Machine Learning Project

Ashutosh
Harshit Goyal
Vishesh Bhati

8th December 2023

1 Introduction

Forecasting stock market movements is difficult due to the numerous factors that can influence a stock's price. While the markets are complex, they are not completely random. As history tends to repeat itself, there are some predictable patterns. Markets, in general, mirror rational human behaviors and emotions. Furthermore, financial theory assumes that capital markets are efficient and incorporate all available information. While perfect predictions are impossible, we can create useful models by taking into account as many relevant variables and human behaviors as possible. We will attempt to forecast returns for each each 500 stocks which make up the SP500 index and provide a trading signal based on our prediction.

2 Data Preparation

The dataset spans a daily frequency, covering the extensive timeframe from year 1992 to 2022. With a vast size of 112 X 2.5 million records. The inclusion of multiple data types ensures a holistic representation, capturing not only market trends but also the financial health of the underlying stock.

2.1 Parameters

We have used the following features as predictive parameters :

1. Prediction Variable: Stock Returns
2. Multi Index: 2 indexes (Date, Permno)
3. Price Action: We harnessed Panel Data, specifically Price Action, sourced from WRDS (Wharton Research Data Services). This rich dataset encompasses six key parameters: Open, High, Low, Close (OHLC), Shares Outstanding and trading volume. For instance, OHLC data provides insights into daily price movements, while volume reflects market activity
4. Financial Statement: Incorporating financial statement data from CapitalIQ, our dataset encapsulates a robust 68 parameters for each Permno (Permanent Number). Parameters include financial ratios and components from income statements like EBIT, Net Income. Examples of financial ratios involve Return on Equity (ROE), Debt-to-Equity ratio, and Earnings per Share (EPS).
5. Technical Indicators: A set of 36 individual technical indicators, meticulously calculated, add another layer of sophistication to our dataset. These indicators, computed from historical price and volume data, include Relative Strength Index (RSI), Moving Averages, and Bollinger Bands. RSI gauges the momentum, Moving Averages provide trend insights, and Bollinger Bands capture volatility. These technical indicators, derived from years of market data, offer a comprehensive view of market trends and patterns. technical indicators.

(Total = 110 predictive parameters)

2.2 Normalization

To make sure that the input features are on a consistent scale, normalization is necessary.

The original dataset is transformed into a new one (data mapping) where the values are ranked between -1 and 1, with the 'TICKER' column kept in its original position through:

1. Data ranking:

$$Rank_{\text{norm}} = \frac{\text{Rank} - 1}{\text{Total Number of Values} - 1} \tag{1}$$

2. Normalization:

$$X_{\text{norm}} = \frac{X_{\max} - X_{\min}}{X - X_{\min}} \tag{2}$$

3 Feature Extraction

3.1 Regression

Regularization techniques used in linear regression models to address multicollinearity and prevent overfitting include Ridge regression, Lasso regression, and Elastic Net regression. To control the magnitude of the coefficients, they add penalty terms to the traditional linear regression cost function.

3.1.1 Ridge Regression

Ridge regression includes a regularization term that is proportional to the square of the coefficient magnitude θ_i. The regularization strength is controlled by the hyperparameter.

3.1.2 Lasso Regression

The regularization term in Lasso regression is proportional to the absolute value of the coefficients. Ridge, like, controls the regularization's strength.

3.1.3 Elastic Net Regression

By introducing a mixing parameter (rho), Elastic Net combines both Ridge and Lasso penalties. When =0, it corresponds to Ridge regression; when =1, it corresponds to Lasso regression.

3.2 Checking for multicollinearity and heteroskedasticity using breusch pagan test

The Breusch-Pagan test is commonly used in regression models to check for heteroskedasticity. It determines whether the error variance is constant across observations. It is not commonly used to test for multicollinearity, which is a different issue related to predictor variable correlation.

3.2.1 Breusch-Pagan Test

Breusch-Pagan Heteroskedasticity Test:Execute a Regression Model: Fit your regression model and calculate the residuals (the differences in observed and predicted values).Residuals Squared: For each observation, compute the squared residuals. Auxiliary Regression: Create a new regression using the squared residuals as the dependent variable and the original predictors as independent variables.

Interpret the Results: If the p-value is less than a predetermined level of significance (e.g., 0.05), you can reject the null hypothesis and conclude that there is evidence of heteroskedasticity.

3.2.2 Huber-White Test

To remove heteroskedasticity, use the Huber white process. The Huber-White sandwich estimator, also known as heteroskedasticity-robust standard errors or White's standard errors, is a method for dealing with heteroskedasticity in a regression model's residuals. When the assumption of homoskedasticity is violated, this method adjusts the standard errors of the estimated coefficients to provide more reliable inference.

4 Light GBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. After runnng the Light GBM we shortlisted 35 indicators that had greatest ranks for training our transformer model.

#	Column	Non-Null Count	Dtype
0	stochrsi_k	2492541 non-null	float64
1	stochrsi	2492541 non-null	float64
2	stoch_signal	2492541 non-null	float64
3	stoch	2492541 non-null	float64
4	stochrsi_d	2492541 non-null	float64
5	rsi_14	2492541 non-null	float64
6	eom	2492541 non-null	float64
7	adx_neg	2492541 non-null	float64
8	adx_pos	2492541 non-null	float64
9	bollinger_pband	2492541 non-null	float64
10	atr	2492541 non-null	float64
11	tsi	2492541 non-null	float64
12	OPENPRC	2492541 non-null	float64
13	bollinger_lband_indicator	2492541 non-null	float64
14	PRC	2492541 non-null	float64
15	bollinger_hband_indicator	2492541 non-null	float64
16	macd_diff	2492541 non-null	float64
17	mfi	2492541 non-null	float64
18	bollinger_wband	2492541 non-null	float64
19	bollinger_lband	2492541 non-null	float64
20	VOL	2492541 non-null	float64
21	roc	2492541 non-null	float64
22	macd	2492541 non-null	float64
23	fi	2492541 non-null	float64
24	ASKHI	2492541 non-null	float64
25	BIDLO	2492541 non-null	float64
26	cmf	2492541 non-null	float64
27	sma_50	2492541 non-null	float64
28	bollinger_hband	2492541 non-null	float64
29	ema_20	2492541 non-null	float64
30	eom_sma	2492541 non-null	float64
31	macd_signal	2492541 non-null	float64
32	adi	2492541 non-null	float64
33	obv	2492541 non-null	float64
34	bollinger_mavg	2492541 non-null	float64
35	RETX	2492541 non-null	float64

Figure 1: 35 extracted features

5 Transformer Model

A Transformer is a type of neural network architecture first described in the paper "Attention is All You Need" by Vaswani et al. It employs a mechanism known as self-attention to efficiently handle sequential data. Transformers are commonly used in natural language processing tasks such as machine translation and text generation, but they can also be used in other domains.

The Encoder and the Decoder are the two primary parts of the Transformer model.

Encoder: A stack of dense (fully connected) layers, each with a ReLU activation function, make up the Encoder class. The num layers parameter determines how many layers the encoder has. These layers receive the input from the encoder in a sequential order.

Decoder: While the dense layers in the Decoder class have a linear activation function, they are otherwise similar to the encoder class. The decoder's output is transformed into a one-dimensional tensor.

Model Collection: The Adam optimizer and the mean squared error loss function are used to compile the model.

Model Construction: In both the encoder and decoder, an instance of the Transformer model with 6 layers is created. We have seen in multiple research papers that shallow layers for financial data is the best for prediction and fitting.

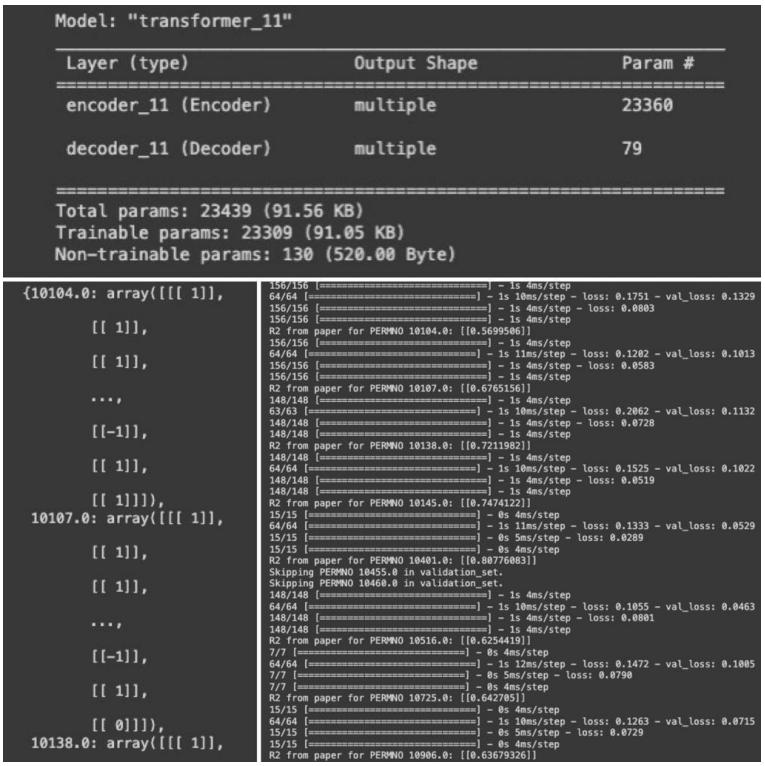

Figure 2: Model Architecture Output