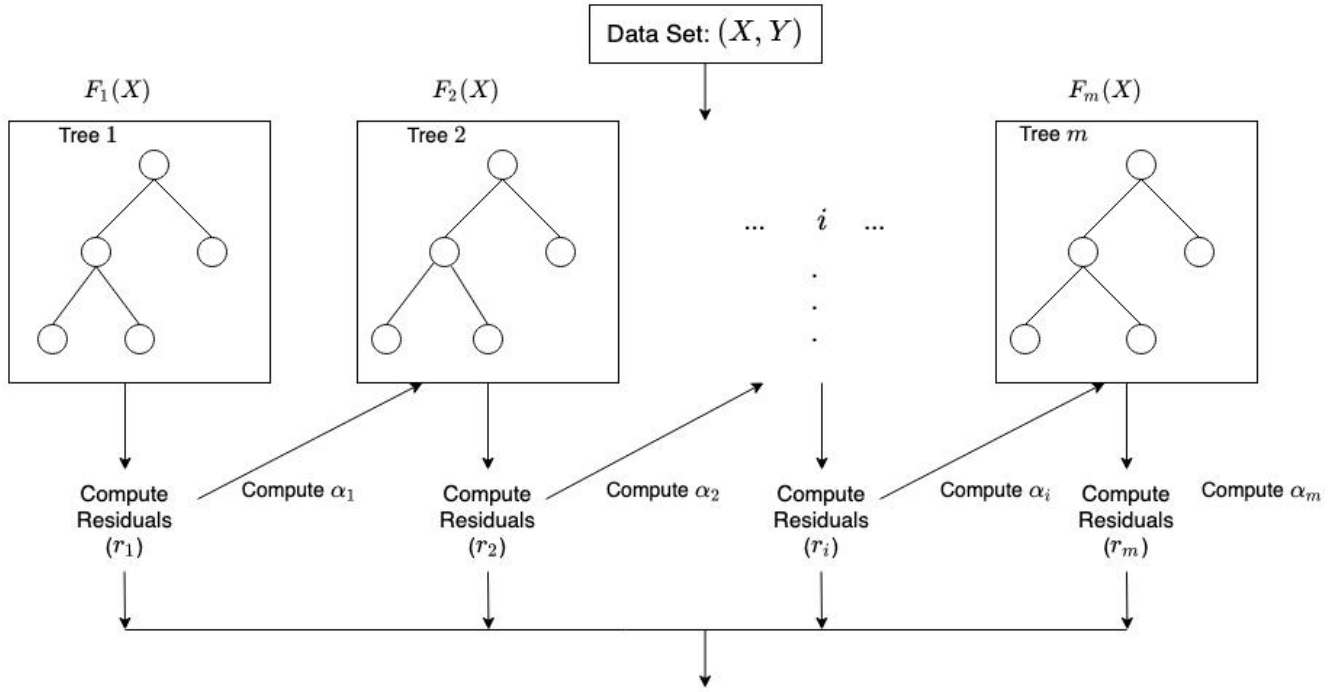
Data Preprocessing

We selected NVIDIA's five-year stock market data from March 18, 2019 to March 18, 2024. In our XGBoost and LSTM model, we firstly split the whole data set into a 70% training set, 25% validation set, and a 5% test set. We used the training set to train our models, then used the validation set to tune the hyper parameters for lower RMSE and MAPE. After selecting the best hyper parameters, we fed the training set and validation set into the model, and finally used the test set to test our predictions and compare the results. In our ARIMA and VAR model, we split the data set into a 90% training set and a 10% test set, for there are not so many complex hyper parameters as XGBoost and LSTM to tune.

After splitting the data set, we scaled the "Adj Close" column in each set respectively to make the observations in each set follow a standard normal distribution (mean=0 and standard deviation=1).

XGBoost

XGBoost (eXtreme Gradient Boosting) is a scalable, portable and distributed gradient-boosted decision tree (GBDT) model. It provides parallel tree boosting and is the leading machine learning library for regression problems. The workflow of XGBoost is shown in the figure below.

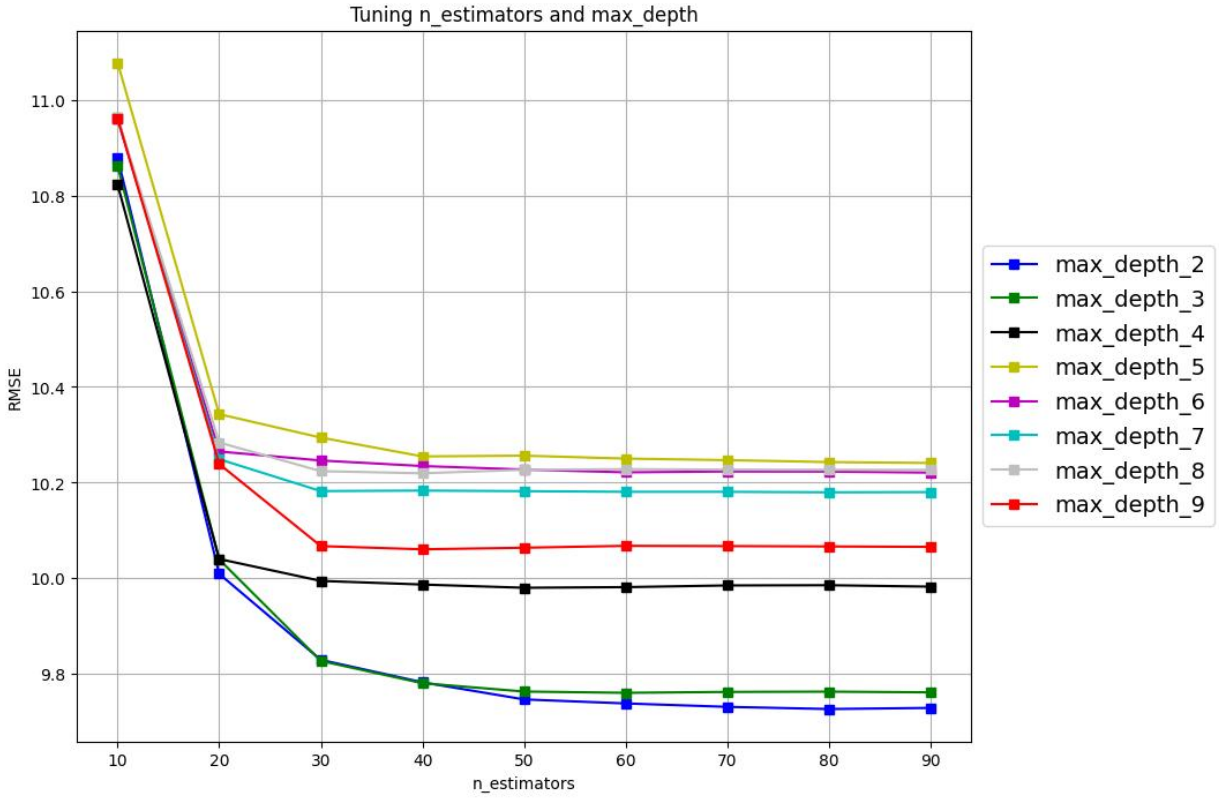


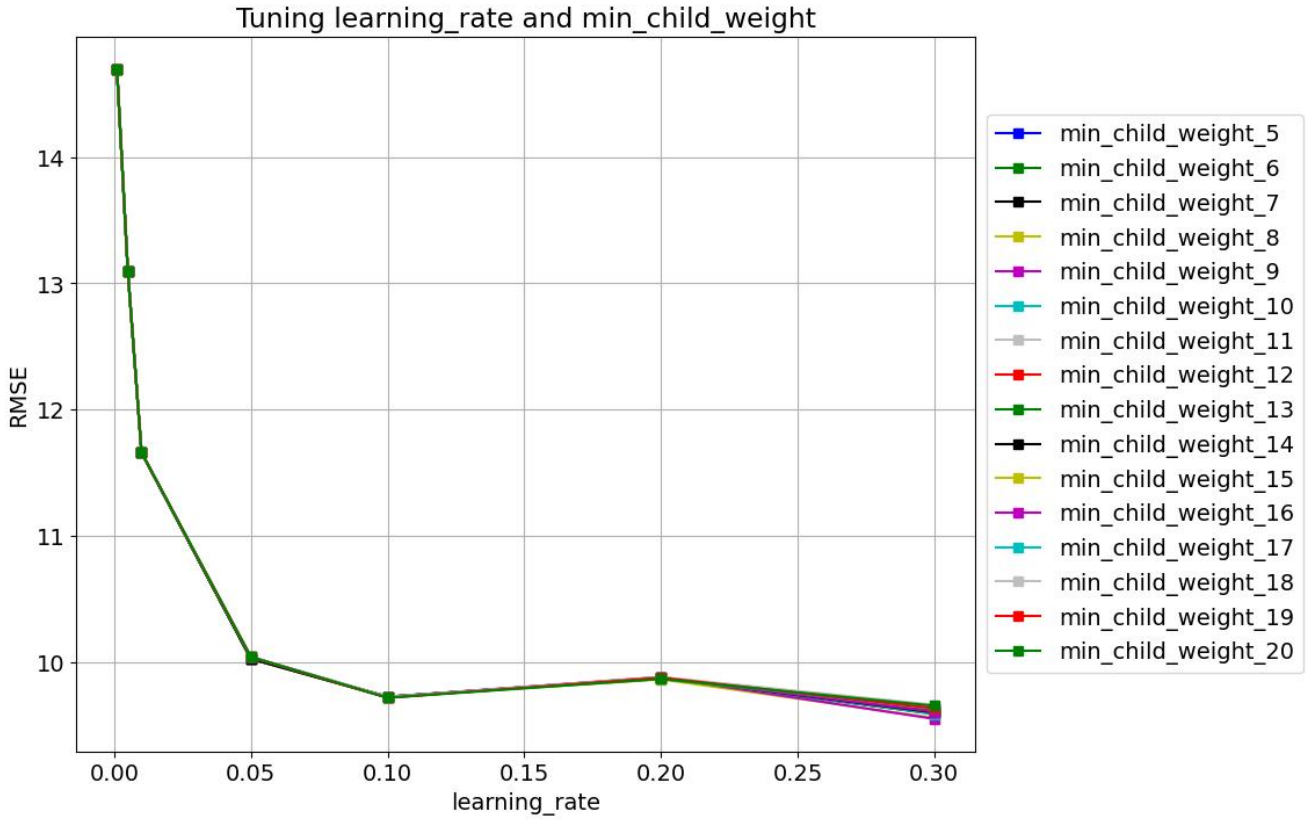
https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html (pic source)

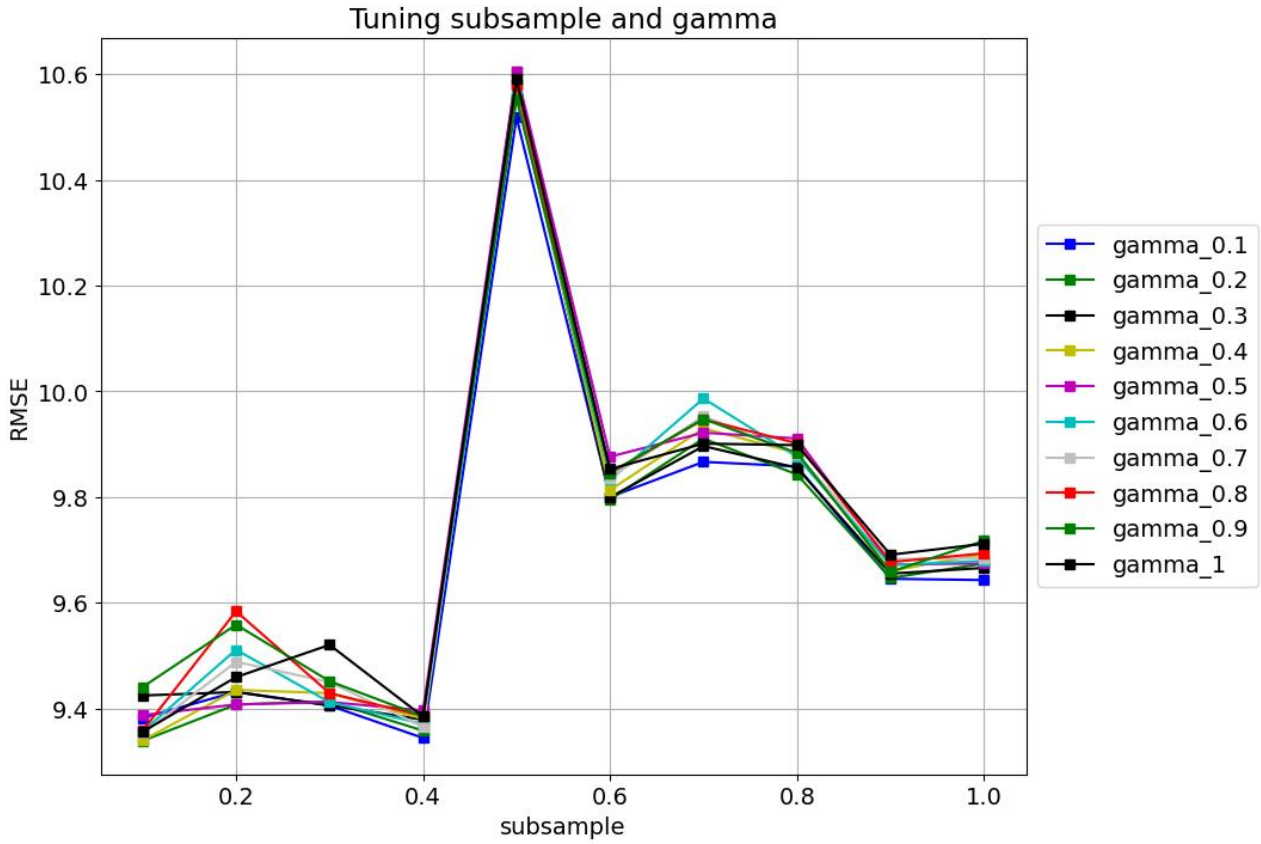
For our XGBoost model, before the data preprocessing mentioned above, we also performed feature engineering. We generated the following new features:

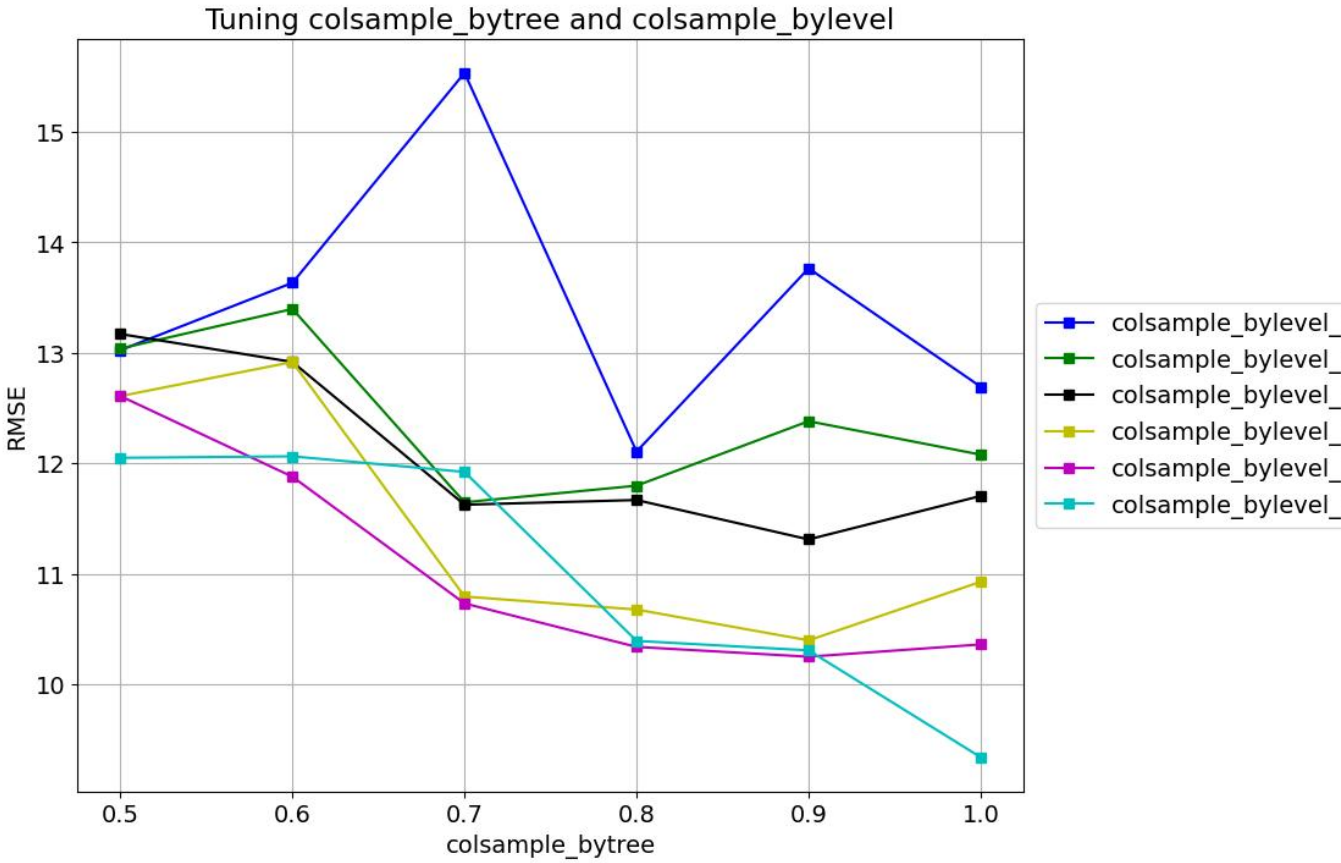
- Mean 'Adj Close' of each month
- Difference between ‘High’ and ‘Low’ of each day
- Difference between ‘Open’ and ‘Close’ of each day
- Mean ‘Volume’ of each month

Then we dropped the ‘High’, ‘Low’, ‘Open’, and ‘Close’ columns, and added other columns indicating lag information. We used a total of 28 features in our XGBoost model to predict the ‘Adj Close’ of each day. Then we tuned the hyper parameters.




Tuning learning_rate and min_child_weight
RMSE
learning_rate
14
13
12
11
10
0.00
0.05
0.10
0.15
0.20
0.25
0.30
min_child_weight_5
min_child_weight_6
min_child_weight_7
min_child_weight_8
min_child_weight_9
min_child_weight_10
min_child_weight_11
min_child_weight_12
min_child_weight_13
min_child_weight_14
min_child_weight_15
min_child_weight_16
min_child_weight_17
min_child_weight_18
min_child_weight_19
min_child_weight_20

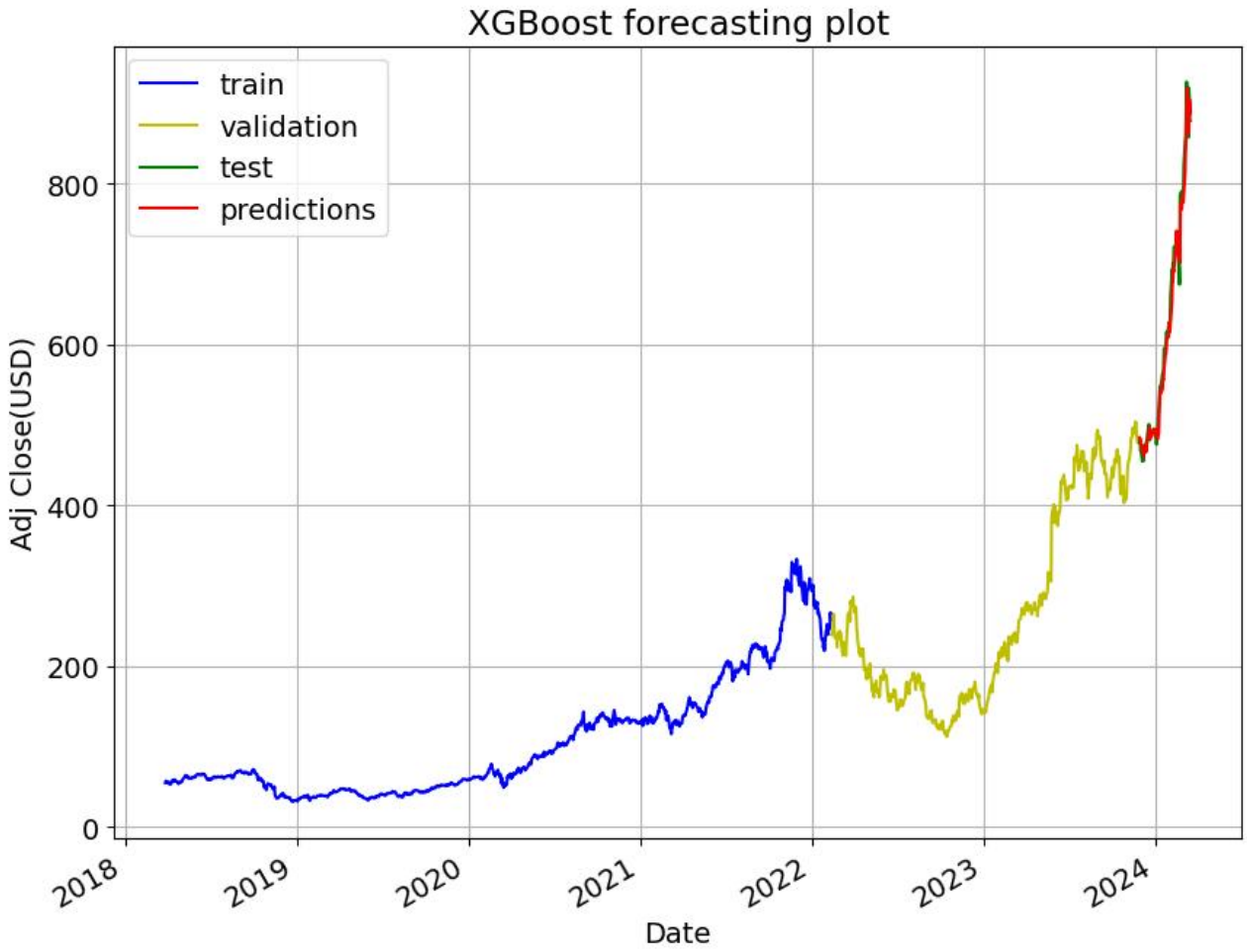

Tuning subsample and gamma
RMSE
subsample
10.6
10.4
10.2
10.0
9.8
9.6
9.4
0.2
0.4
0.6
0.8
1.0
gamma_0.1
gamma_0.2
gamma_0.3
gamma_0.4
gamma_0.5
gamma_0.6
gamma_0.7
gamma_0.8
gamma_0.9
gamma_1

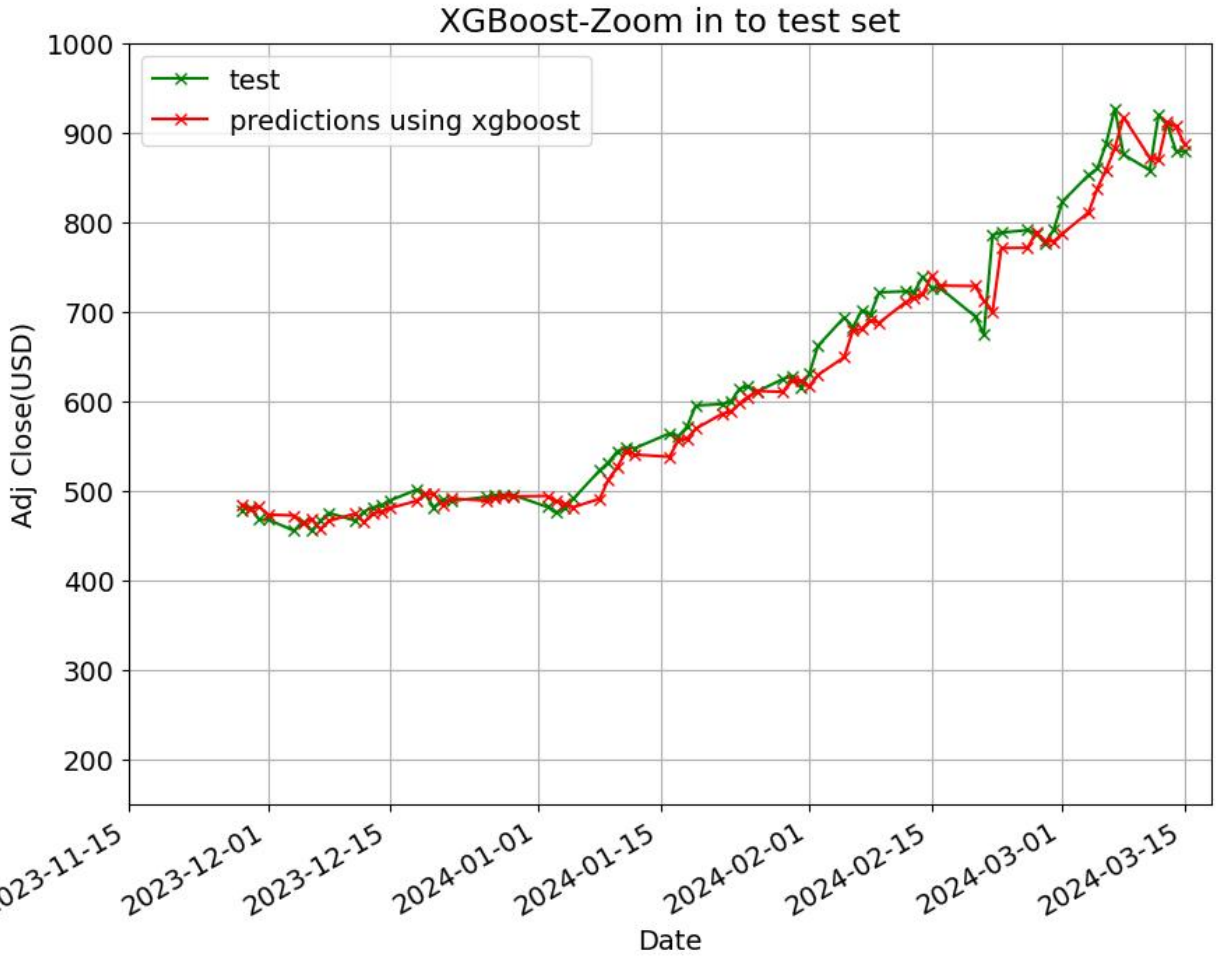


	param	original	after_tuning
0	n_estimators	100.000	80.000
1	max_depth	3.000	2.000
2	learning_rate	0.100	0.300
3	min_child_weight	1.000	9.000
4	subsample	1.000	0.100
5	colsample_bytree	1.000	1.000
6	colsample_bylevel	1.000	1.000
7	gamma	0.000	0.200
8	rmse	9.762	9.339
9	mape_pct	2.808	2.700

After tuning hyper parameters, our final model yielded a relatively good result:

```
RMSE on test set = 21.366
MAPE on test set = 2.355%
```


XGBoost forecasting plot
train
validation
test
predictions
Adj Close(USD)
0
200
400
600
800
2018
2019
2020
2021
2022
2023
2024
Date



ARIMA

ARIMA (Autoregressive Integrated Moving Average) is a classic time series forecasting model used to analyze and predict data with temporal dependencies. It combines three components: autoregression (AR), differencing (I), and moving average (MA).

Autoregression captures the relationship between an observation and a certain number of lagged observations. It assumes that the current value of a variable can be predicted using its own previous values. The order of autoregression, denoted as **AR(p)**, specifies the number of lagged terms considered in the model.

The differencing component is used to make the time series data stationary, which means removing any trend or seasonality. Differencing involves subtracting the previous observation from the current observation. The order of differencing, denoted as **I(d)**, indicates the number of times differencing is applied to achieve stationarity.

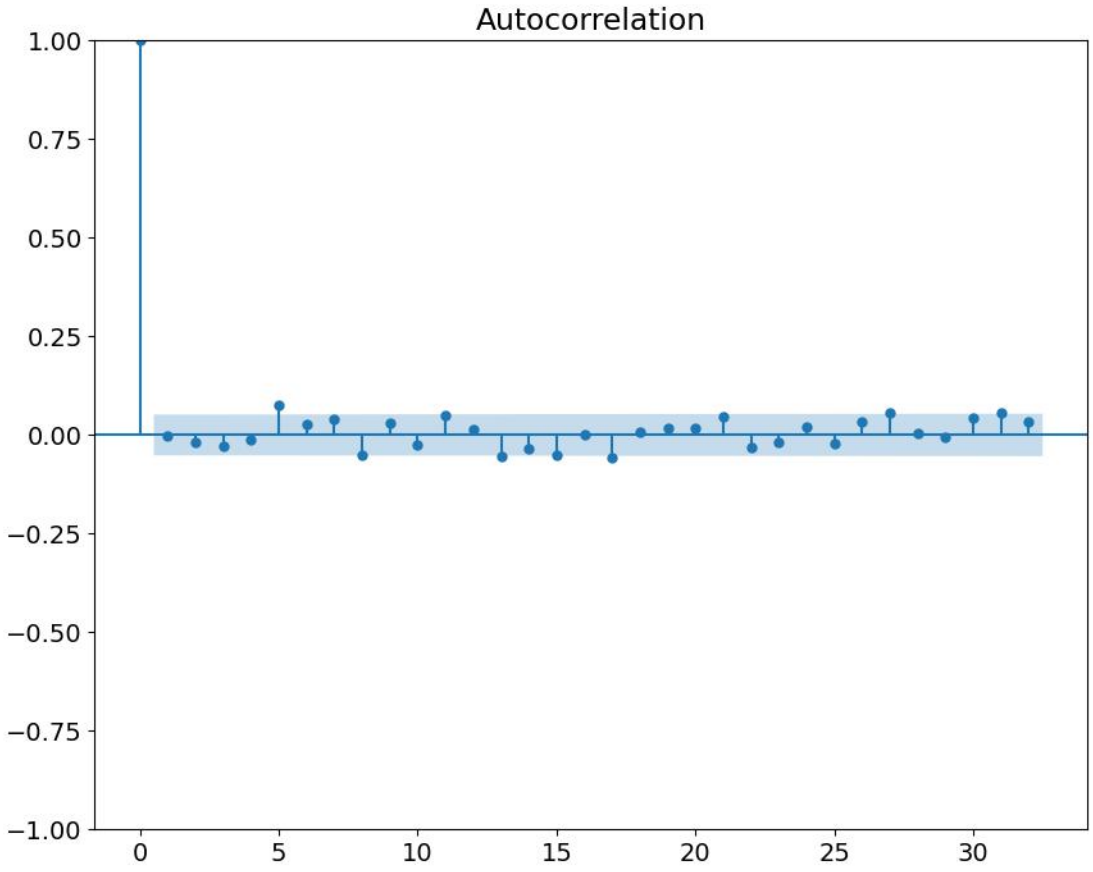
Moving average considers the dependency between an observation and a residual error from a moving average model applied to lagged observations. It helps to capture the short-term fluctuations in the data. The order of the moving average, denoted as **MA(q)**, specifies the

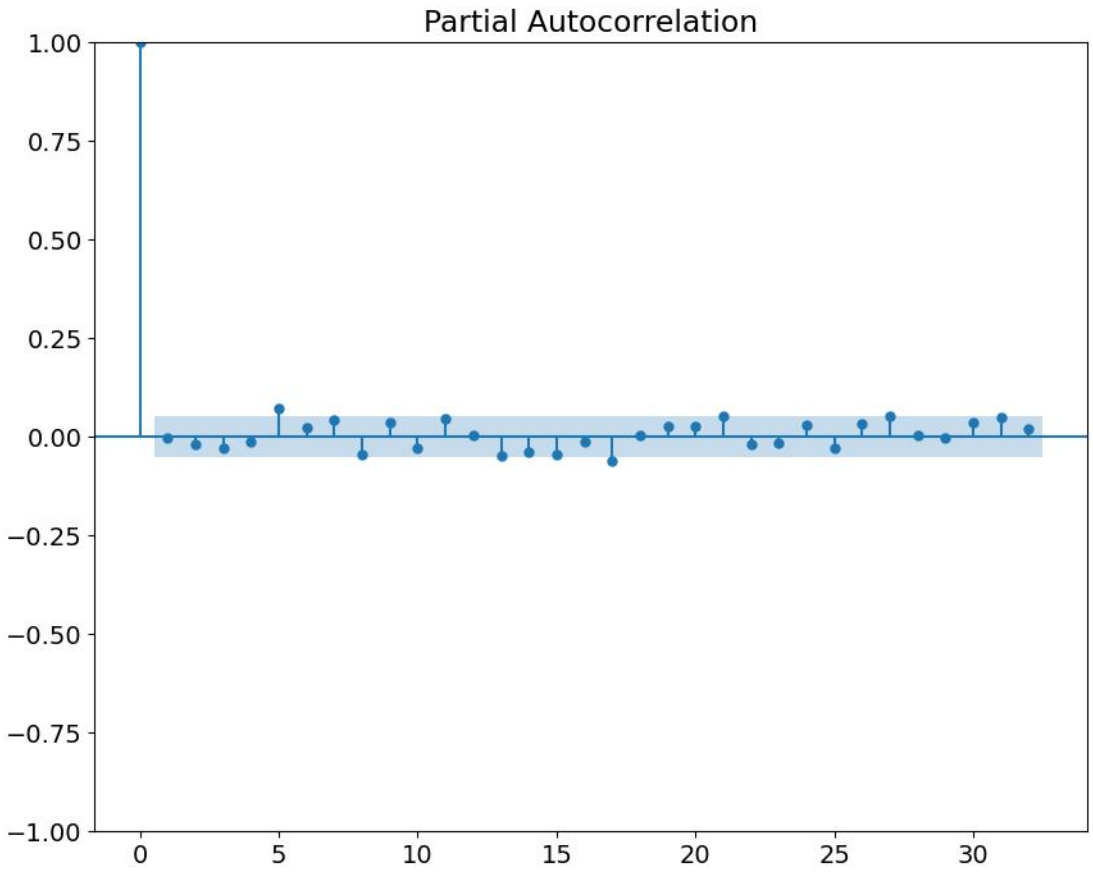
number of lagged forecast errors considered in the model.

The ARIMA model combines these three components as **ARIMA(p, d, q)**. It represents a linear regression model where the dependent variable is differenced to achieve stationarity and is then regressed on the lagged values of itself and the lagged forecast errors.

In our ARIMA forecasting model, in addition to data preprocessing mentioned above, we do the ADF (Augmented Dickey-Fuller) test to check whether the data is stationary. ADF test shows that p-value is about 0.995 which is much greater than 0.05, indicating the data is not stationary. So we do the differencing.

After differencing, p-value from ADF test became very close to 0. We also examined the ACF (autocorrelation function) and PACF (partial autocorrelation function) plots after differencing.





From the above results, we can determine that the order of differencing (d) equal to 1.

The parameters p and q also need to be tuned based on the characteristics of the time series data. This is done automatically using the auto_arima function which fit the model many times to find the best model which yield the lowest AIC (Akaike Information Criterion).

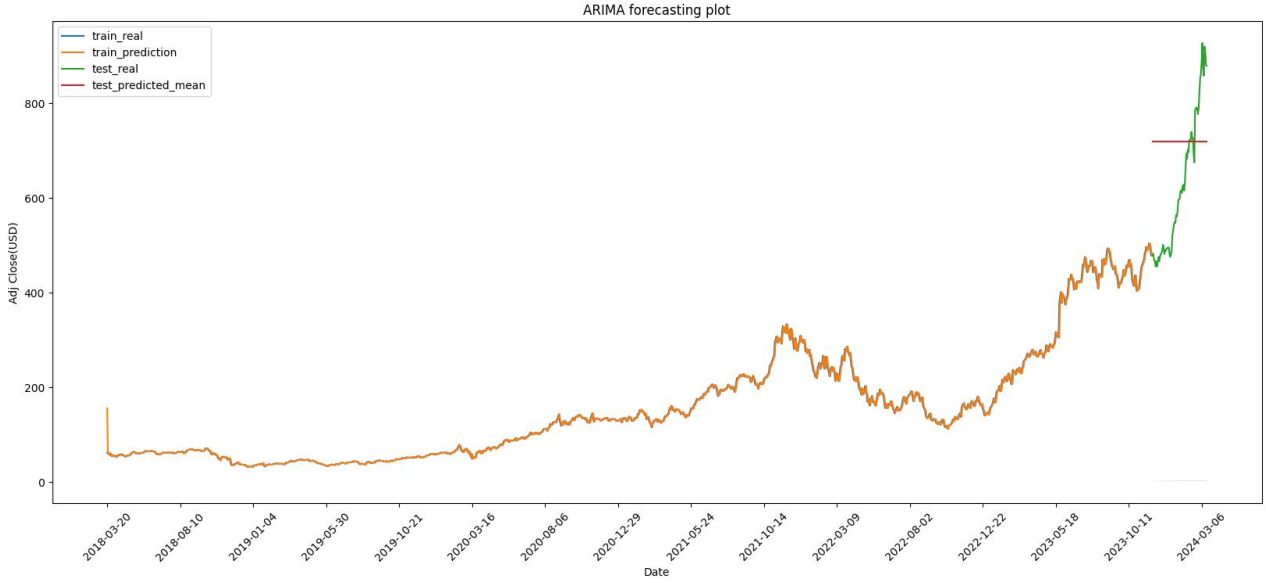
Our final ARIMA model result：

SARIMAX Results

Dep. Variable:	adj close	No. Observations:	1434
Model:	ARIMA(0, 1, 0)	Log Likelihood	2212.487
Date:	Sun, 24 Mar 2024	AIC	-4422.973
Time:	12:06:48	BIC	-4417.705
Sample:	0	HQIC	-4421.006
	- 1434		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
sigma2	0.0027	3.04e-05	87.751	0.000	0.003	0.003

Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	22790.14
Prob(Q):	0.90	Prob(JB):	0.00
Heteroskedasticity (H):	44.86	Skew:	1.64
Prob(H) (two-sided):	0.00	Kurtosis:	22.26



RMSE on test set = 430.646

MAPE on test set = 73.176%