Predicting Stock Prices

- Sam Stoltenberg

Questions

I. How can we clean our null littered data for feeding it into a neural network?

II. What feature changes have the most impact on changes in tomorrow's price?

III. What are the limitations faced when predicting stock data?

IV. How can we best build an array of networks for predicting stock data?

About the Data

Scraped from an investment firm consisting of

- Daily data from August 9th, 2019 -> Today
- 7,681 symbols
- Five tables
  - Prices
  - Analyst Opinions
  - Performance Stats
  - Company Information
  - Stock Split Data

Scrubbing the Data

- All choices made during the scrubbing process have a direct effect on the network's predictions, thus they and their effects must be carefully thought out.
- We remove many features and symbols where data could potentially be negative, as filling it with zero could have drastic consequences.
- After scrubbing we ended up with roughly 2,000 symbols, and predicted with only 7 from the S&P 500.

Scrubbing the Data Part 2

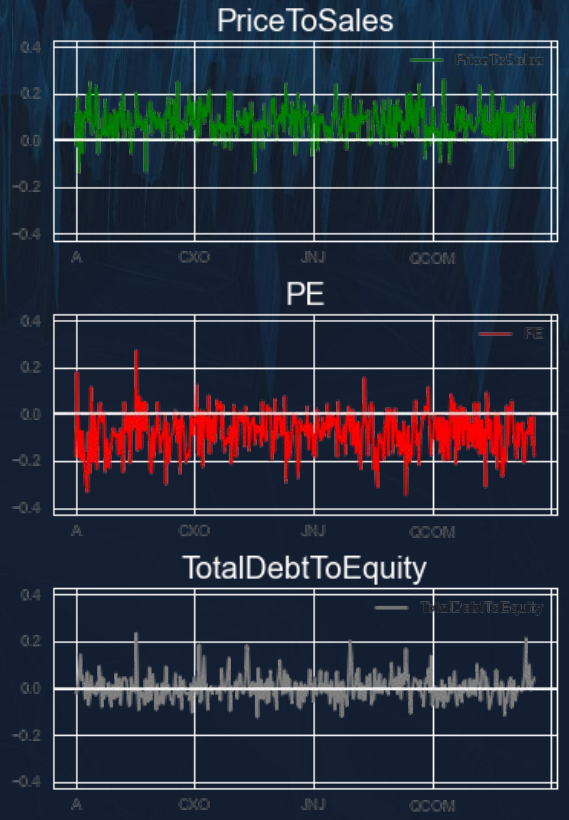
- Match all table's indexes
- Performance and Pricing data were filled by the moving average with a maximum of three consecutive null values.
- Removed roughly 5,000 symbols that still contained null values.
- Removed six features of performance that contained sparse data.
- Analyst data was numerically mapped. Ex:
  - Buy -> 5, Sell -> 1
- Null company data was filled with"unknown" or 0.

Correlation Search

We searched for which features had the greatest correlation to price tomorrow and the next day. By finding which features correlated we could remove 20 features that did not.

On the right you'll see three separate correlations indicating what price is most likely to do the next day:

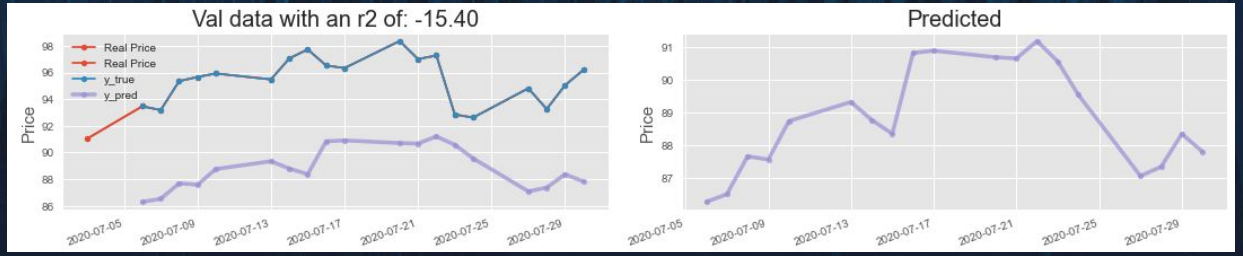
- In green PriceToSales has a positive correlation to price.
- In red PE has a negative correlation to price.
- In gray TotalDebtToEquity has no positive or negative correlation thus we would remove this feature



Base Network and Hand Tuning

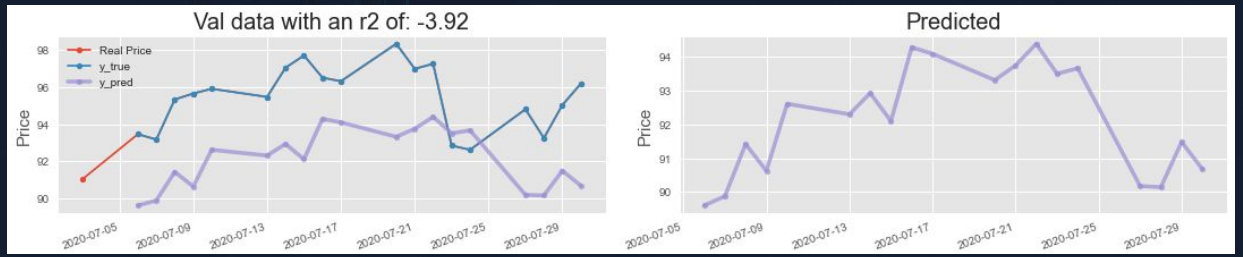
Our first base network was predicting tomorrow's price of Apple with all of the previous day's data of Apple.

Base Network



You can see the hand tuned network is much closer to the actual data with an R-squared of -3.92.

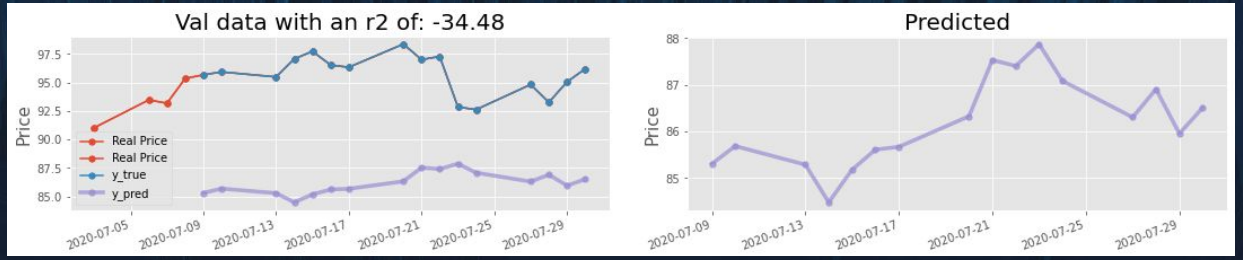
Hand Tuned Network



Base Network and Hand Tuning (cont…)

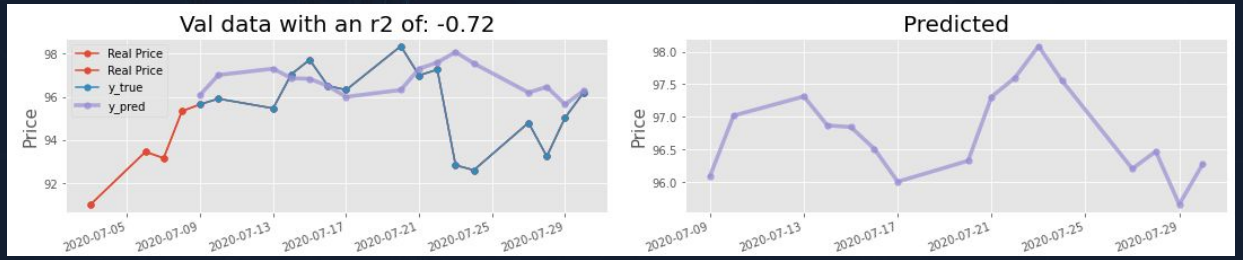
Our next base network was for predicting Apples price with four days of all other stocks' prices.

Base Network



The results of our hand tuned network greatly improved with an R-squared of -0.72

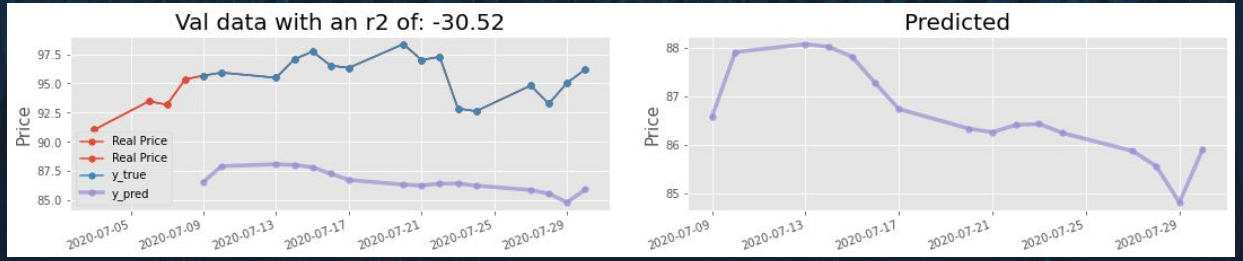
Hand Tuned Network



Base Network and Hand Tuning (cont…)

Our final base network was for predicting Apple's price with four days of data from Apple's industry

Base Network



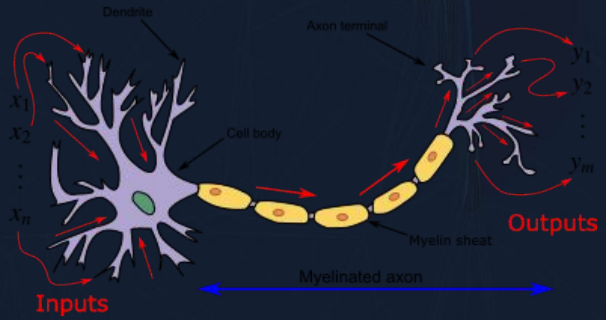
Hand Tuned Network

The results of our hand tuned network improved with an R-squared of -11.23

Final Network

We first attempted to build an array of networks to predict all of the data. After that didn't work due to processing limitations we built the following networks to predict two days of a single sector's prices:

- A network to predict tomorrow's prices
- A network to predict prices of the day after
- Combining these networks we can forecast the next two day's prices of a sector.



Hyper Parameter Tuning

We tuned our two networks by:

- Breaking down the creation of a network to variables such as:
  - How many hidden layers to use
  - How many previous days to use for each prediction
- Grid searched 1.2 million hyper-parameters combinations for each network taking roughly 5 hours each.
- Implemented k-fold cross-validation.

Cross-Validation

Final Network Scores

For predicting tomorrow's prices for "Technology Hardware" we achieved an R-squared of -2.63 vs -5.72 of our base network.

For the following day's prices we achieved an R-squared of -3.58 vs -9.43 of our base network.

Stock data is almost random, so we will never achieve a prediction accuracy anywhere close to a perfect R-squared of 1.0. We'll have to see how well it does forecasting for real results!

Conclusion

I. For cleaning the null data we can interpolate null values, and drop symbols and features that still contain nulls.

II. Changes in PE and PriceToSales have the greatest effect on tomorrow's price, although they are equations of price so changes in price could just be affecting price the next day.

III. Our number one limitation for predicting the data was GPU RAM. We ran into out of memory errors quickly if we tried predicting on all of the data.

IV. The best array of networks for predicting stock data we found was training a network for each of the x_days you want to predict into the future.

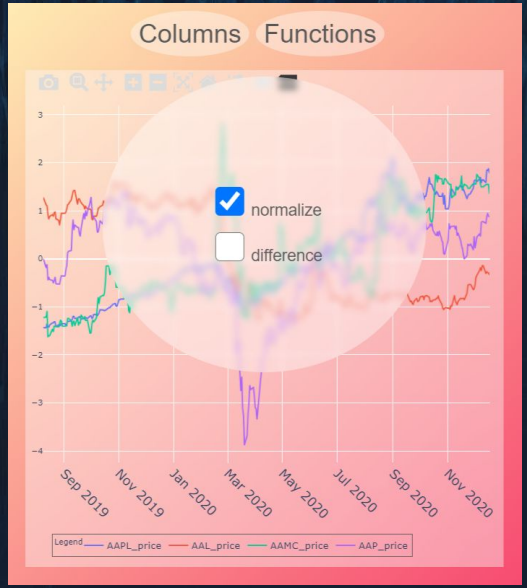
Next Steps

- Cluster data on absolute correlation
- iteratively add columns and test prediction quality
- Try with and without differencing


Thank you!

Appendix

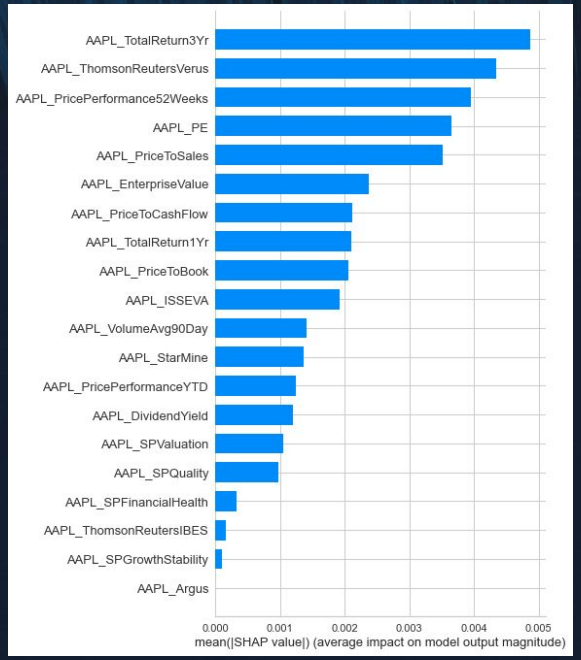
View our cleaned data [Here]



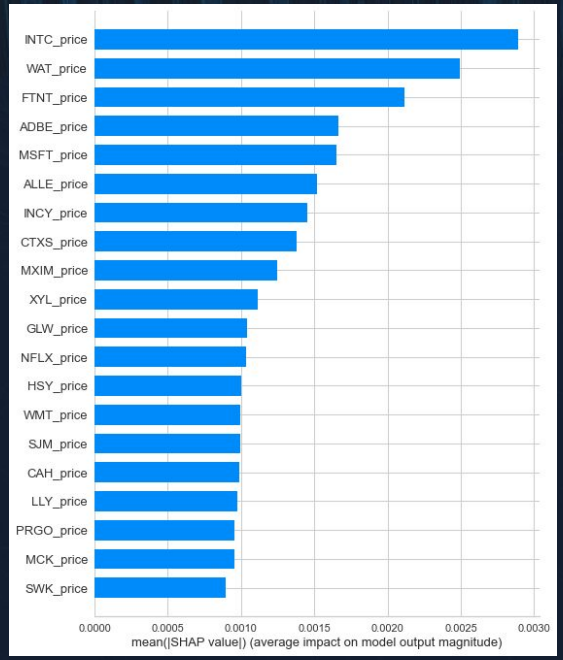
Appendix (cont…) Interpretations

Here we have model interpretations for three of our models showing how much each feature affected the prediction.

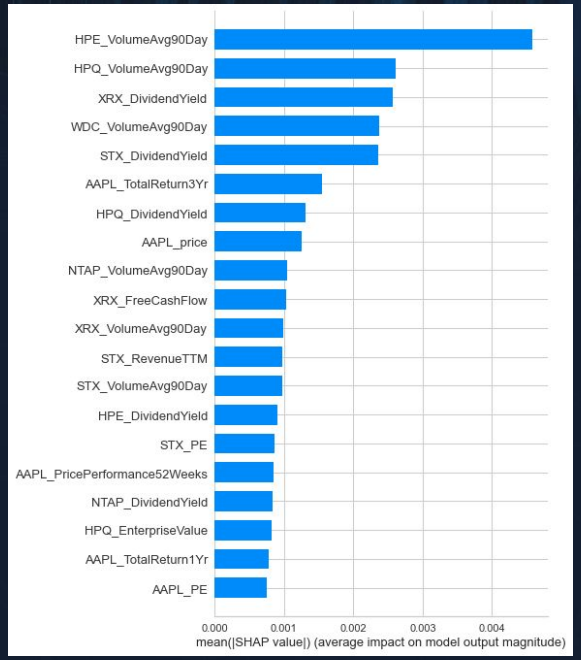
AAPL with AAPL data



AAPL with all prices



AAPL with sector data



Appendix (cont…) Hyper parameters

Here we show the best hyper parameters from our two auto-tuned networks.

Tomorrow

```
parameters = {
    'n_days': 3,
    'add_hidden_lstm': 1,
    'use_input_regularizer': 0,
    'input_dropout_rate': 0.1,
    'add_gaussian_noise': 0,
    'use_hidden_regularizer': 0,
    'hidden_dropout_rate': 0.0,
    'n_hidden_layers': 1,
    'hidden_neurons': 64,
    'optimizer': 'rmsprop',
    'patience': 0,
    'use_early_stopping': 0,
    'batch_size': 64,
    'gaussian_noise_quotient': 1.0,
    'input_regularizer_penalty': 0.01,
    'hidden_lstm_neurons': 64,
    'hidden_regularizer_penalty': 0.1}
```

Day After Tomorrow

```
parameters = {
    'n_days': 3,
    'add_hidden_lstm': 0,
    'use_input_regularizer': 0,
    'input_dropout_rate': 0.3,
    'add_gaussian_noise': 0,
    'use_hidden_regularizer': 0,
    'hidden_dropout_rate': 0.1,
    'n_hidden_layers': 1,
    'hidden_neurons': 32,
    'optimizer': 'adam',
    'patience': 0,
    'use_early_stopping': 0,
    'batch_size': 64,
    'gaussian_noise_quotient': 3.0,
    'input_regularizer_penalty': 0.01,
    'hidden_regularizer_penalty': 0.01,
    'hidden_lstm_neurons': 64}
```