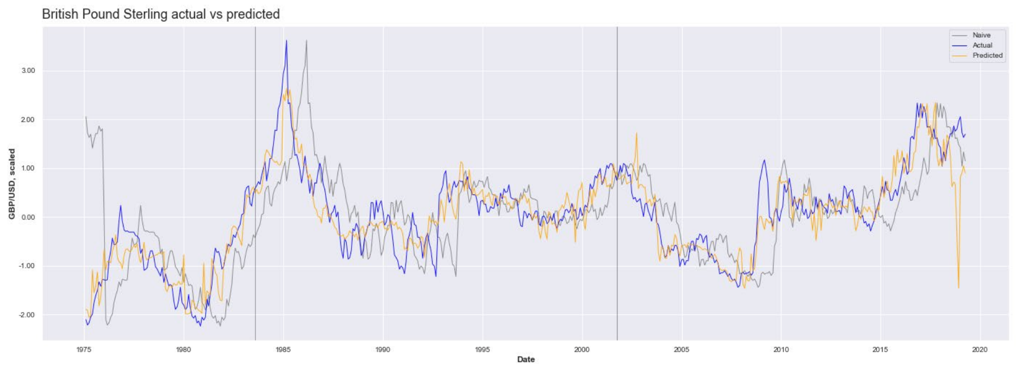
5) Great British Pounds, Monthly

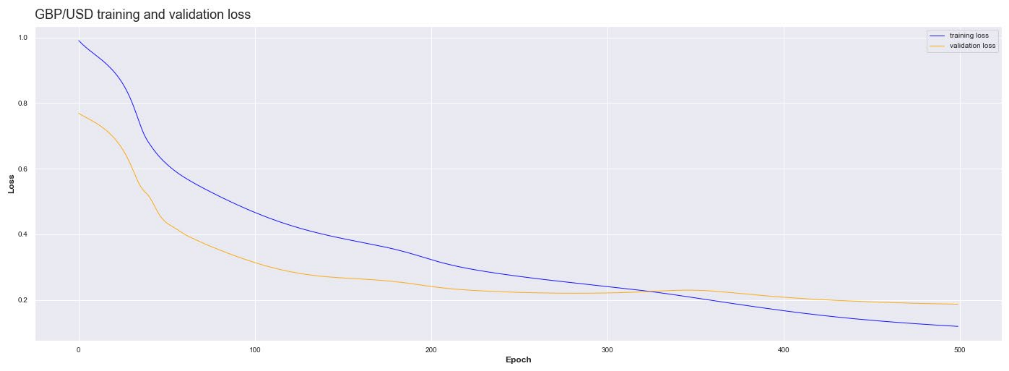
Great British Pounds beats the naïve model, but shows lower predictive power than the other models. Test error is much higher than validation and train. The large swings at the end of the test set appear to be responsible for the large test loss.

The training plot shows that validation error is not decreasing at the same rate at the training error, indicating there may be opportunity to optimize the training hyper parameters.

- Train Loss = 0.34, Val Loss = 0.28
- Test Loss = 0.41
- Baseline = 0.78

```
Epoch 0, Train Loss: 0.9908380508422852, Validation Loss: 0.7690363526344299
Epoch 250, Train Loss: 0.4865972307692486, Validation Loss: 0.3540098104343946
Epoch 499, Train Loss: 0.33846748413145544, Validation Loss: 0.2829964399933815
Training completed at 2019-04-28 21:18:04.630447, taking: 12.936262 seconds. Test error: 0.40815994143486023 (vs baseline: 0.7834954261779785)
```





6) Japanese Yen, Monthly

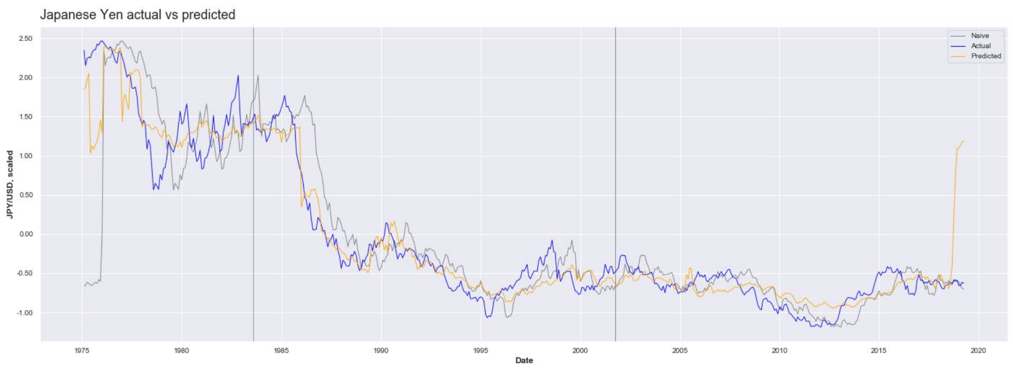
Japanese Yen beats the naïve model. The validation loss is halfway between the test loss and training.

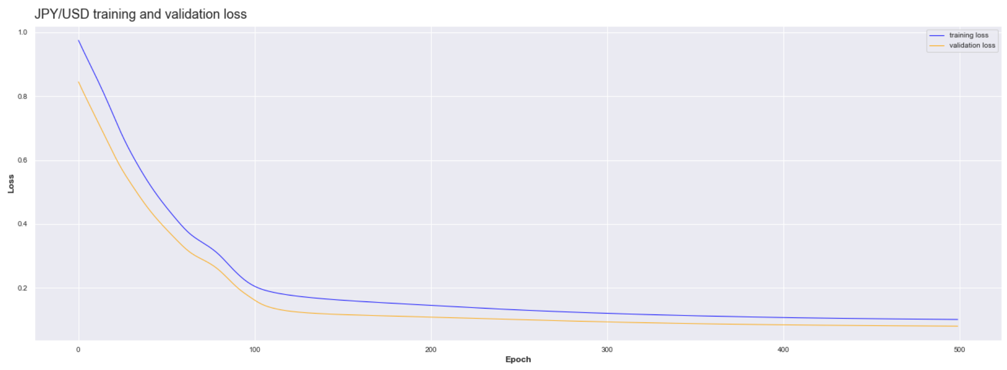
- Train Loss = 0.20, Val Loss = 0.16
- Test Loss = 0.12
- Baseline = 0.27

At the end of our training data, we see some unacceptable jumps in the prediction. These will need to be explored further to identify the root cause.

The training plot shows a gradual decrease in loss over the number of epochs and is decreasing through the end. Expanding the number of epochs will increase accuracy.

```
Epoch 0, Train Loss: 0.974874754746755, Validation Loss: 0.8449293971061707
Epoch 250, Train Loss: 0.29670964583141074, Validation Loss: 0.24071864510674876
Epoch 499, Train Loss: 0.20456986664732296, Validation Loss: 0.16439435255527496
Training completed at 2019-04-28 21:17:49.659682, taking: 13.007106 seconds. Test error: 0.12166959047317505 (vs baseline: 0.2723158895969391)
```



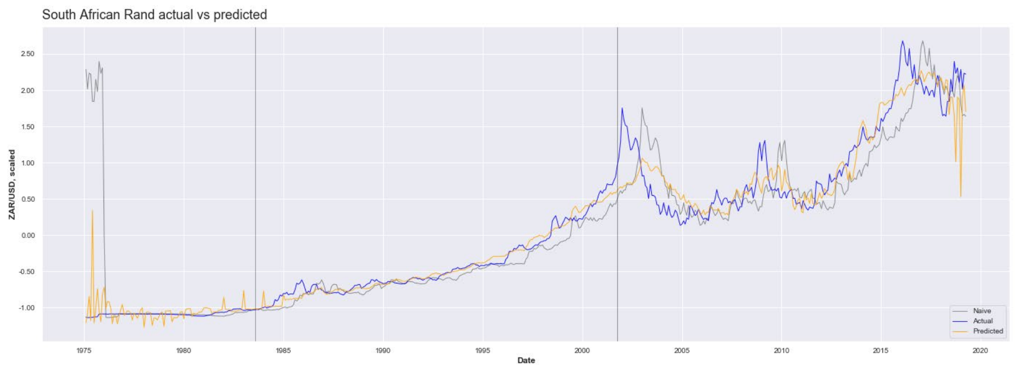


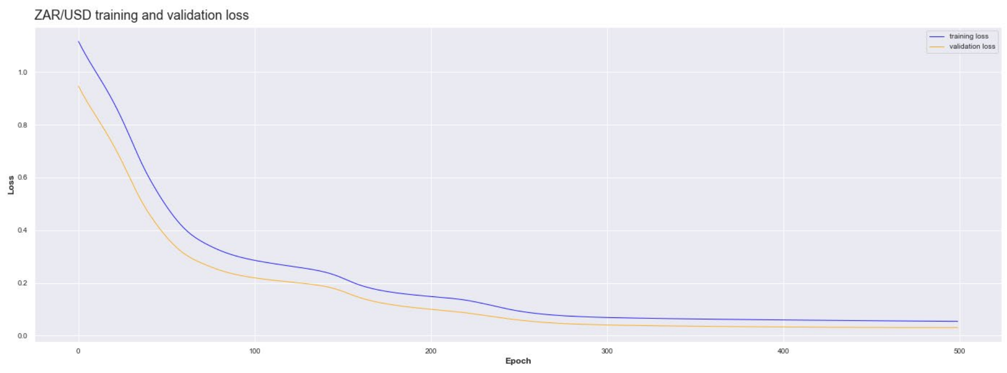
7) South African Rand, Monthly

South African Rand also beats the naïve model, and shows unexpectedly accurate test results. The training loss is the worst, followed by the validation loss. The plot shows the predicted value performs poorly for recent data. The training plot shows a good gradual decrease in training and validation loss, confirming solid parameter choice.

- Train Loss = 0.20, Val Loss = 0.15
- Test Loss = 0.06
- Baseline = 0.30

```
Epoch 0, Train Loss: 1.1163103183110554, Validation Loss: 0.9466506242752075
Epoch 250, Train Loss: 0.3437796458700897, Validation Loss: 0.26632237527890507
Epoch 499, Train Loss: 0.2041646212823689, Validation Loss: 0.1517663895227015
Training completed at 2019-04-28 21:18:19.479573, taking: 12.665988 seconds. Test error: 0.058195389807224274 (vs baseline: 0.30056634545326233)
```





Conclusion

In the end, problem definition provided better results than optimizing the model. However, we do still have opportunity to increase our accuracy for recent data. In order to achieve our original goal of creating an optimal portfolio, accurate predictions multiple time steps into the future are needed. Hyperparameter optimization shows promise in finding optimal solutions and avoiding poor local optima. Predicting one day ahead shows little promise in beating the naïve model.

We find it interesting that exogenous variables do not improve the model, but it follows that we are unable to capture the range of information in an efficient market. We are happy with the results of the LSTM looking forward one month, and look forward to implementing our next steps.