Time Series Analysis: **Predicting Stock Returns** with Recurrent Neural Networks (RNN)

Adrian Halarewicz
Springboard Capstone Project 2

Problem Statement:

Investing in the stock market would be easy if an investor could predict changes in stock price.

A stock's price can be influenced by many factors:

- economic trends
- related or unrelated companies
- expected returns

Is it possible to predict future changes in a stock's price exclusively from trends in a stock's own historical data?

The Client & Possible Implementations

A single investor with a particular interest in a single stock

The model will decide whether to buy, hold, or sell a particular stock.

The model can serve as the foundation of robotic trading or financial advising software and remove the need to manually manage a stock portfolio.

The Data Set

Historical stock data can be easily collected with the **yfinance** library.

The Recurrent Neural Network will utilize Long Short Term Memory (LSTM) to look for trends in a stock's **Adjusted Closing** price.

Exponential behaviour needs to be reduced to normalize changed in closing price.

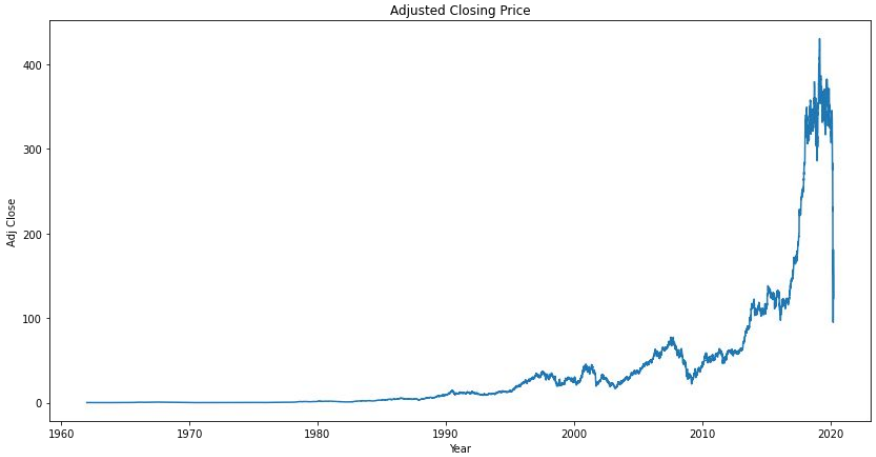
The change (delta) between two consecutive days need to be calculated.

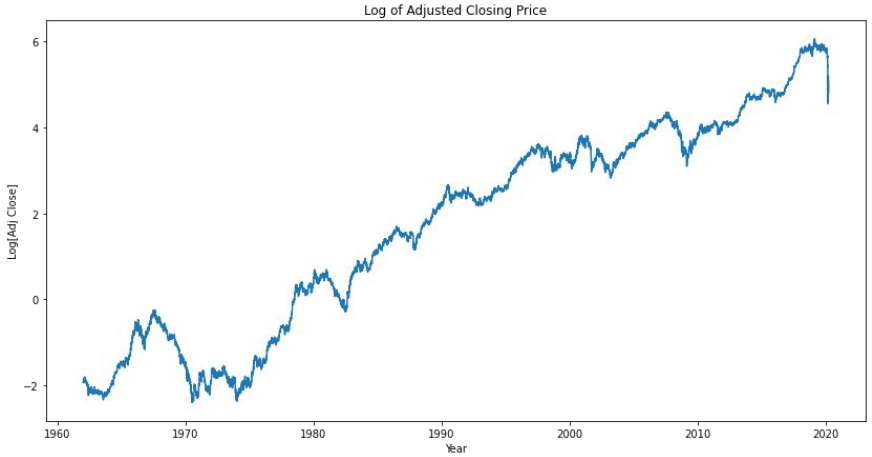
The resulting values need to be formatted for Time Series Analysis.

Downloading data with **yfinance**

```
import yfinance as yf

import pandas as pd

from datetime import datetime

today = datetime.date(datetime.now())

df = yf.download("GOOG",  start="2004-08-19", end= today)

type(df) → pandas.DataFrame
```

Eliminating Exponential Behaviour






```
log_scaled_df
```

```
log_scaled_df =
np.log(adj_close_df)
```

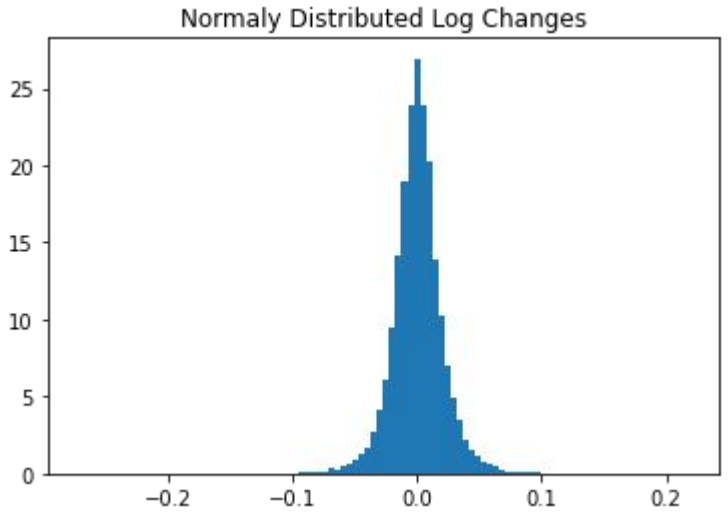
Changes in recent years will dominate changes when price was lower.

Obtaining a Normal Distribution

The model will predict the log of expected returns

$$delta_i = Log_e\left(\frac{price_{i+1}}{price_i}\right)$$

This results in Normally Distributed Data



Formatting for Time Series Analysis

DateTime Index	back_5 (predictor)	back_4 (predictor)	back_3 (predictor)	back_2 (predictor)	back_1 (predictor)	Log[delta(Adj Close)] **(target)**
date	`delta_i-5`	`delta_i-4`	`delta_i-3`	`delta_i-2`	`delta_i-1`	`delta_i`

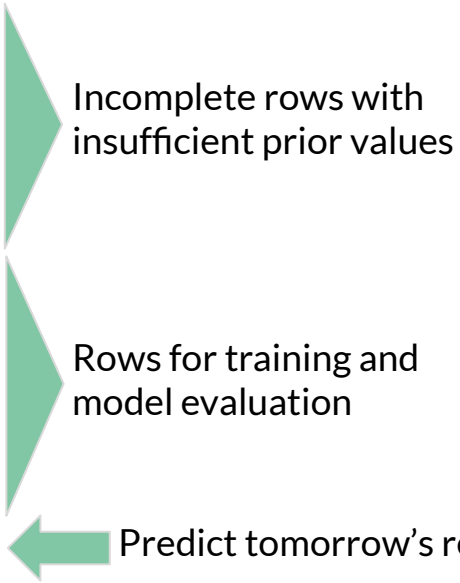
Columns are created from previous target values.

LSTM remembers the returns from the 5 previous trading days

The RNN layers identify internal fluctuation trends.

The complete data set

	back_5	back_4	back_3	back_2	back_1	Adj Close	
Date							
1966-07-05	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	0.052056	**drop row**
1966-07-06	**NaN**	**NaN**	**NaN**	**NaN**	0.052056	-0.036905	**drop row**
1966-07-07	**NaN**	**NaN**	**NaN**	0.052056	-0.036905	0.011215	**drop row**
1966-07-08	**NaN**	**NaN**	0.052056	-0.036905	0.011215	-0.003724	**drop row**
1966-07-11	**NaN**	0.052056	-0.036905	0.011215	-0.003724	-0.015037	**drop row**
...	**...**	...	...	...	...	...	**train/test**
2020-03-27	-0.079807	0.166577	0.006340	0.026460	-0.020160	0.024810	**train/test**
2020-03-30	0.166577	0.006340	0.026460	-0.020160	0.024810	-0.016673	**train/test**
2020-03-31	0.006340	0.026460	-0.020160	0.024810	-0.016673	-0.044394	**train/test**
2020-04-01	0.026460	-0.020160	0.024810	-0.016673	-0.044394	0.020835	**train/test**
2020-04-02	-0.020160	0.024810	-0.016673	-0.044394	0.020835	**NaN**	**prediction**



Interpreting Predictions

Predicted Return

$$e^{prediction} = \frac{Price_{today}}{Price_{tomorrow}}$$

Predicted Closing Price

$$Price_{tomorrow} = Price_{today} * e^{prediction}$$

$$e^{prediction} > 1.0 \Rightarrow +\ positive\ returns\ expected$$

$$e^{prediction} = 1.0 \Rightarrow Price_{tomorrow} = Price_{today}$$

$$e^{prediction} < 1.0 \Rightarrow -\ negative\ returns\ expected$$

Model Architecture

Optimized with Cross-Validation

Layer Type:	Dense
Layers:	10
Nodes in each Layer:	25
Epochs:	2
Optimizer:	“adam”
Loss function:	“mean_squared_error”
Mean accuracy of 10 samples:	53.23%

Significance of Accuracy

Results greater the 50.0% are acceptable.

It is extremely rare for a stock predictor to exceed 60%

Any slight advantage can be scaled to make tremendous profit.

Consider the wealth Las Vegas casinos generate from games where the house win probability is only ~52%.

Thank you

https://github.com/AHalarewicz/TimeSeriesAnalysis