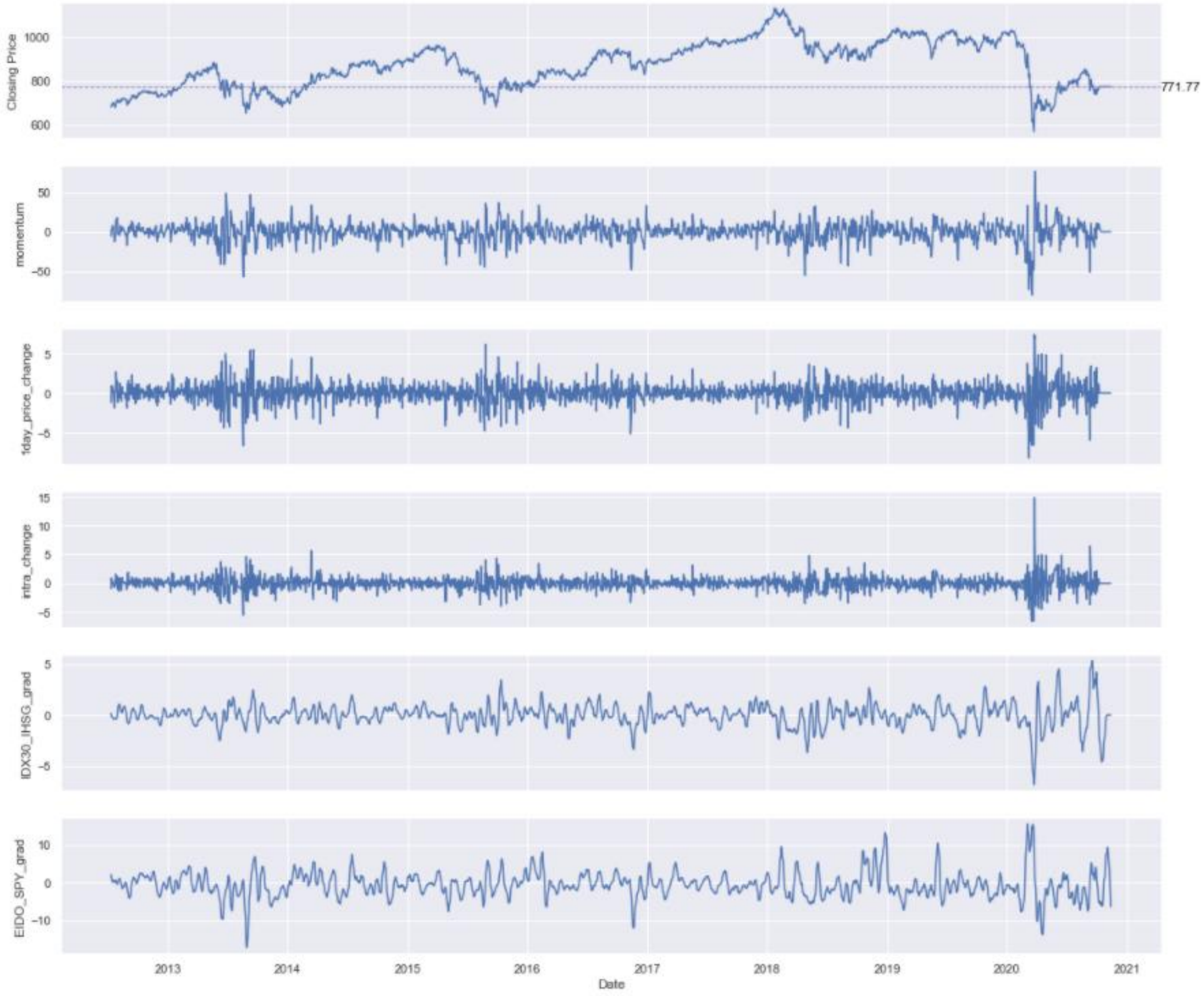
LSTM for IDX30 Momentum Trading

Hinge Loss function, rescaling, IDX30-IHSG gradient

Panin Asset Management
Values Come First

Input Data

Input variables for LSTM model



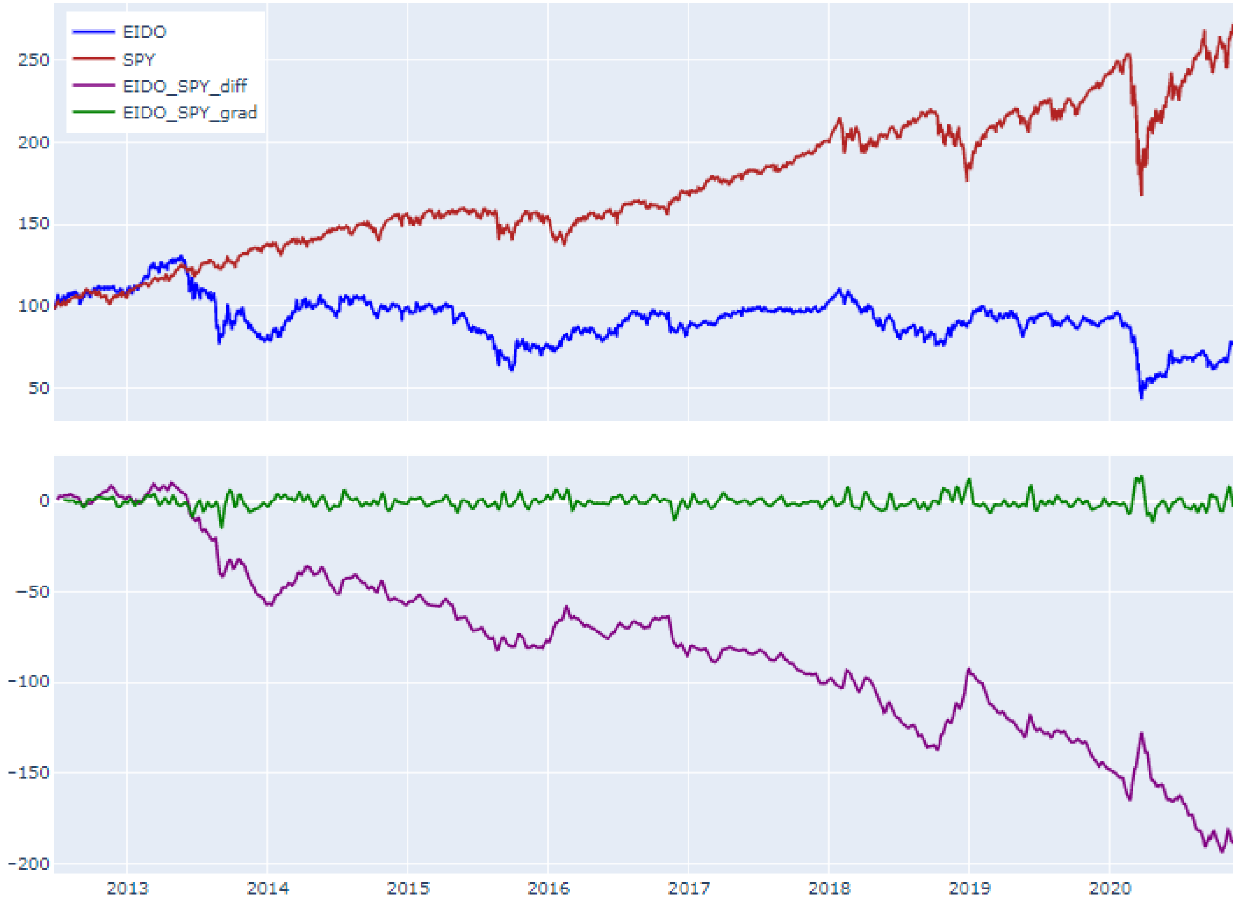
	Parameters	Training Data	Testing Data
0	Start Date	2012-07-04 00:00:00	2019-01-01 00:00:00
1	End Date	2018-12-31 00:00:00	2019-12-31 00:00:00
2	Data Points	1694 Days	261 Days

- For training data we will use LQ45 ticker from 04-07-2012 to 2018-12-31 (1694 Days) and validation dataset will also come from this sample at 10% of total dataset (170 Days)
- For holdout test data we will use LQ45 ticker from 01-01-2019 to 31-12-2019 (261 Days), this data will not be used in training and acts as a benchmark from previously unseen data
- There are 5 input variable in this model: momentum, 1 Day Price Change, Intraday Change and grad
- Momentum is the difference between current price and the past 7-Day Moving Average
- Intraday Change is the percentage difference between Closing Price and Opening Price

Difference of Difference – 7-Day Difference of 2 Moving Average

IHSG vs IDX30 and EIDO vs SPY

EIDO vs SPY

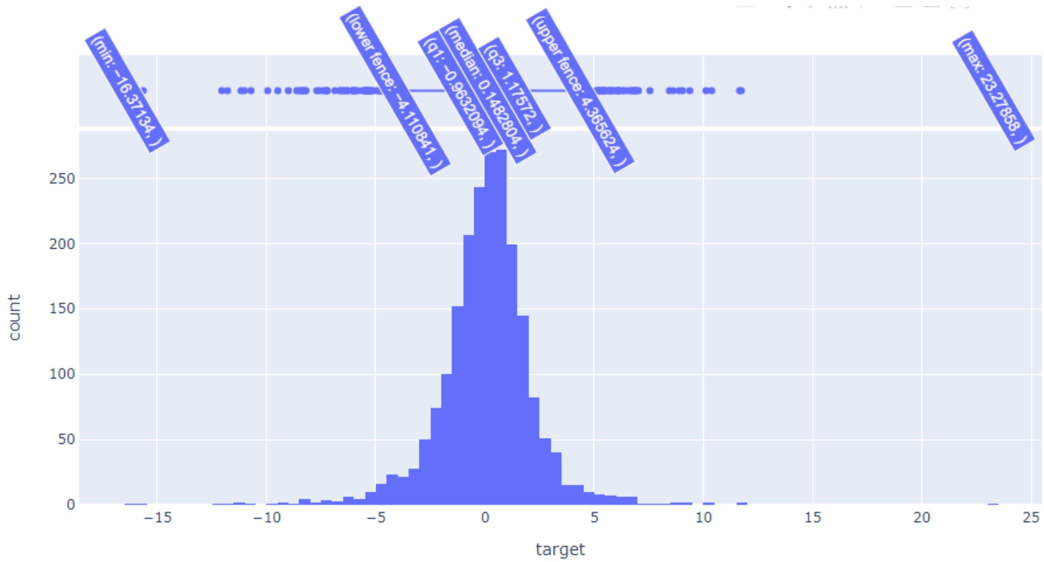


IDX30 vs IHSG

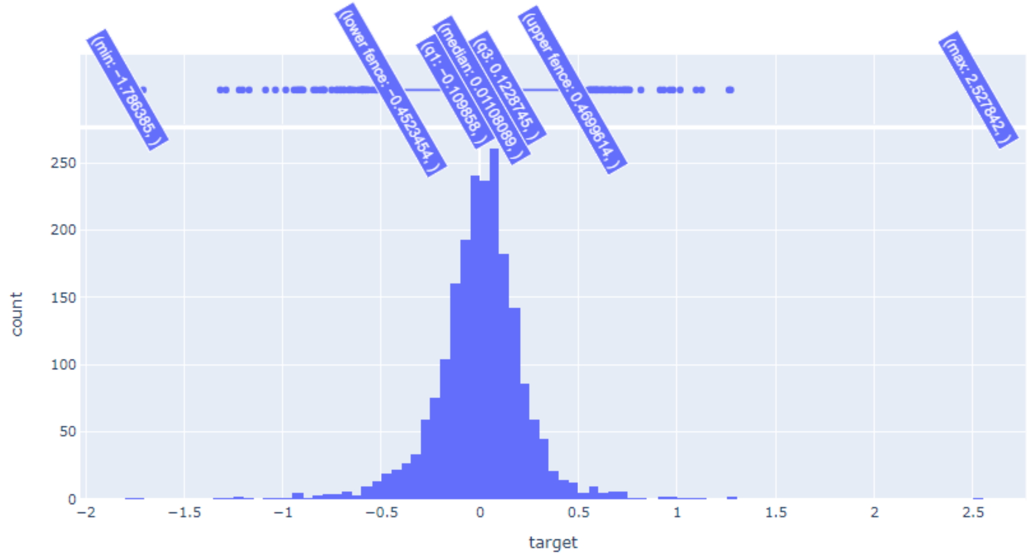
Standardizing Target Variables

Rescales Target variables (3-Day price change) using standardization

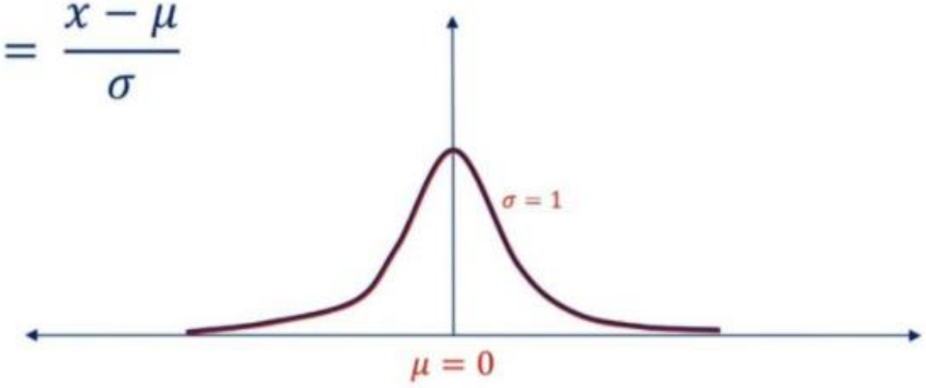
Original Target



Standardized Target



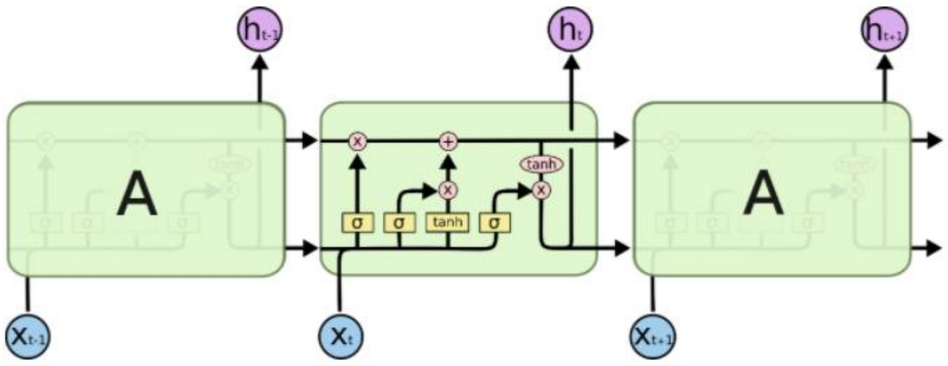
$$z = \frac{x - \mu}{\sigma}$$



- To make it easier for the model, we will rescale the target variables (3-Day price change) using Standard Scaling method.
- Standardizing is done by subtracting the mean and then dividing all the values by the standard deviation. Standardization results in a distribution with a standard deviation equal to 1 and variance equal to 1
- The output layer of the model will also have a tanh layer which outputs values ranging from -1 to 1

The Model - Overview

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)



The repeating module in an LSTM contains four interacting layers.

LSTM Unit

$$\tilde{c}_t = \tanh(W_c[a_{t-1}, x_t] + b_c)$$

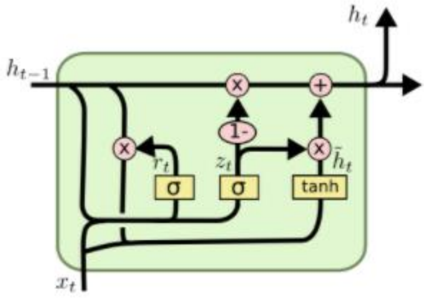
$$G_u = \sigma(W_u[a_{t-1}, x_t] + b_u)$$

$$G_f = \sigma(W_f[a_{t-1}, x_t] + b_f)$$

$$G_o = \sigma(W_o[a_{t-1}, x_t] + b_o)$$

$$c_t = G_u * \tilde{c}_t + G_f * c_{t-1}$$

$$a_t = G_o * tanh(c_t)$$



FULL GRU Unit

$$\tilde{c}_t = \tanh(W_c[G_r * c_{t-1}, x_t] + b_c)$$

$$G_u = \sigma(W_u[c_{t-1}, x_t] + b_u)$$

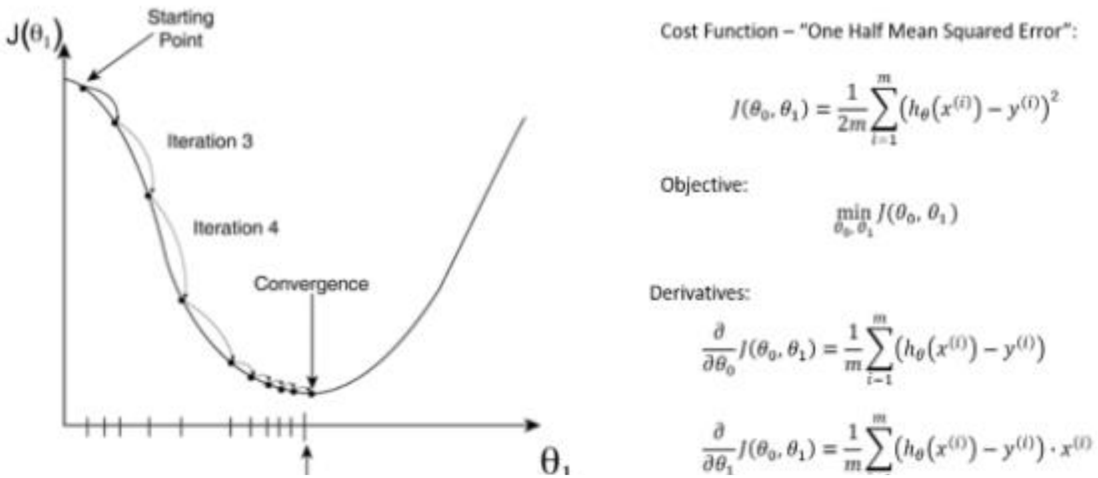
$$G_r = \sigma(W_r[c_{t-1}, x_t] + b_r)$$

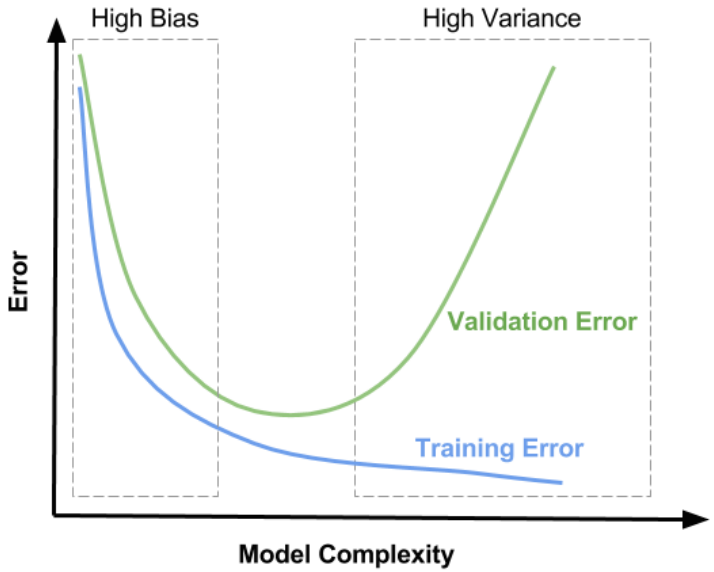
$$c_t = G_u * \tilde{c}_t + (1 - G_u) * c_{t-1}$$

$$a_t = c_t$$

- Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.
- Consider a simple RNN processes more steps, it has troubles retaining information from previous steps. layers that get a small gradient update stops learning. Those are usually the earlier layers. So because these layers don’t learn, RNN’s can forget what it has seen in longer sequences, thus having a short-term memory.
- Unlike LSTM, Gated Recurrent Unit (GRU) combines the forget and input gates into a single “update gate.” It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.
- We will train ML model using LSTM and GRU to predict the price change of LQ45. Given the previous data at yesterday (t-1) to 20 days back (t-20) as input, we want to predict the price change of LQ45 from today to three days forward, (t+3) – (t).

Loss Function and Gradient Descent





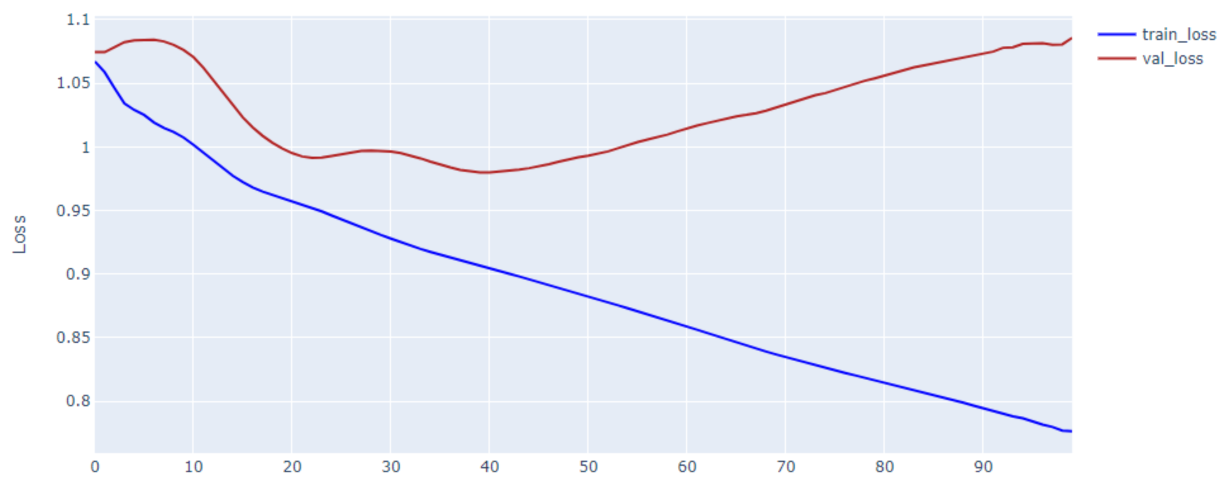
- The goal any machine learning model is to move the total loss of the prediction to the lowest possible value. In order to get the lowest error value, we need to adjust the *weights* ***W*** and biases ***b*** to reach the smallest possible loss value (error) by each iteration.
- Gradient descent is an iterative optimization algorithm used in machine learning to minimize a loss function. The loss function describes how well the model will perform given the current set of parameters (weights and biases), and gradient descent is used to find the best set of parameters.
- The primary set-up for learning neural networks is to define a loss function that measures how well the network predicts outputs on the test set. One common function that is often used is the mean squared error, which measures the difference between the actual value of *y* and the estimated value of *y* (the prediction).
- In this model we will use hinge function as a loss function since the output of the model is a value ranging from 1 to -1 and we want to penalize wrong sign (+/-) more severely in our prediction.

$$\ell(y) = \max(0, 1 - t \cdot y)$$

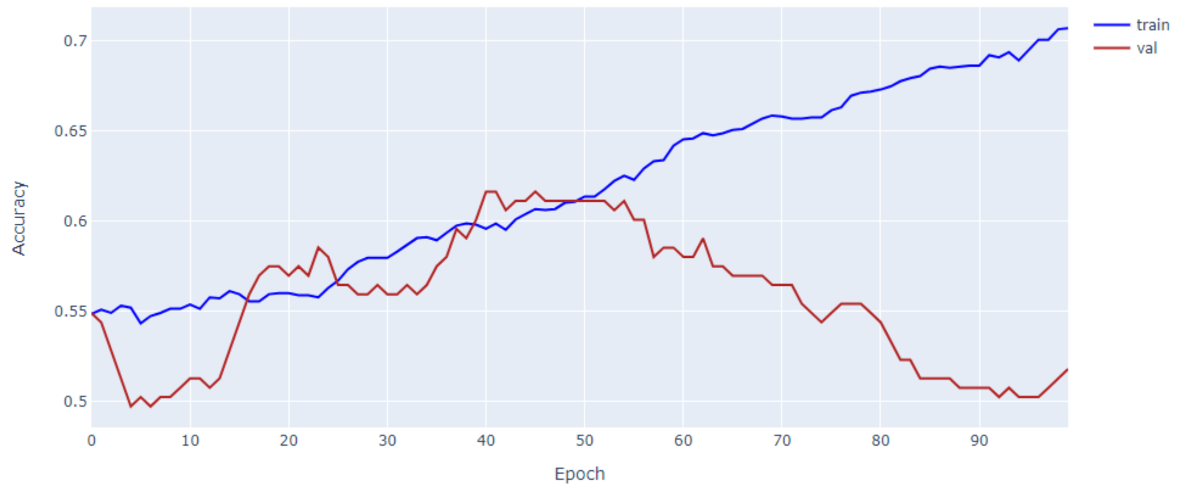
Model Training

Loss and accuracy improvement by epochs

Train and Validation Loss per epoch



Train and Validation Accuracy per epoch

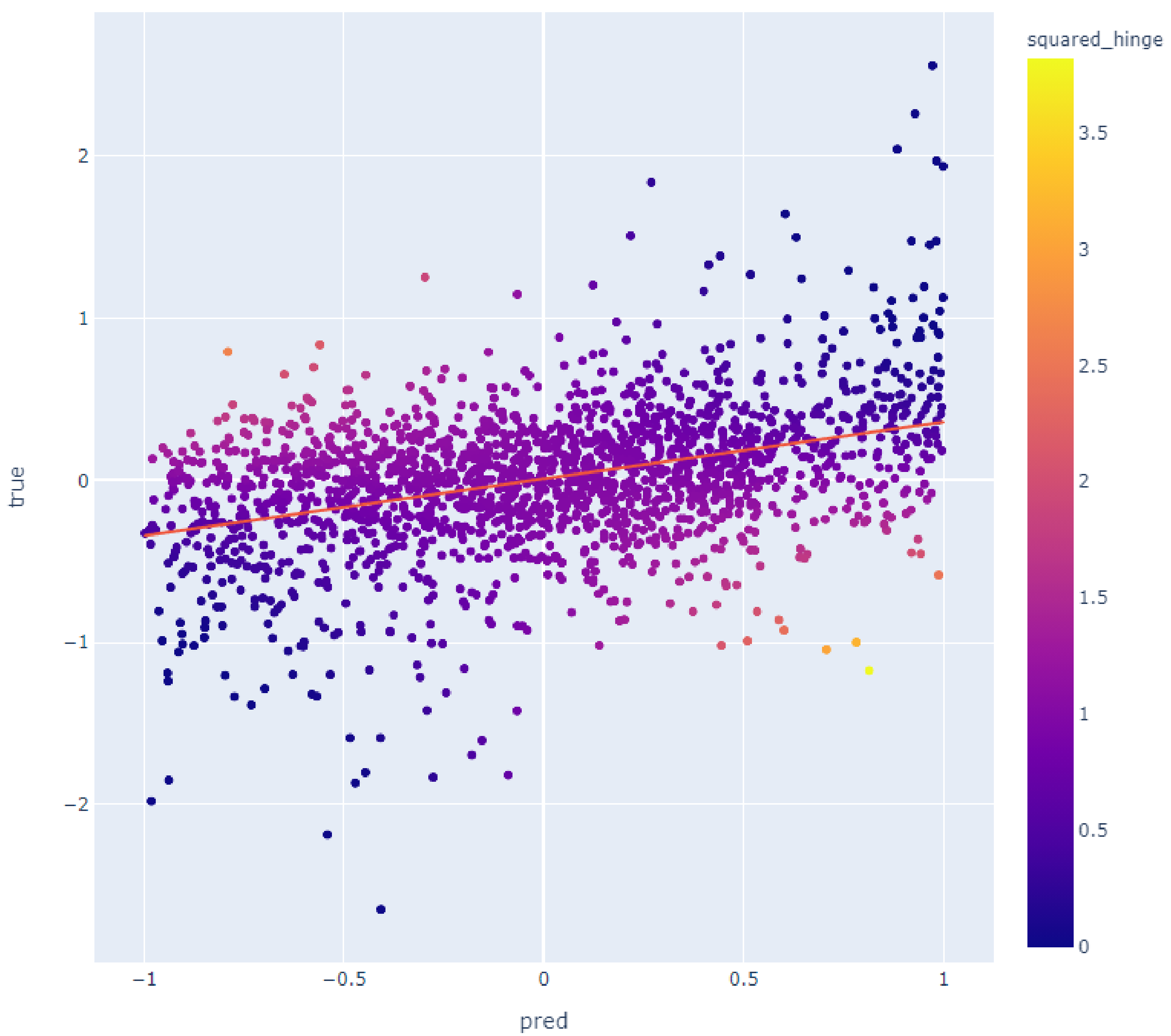


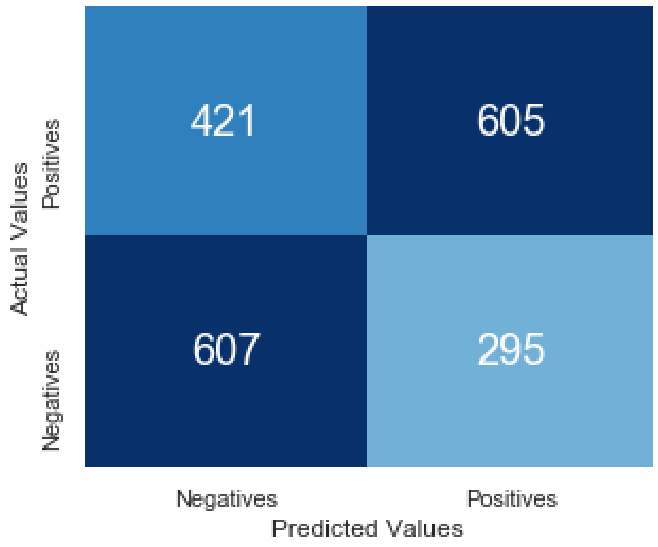
- The top left plot is the gradient descent of the training algorithm, which is the loss function (error function) plotted over training iteration.
- The bottom left plot is the accuracy of the model on each training iteration. We expect the accuracy to simultaneously increases as the loss function decreases.
- As we can see from the learning rate chart on the left, the loss function decreases over time and the minimum is on the 38th epoch, while the accuracy maximum is on the 42nd epoch.
- This means that the model is learning something from data as it gets better results with each training iterations (epochs).
- The validation data (10% of training data) also increases while at the same time the loss decreases
- We will pick the model with the highest accuracy (+ / - sign accuracy). Ideally, the model with the lowest loss function will also have the highest accuracy.

GRU Training Results

Results with Training Data (2012-01-01 to 2018-12-31)

Training - Best Output



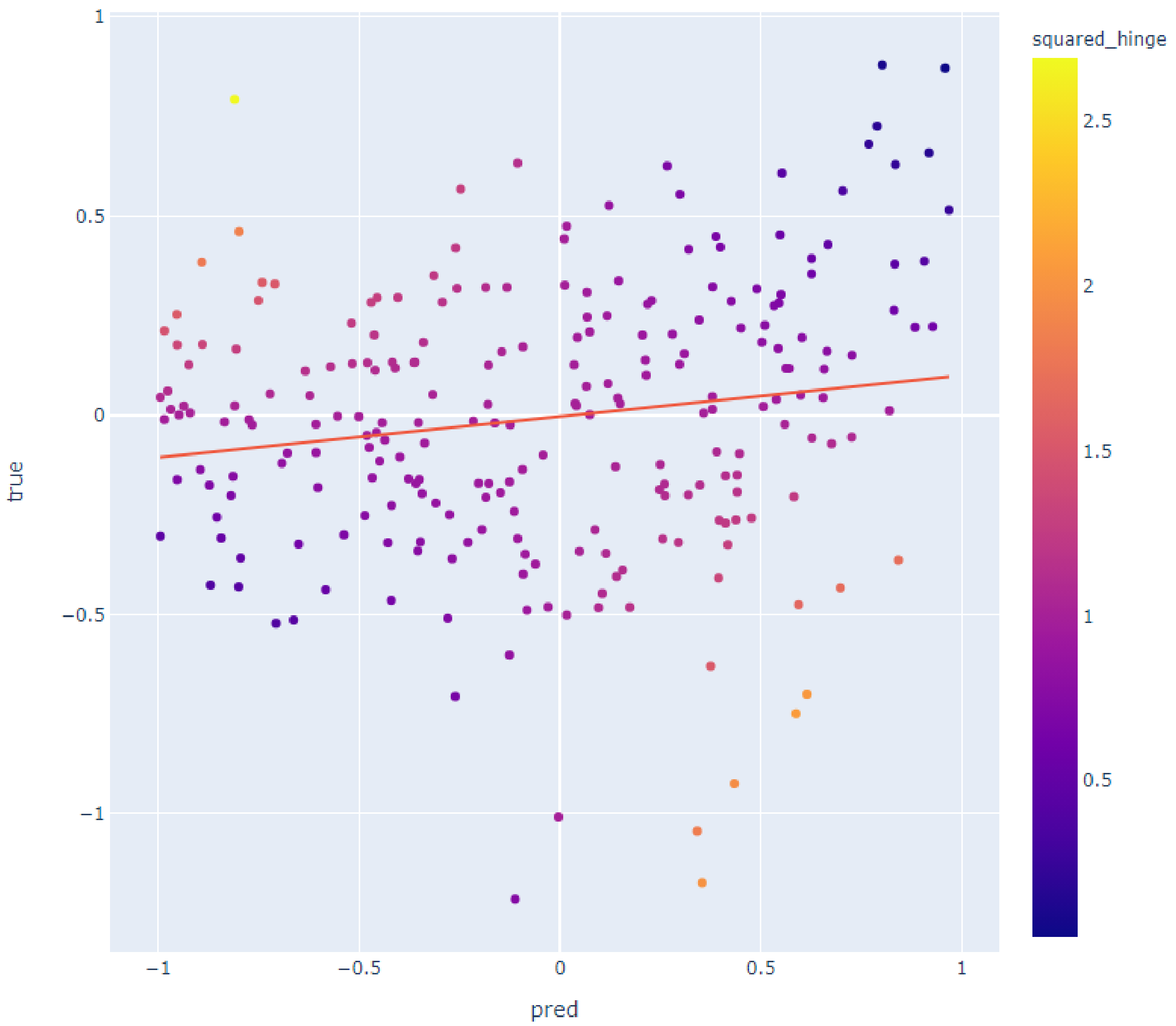


	precision	recall	f1-score	support
Negatives	0.59	0.67	0.63	902
Positives	0.67	0.59	0.63	1026
accuracy			0.63	1928
macro avg	0.63	0.63	0.63	1928
weighted avg	0.63	0.63	0.63	1928

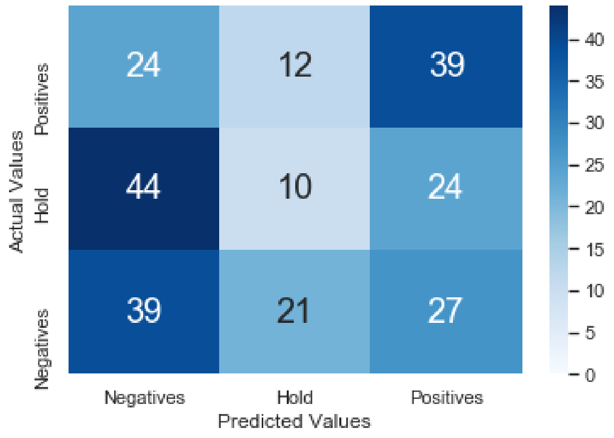
GRU Prediction Results

Prediction Results with Test Data (2019-01-01 to 2019-12-31)

Test - Best Output

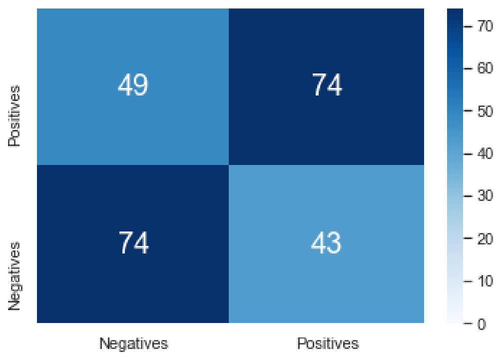


High Conviction Accuracy



	precision	recall	f1-score	support
Negatives	0.59	0.44	0.50	117
Hold	0.21	0.41	0.27	39
Positives	0.51	0.45	0.48	84
accuracy			0.44	240
macro avg	0.43	0.43	0.42	240
weighted avg	0.50	0.44	0.46	240

Overall Accuracy



	precision	recall	f1-score	support
Negatives	0.63	0.60	0.62	123
Positives	0.60	0.63	0.62	117
accuracy			0.62	240
macro avg	0.62	0.62	0.62	240
weighted avg	0.62	0.62	0.62	240

Predictive power over time

Error values spikes in accordance with big news / events

Demo 22 Mei: Korban meninggal, dalang kerusuhan dan 'ada settingan menciptakan martir'

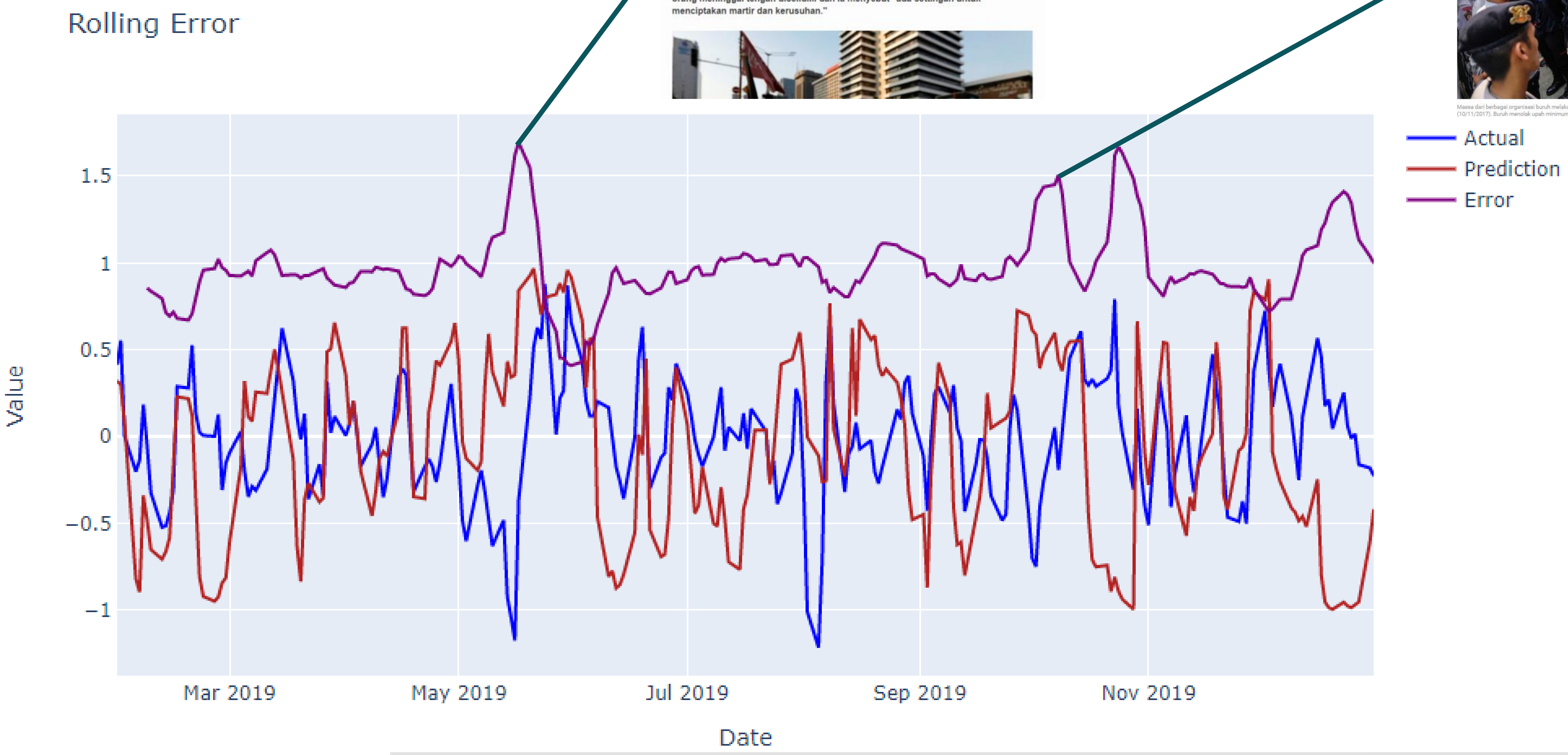
21 Mei 2019
Diperbarui 22 Mei 2019

Kapolri Jendral Tito Karnavian menyatakan kerusuhan yang menyebabkan enam orang meninggal tengah diselidiki dan ia menyebut "ada settingan untuk menciptakan martir dan kerusuhan."

Hari Ini, Sekitar 30.000 Buruh Akan Berunjuk Rasa di Gedung DPR

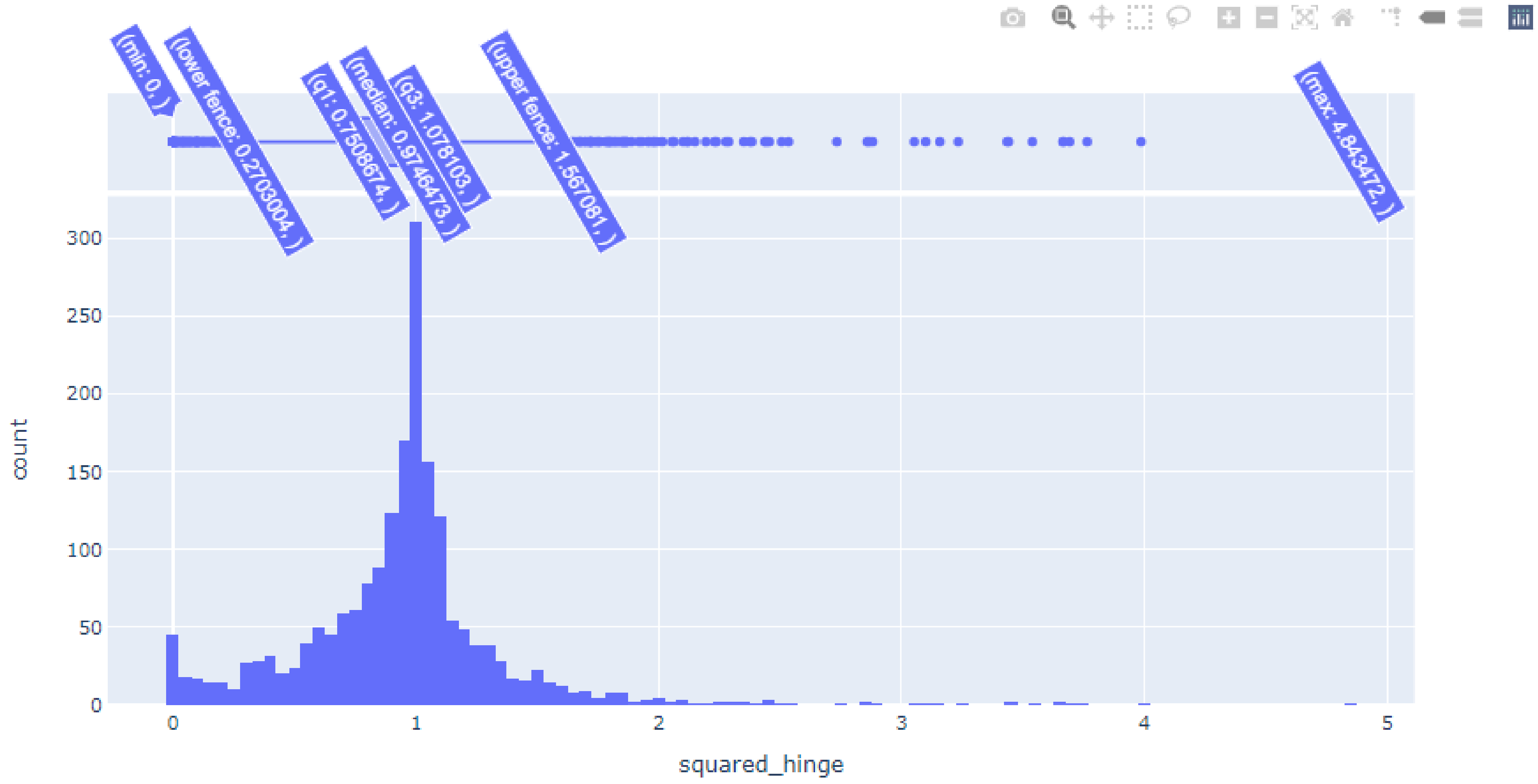
Kompas.com - 02/10/2019, 06:58 WIB

BAGIKAN:



Error Distribution

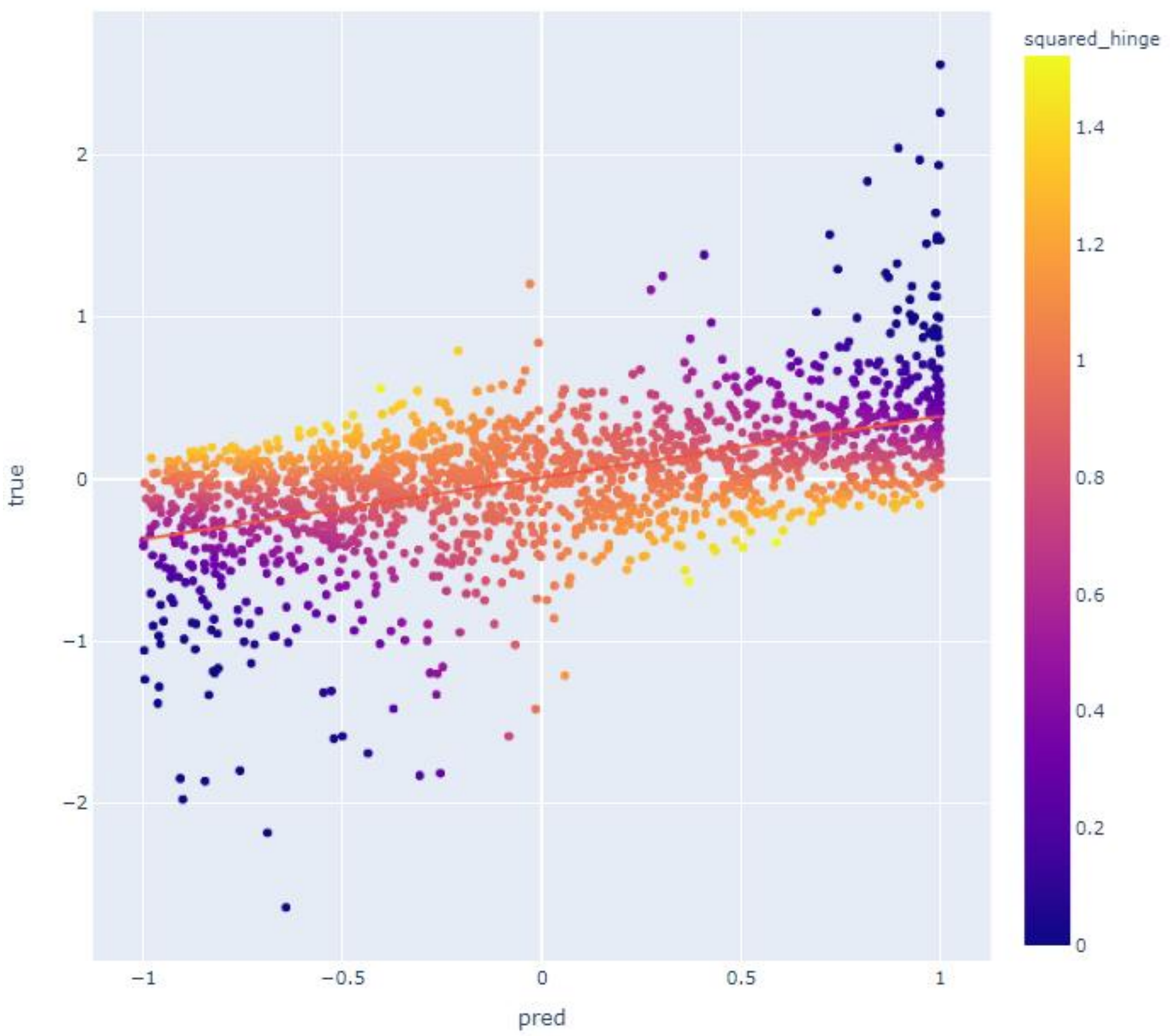
Error value distribution of Training data

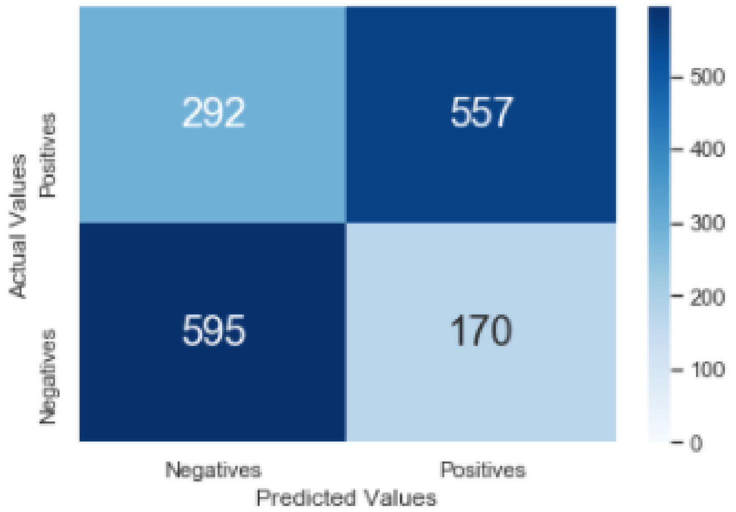


GRU Training Results – Without Outliers (Loss < 1.3)

Filtering high loss prediction back to training dataset

Training - Best Output



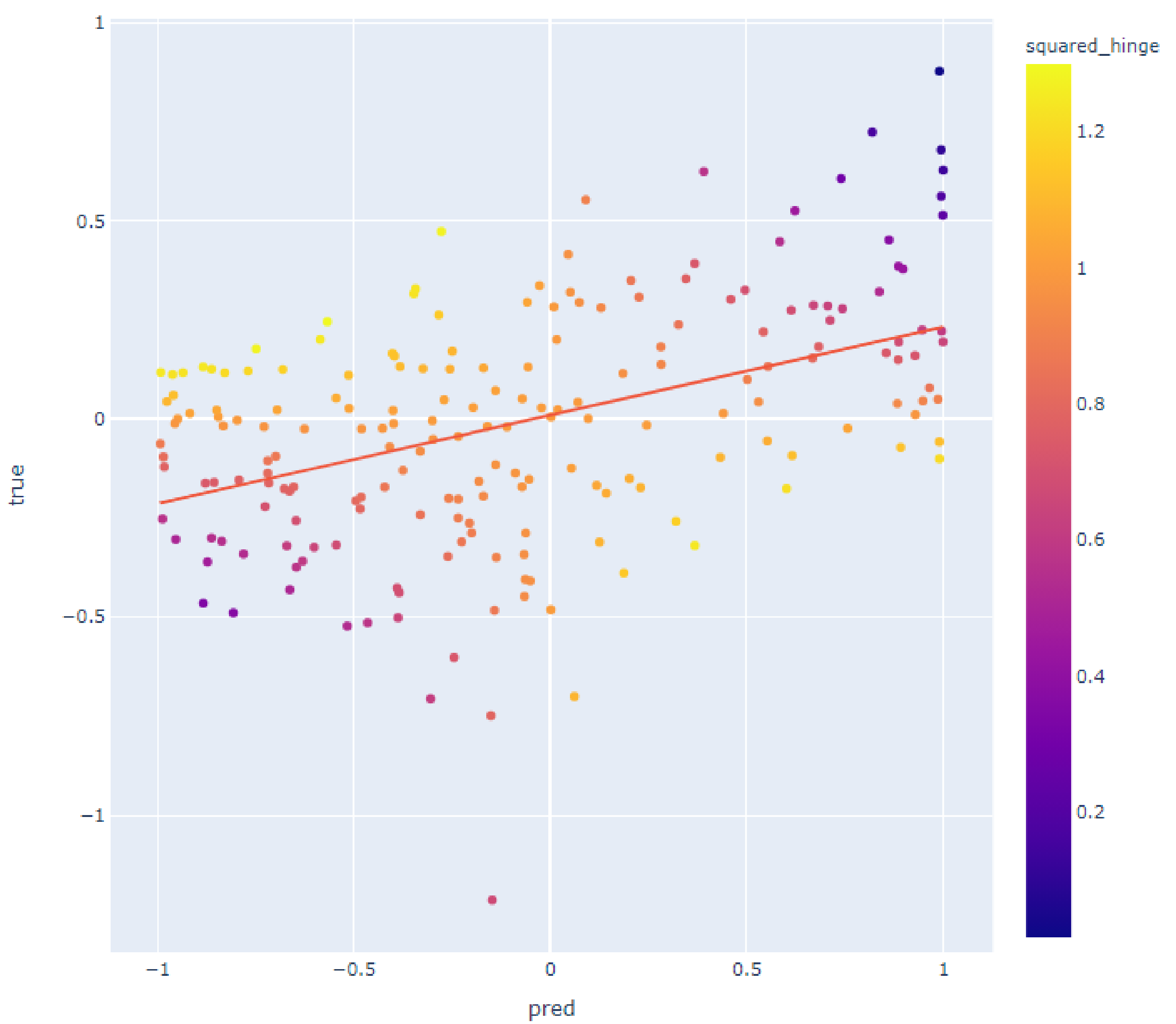


	precision	recall	f1-score	support
Negatives	0.67	0.78	0.72	765
Positives	0.77	0.66	0.71	849
accuracy			0.71	1614
macro avg	0.72	0.72	0.71	1614
weighted avg	0.72	0.71	0.71	1614

GRU Prediction Results – Without Outliers (Loss < 1.3)

Prediction Results with Holdout Test Data (2019-01-01 to 2019-12-31)

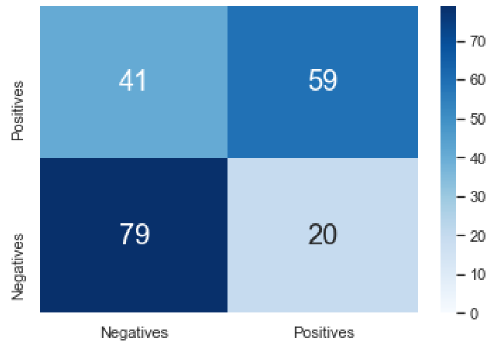
Test - Best Output



High Conviction Accuracy (pred > 0.15 or <-.15)

	precision	recall	f1-score	support
Negatives	0.68	0.45	0.54	101
Hold	0.16	0.33	0.21	36
Positives	0.66	0.60	0.63	62
accuracy			0.47	199
macro avg	0.50	0.46	0.46	199
weighted avg	0.58	0.47	0.51	199

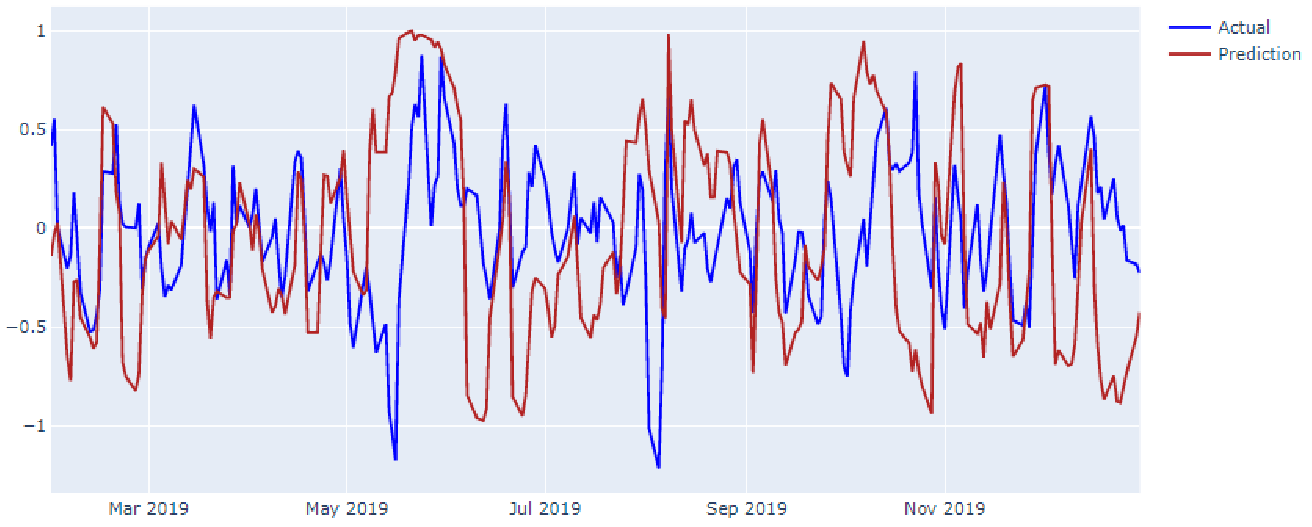
Overall Accuracy

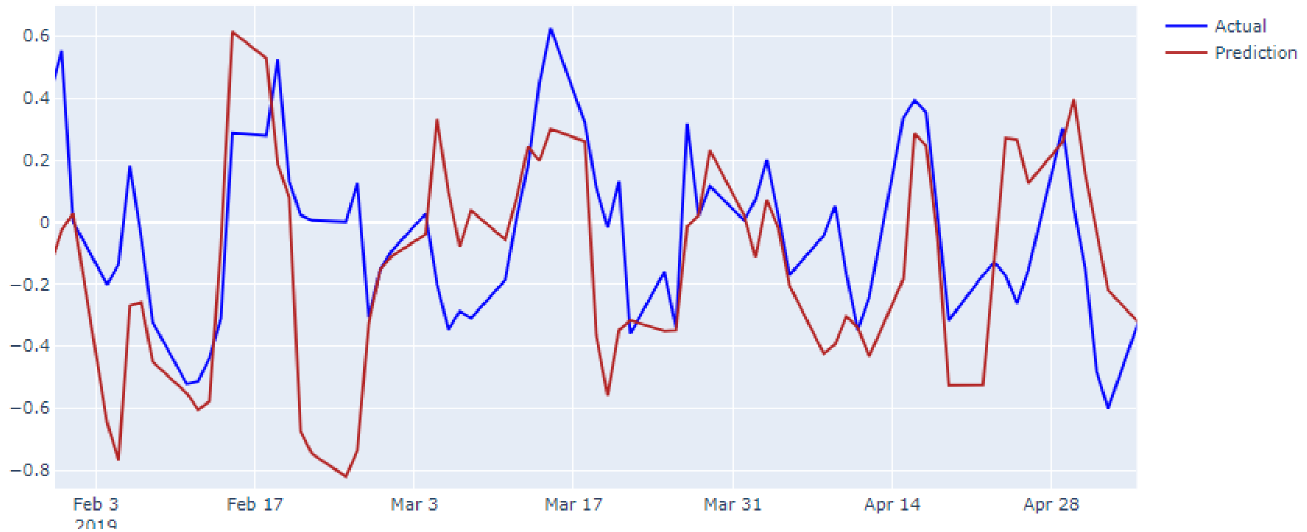


	precision	recall	f1-score	support
Negatives	0.80	0.66	0.72	120
Positives	0.59	0.75	0.66	79
accuracy			0.69	199
macro avg	0.69	0.70	0.69	199
weighted avg	0.72	0.69	0.70	199

Test Result Snapshot

With outliers Removed





- The top left plot is the prediction overlayed by the actual value of the target
- The bottom plot is the zoomed in plot of prediction and actual value from February 3rd 2019 to Apr 28th 2019.
- As we can see, the prediction and actual value are not far apart (in periods with no big events). The most important thing is that the prediction vs actual sign is accurate most of the time.

Trading Simulation (30-01-2019 – 01-01-2020)

Outperforms Buy & Hold Strategy with 2.33% returns over 1 year period

Average *Buys* and *Sells*

position	Close	Open	High	Low
Buy	981.5215	980.0937	986.6265	974.4929
Hold	996.969	997.844	1002.428	991.4431
Sell	1004.396	1006.808	1010.419	999.4105

Projected Returns

	Avg. Buy Price	Avg. Sell Price	Returns	Buy & Hold
Ideal	981.5215	1004.396	2.33%	-0.80%
Best Case	974.4929	1010.419	3.69%	
Worst Case	986.6265	999.4105	1.30%	

- Ideal : Buying all the average *Buys* and selling the average *Sells*
- Best Case : Buying the Low *Buys* and Selling at the High *Sells*
- Worst Case : Buying the High *Buys* and Selling at the Low *Sells*