Recurrent Neural Network for Gold Price Prediction

Jevani Nikitha Chittaluri
University Of Texas at Dallas
Richardson, USA
nikitha.chittaluri@gmail.com

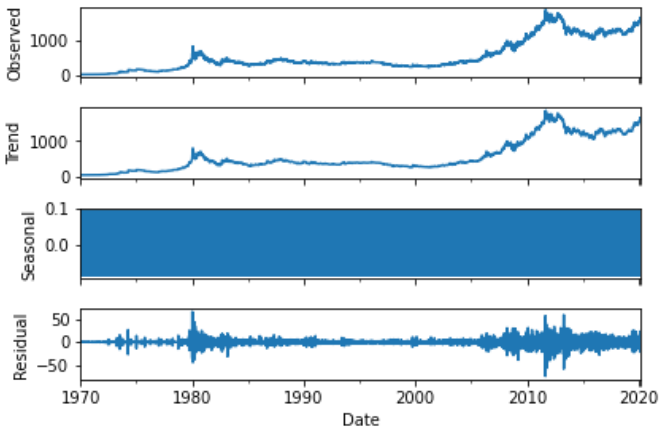
Shreeya Thatipalli
University Of Texas at Dallas
Richardson, USA
shreeyathatipally@gmail.com

***Abstract*—This project is an implementation of Recurrent Neural Network model for Time Series Prediction on Gold price Dataset. Most people consider investing in gold as good investment. Where many people could use our model to check what can be the gold rate in future. Gold price graph is not linear, so Convolutional Neural Networks can be used to solve non linearities in our data and among them Recurrent Neural Network is proved to suitable for any time series prediction. We have obtained gold Price dataset from Kaggle.**

Keywords— time series prediction, Recurrent Neural Network

I. Introduction

Gold is a precious metal and is sensitive to price changes. We call gold price prediction as time series forecasting because time series can be assumed as a set of sequential data points which are placed in an order and are collected at a regular time intervals. So, this approach of time series can be used on any dataset where value of our dependent attribute changes with change in time. Time series models are found to be effective in cases where our independent attribute values depend on the factor of uncertainty in the future. Time Series analysis process decomposes the data into Trend of the data, Seasonality of the data, cyclic and residual component of the data.



There are various techniques and methods that can be used to predict gold Price. For example, ARIMA (auto regressive integrated moving average), ARMA (auto regressive moving average) etc., But all these methods can only work on Linear Data, but they can't help us with nonlinear data. So, we must use Artificial Intelligence models to predict our data. Methods like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) Models are widely used in this area. Among all these, we have chosen to use Recurrent Neural Networks (RNN).

II. Background Work

Over the period of time different researchers have developed different model for gold price prediction. Parisi et al. has developed gold rate prediction model using iterative and rolling neural network model in 2008. In 2012 Yazdani Chamzini et al. has created a model to predict gold price using Artificial Neural Networks and Adaptive Neural Inference system. Which then was compared to ARIMA model. In 2014 Li used new artificial bee colony algorithm with wavelet neural network to predict gold prices. In 2016 Xian et al. created a gold price prediction model using ensemble empirical model decomposition and also performed independent component analysis. And Gangopadhyay et al. used a new model called vector error correction model to predict the gold prices. In 2016, Sivalingam et al. developed an extreme machine leaner where previous prices of silver, gold, oil, S&P 500 index and exchange rate were used for training. In 2019, Jianwei et al. has introduced a new combination of independent component analysis and Gated Recurrent Unit Neural Network Methods.

Alameer et al. used advanced multilayer perceptron neural network and whale optimization algorithm to predict gold prices. In 2020, Zhang and Ci used the deep belief networks. And Weng et al improve extreme machine learning by reducing the effect of randomness on prediction results. He achieved this with the use of genetic algorithm.

Current State value can be calculated using the below formula,

```
Current_state = f(previous_state, input_state)
```

If we want to use tanh as activation function, we can use below formula,

But, in our model we are using sigmoid as activation function using the below function,

```
def sigmoid_activation(self,i):
        return 1 / (1 + np.exp(-i))
```

In RNNs trained on long sequences the gradients will easily explode or vanish. This can be avoided by initializing weights carefully. Sometimes, with good initial weights also when we are dealing with long range dependencies.

While back propagating through many layers, if weights are small, gradients will shrink exponentially and vanish. This is called Vanishing gradient problem.

If weights are large gradients will grow exponentially.

a) Overfitting: Sometimes, we encounter a problem of overfitting while training our model. We can detect this by observing training and validation loss. If our validation loss is much higher than the training loss then we should understand that out model is overfitting. Then we can reduce our network size to overcome this. Or we can even increase the dropout to avoid this.

b) Underfitting: Sometimes, we encounter a problem of underfitting while training our model. We can detect this by observing training and validation loss. If our training and validation loss are almost equal then we should understand that our model is underfitting. We can overcome this by increasing number of neurons per layer or increase the number of layers.

III. Dataset

We are using gold price dataset from Kaggle. Data Statistics of our dataset are as follows.

[10787 rows x 2 columns]

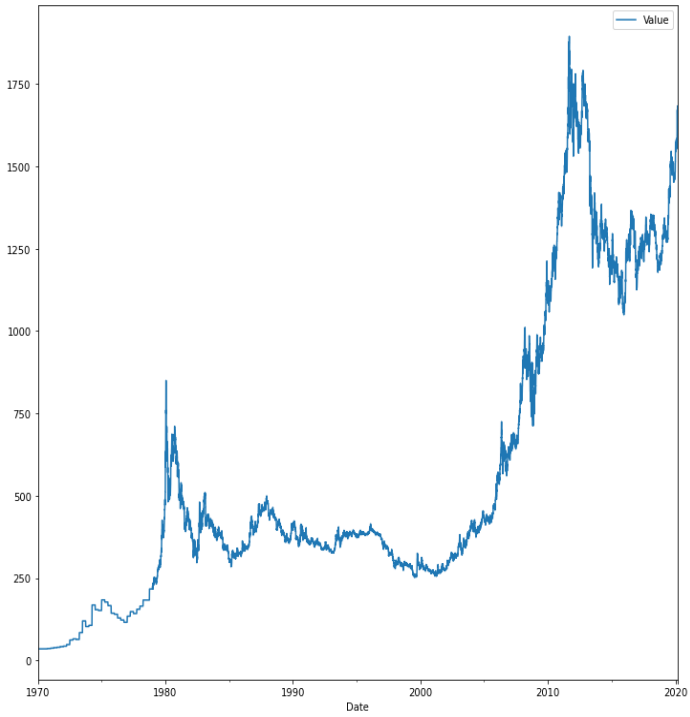
	Value
count	10787.000000
mean	653.596634
std	434.030848
min	35.100000
25%	349.200000
50%	409.350000
75%	1061.625000
max	1895.000000

We are having 2 columns date and value. We are making date as index and giving frequency as Day. But in our dataset few dates are missing. We are filling them using ffill i.e., front fill. For Example, if 31st December 2001 value is missing we are giving it same value as 30th December 2001. We are also filling null values with ffill.

Here our other options were back fill or average value. We can even pick back fill here, which when we tried also gave same set of results as of front fill. But we are against using average value here because gold value either increases or decreases to know that trend at a period of time would be hard if we use average value. So, if we use front fill or back fill our trend wont get disturbed. Whereas if we use average value for every missing value we would end up with a disturbed trend graph.

Later on while training the data we are standardizing our dataset using MiniMaxScaler which would convert our values ranges between 0 and 1. We are directly importing this library into our program.

Below is the plotted graph of Date vs Value for our dataset,



IV. RMSE

Root Mean Square Error (RMSE) is a standard measure of error in a model.

Root Mean Square Error can be calculated using the below formula, As RMSE is a measure of error we want it to be as low as possible.

So, we can scale accuracy of our model by looking at RMSE values. If they are high, that means that our model is having less accuracy. If RMSE values are low, then our model is having high accuracy.

RMSE is also defined as the standard deviation of all the residuals. Which is a measure of how spread-out data is.

Value of standard deviation is the square of variance value. Where variance is the average of squared differences from the mean value.

Residual means prediction error. Which can be easily calculated by subtracting the predicted value from the actual value.

In RMSE, errors are squared before calculating their averages. Therefore, it gives large weight to high errors.

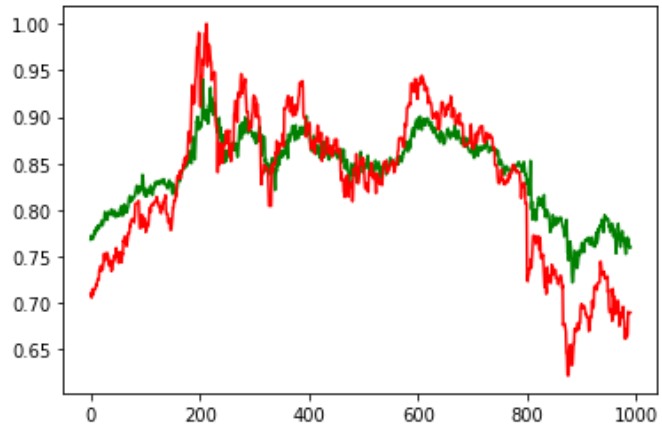
V. Results and Analysis

We are taking first 16,000 records as train data and rest all as validation data. While training with different hyper parameters we were getting different issues. Sometimes, it's over fitting and sometimes under fitting.

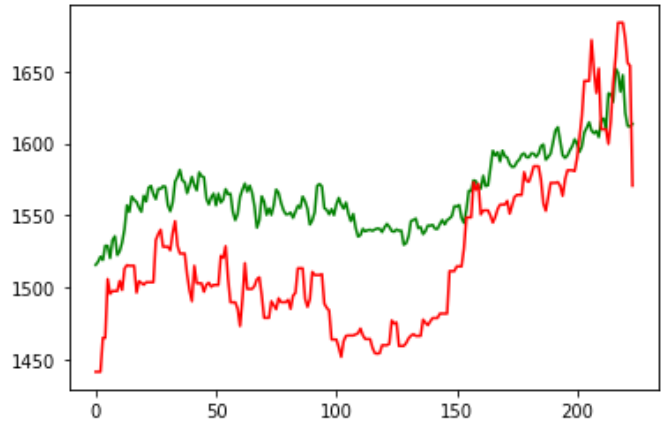
To avoid gradient explosion, we are keeping some limits. Which are also considered as hyper parameters.

After altering our hyper parameters best model we were able to create gave the below results for training and validation set respectively.

For Training Data:



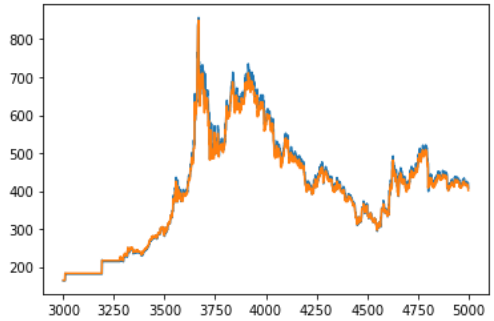
For Validation Data:



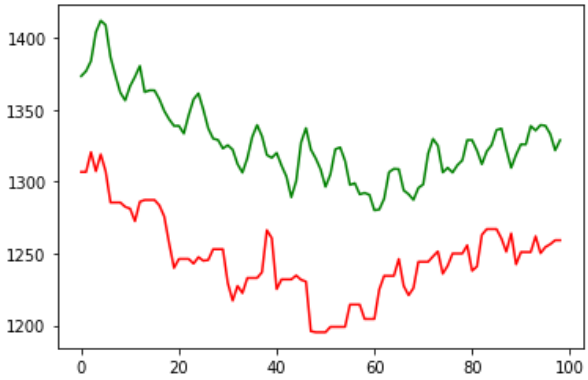
RMSE value we got for this is 0.00821.

When we gave a higher learning rate, we encountered a problem of overfitting where graph for training, looked as below, a clear example of overfitting because our test set has huge residues,

Graph for Training Data



For Validation Data:



Clear Case of overfitting.

VI. Conclusion and Future Work

Here we are only using one independent attribute i.e., date. But research shows that gold rate depends on different values like oil price, stock price, inflation rate and so on. So we can extend our project by adding more independent attributes. And definitely using LSTM model with different layers.

VII. References

[1] https://www.analyticsvidhya.com/blog/2019/01/fundamentals-deep-learning-recurrent-neural-networks-scratch-python/?#

[2] https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e