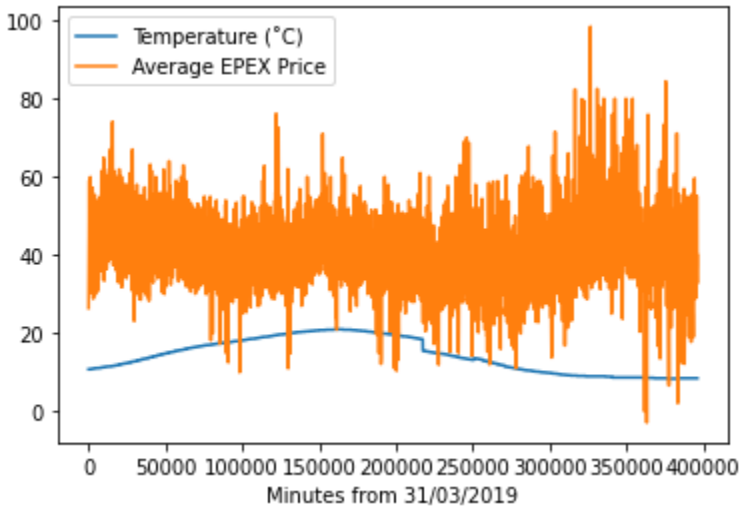
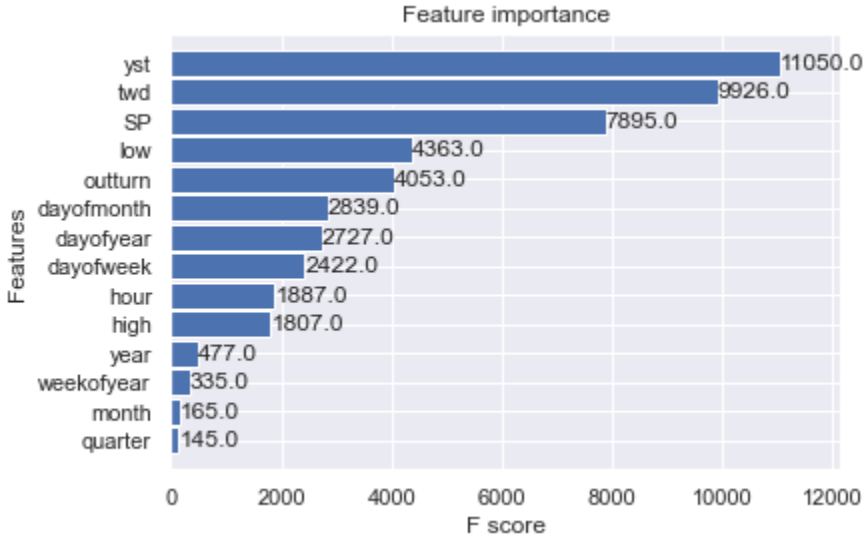
Report
Code link: https://github.com/hanzhibao/AIHack/blob/main/AIHACK_Copy1.ipynb

Exploratory Analysis

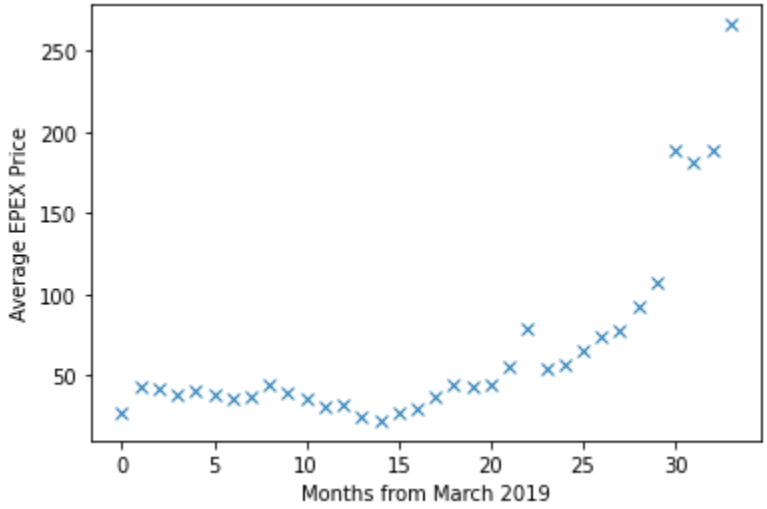
- We first plotted EPEX prices over time
- We then compared EPEX prices with spot prices
- Attempted to predict prices with vector autoregression
- Temperature data was collected and added as features (low, high and outturn).
    - This was done as colder temperatures may lead to higher energy demand (hence prices) for heating.



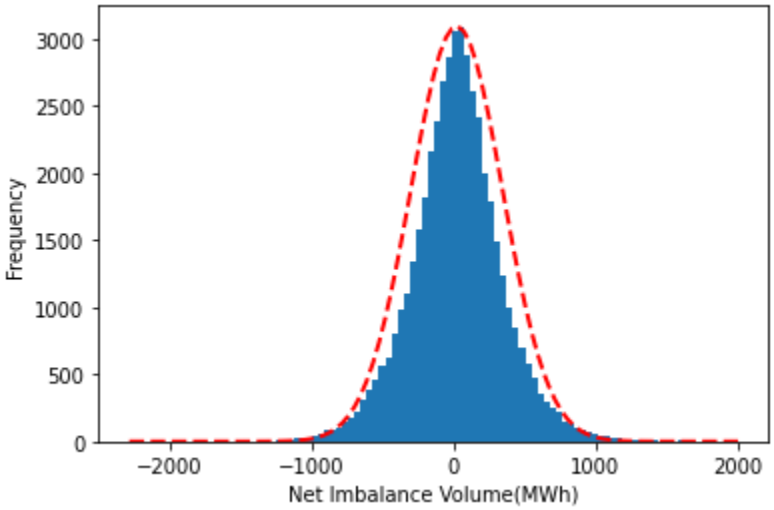
- There doesn't seem to be a clear correlation between EPEX price and Temperature. However, feature importances from our selected model suggests that the low temperature was significant in influencing pricing data.
- This may be an interesting area to explore.



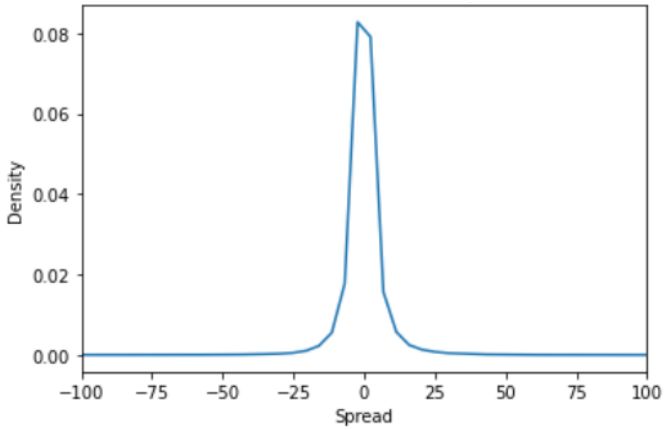
- Ordered features that influence the EPEX price by importance, and found that yesterday, two days ago and SP(30 min increments of day) were the most important. On the other hand, the model found it difficult to observe correlation season/month and EPEX price which was expected due to the large increase in EPEX price over time.



- Very strong increase in price with month is seen, meaning that seasonal trends are drowned, on top of the fact that we get 3 rounds of 1 month at maximum meaning it is much more difficult to get useful data.



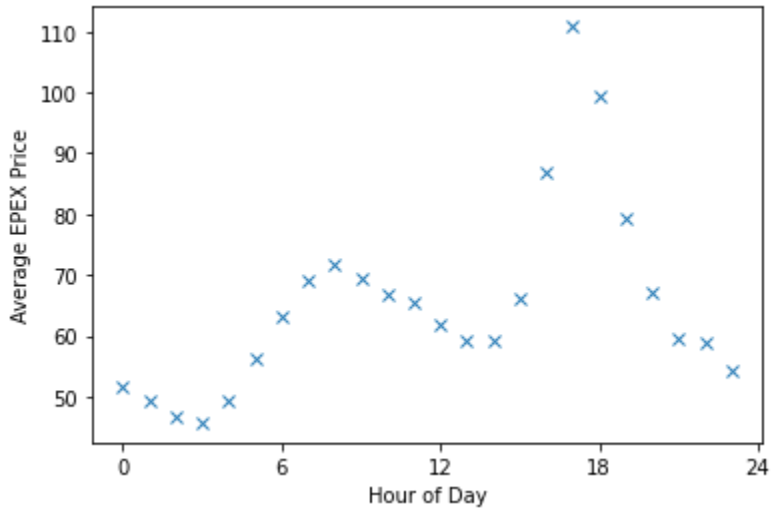
- Net imbalance was plotted as a histogram as well. May suggest net imbalance comes from random factors which may be difficult to predict.



- The distribution of price spreads (calculated as the difference between hourly EPEX prices). The range of spreads is very large at around 2700.
- However, most spreads are focused around 0 (no change in prices), with a mean of around 0.06. This implies the general uptrend of EPEX prices as observed in real data while having highly variable prices over time.

ML Method

- The current method used is the XGBoost (extreme gradient boost)
- The Inputs do not contain current price movement, and aims to use data available from the past day and predict future market movements
    - Temperature (assuming forecasts are quite accurate), time of day, prices at same time of previous days.
    - We chose to include a temperature dataset as energy consumption is largely affected by temperature (due to heating and air-conditioning, and possible living patterns of the general public)



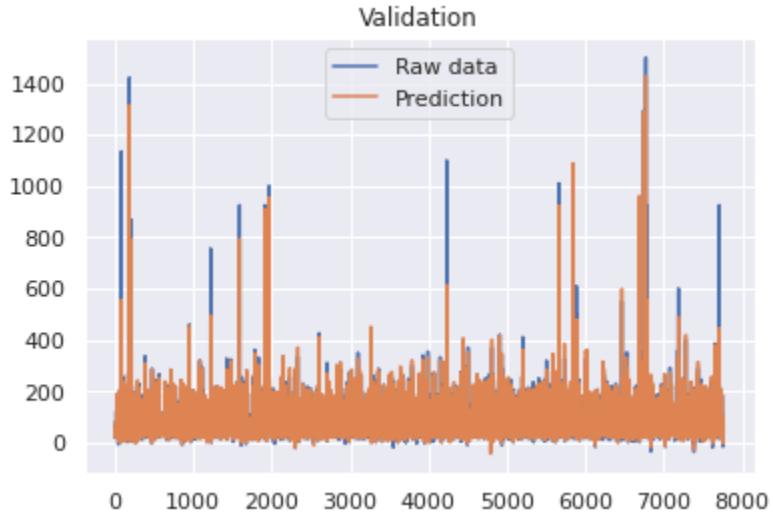
    - Clear correlation between hour of day and EPEX price, thus thought that information was important. We also thought that it was important to include information of prices from a few days back to keep up with the trends.

Our models

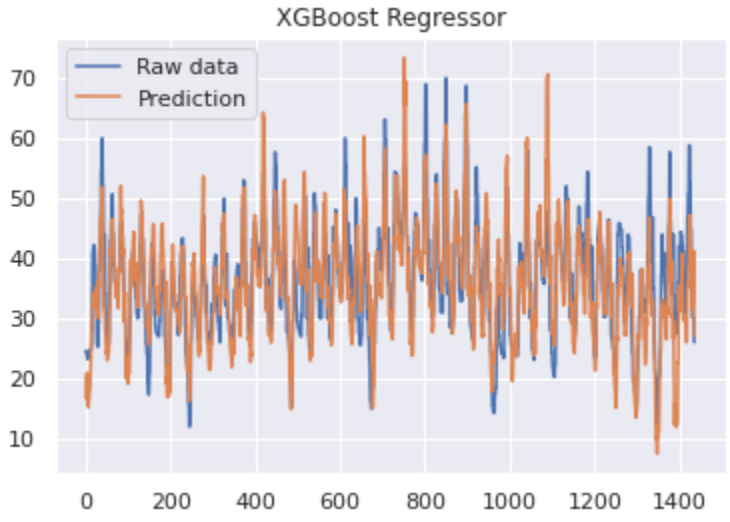
XGBoost:
parameters: objective='reg:squarederror', alpha= 0.5, learning_rate= 0.1, n_estimators=1000, min_child_weight= 0.5, max_depth= 5

Sklearn.model_selection and tune_sklearn packages were used for tuning the parameter.

Validation dataset: 17% of the training data (randomized)
X-axis: datapoints, y-axis: 'apx_da_hourly'



Final test on 2019 september:
X-axis: datapoints, y-axis: 'apx_da_hourly'



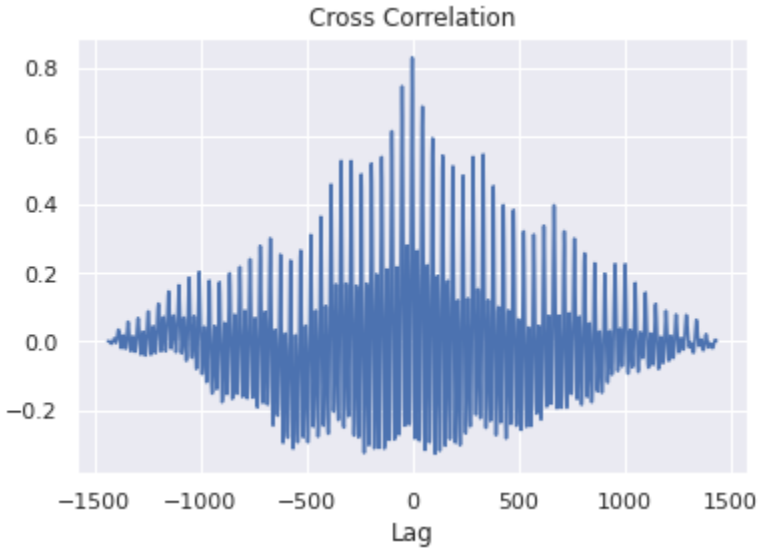
Best results:
MAE for XGBoost: 4.346188596405437
Accuracy for XGBoost: 67.35115629064346 %
MAE for Random Forest Model: 4.353819341760566

Accuracy for Random Forest Model: 67.353985340526 %

actually very close but XGBoost has many more parameters to be tuned, so it is potentially better if we can tune the parameters better

Solving the trading problem

- A basic system was used Peaks and troughs were identified using the scipy signal.find_peak function.
- Continuous peaks/troughs were identified, and bought around the troughs and sold around the peaks
- Used to determine times to buy and sell energy to charge the battery in 30 minute increments
- These times were saved and back tested against unseen data
- *Our simulation returned £1217.27 made over the month of september 2019.*
- This method only requires us to match peaks and troughs with the simulated data. Thus cross correlation is a valid metric to assess fit.



- Maximum cross-correlation near the center == good fit

Possible Improvements to the model

- Improving algorithm
    - Incorporate peak height, look at future peaks /troughs and choose the option with highest profit
- Tune the models better, especially XGBoost

- Construct a stack model by combining several finely tuned models, then train together with a part of the training data (meta training data) and predict a common answer in the end.