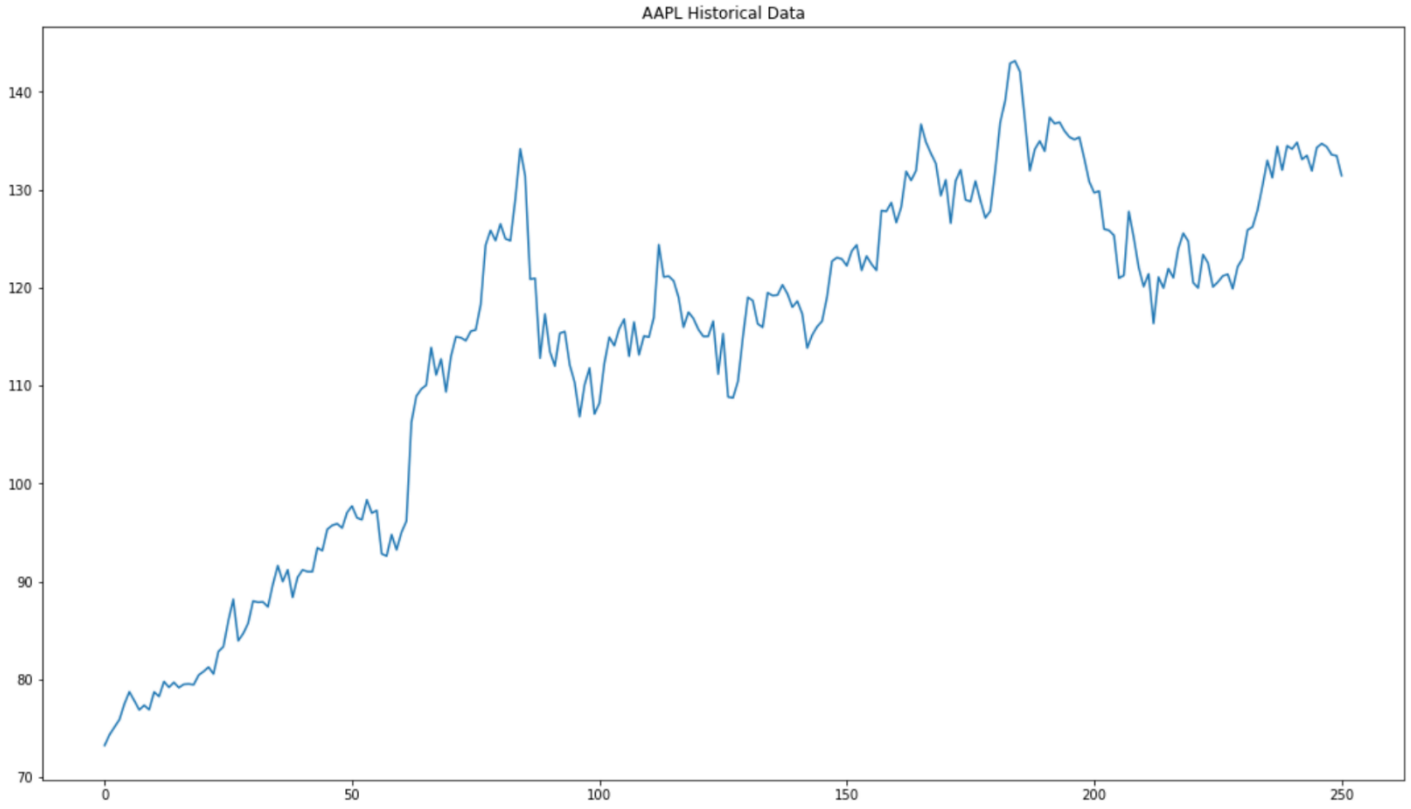
Scientific Computing Project 4: Neural Network Stock Predictor

Brian Livian

Introduction

Neural networks is a powerful technique used in numerical computing. It can be applied to many areas in data science to make predictions based on training data. In this project, I use a neural network via the TensorFlow/ Keras python library to predict the price of apple stock. I use a long short term memory neural network with parameters for gradient descent and mean square error to fit the data for a prediction. An illustration of a AAPL's historical close price is shown below.



Methodology

To obtain AAPL's historical data, I downloaded a csv of the company's stock price via yahoo finance. To make the program reproducible, I upload the data into my GitHub repository and import the data into a pandas dataframe. The data shows the stock's price between the dates 5/4/2020 to 4/26/2021. The historical data has columns for open, high, low, close, adjusted close, and volume. There are 251 rows in this dataframe, correlating to 251 trading days of AAPL stock. For this neural network, I use the average of Apple's Open and Close price as data to train the neural network. The average of this dataset is $114, with the standard deviation being 17.8.

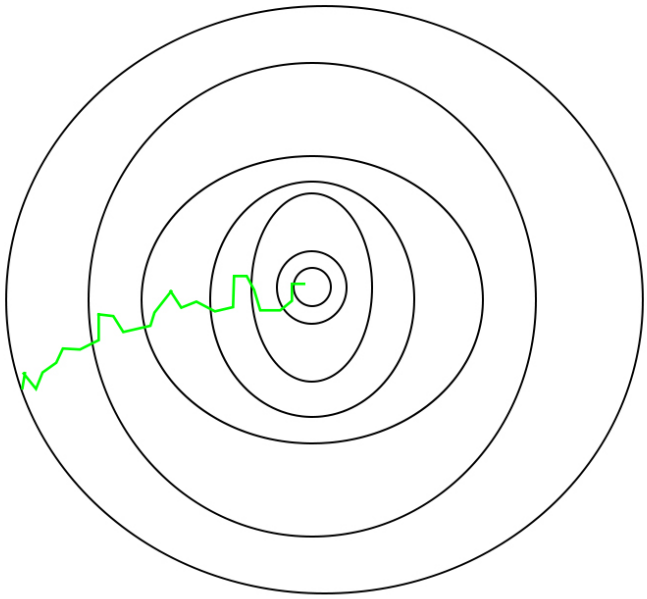
To train the neural network model, I first convert AAPL's price data to a NumPy array of values. Then, I normalize the array, allowing the model to have optimal performance. After normalizing the historical data, I split the data into a training set and a testing set. Finally, I populate the training and testing data into arrays X and Y. These arrays are separated by 3 days, allowing the neural network to take input data based on prices spaced 3 days apart.

Now that the training and testing data is assigned, I create the neural network via the Keras library. I use Keras to compile a long short term memory neural network with a hidden layer of 10 neurons using the tanh activation function. I then create the output layer for the network via Dense. The neural network measures accuracy based on the mean squared error of the test data. I design the network to optimize based on stochastic gradient descent, with 100 iterations (epochs). In other words, the neural network optimizes its prediction by minimizing the mean square error of the training data. The program updates the appropriate weights and biases of the input layer automatically for each step in the gradient descent. A screen snip of the code is shown below, as well as a generic illustration of gradient descent:

```
# Create neural network model via keras
model = Sequential()
# Create lstm layer of 10 neurons and tanh activation function
# Hard sigmoid recurrent activation by default
model.add(LSTM(10,activation = 'tanh'))
```

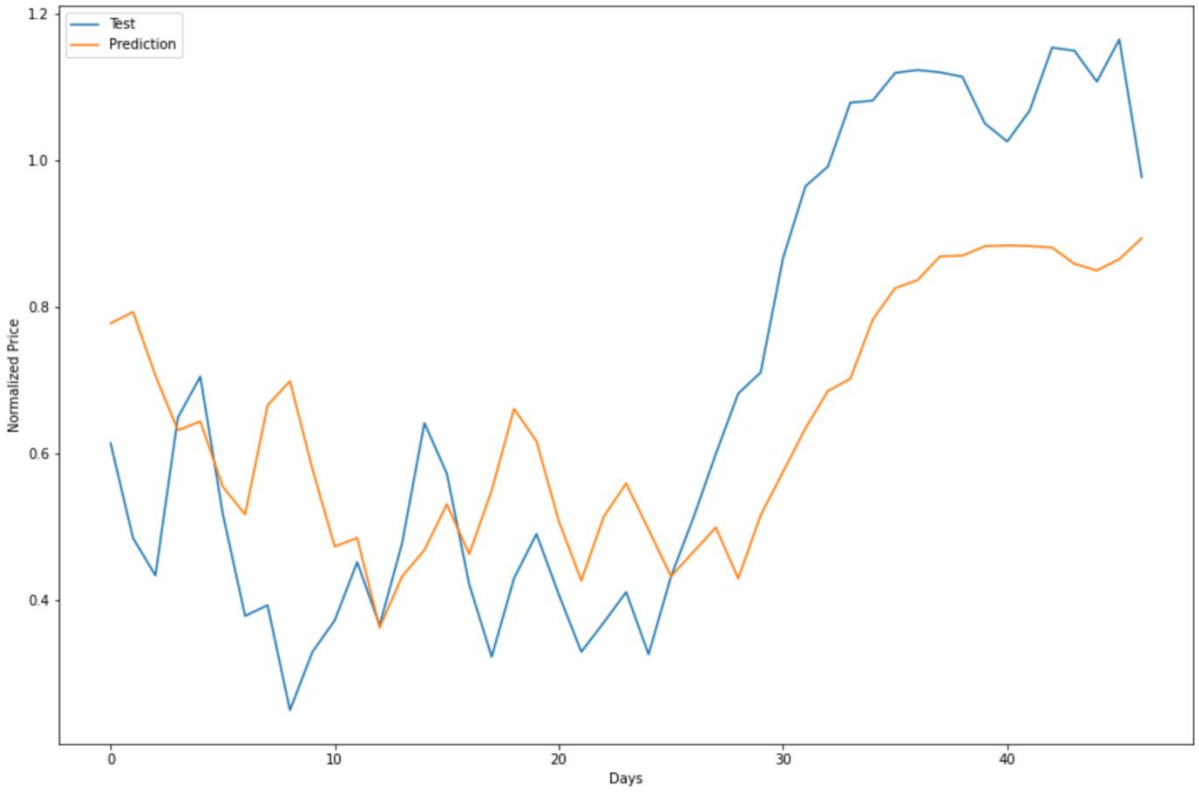
```
# Add dense layer
model.add(Dense(1))
```

```
# Compile model with metrics for mean square error and stochastic gradient descent
model.compile(loss= 'mean_squared_error',optimizer = 'SGD')
model.fit(X_train,Y_train, epochs=100)
Predict = model.predict(X_test)
```

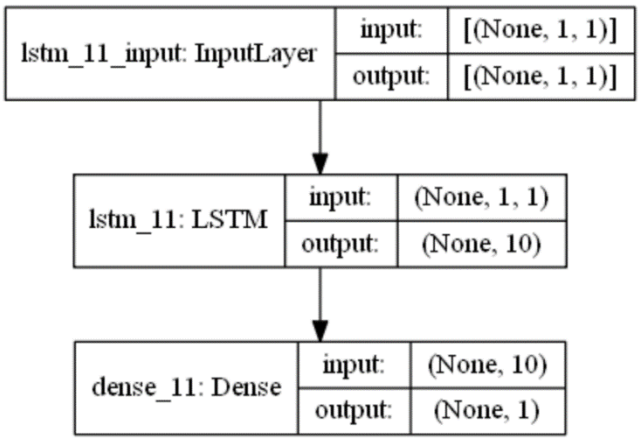


Computer Simulation and Result

Below is a graph comparing the test data to the predicted price. Initially the prediction is very accurate. As time goes on, the prediction becomes less accurate, yet still predicts the positive trend correctly.



Below is a visual describing the neural network model.



Conclusion

Neural networks is a strong tool in machine learning/ scientific computing to make prediction based on an input of training and testing data. In this project, I use a long short term memory neural network with the Keras library to predict the change in stock price in Apple. Although a neural network with weights and biases can be constructed manually through NumPy, I chose to explore the Keras due to its simplification of neural networks and its application to possible projects of mine in the future. The example of the prediction shown in this project takes an input of training data (X and Y) spaced 3 days apart. The training data representing 80% of the total data, while the testing data represents 20%. Note that these metrics can easily be changed through the python file. These metrics inputted into the neural network produce a fairly accurate prediction, especially at the begging of the graph. As time goes on, the prediction becomes less accurate, however it still correctly predicts the positive change in apple stock.

References Cited

[1] Keras documentation (LSTM), https://keras.io/api/layers/recurrent_layers/lstm/

[2] Keras activation documentation, https://keras.io/api/layers/activations/

[3] Keras losses documentation, https://keras.io/api/losses/

[4] Keras optimizers documentation, https://keras.io/api/optimizers/

[5] Marianne Benkamoun, Time Series Prediction Using Reccurent Neural Networks, https://medium.com/@marianne.benkamoun/stock-price-prediction-using-recurrent-neural-networks-369c21817da8