DAV Team Technical Assignment

Pratik Sahoo
22B0968

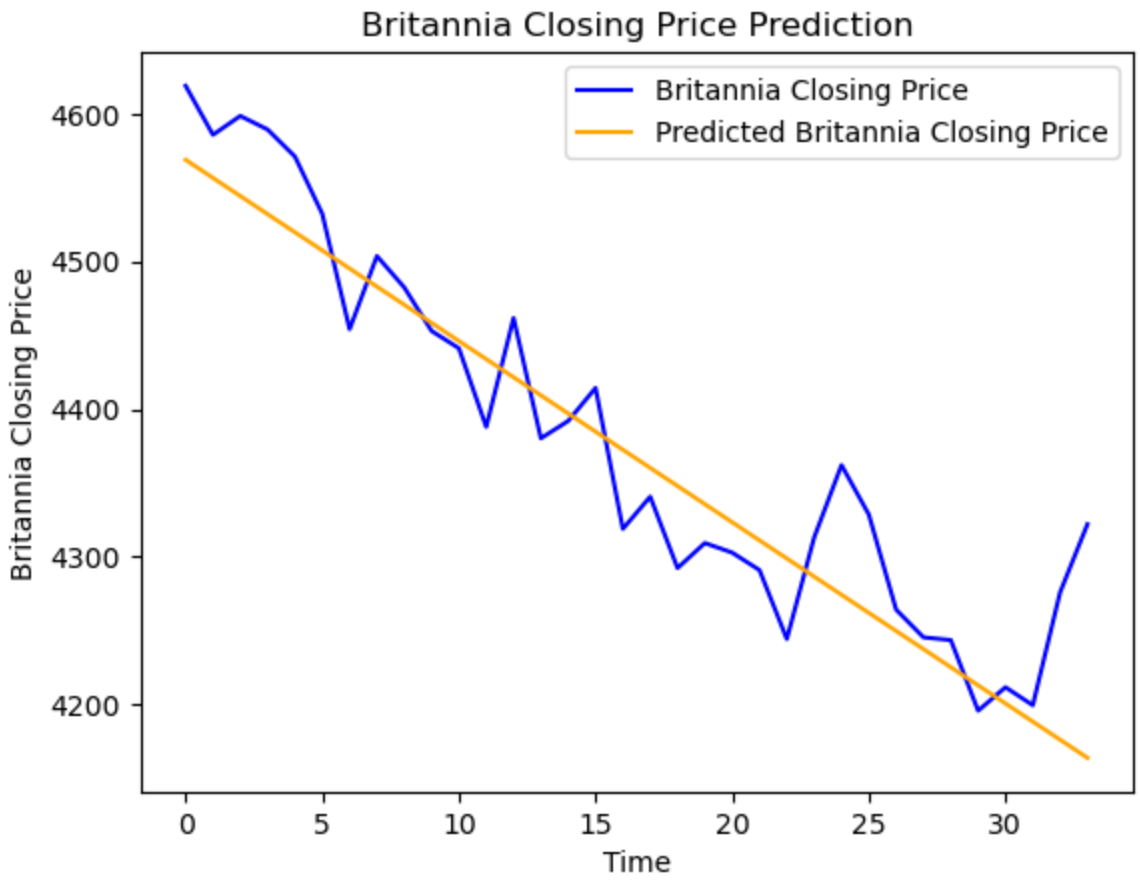
Q2]

In my Q2.ipynb file, there are 2 approaches outlined. First is a deep learning approach (which i found at https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233). I implemented this because I was interested in how it worked, as my intuition told me that using deep learning to predict a stock price would result in heavy overfitting. However, it seems that the dropout layers prevent that, as the result matches the stock trends nicely.

However, I realized that this model is not what was required, because the problem statement asks for us to predict the upcoming month's prices using whatever data is already given, while this model only predicts the next day's values. I thought of bootstrapping and building the next month's data day-by-day, but I realized that using estimates to make other estimates in a recursive manner would only amplify the errors in the estimations, until it no longer resembled the actual data, especially since we would be bootstrapping dozens of layers deep.

How do we predict, given previous prices of a stock, the upcoming prices for a *month* with absolutely no idea of what happens in that month? Well, we can't really predict with any reasonable accuracy since we lack real-world data in the required time frame. However, the most basic solution, which seems like the best considering we have no other data, is to assume the stock will continue to move the way it currently is. Hence, I decided to fit a linear regression model onto the data (with time as a feature), to find the current trend and extrapolate it.

That is exactly what I did, and while it should (in theory) give reasonable results, the last day of given data (9th Feb 2023) is a day of reversal of the prices for Britannia Industries, ruining any linear model I ran on it. To make the results more accurate, I committed one of the greatest sins a data scientist can do: hand-pick the data. I understand that this is something we must always avoid; I only did so to show that the linear model can, in most scenarios, act as a reasonable estimator. Hence, I excluded the last day from the training data, and trained it on the 4 days before that, and it gave quite a nice result.



However, this can never be a valid example for showing that we should use this method to predict prices, as I hand-picked the data, and also because I got lucky with the number of days I measured the trend over (using 10 instead of 5 predicts the opposite trend).

Using Mean Squared Error (MSE) loss, we got a loss of 2368.15, showing that the typical error is of the magnitude ~50 which is quite good considering the prices are ~4500.

Q3]

Clearly, the model generates satisfactory predictions on the training data but fails miserably on the testing data. This is a very clear example of overfitting, indicating that this model will perform quite poorly on real data. Some solutions are discussed ahead.

To reduce overfitting, we can apply regularization on a regression model, or dropout layers on a deep learning model. These are standard ways of dealing with overfitting, and often improve performance on testing data.
If one has more data available, that would also work to reduce overfitting. One may also manufacture synthetic data.

Looking at the diagram given, one can notice that the actual data to be modeled is usually at 0, and has peaks in between. The model has issues with recognizing where the peaks occur, and has false peaks on the testing data. There are 2 approaches that come to mind: either make the dataset (for peaks or 0) less sparse by applying SMOTE (Synthetic Minority Oversampling Technique), or multiply the result of this model with a logistic regressor that identifies whether there is a peak or not, and tune the threshold/shape of the logistic regressor to prevent false positives for peaks. However, these are more complicated techniques that have a higher chance of behaving unexpectedly, and thus more data and regularization seem like the best solutions to try.