Data

Available Features: What available features are interesting to investigate, and why?

As we are interested in the future open price, the two most interesting features might be previous open and close prices (especially the open and close of the previous day). The close the day before is also the closest in time to the open. Volume is a metric of how much the stock is traded that day, where a high number means a lot of activity and therefore probably more volatility. Adjusted close seems to mainly account for dividends, and could possibly replace close price.

Other Features, What other features that are not included or can be inferred from available features would be relevant to include, and why?

The stock related features could possibly be combined to form different financial metrics that could reflect volatility and trend of the price. A lot of information could be retrieved from the Date variable, like how many days since the market was last open (has there been a holiday, or weekend), what day of the week it is, how many days it is after the latest quarterly report of Microsoft, how many days it is after the last big payday in America. These features are at least worth investigating.

Corrupt Data: The data has missing values, how would we go about to handle this?

I would not say the data has any missing values, if this is the case I have missed it. Although, there are a few breaks sometimes between market opens (weekends, holidays etc), with the biggest break being 4 days. In this case I think the best solution is to include a feature that contains the amount of days since the market was last open.

Processing: How should be process the data if we want to train a model that can forecast future daily opening stock-prices?

I don’t think that the next day’s open price is a good target for training since it is extremely dependent on the open price of the previous day. In addition, the higher the stock price is, and the less volatile it is, the more auto-correlated this variable becomes. My solution was to create a target variable of the difference in open price between current day and previous day instead. The data should also be scaled, I opted for standardized scaling, removing the mean and scaling to unit variance, instead of min max scaling since there is no theoretical minimum or maximum value that a stock price can have.

Insights: Besides predicting stock-prices, what other insights can we get from the data and why would these be interesting?

It depends on what insights a customer would want. For investing purposes, it would be interesting to estimate the volatility (variance) of the stock, and the average gain the last few years. If it is less volatile and has higher average gain than similar tech stocks it could be argued that it is undervalued. The volume data could also be used to analyse investor trends, especially in conjunction with relevant dates like quarterly reports or major world events. For a hedge fund, the stocks correlation with other stocks is very relevant, as it could be purchased together with stocks that are negatively correlated with it, to reduce risk.

2. Modeling

2.1 Forecast What model did you use to forecast future opening stock-prices, and can you think of other models you could have used? Mention some pros and cons with your choice.

I used a two layer LSTM pipelined with a dense layer with one neuron. Since it is sequential data recurrent neural networks are very powerful, especially at regression tasks like this one, where all features are numerical (except Date), since you might want to look at longer timespans when analysing (a few weeks at least), I chose the LSTM because it doesn't have a vanishing gradient problem like pure RNNs. The two main problems I see with LSTM is first that the dataset is quite small (a few thousand samples), and secondly that the target variable (of open gain/loss) has a lot of noise (or at least it looks like noise to the model with so few good features to work with), this is a problem since the model will overfit on the data. Dropout and regularization would have been good to implement, although my guess is that that would barely mitigate the problem. Perhaps the LSTM could also be scaled down to just a few units in each layer, while implementing dropout and regularization, and while cross validating to ensure good practice. At this point it is probably worth considering more simple models that are less prone to overfit. Maybe a classic times series SARIMA model could yield as good results as the LSTM and be much more lightweight and interpretable. A lightweight RNN with short memory could also work. Or even a regression SVM as they can be somewhat resistant to overfitting. Linear regression could be a good last resort as well.

2.2 Performance How did you measure the performance of your model and what conclusions can you draw from these measurements?

I mentioned this in the data section, but I trained on the difference between open prices, with mse. I had a small validation dataset to pick the best model checkpoint, and to see the degree of overfitting of the model. I then calculated the rmse of the predictions, and compared it to the rmse of the test dataset of the open price difference. The rmse of the predictions was slightly higher, which meant my model would have lost me money, if deployed on the market.

2.3 Stability Is there anything to be said about the stability of your trained model, for example in terms of volatile predictions?

In my case the model made very small predictions on the price change and generally guessed that the open price would be about the same as the day before. One problem would be if the input data would change quite drastically, then the model could potentially predict more extreme values which could be problematic. This is the reason I used standardization instead of min max scaling, as if I would use the highest open price value as the max value during scaling, and the input data a few days after would surpass this maximum value, the predictions could become unstable.

ETL Briefly explain how you would setup an ETL pipeline for fetching, transforming and storing daily stock data

The best way to fetch the data would probably be some financial services API functions, one way of doing this would be through Apache Airflow, where first a Python task could run at a set interval, and retrieve new data. This task could be connected to another Python script that cleans up this retrieved data, transforms it into a good storable format, and finally inserts it either into some SQL based database, or some simple storage solution like an S3 bucket, depending on how the data looks like, if it does not change from the task, SQL database seems like a good storage solution. Some checks could be good to implement in each step, to make sure that there's no errors, and that the API or database could be properly fetched.

3.2 Model: What tools and programming languages would you use to train & deploy your model, given how you would setuo the ETL pipeline?

This would depend on which platform my team works, and what deployment platform the customer would prefer.

It could be done without cloud tools by using containerized ML models using tensorflow serving with Kubernetes. The model can then be retrained at any time, and tensorflow serving will simply chose the last model trained.

Using cloud, the same thing could be done in for example AWS Sagemaker. A pipeline could be constructed where a Sagemaker notebook could fetch the data from an S3 bucket or AWS SQL database (AWS RDS) and do a prediction whenever there's new data, there's a lot of deployment tools in Sagemaker that could be used.

Training: How can we train the machine learning model and should we set up an automated pipeline for this or is it better to do it manually?

Automated pipeline is better, as the model would need to be re-trained quite often. It could either be done in the cloud with a platform like Amazon Sagemaker, or more locally with tensorflow serving, where airflow could again be used to gather all new data since the model was trained last, and re train the model.

Reporting: How would you report the results from the model such that the a company using it can make quick decisions?

Regression prediction results could be inserted into a database the company has as soon as they are computed. The inference results could also be visualized in a dashboard that they have access to, or updated in some web interface.