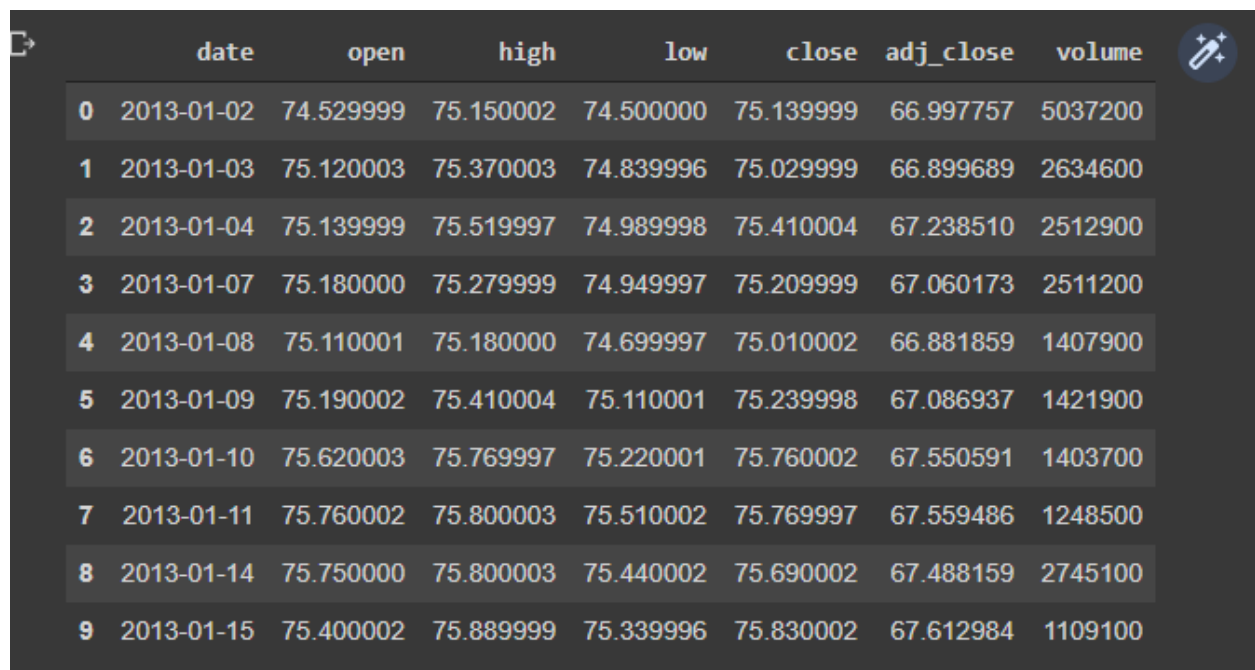


XGBOOST Implementation

By Lubdhak Mondal and Kapil Chandak

Rough Problem Statement

we aim to predict the **daily returns** of Vanguard Total Stock Market ETF (VTI), using data from the previous N days. In this experiment, we will use 6 years of historical prices for VTI from 2013-01-02 to 2018-12-28, which can be easily downloaded from yahoo finance.



	date	open	high	low	close	adj_close	volume
0	2013-01-02	74.529999	75.150002	74.500000	75.139999	66.997757	5037200
1	2013-01-03	75.120003	75.370003	74.839996	75.029999	66.899689	2634600
2	2013-01-04	75.139999	75.519997	74.989998	75.410004	67.238510	2512900
3	2013-01-07	75.180000	75.279999	74.949997	75.209999	67.060173	2511200
4	2013-01-08	75.110001	75.180000	74.699997	75.010002	66.881859	1407900
5	2013-01-09	75.190002	75.410004	75.110001	75.239998	67.086937	1421900
6	2013-01-10	75.620003	75.769997	75.220001	75.760002	67.550591	1403700
7	2013-01-11	75.760002	75.800003	75.510002	75.769997	67.559486	1248500
8	2013-01-14	75.750000	75.800003	75.440002	75.690002	67.488159	2745100
9	2013-01-15	75.400002	75.889999	75.339996	75.830002	67.612984	1109100

Raw Data

Steps Involved

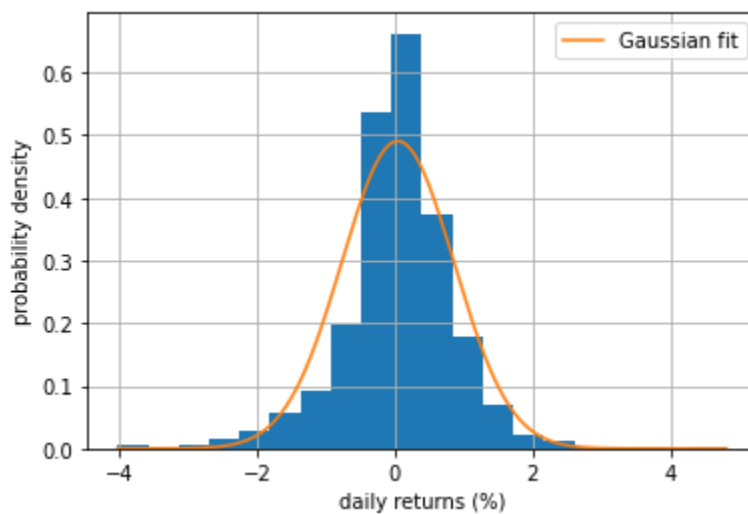
- Problem Statement
- Exploratory Data Analysis
- Feature Engineering
- Training, Validation, Test split
- Hyperparameter Tuning
- Applying the Model
- Findings

Altogether, we have 1509 days to work with. (Saturdays and Sundays are not included). A plot of the adjusted closing price in the entire dataset.



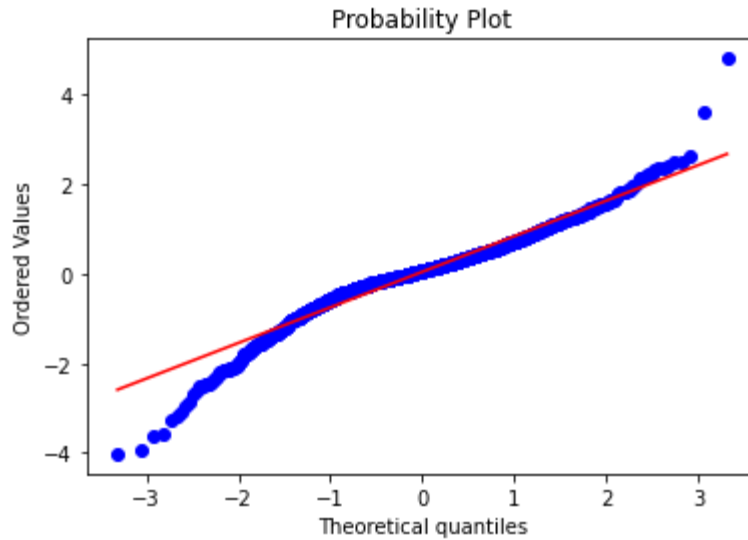
Daily Returns $r(t) = (p(t)/p(t-1) - 1) \times 100$

where $r(t)$ and $p(t)$ denote the daily returns and adjusted closing price on day t respectively.



Distribution plot of the daily returns.

Observation: the distribution plot approximates quite well to a Gaussian distribution.



Observation: The distribution of the daily returns fits the Gaussian distribution pretty well, except at the extreme values when the sample distribution is compared with the Gaussian distribution by default.

Explanation: The better the fit to the straight red line, the better the fit to the Gaussian distribution.

Evaluation of the effectiveness of implemented methods

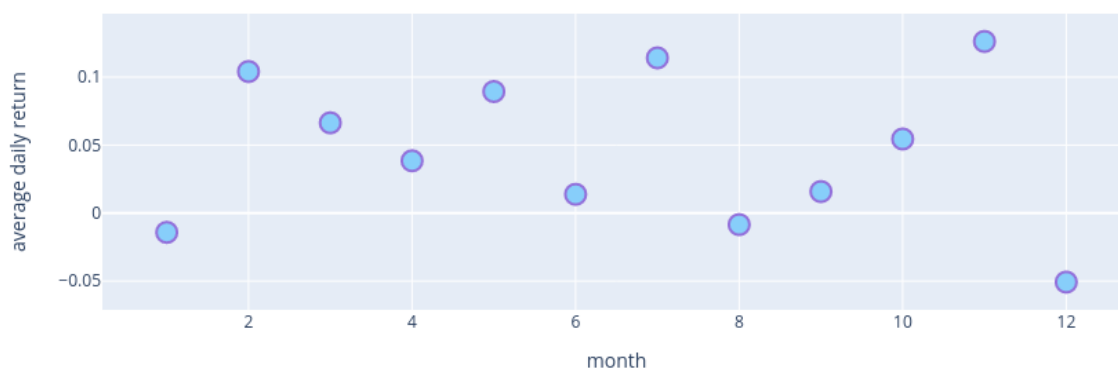
- root mean square error (RMSE)
- mean absolute percentage error (MAPE)
- mean absolute error (MAE)

We first predict the daily returns, then convert back to price based on the daily returns formula above, and compute the metrics with respect to the actual prices. For all metrics, the lower the value, the better the prediction.

Exploratory Data Analysis (EDA)

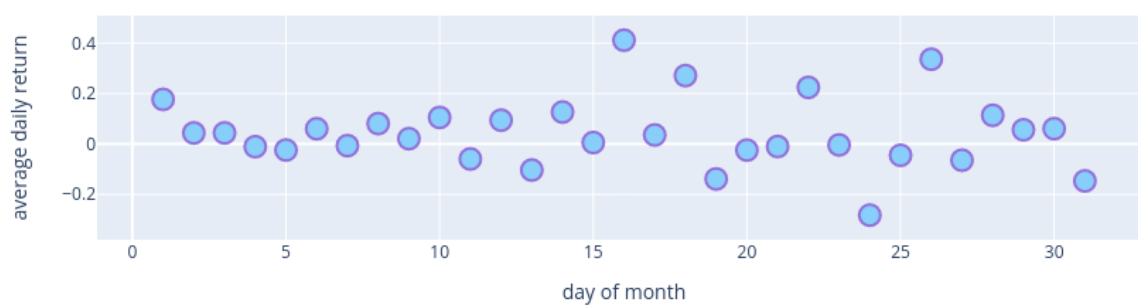
The plot below shows the average daily returns for each month. We can infer that based on our dataset, most months have positive daily returns (only Jan, Aug, and Dec have a negative average daily return, "Calendar Effect").

Average daily returns by month



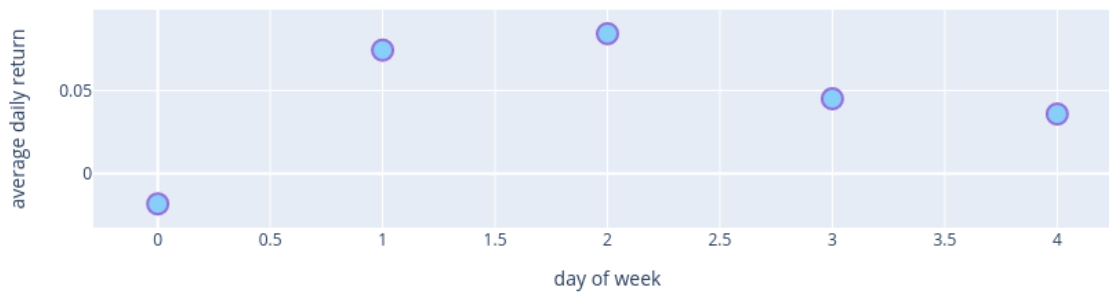
The plot below shows the average daily returns for each day of the month. Most days have positive daily returns.

Average daily returns by day of the month



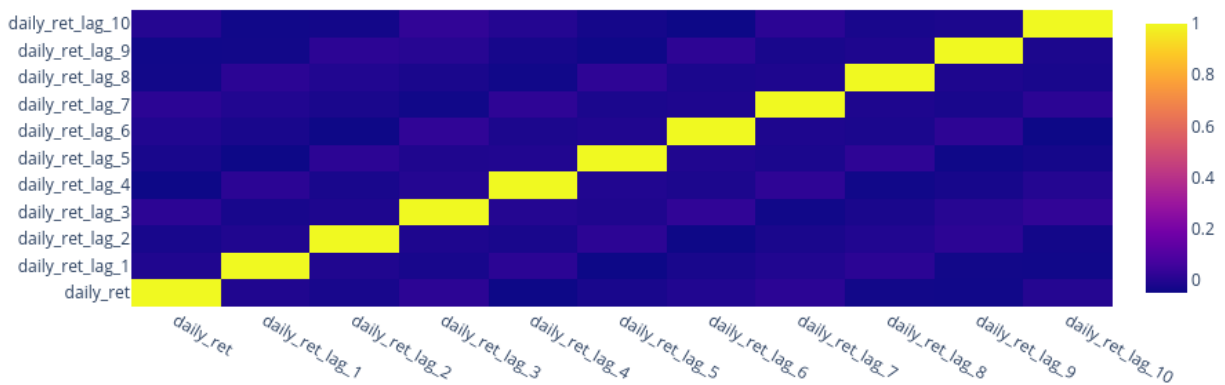
The plot below shows the average daily returns for each day of the week. Mondays on average have a negative daily return. (*Weekend Effect*)

Average daily returns by day of the week



The heatmap below shows the correlation of previous days' daily returns with the current day's. Interestingly, there is very little correlation between the day-to-day daily returns.

Correlation heatmap for the lag features



Feature Engineering

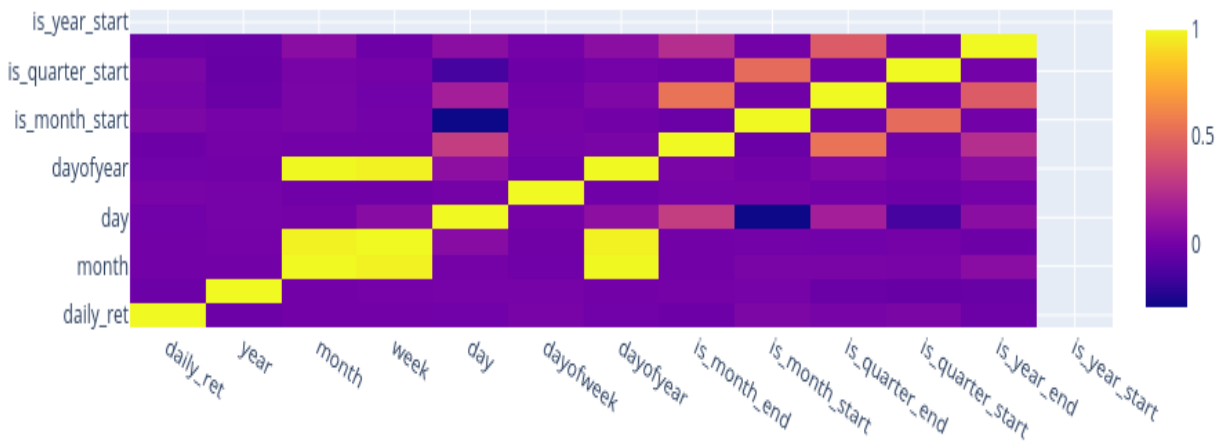
Based on simple observation and focusing on learning the essence of the algorithm first completely we constructed very easy features to work on, FASTAI module helped us to construct these features without much hassle.

Features are,

- daily returns of the last N=10 days
- Year
- Month
- Week
- Dayofmonth

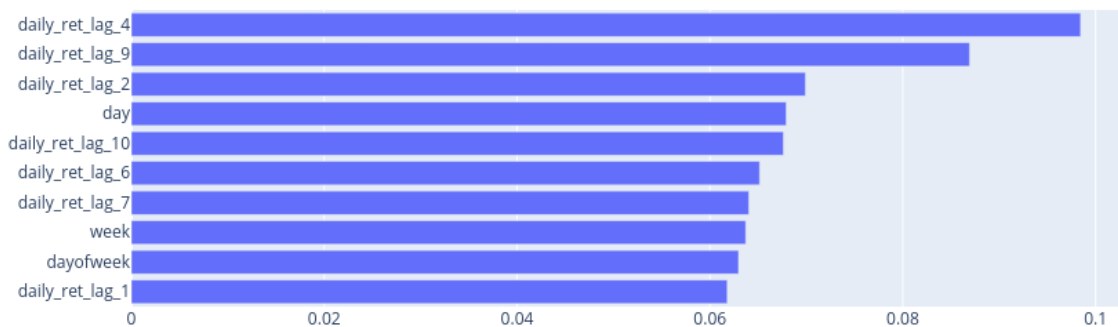
- Dayofweek
- Dayofyear
- Is_month_end
- Is_month_start
- Is_quarter_end
- Is_quarter_start
- Is_year_end
- Is_year_start (**removed** as we also found that the feature is_year_start has all NaNs. This is because the first day of the year is never a trading day, and so we remove this feature from the model.)

Correlation heatmap for date features



Below is a bar chart showing the importance scores of the top 10 most important features.

top 10 most important features



Training, Validation, and Test

To perform a forecast, we need training and validation data. We will use 3 years of data as the train set, which corresponds to 756 days since there are about 252 trading days in a year ($252 \times 3 = 756$). We will use the next 1 year of data to perform validation, which corresponds to 252 days. In other words, for each forecast we make, we need $756 + 252 = 1,008$ days of data for model training and validation. The model will be trained using the train set, and model hyperparameters will be tuned using the validation set.

It is very important in time series prediction that the train, validation, test splits have to be in chronological order to prevent from '**information leak**' in the model, which is defined as the scenario where the model is trained on data that provides information about the test set.

We used **XGBoost** to perform forecasts on several days in our test set, namely:

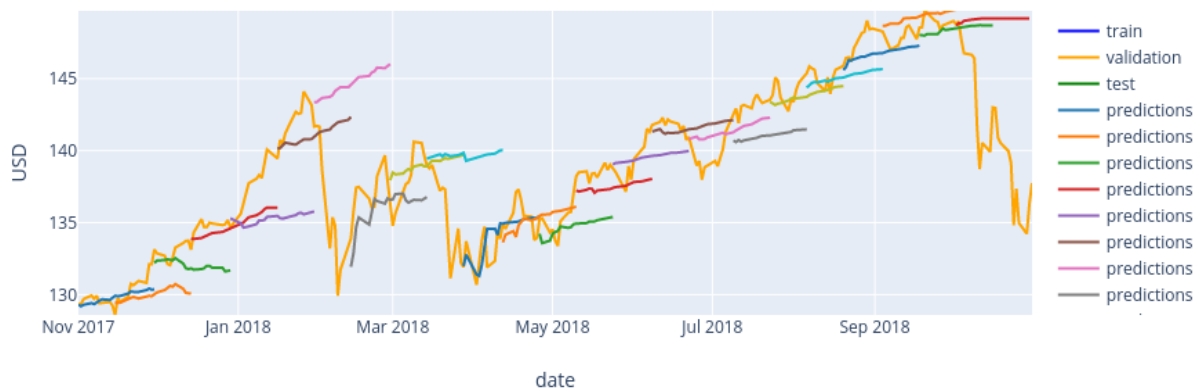
2017-01-03, 2017-03-06, 2017-05-04, 2017-07-05, 2017-09-01, 2017-11-01, 2018-01-03, 2018-03-06, 2018-05-04, 2018-07-05, 2018-09-04, 2018-11-01

For each of the 12 forecasts above, we will use a forecast horizon of 21 days. We will use the 1008 days immediately before the forecast date as the training and validation set, with a 756:252 split as mentioned above.

Hyperparameter Tuning

We perform hyperparameter tuning on the validation set. For **XGBoost**, several hyperparameters can be tuned including `n_estimators`, `max_depth`, `learning_rate`, `min_child_weight`, `subsample`, `gamma`, `colsample_bytree`, and `colsample_bylevel`.

Predictions on the validation set without hyperparameter tuning



Predictions on the validation set with hyperparameter tuning



param	before_tuning	after_tuning
n_estimators	100.000	59.000
max_depth	3.000	5.000
learning_rate	0.100	0.300
min_child_weight	1.000	9.000
subsample	1.000	1.000
colsample_bytree	1.000	1.000
colsample_bylevel	1.000	1.000
gamma	0.000	0.900
val_rmse	2.961	2.942
val_mape	1.804	1.799
val_mae	2.488	2.476

On tuning the hyperparameters the RMSE, MAPE, MAE drops and these values differ a lot from the default values too.

Applying the Model

We're done with EDA, Feature Engineering and Hyper Parameter Tuning. And we're now going to forecast using the test set. In this case, we have a forecast horizon of 21 days which means we need to generate 21 predictions for each forecast. We cannot generate all 21 predictions at one go, because after generating the prediction for day T, we need feedback on this prediction into our model to generate the prediction for day T+1, and so on until we have all 21 predictions. This is known as **recursive forecasting**.

Findings

Below shows the RMSE, MAPE, and MAE of each forecast, along with their corresponding (selected) optimum hyperparameters tuned using their respective validation sets.

date	RMSE	MAPE(%)	MAE	n_estimators	max_depth	learning_rate	min_child_weight
2017-01-03	4.069181	3.475259	3.927979	35.0	2.0	0.1	8.0
2017-03-06	2.292502	1.789292	2.092942	17.0	6.0	0.1	5.0
2017-05-04	0.785042	0.441047	0.524416	23.0	2.0	0.1	11.0
2017-07-05	0.920547	0.674549	0.829225	25.0	2.0	0.1	9.0
2017-09-01	1.045488	0.755843	0.946815	25.0	2.0	0.1	8.0
2017-11-01	0.914314	0.553748	0.719505	17.0	2.0	0.1	7.0
2018-01-03	3.634313	2.352507	3.325212	17.0	2.0	0.1	5.0
2018-03-06	4.182646	2.478044	3.318856	19.0	8.0	0.1	5.0
2018-05-04	3.296143	2.300156	3.181564	21.0	4.0	0.1	5.0
2018-07-05	3.024015	2.036865	2.921950	19.0	8.0	0.1	5.0
2018-09-04	1.535855	0.963535	1.428967	19.0	5.0	0.1	5.0
2018-11-01	2.755816	1.551702	2.135047	59.0	5.0	0.3	9.0

RMSE, MAPE, and MAE of each forecast made with XGBoost

To visualize the forecasts is to plot each forecast with its actual value. If we have perfect accuracy, each forecast should lie on the diagonal **y=x** line.

Comparing the forecasts using XGBoost with their actual values

