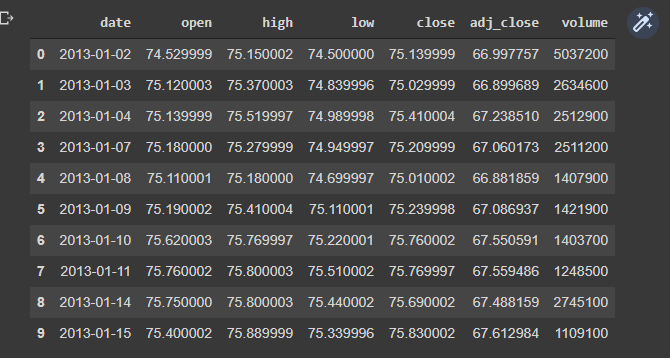
## XGBOOST Implementation

By Lubdhak Mondal and [Kapil Chandak](mailto:kapil.chandak@students.iiserpune.ac.in)

## 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.

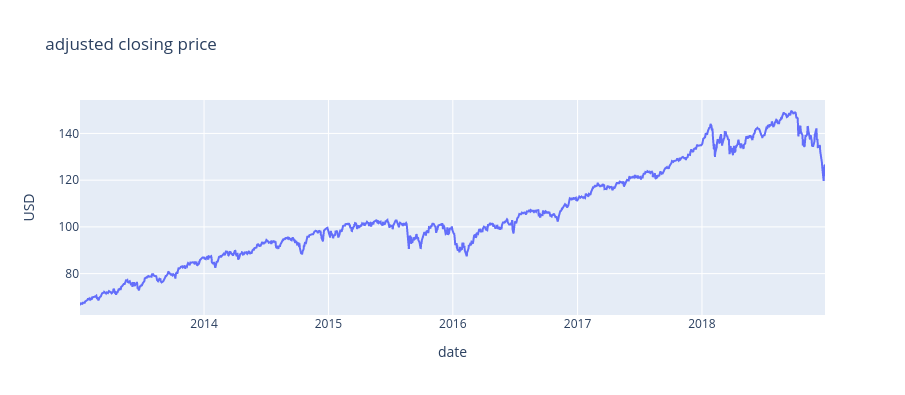


Raw Data

## Steps Involved

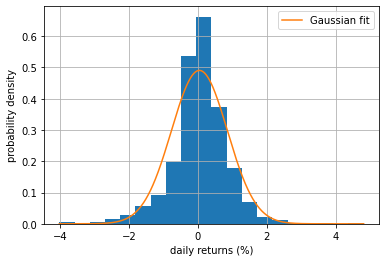
* Problem Statement
* Exploratory Data Analysis
* Feature Engineering
* Training, Validation, Test split
* Hyperparameter Tuning
* Applying the Model
* Findings

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.



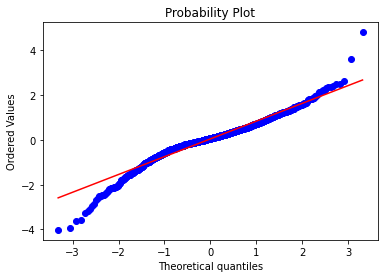
### Daily Returns **r(t)=(p(t)/p(t-1) -1)×100**

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.

### 

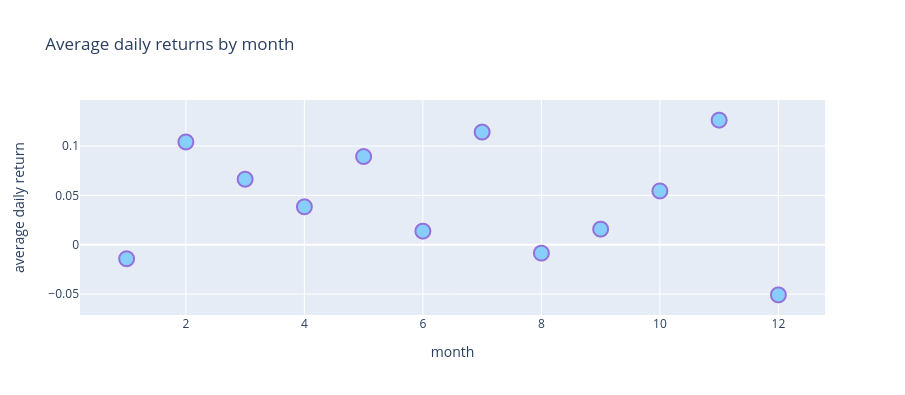
### Evaluation of the effectiveness of implemented methods

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

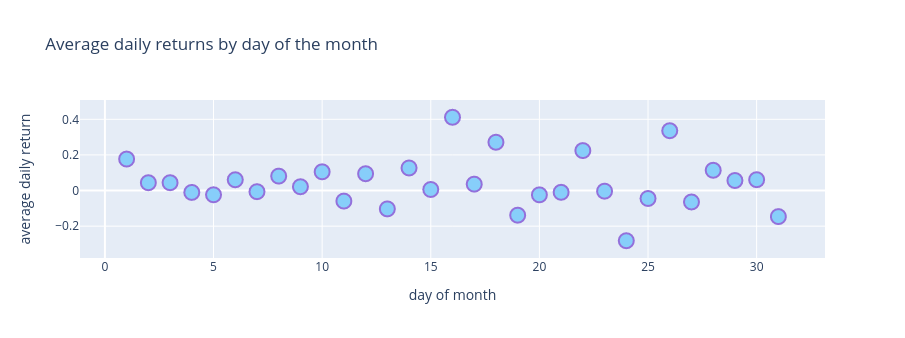
We first predict the daily returns, then convert back to price based on the daily returns formula above, and compute the metrics with respect to the actual prices. For all metrics, the lower the value, the better the prediction.

## Exploratory Data Analysis (EDA)

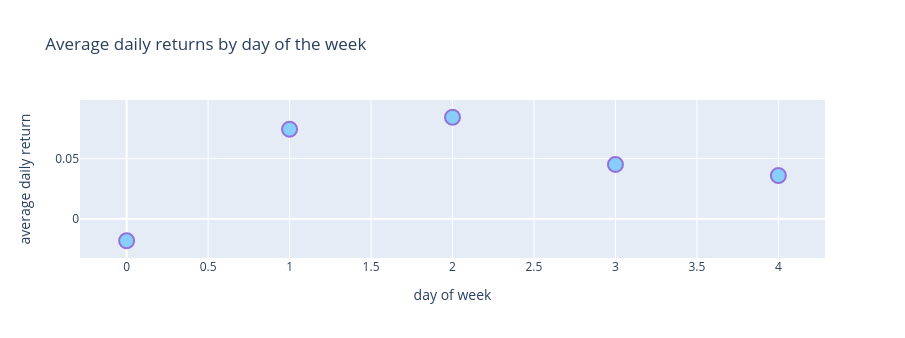
The plot below shows the average daily returns for each month. We can infer that based on our dataset, most months have positive daily returns (only Jan, Aug, and Dec have a negative average daily return, “Calendar Effect” ).



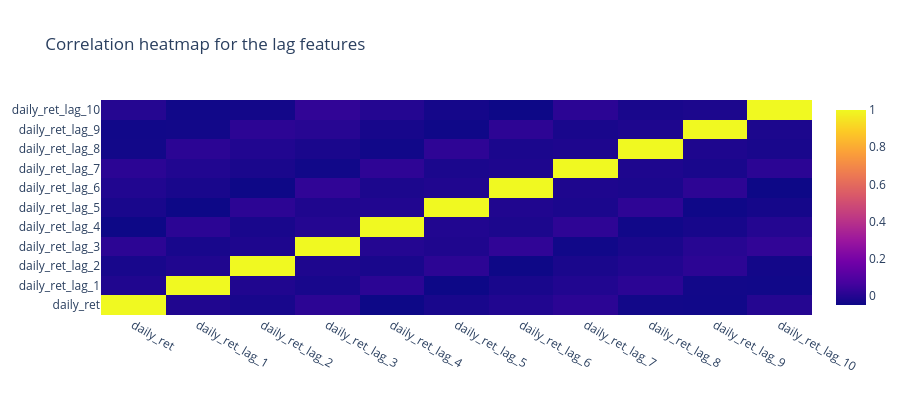
The plot below shows the average daily returns for each day of the month. Most days have positive daily returns.



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



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.

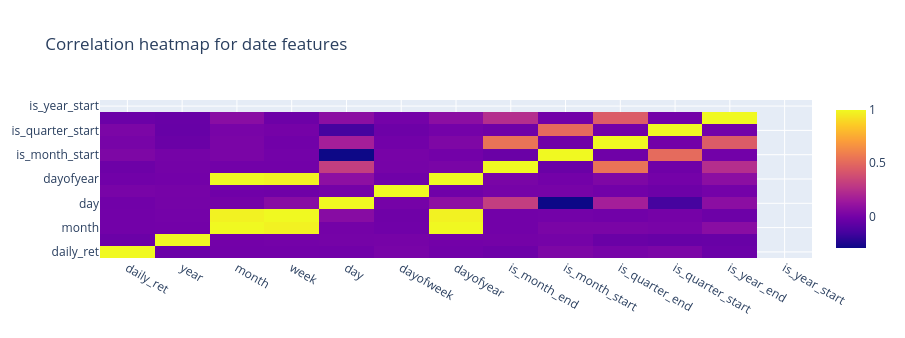


## Feature Engineering

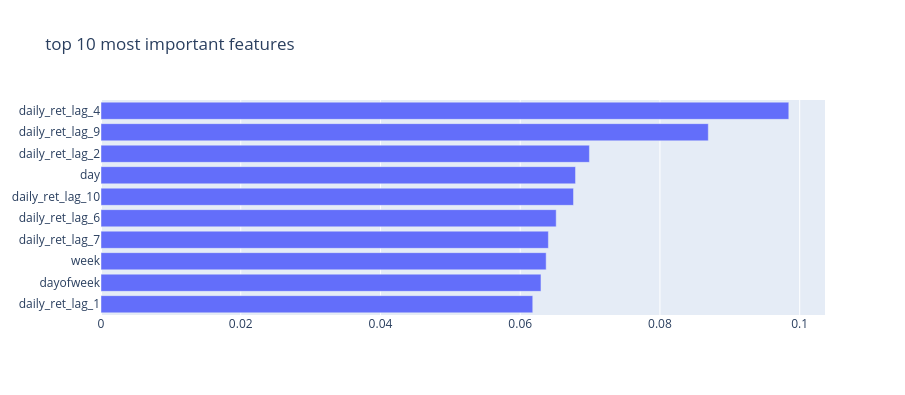
Based on simple observation and focusing on learning the essence of the algorithm first completely we constructed very easy features to work on, FASTAI module helped us to construct these features without much hassle.

**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.)



Below is a bar chart showing the importance scores of the 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\*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.

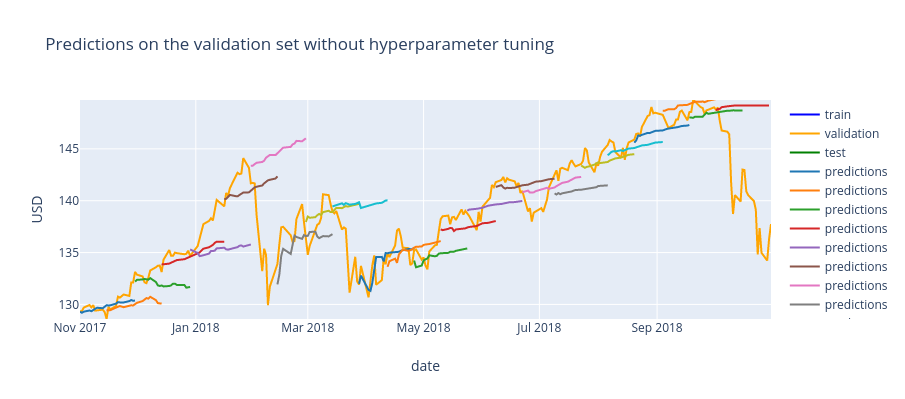
We used **XGBoost** to perform forecasts on several days in our test set, namely:

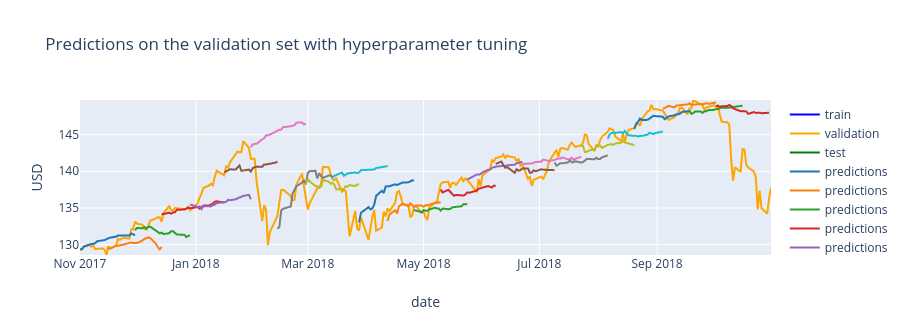
2017–01–03, 2017–03–06, 2017–05–04, 2017–07–05, 2017–09–01, 2017–11–01, 2018–01–03, 2018–03–06, 2018–05–04, 2018–07–05, 2018–09–04, 2018–11–01

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.







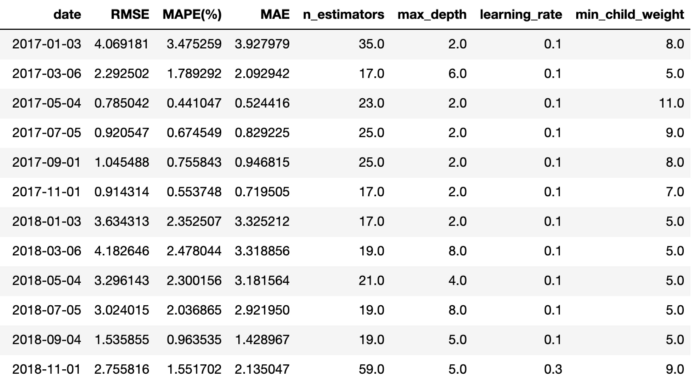
On tuning the hyperparameters the RMSE, MAPE, MAE drops and these values differ a lot from the default values too.

## 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.



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.

