My background is in software and financial engineer and for my capstone project have chosen

“Predicting the Stock Trend with News Sentiment Analysis and Technical Indicators”. Those are summary and trends which I currently see.

Machine learning can be applied in several different cases for this one scenario.

* Pattern recognition from candle data to identify levels of significance
* Creating specialised indicators to add to a simple rule based strategy
* A final processing and aggregation layer to make a prediction from your set of indicators.

Machine learning can also be applied in slightly more exotic ways to help refine further information:

* Denoising and auto-encoding — used to remove some of the random noise of a price feed to help distill the underlying trend or specifics of the market sentiment.
* Clustering — group together different equities and financial instruments to streamline the value of a portfolio. Or it could be used to evaluate and reduce the risk of a portfolio.
* Regression — often used to try to predict the price at the next time step, however it can be applied to a range of abstracted indicators to help predict trading signals earlier.

Hey Arron,

That is a big challenge, using the regression models to predict stock price and trends from OHLC data, the problem here is the correlation, usually you try to predict the close price from the historical data (open, hight, low) have correlation 1 to close price, if you drop those high correlation features you loose the input historical data, what are the other option using averages (H+L+C), or percent changes.

I used VWAP, to keep the historical data, which also has correlation 1 to close price

Traders use the VWAP to combine price and trading volume and make important decisions about whether to make an entry or exit point in a specific security.

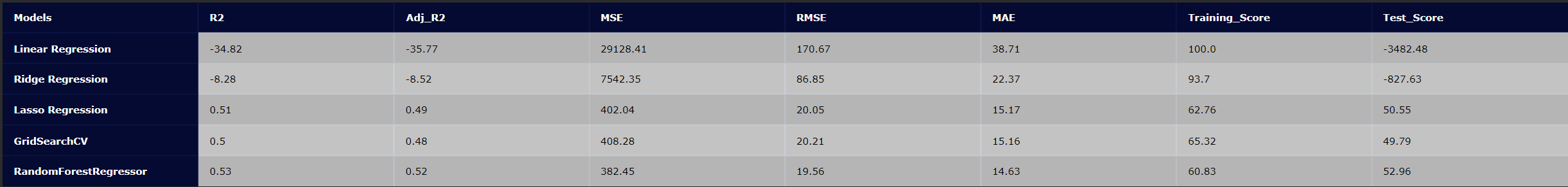
Another challenge is sentiment of the news, finding the right pre-train model, for positive and negative, or positive neutral and negative

Those are some early findings, from my capstone, for time being, correlation has been ignored

**Findings:** The best model for predicting the stock trend RandomForestRegressor model, which

has the lowest Root Mean Square Error (RMSE) 19.56, It shows how far predictions fall from measured true values using Euclidean distance.

A R-squared between 0.50 to 0.99 is acceptable in social science research especially when most of the explanatory variables are statistically significant.



**Results and conclusion**: At the moment of developing the RandomForestRegressor model is the best performer, but next development is the Long Short-Term Memory Networks model.

The Long Short-Term Memory Networks is a deep learning, sequential neural network that allows information to persist. It is a special type of Recurrent Neural Network which is capable of handling the vanishing gradient problem faced by RNN. LSTM was designed by Hochreiter and Schmidhuber that resolves the problem caused by traditional RNNs and machine learning algorithms. LSTM Model can be implemented in Python using the Keras library.

The LSTM model is then trained to predict the target variable (e.g., the future stock price) based on the historical information.

Hey Vincent,

You made good points, what I see with OHLC historical data is a big challenge, not the models but the data in its own.

To expand those predictive features, you might do.

1. News sentiment, hope you got the right pre-train model (scores and classification of it, as positive and negative, or with neutral
2. Add Technical Indicators as RSI, MCAD, and etc

But then, if the historical data [vwap, open or avg(C+H+L)] has corelation 1 to close price the target, those news and technical indicator have 0.20 or 0.40, are almost ignored.

So the regression is not a go, but forecasting of time-series might LSTM or ARIMA might work.

Hey Vincent,

Good work!

The ration epochs to loss is really good.

Just curious what input futures have you used of OHLC historical data.

model.compile("adam", "binary\_crossentropy", metrics=["accuracy"])

The passing parameters of the compile method need to be reordered to be explicitely specified, and the nan will disappear.

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=["accuracy"])