# Equity Price Prediction with LSTM

Anthony Chi

**March 23rd, 2018**

**Project Proposal**

**Data Examination**

**Modeling and Optimization**

## I. Definition

### Project Overview

Equity price prediction has always been a tough task. Using traditional machine learning, ie. classification or regression, the results are often mediocre. Reasons could be: 1. unable to extract meaningful features that correlates with desired targets; 2. traditional machine learning algorithms do not incorporate all available information. The goal for this exercise is to explore what are the predictive power we can achieve with additional means. With added dimension (time), would we be able to achieve better predictive power.

### Problem Statement

To explore the exercise, here were the steps that I took:

* Explore data – there were a few data sources available for use. I needed to select a source.
* Clean data –
  + Outliers: It was important to investigate the outliers. If outliers were not due to mistakes, it were important to include outliers as the model needed to be able to fit and predict outliers and behaviors.
  + Missing data: Explored the reasons for missing data. Database would unlikely be leverage if data was missing and no reasonable explanation was uncovered.
* Feature treatments – by looking at the features’ distribution as well as their correlation with the targets, did we need to implement special treatments? Here were the questions we needed to answer:
  + Was PCA necessary, for dimensionality reduction and improved model performance?
  + Did we need transformations to make the distribution more normal, and thereby improving model performance?
  + Were there correlation between features and it was difficult to select features to use in modeling algorithms?
* Prepare data – How should the data structure be treated, as they were fed into modeling algorithms? LSTM might need some special attention.
* Model selection – there were two benchmark models: Lasso Regression, and XGBoost Classification, as they were historically higher performance modeling algorithms.
* My goal was to make find optimized LSTMs that would beat the benchmarks, in both classification and regression.

### Metrics

For classification exercise, I used accuracy as my metric. For the regression exercise, negative root mean square error was used. NRMSE was compared to the target mean (as a percentage), to get an idea of the magnitude of the error.

## II. Analysis

### Data Exploration

The goal here was to examine the available data, and achieve the following objectives:

Compare Data Sources: many data sources are becoming unavailable, turned into paid services, or only available on proprietary platforms. Many APIs were also inactivated, and webpages were inactivated or unable to be scrapped. I was only able to access three data sources: Quandl API, NASDAQ, and finance.yahoo.com.

After I compared the three data sources, there were visible differences in Adjusted Closing and Adjusted Volume. There were also occasional missing data from Quandl API; the NASDAQ maximum historical data only spanned 10 years. Therefore, I decided to rely soley on finance.yahoo.com’s data.

Using the available features, including open, close, high, low, volume and adj. close, I generated additional features:

* High-Low Range, as a percentage of previous day’s close: this described how volatile the day’s movements were
* Open-Close Range, as percentage of same day’s open: this described the general movement of the day’s activities
* Moving average (adjusted close & volume) for a specific time period.

The features were generated to answer the following questions or explore ideas:

* What am I trying to predict? Specific price, change in price, simple direction, or general trend/direction? This will determine if this is a classification (bi or multi) or a regression problem.
* Are there any relationship between the independent and dependent features?
* What do the feature distributions look like? Do I need to make any transformation?
* Are there any missing or outliers that I need to take care of?
* Do I need to use scalers, or other feature engineering techniques such as PCA?
* Is there a model that can be applied to all equities? Or do I need to train for each equity?

### Exploratory Visualization

**Target**

**Outliers**

**Missing Data**

**Delete Feature**

**Feature Types Transformed**

**One-hot Features**

**New Features created**

**Features Transformed**

**PCA**

### Algorithms and Techniques

### Benchmark

1. I used two benchmark models, namely XGBoost Classifier and Lasso Regression. The ultimate goal was to use LSTM to beat these benchmark models. Supposedly, LSTM should be able to incorporate long and short term memories to generate additional insights, and further improve the predictive power of resulting models.
2. During the LSTM optimization exercise, I attempted to explore how the hyperparameters such as window, epoch, batch size, and LSTM constructs affected the model performances. This surely helped me tune the model for better performances.
3. Also note this was a regression, as well as, a classification exercise.
   * 1. As the exercise became computationally expensive, I moved computational and storage needs onto AWS.

**III. Methodology**

**Data Preprocessing**

**Target**

**Outliers**

**Missing Data**

**Delete Feature**

**Feature Types Transformed**

**One-hot Features**

**New Features created**

**Features Transformed**

**PCA**

### Implementation

### Refinement

### **Lasso Regression (benchmark)**

### **XGBoost**

### **LSTM**

### **KNeighborsRegressor:**

## IV. Results

### Model Evaluation and Validation

### Justification

## V. Conclusion

### Free-Form Visualization

### Reflection

# Findings and questions

# Questions:

# Predicting next day price may not be the best thing to do. We may want to predict price in 5 days, or even much more ahead? What’s a reasonable window to predict price before randomness takes over?

# Should I predict volume?

# Should I include other features to predict price?

# Does dropout improve performance?

# Does multiple layers improve performance?

# Should we actually try to predict price? Or should we predict a range? For example, -1~1% is neutral, 1~5% is good, 5%+ is excellent, etc.

# Findings:

# Best results is with window 15 (about 3 weeks), epoch 2000, and batch-size 50-250.

# There may be upward bias, meaning all that we have seen is upward trend (most of the time).

# When selecting hyperparameters, training predict time, testing predict time, training error eval time, and testing error eval time are negligible.

# The larger the window, the longer the training time. The higher the epoch, and the smaller the batch size, the longer the training time.

# Best training error achieved is with window 15, batch size 10, epoch 500: 2.5616e-06

# Best testing error achieved is with window 10, batch size 500, epoch 2000: 7.58083e-05

### Citation and Sources

**Relevant Files and Folders**