Model: XGBoost

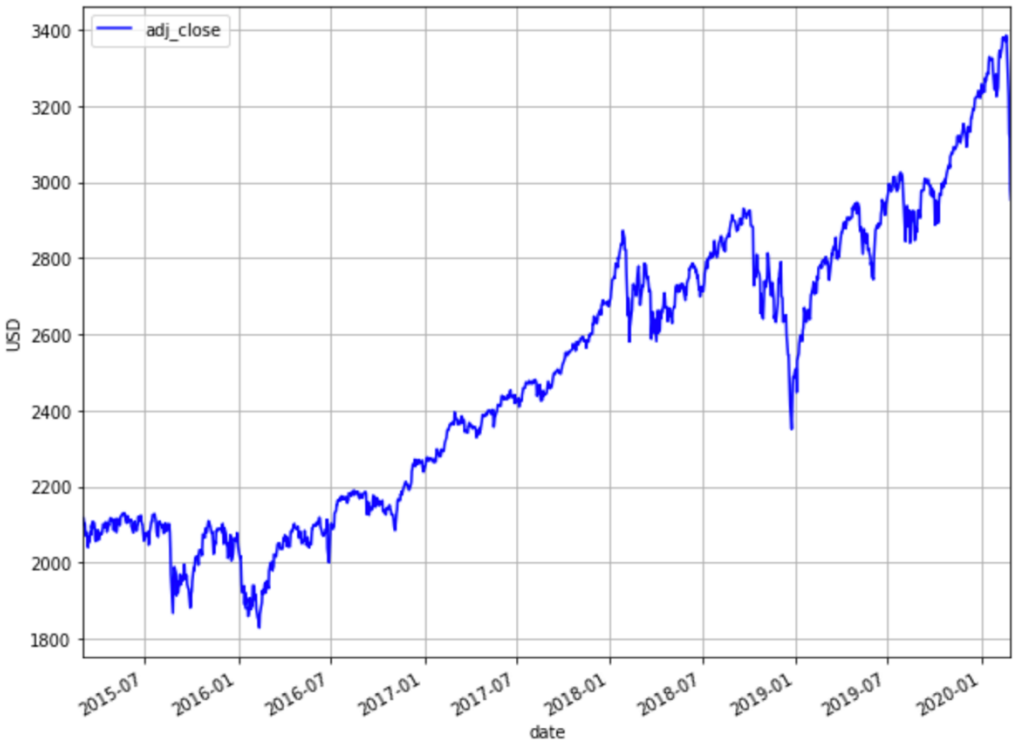
Table of Contents

Problem Statement:

We aim to predict the daily adjusted closing prices of S&P500 (^GSPC), using data from the previous N days (ie. forecast horizon=1). We will use five years of historical prices for S&P500 from '2015-02-28' to '2020-02-28', which can be easily downloaded using **pandas_datareader** from yahoo finance. After downloading, the dataset looks like this:

date	high	low	open	close	volume	adj_close
2015-03-02	2117.520020	2104.500000	2105.229980	2117.389893	3409490000	2117.389893
2015-03-03	2115.760010	2098.260010	2115.760010	2107.780029	3262300000	2107.780029
2015-03-04	2107.719971	2094.489990	2107.719971	2098.530029	3421110000	2098.530029
2015-03-05	2104.250000	2095.219971	2098.540039	2101.040039	3103030000	2101.040039
2015-03-06	2100.909912	2067.270020	2100.909912	2071.260010	3853570000	2071.260010

- Plot adjusted close over time.



Split Train, validation, test

We will split this dataset into train ('2015-02-28' to '2019-06-03'), validation ('2019-06-03' to '2019-12-31), and test ('2020-01-01' to '2020-02-28'). The model will be trained using the train set, model hyperparameters will be tuned using the validation set, and finally the performance of the model will be reported using the test set. Below plot shows the adjusted closing price split up into the respective train, validation and test sets.

Model Evaluation

To evaluate the effectiveness of our methods, we will use the root mean square error (RMSE) and mean absolute percentage error (MAPE) metrics. For both metrics, the lower the value, the better the prediction.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

XGBoost Model

features

We will train the XGBoost model on the train set, tune its hyperparameters using the validation set, and finally apply the XGBoost model on the test set and report the results. Obvious features to use are the adjusted closing prices of the last N days, as well as the volume of the last N days. In addition to these features, we can do some feature engineering. The additional features we will construct are:

1. Difference between high and low for each day of the last N days;
2. Difference between open and close for each day of the last N days.

Feature Scaling

We know that when constructing XGBoost, feature scaling is very important for the model to work properly. If the model trained on adjusted closing price within a certain range, and then, the model can only output predictions among this range. When the model is trying to predict the validation set and it saw values out of this range, it is not able to generalize well.

After conducting some research online, and tried several scaling methods, we scaled the train set to have mean 0 and variance 1, and use this to train the model. Subsequently, when we are doing predictions on the validation set, for each feature group of each sample, we will scale them to have mean 0 and variance 1.

For example, if we are doing predictions on day T, we will take the adjusted closing prices of the last N days (days T-N to T-1) and scale them to have mean 0 and variance 1. The same is done for the volume features, where we will take the volume of the last N days and scale them to have mean 0 and variance 1. Repeat the same for the additional features we constructed above. We then use these scaled features to do prediction. The predicted values will also be scaled and we inverse transform them using their corresponding mean and variance. We found that this way of scaling gives the best performance.

Tuning Hyperparameters

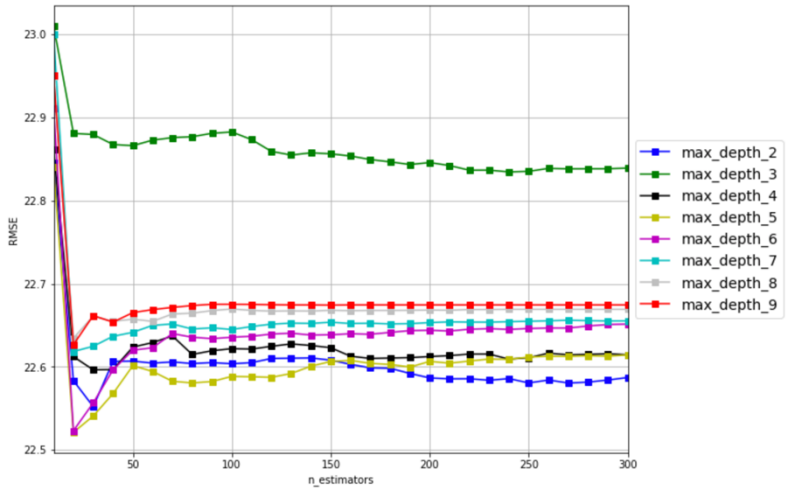
1. Determine Feature – N days

Below plot shows the RMSE between the actual and predicted values on the validation set, for various values of N. We will use N=3 since it gives the lowest RMSE.

	N	rmse_dev_set	mape_pct_dev_set
0	2	1.225	0.585
1	3	1.214	0.581
2	4	1.231	0.590
3	5	1.249	0.601
4	6	1.254	0.609
5	7	1.251	0.612
6	14	1.498	0.763

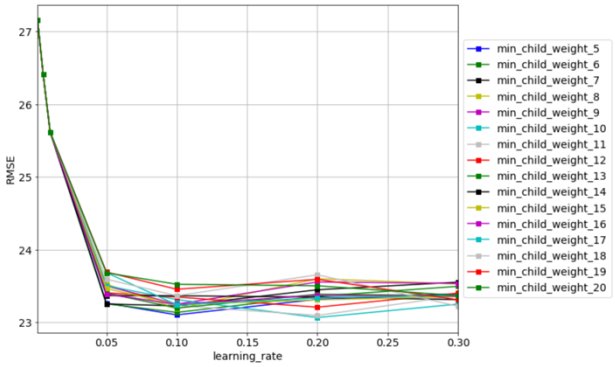
Use N = 3 for lowest RMSE and MAPE

2. Tuning XGBoost - n_estimators (default=100) and max_depth (default=3)



Get optimum value: min RMSE = 22.521 => optimum params = (20, 5)
=> choose: n_estimators=20 and max_depth=5

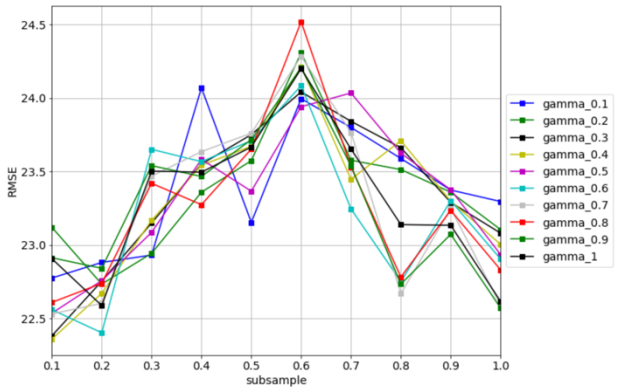
3. Tuning XGBoost - learning_rate(default=0.1) and min_child_weight(default=1)



Get optimum value: min RMSE = 23.064 => optimum params = (0.2,10)

=> choose: learning_rate =0.2 and min_child_weight =10

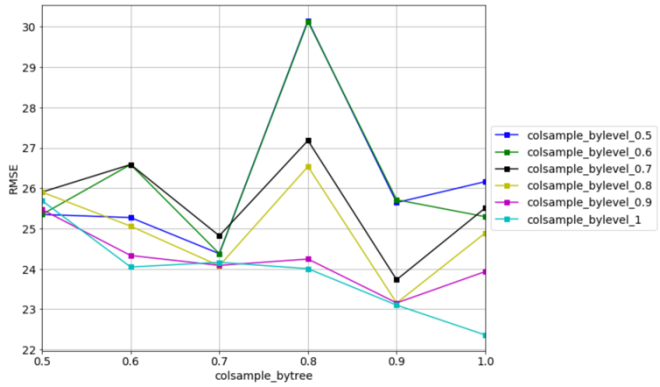
4. Tuning XGBoost - subsample(default=1) and gamma(default=0)



Get optimum value: min RMSE = 22.358 => optimum params = (0.1,0.4)

=> choose: subsample =0.1 and gamma =0.4

5. Tuning XGBoost - colsample_bytree(default=1) and colsample_bylevel(default=1)



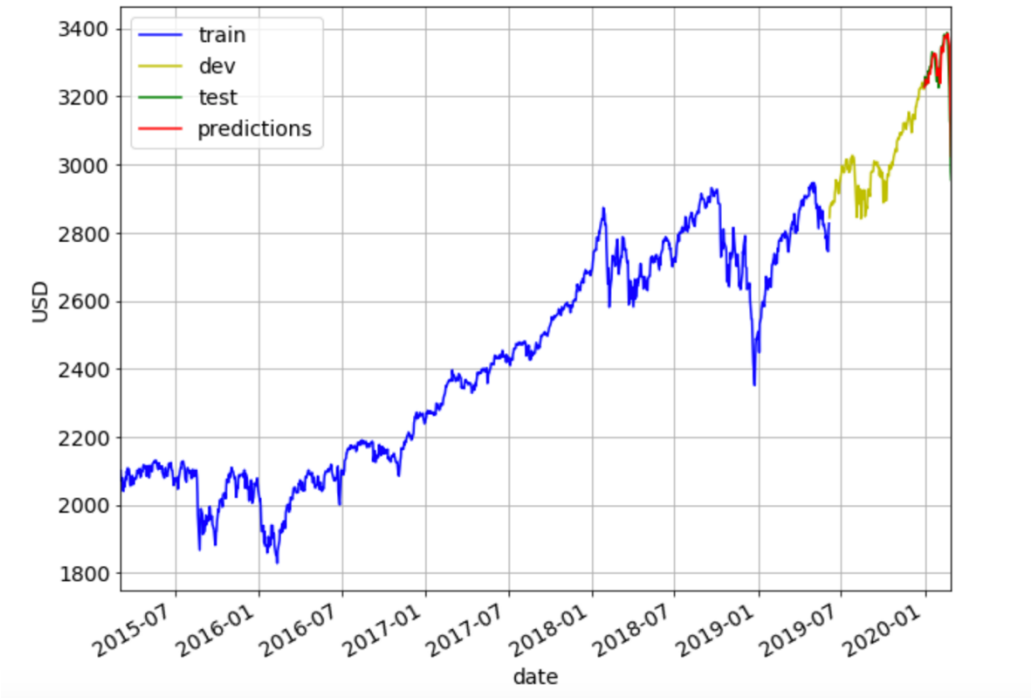
Get optimum value: min RMSE = 22.358 => optimum params = (1.0, 1.0)

=> choose: colsample_bytree =1 and colsample_bylevel =1

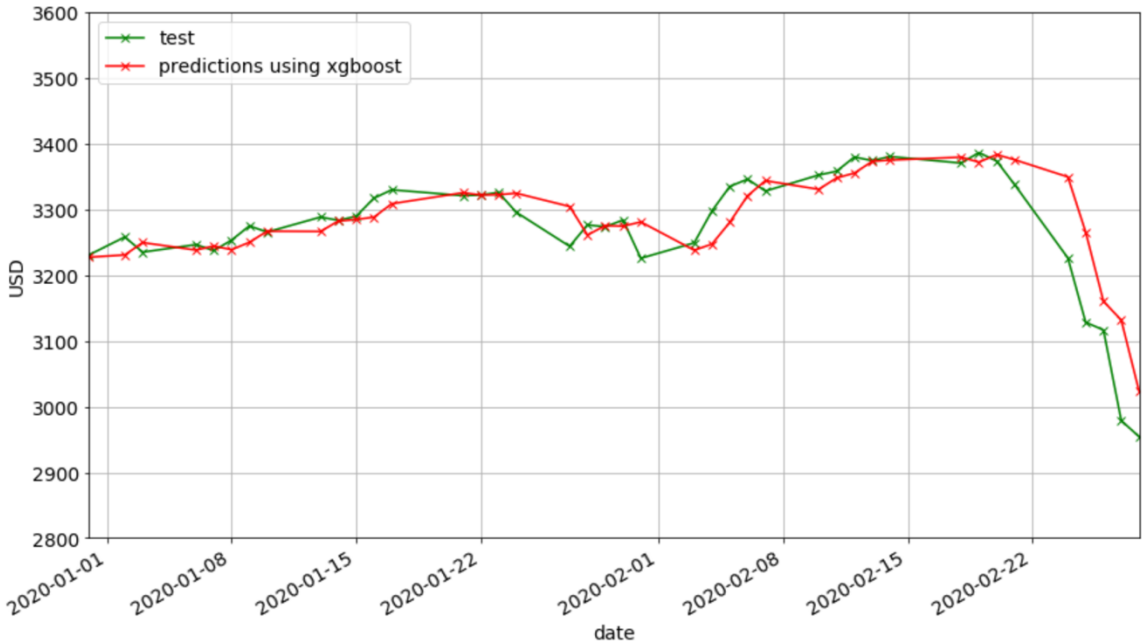
- Tuned params Summary (Final Model)

	param	original	after_tuning
0	n_estimators	100.000	20.000
1	max_depth	3.000	5.000
2	learning_rate	0.100	0.200
3	min_child_weight	1.000	10.000
4	subsample	1.000	0.100
5	colsample_bytree	1.000	1.000
6	colsample_bylevel	1.000	1.000
7	gamma	0.000	0.400
8	rmse	22.882	22.358
9	mape_pct	0.554	0.544

Prediction result

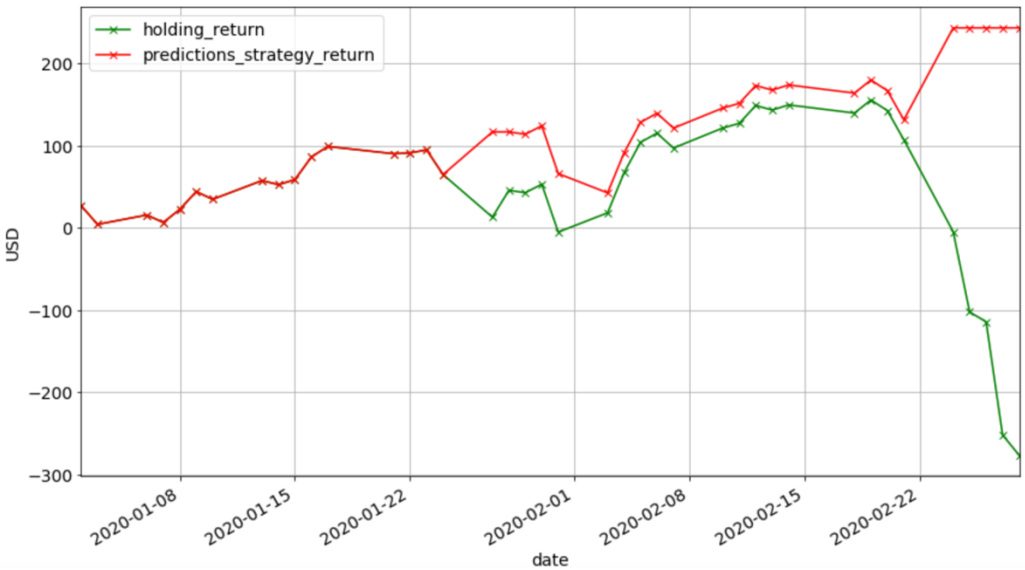


Zoom in to test set



Trading Strategy based on Prediction Result

- State 1 (buy): if prediction price tomorrow is higher than today actual price within the range 10;
- State 2: (sell): if prediction price tomorrow is lower than today actual price within the range 10;



From our return results, we notice that, with the accurate prediction result, we can prevent loss when the price is dramatically decreasing.