ORIE 5741 Final Project Report

Oil Price Prediction

Harshavardhan Bapat (hsb57); Joel Dsouza (jnd74); Yash Ganatra (ybg3)

Abstract

This project employs machine learning techniques using financial and economic data to attempt to predict crude oil prices. Prediction of up/down prices movements is attempted using both linear classifiers (SVM, Logistic Classification) and non-linear classifiers (Decision Trees). Price prediction is attempted using Quadratic and Huber loss models with regularization and non-linear ensemble methods such as Random Forest, Boosting and Stacking. Finally, a modified ensemble method is proposed and executed, which delivers promising results. At the end, model limitations and possible improvements are discussed.

1 Data Analysis

1.1 Background

The purpose of this project is to train a machine learning model to predict crude oil prices based on various financial and economic data. Crude oil is one of the most important commodities for the global economy, and while humankind will eventually shift to more carbon-neutral sources of energy, currently the world is heavily reliant on crude oil. Since crude oil is used so extensively, and accounts for billions of dollars in input costs for industries, a price prediction model could be extremely useful for companies to lower and hedge costs. Further, commodities trading strategies can be derived from such a model to gain profits.

1.2 Data Description

Like most other commodities, oil prices are a function of supply and demand. Supply of global crude oil is largely in the hands of a few top nation producers with access to the largest oil fields in the world – the OPEC coalition, United States of America, Russia, Canada, and China. Meanwhile, demand for crude oil is tougher to estimate accurately due to its ubiquitous use, but due to the importance of crude oil as an energy source driving global economies, we proxy global demand for crude oil using indices reflecting the economic and financial health of major economies. An important factor to consider as an intermediary between supply and demand for oil is the cost to store and ship oil barrels, and we proxy this by using a freight cost index.

The data that has been considered can be classified into three categories:

1. Supply Side Factors
2. Demand Side Factors
3. Other Economic and Financial Data

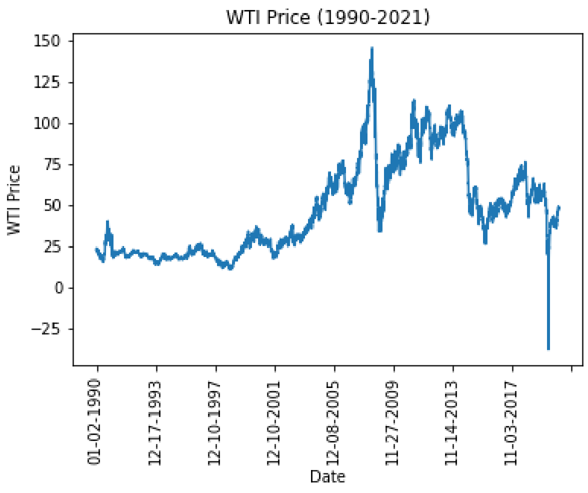


Figure 1: WTI Crude Futures Price

The dependent variable considered is the WTI Crude Futures Price, which is the daily closing price for futures of the West Texas Intermediate Crude Oil delivery contract with nearest maturity. For supply side factors, we consider the daily oil production of OPEC, US, Russia, Canada, and China as reported by the DOE, and the CASS Freight Rate Index which is a proxy for shipping costs. To proxy for demand side factors, we use the YoY GDP of some of the largest countries in the world, ensuring we capture distribution across all regions, while we also consider the Industrial production output of US, Russia, and the Euro Area, which may be a more concise proxy for crude oil demand, since industrial businesses are the major sources of demand (as opposed to industries such as software etc.). Commodities such as crude oil are also closely related to inflation, but the relationship may be a two-way causation. In addition to the above data, we also consider 5 major global stock indices as proxies for 'economic health' of the world, namely S&P500 (US), Dow Jones Industrial Average (US), DAX (Germany), Hang Seng (Hong Kong) and Nikkei 225 (Japan). We also consider US 10- year government bond yields as another indicator of economic health, and the USD/EUR exchange rate as a proxy for relative strength of USD and EUR, two of the most prominently used global currencies. All the above data is sourced from Bloomberg LP.

Historical data for the considered features is available for different time horizons. Most of the features' data is available from 1992-2020, but some features' data is only available from 2002. We account for this in our regime-based modelling (discussed later), but we limit our overall horizon to 2002-2020, to ensure we include important features such as OPEC oil production.

The below table summarizes some of the features considered and their corresponding frequency:

Data	Frequency
WTI Crude Oil Futures	Daily
OPEC Oil Production per day	Monthly
CASS Freight Rates	Monthly
US Oil Production per day	Monthly
US GDP	Quarterly
Canada Oil Prod per day	Monthly
China GDP	Quarterly
Russia Oil Production per day	Monthly
Russia GDP	Quarterly
US Inflation YoY	Monthly
Japan GDP	Quarterly
Euro Area Inflation YoY	Monthly
Germany GDP	Quarterly
Natural Gas Futures	Daily
USD/EUR Exchange Rate	Daily
Euro Area Ind Prod Index	Monthly
US Ind Production Index	Monthly
Russia Ind Production Index	Monthly

Also shown is the correlation matrix plot of the initial feature set (Figure 2):

1.3 Data Cleaning and Processing

As all the data considered is time-series, the issue of look-ahead bias must be addressed during data processing before proceeding, to ensure this bias is not incorporated into the analysis.

The features for each target datapoint must be lagged values, available prior to the attempted prediction date. Furthermore, while data on financial indices like stock indices and exchange rates is available after the market closes daily, economic data is released at either a monthly or quarterly frequency, and the release of such data is usually lagged by some time as well. Therefore, to ensure we account for only data that is available at the time of prediction, we must lag our data appropriately. We lag each datapoint by the appropriate release frequency, for e.g., GDP data for Q1 is labeled for the quarter ending in March, but this data is typically released sometime in Q2, therefore it is important to only consider this data after the end of Q2, to avoid any look-ahead bias. Therefore, for each prediction target, the considered economic features lag by one month/ quarter.

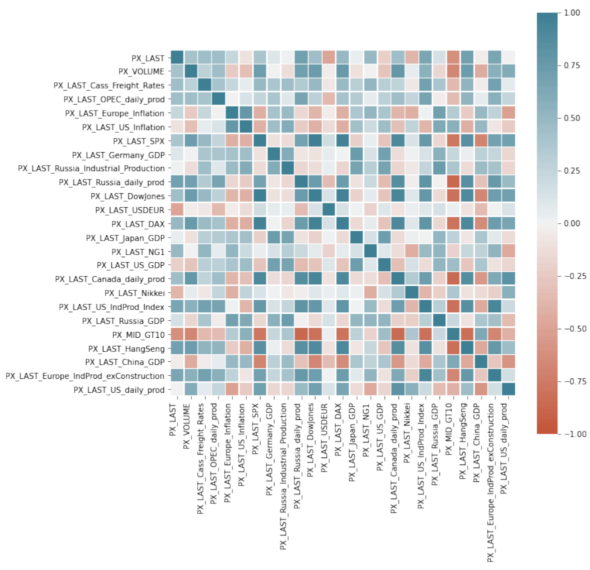


Figure 2: Correlation Plot

1.4 Feature Engineering

1.4.1 SMA of Economic Data

Economic data (representing demand factors for oil) change on a monthly/quarterly basis, while the frequency of our target variable is daily. To proxy for changing economic conditions between the release dates of economic data, we decide to consider as features the 22-day and 65-day smooth moving averages of monthly and quarterly data respectively. This is done after the aforementioned lagging, to ensure consistency and avoid look-ahead bias. We believe considering this for features rather than spot values would better capture the changes in economic conditions that would affect oil prices.

1.4.2 Target Variables

Initially, we attempt to predict daily up/down movements in oil prices, therefore modeling it as a classification problem. Here, we take the 'sign' of the change in the 22-day smooth moving average of oil prices, categorizing a down movement as '-1', and an up or no movement as '+1'. We decide to take a 22-day moving average to remove some noise from the daily data, which holds less valuable information for this objective.

For prediction of oil prices using regression and ensemble techniques, we consider the daily oil price as our target variable, since this is a more useful problem to solve.

1.4.3 Differencing

The variation of oil prices is often due to changes in economic and financial conditions rather than the absolute values. Therefore, we decide to consider the first difference of each feature as the feature value rather than the absolute value, to capture the impact of changes in the features on the target variable rather than the impact of the absolute value. Since oil prices are fairly continuous, previous day oil prices are an important starting point for any estimation, therefore is also considered as a feature. Within the classification problem, the differencing leads to some information regarding the momentum of the respective horizon as well.

For data available daily, we consider the 1-day, 2-week and 1-month change in value. For data available monthly, we consider the 2-week and 1-month change, while for quarterly data, we consider the 1-month change in value. Differencing the data also reduces correlation between features, as the below plot indicates. Some correlation is observed among financial stock indices, and within GDP data between countries, but overall, the correlation between features has reduced.

Our final feature set contains 4,718 data points and 53 features for each data point. Refer Appendix for correlation matrix plot of the full set of features.

2 Model Selection

The purpose of time series forecasting is to produce precise future forecasts. In the case of time-series data, the rapid and powerful methods we rely on in machine learning, such as train-test splits and k-fold cross-validation, do not work. This is due to the fact that they overlook the problem's time aspects. Therefore, we decided to use TimeSeriesSplits to choose different training and validation datasets that are contiguous in order to tune hyperparameters and use them for training the entire train dataset. We use this model and test it on the test dataset to get an unbiased evaluation of the model.

Standardization of a dataset is a common requirement for many machine learning estimators. For all the models mentioned below, we have scaled the data using Robust Scalar which is robust to outliers. Robust Scaler removes the median and scales the data according to the quantile range.

2.1 Classification

2.1.1 Support Vector Machine(SVM)

The SVM algorithm uses hinge loss for training classifiers. The benefit of such a loss function is that for correctly classified points, it will have small or no loss while incorrectly classified instances will have high loss. One key characteristic of the SVM and the Hinge loss is that the boundary separates negative(down movement) and positive(up movement) instances with a (d-1) dimensional hyperplane. Support vectors are data points that are closer to the hyperplane and have an influence on the hyperplane's position and orientation. We maximize the classifier's margin by using these support vectors. These points assist us in constructing our SVM.

2.1.2 Logistic Regression

Logistic regression is a classification method for determining the likelihood of an event's success or failure. It aids in the classification of data into discrete classes by examining the link between a set of labeled data. It is based on the sigmoid function, with probability as the output and input ranging from -infinity to infinity.

2.1.3 Decision Tree Classifier

By learning simple choice rules derived from prior data, Decision Tree builds a training model that can be used to forecast the class (training data). The values of the root attribute and the record's attribute are compared. We follow the branch that corresponds to that value and jump to the next node based on the comparison. Decision trees need less effort for data preparation during pre-processing than other methods.

2.2 Regression

2.2.1 Lasso Regression

We found that some of the features from our dataset might be heavily correlated which can affect the model's interpretability adversely. To avoid the problem of multicollinearity, we use regularization to produce a unique solution. We use Lasso regression which uses l_1 regularization. The idea is to investigate if our model can perform better with regularization, and then to see if eliminating some variables will make it easier for us to comprehend the variables.

2.2.2 Huber Regression

After regularization, we try to fit a linear model to this data set using a different loss function. There are outliers in the oil prices time-series data during a crisis, imbalance in the demand and supply, etc. We don't want to overly distort our model to account for those outliers. Our idea is that when the errors are too large, we should punish less, and when the errors are within a fair range, we should punish the same as quadratic loss. The Huber Loss function combines the l_1 and quadratic loss functions where small errors are punished with Quadratic Loss and while excessive errors are punished with l_1 Loss (the robust part).

2.2.3 Random Forest

We also tried fitting non-linear models to our dataset. The intuition behind employing a non-linear model was to capture non-linear relations between features and the target. One of the known non-linear models is the decision tree. Random Forests build uncorrelated decision trees by performing feature selection implicitly. It accomplishes this by constructing each decision tree using a random collection of features. This makes it an excellent model for dealing with data with a large number of features. Also, Random Forests are not influenced by outliers to a fair degree by binning the variables. Random Forests are known for their high accuracy and ability to manage the bias-variance trade-off. The variance is averaged as well because the model's premise is to average the results over the various decision trees it creates.

2.2.4 Boosting

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added until the training set is predicted perfectly or a maximum number of models are added. Boosting technique helps when we are dealing with bias or underfitting in the data set. Here, we use XGBoost that builds trees using Gradient boosting, a technique that involves creating new models that forecast the residuals or errors of previous models, which are then combined to form the final prediction.

2.2.5 Stacking

Stacking or Stacked Generalization is another ensemble machine-learning algorithm that uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms. The advantage of stacking is that it can combine the capabilities of a number of high-performing models to create predictions that outperform any single model in the ensemble on classification or regression challenges. For our purposes, we used Lasso regression, Huber regression, Random Forest and Boosting as our base models. All of these models fit on the training data and their predictions are compiled. We use linear regression as our meta-model to learn how to best combine the predictions of the base models.

2.3 Modified Ensemble Method

We believe that time series data can have many factors leading to their evolution like seasonality, long-term trend, short-term trend, sudden fluctuations, etc. Hence it becomes very difficult for a single model to capture all these factors and forecast. To check if different models can perform differently depending on the data itself, we trained all the above models in various regimes to account for different trends, crises, economic conditions, etc. The four regimes we chose were: 1992-1999, 2000-2006, 2007-2012, and 2013-2020. We observed that all models performed differently on various regimes. This is the motivation for such a model that can predict depending on the data. We, therefore, propose a modified ensemble method to tackle this problem.

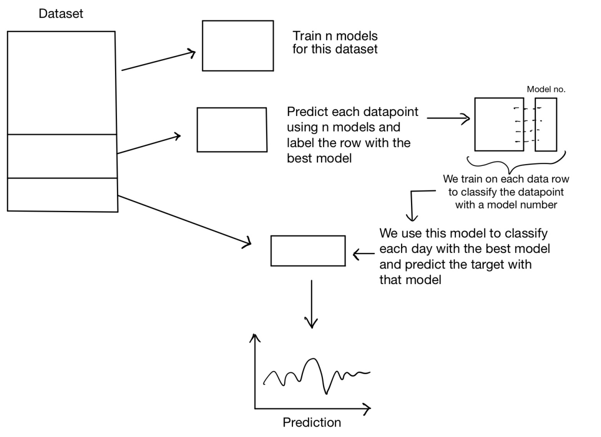

Dataset
Train n models for this dataset
Predict each datapoint using n models and label the row with the best model
Model no.
We train on each data row to classify the datapoint with a model number
We use this model to classify each day with the best model and predict the target with that model
Prediction

Figure 3: Modified Ensemble Method Working

As our aim is to check which model will perform well on the given data and then predict the price, it will require a model which classifies and regresses as well. We divide the data into two train sets and a test set. We train Lasso regressor, Huber regressor, Random Forest regressor and Boosting regressor using walk-forward validation to tune the hyperparameters. Using these models, we predict each data point in the second train set to check which model performs the best. We then label that particular data point with the model number. After this step, we will have a training dataset for the classification problem. We fit a decision tree to classify the datapoint into the model number. Now, on the test set, we use our classification model to predict which model will suit each data point and then use that particular model for its price prediction to get an unbiased evaluation of the modified ensemble method.

3 Results

Refer Appendix for full tabulated results.

3.1 Classification

We train three classification models: SVM, Logistic Regression and Decision Tree Classifier to predict the Up or Down movement of the oil prices. The accuracy of SVM on our dataset is 0.9226 and that of Decision tree is 0.9247, both higher than that Logistic Regression which is 0.3368.

Logistic loss in general diverges faster than hinge loss, which makes it sensitive to outliers Secondly, Logistic loss does not go to zero if the point is classified correctly, but hinge loss does.

SVM finds the optimal hyperplane which separates the data. Therefore, we can see that SVM performs better than Logistic regression.

On the other hand, Decision can capture linear as well as non-linear relations between features and the target which makes it robust for the classification problem. Hence, we can observe that Decision Tree accuracy is on the higher side.

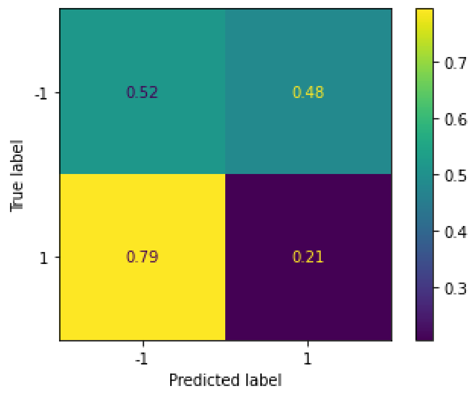


Figure 4: Logistic Regression Confusion Matrix

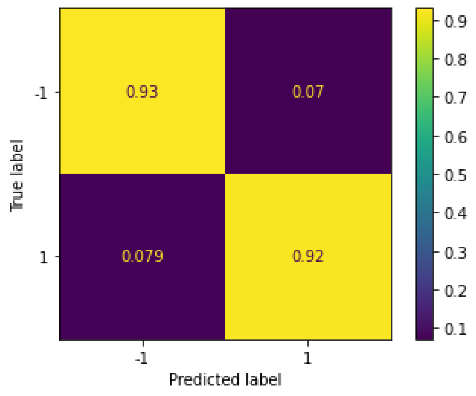


Figure 5: Decision Tree Confusion Matrix

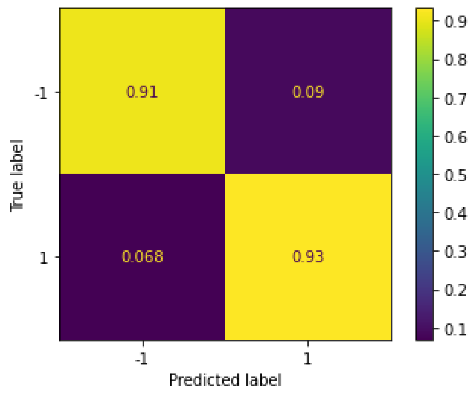


Figure 6: SVM Confusion Matrix

3.2 Regression

3.2.1 l_1 (Lasso) Regression

After data cleaning and feature preparation, we have a final feature set of size 53, and N data entries. We also use walk-forward validation to find the tuning hyper-parameter α that gives lowest RMSE for the model.

Results: Initially the model seems to perform extremely well with train and test RMSE of 1.52 and 2.71 respectively, but upon deeper inspection we realise that the model is memorising the previous day data, and does not capture any other economical moves. We can see this in the plot below where it predicts 20 when the real price was -40, and -40 the next day when the real price was 20.

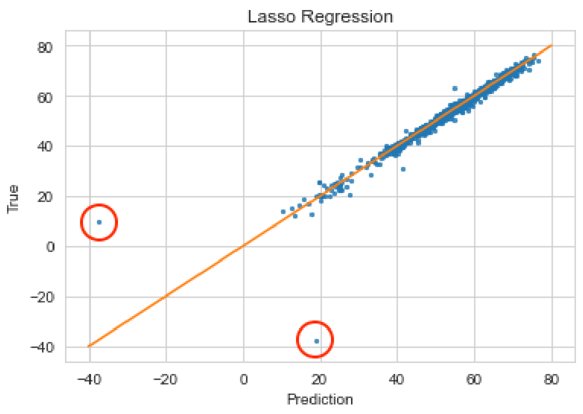


Figure 7: Lasso Regression scatter plot

3.2.2 Huber Regression

Since Lasso regression is unable to accurately capture the economic moves, we use Huber regression next since it's more robust and works better with outliers. *Results:* With train and test RMSE of 21.9 and 43.62, Huber regression performs worse than Lasso, and creates more noise than the actual data.

3.2.3 Random Forest Regressor

We now try an ensemble method to predict the price as it implicitly performs feature selection to generate uncorrelated random trees. This method is expected to perform better than Lasso and Huber regressions. *Results:* Random Forests give us a much better fitting result, with a train RMSE of 3.39 and test RMSE of 16.54. Although the test error is much higher than train, this is not due to over fitting, but due to the oil price dip to -40 during the COVID19 crisis. The model does predict a dip around that period, as it should, but not of the magnitude of -40, thus giving a high test error.

3.2.4 Extreme Gradient Boosting Regressor (XGBoost)

We now employ another ensemble method following the Random Forest Regressor. XGBoost makes the model slightly more complex and reduces the bias of the model significantly.
Results: XGBoost performs the best yet in all of our models, with a train and test RMSE of 0.29 and 2.9 respectively. While this model may also suggest over fitting to some extent, the test error is lowest of all models used.

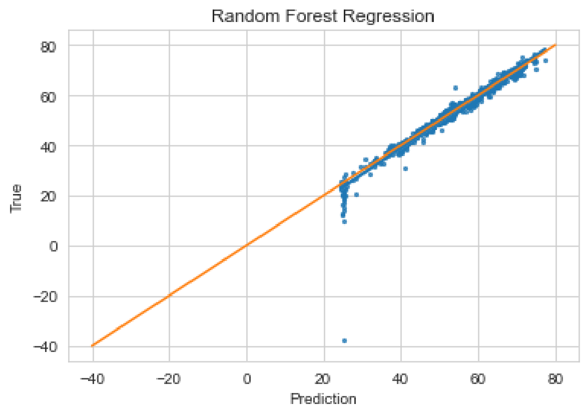


Figure 8: Random Forest Regression scatter plot

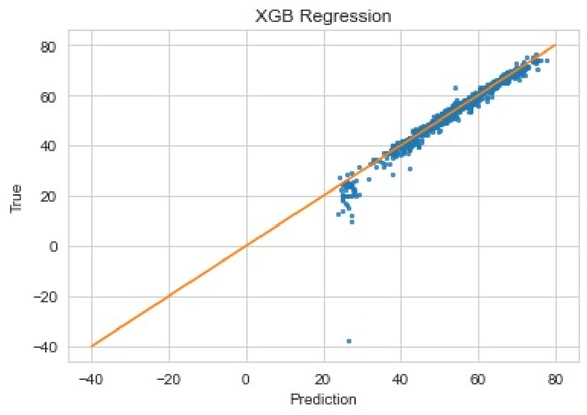


Figure 9: XGBoost Regression scatter plot

3.2.5 Stacking

Another ensemble model we employ is the Stacking of all our models to learn how to best combine the predictions of our existing models. Unfortunately, this model does not learn efficiently to solve our problem.
Results: Stacking gives a train and test RMSE of 37.04 and 17.14 respectively. We can see from this result that stacking does not overfit the data, but it's unable to achieve the required level accuracy.

3.3 Regressions with PCA

PCA (Principal Component Analysis) is a dimensionality reduction technique that reduces the feature set to a few of it's principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. We use PCA to reduce our feature set to one-third of its size, and run the different regressions we perform earlier on this new feature set, to check if that improves our results from any model.
Results: Lasso Regression is unable to memorise the data since the features have been changed to the principal components, hence it does not show an improvement. Random Forest Regressor and Stacking method are also unable to perform well on the PCA feature set. However, we see a significant improvement in the results from Huber Regression, since dimensionality reduction reduces the noise from the Huber Regression model. The train and test RMSE change to 17.08 and 36.26 respectively. While Huber Regression still doesn't provide a good fit, it significantly improves with PCA.

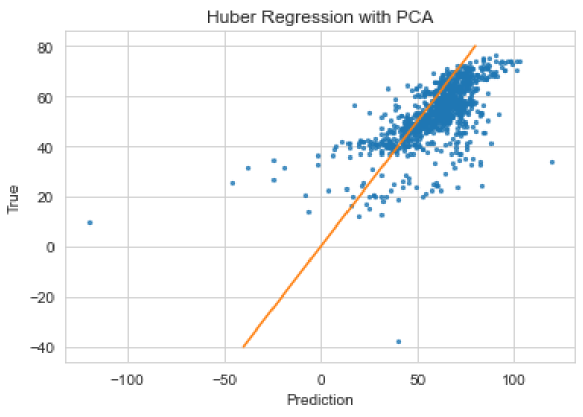


Figure 10: Huber Regression scatter plot with PCA

XGBoost still suffers from overfitting even after using PCA, but is still the best performer out of all models used.

3.4 Modified Ensemble Method

We finally see how the data performs with our proposed Modified Ensemble Method, which uses the most optimal model for each data point, and compare the results to the RMSE of the ARIMA (5,2,5) predictions as a benchmark for our model performance.
Results: With a test RMSE of 2.73 as compared to an RMSE 3.12 for the ARIMA (5,2,5) predictions, the model ensemble method performs better than all the other regression models used for prediction. This is expected, as the modified ensemble uses the most optimal model for predicting each point.

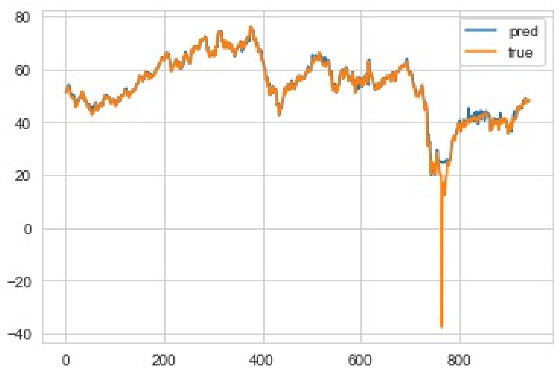


Figure 11: Modified Ensemble Pred. vs True Price for Test set data

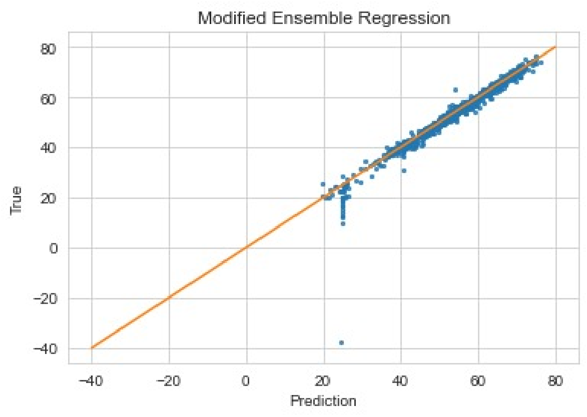


Figure 12: Modified Ensemble scatter plot

4 Future Scope

4.1 Future Improvements

Improve Ensemble Clustering

Within the modified ensemble method, the classification of data points into best fit models is performed by a decision tree. To do this, we divide our data into three parts as mentioned earlier. The validation accuracy of this tree is 60%, which gives room for improvement. We would like to explore if we can use unsupervised learning techniques, so that we can cluster our time series data into different section dynamically and train different models to check which one performs the best.

Forward Looking Models

Another approach to solving the problem of using lagged data to predict current oil prices could be to model out the future (i.e current) values of features based on previous trends and using these predictions to model oil prices. While this adds a layer of estimation to the target variable, we felt this may be an interesting approach to consider if short term and long term trends in the feature set could be captured.

Natural Language Processing (Text input)

While long term trends in oil prices are likely more heavily influenced by economic and fundamental data, short term fluctuations and daily movements are governed more by headlines and news statements, which either take time to be reflected in data, or are not reflected at all. Using natural language processing techniques, this information can be quantified and included in the feature set, and possibly trained to capture short term fluctuations better in the target variable.

Neural Networks (LSTM)

Recurrent neural networks such as Long Short-Term Memory networks are capable of learning order dependence in prediction problems, by giving importance to the information about past inputs for a variable amount of time depending on the weights and input data. Such a network may be trained to capture short & long-term trends in the feature set, which could vastly improve the prediction capabilities. This may be used to improve the previous method.

4.2 This project is not a WMD!

- Our model outputs price predictions and the test errors have the same unit as the output, hence the outcomes of our model are not hard to measure.
- The predictions from our model are not accurate enough to be able to consistently abuse the market, hence they cannot harm anyone.
- Models finance often run the risk of creating a feedback loop if a large part of the market takes positions based on a similar price based strategy. But we attempt to avoid this by considering fundamental features, thus not basing the entire prediction on solely non-intrinsic features like closing price and trading volume.

5 Conclusion

Initially, we attempted to predict the direction of daily movements in the 22-day moving average price of oil using linear and non-linear classifiers such as SVM, Logistic Classification, and Decision Trees. SVM and decision trees were able to classify data with a high level of accuracy, while logistic classification underperformed. Next, we attempted to predict oil prices initially using a quadratic loss and lasso regularized model. This model likely memorized the data in the manner of allocating too much weight to the previous date's price, thereby replicating that price as the prediction, indicating that it may be a model with minimal prediction capability. Huber regression underperformed severely with extremely noisy predictions but displayed some improvement after performing PCA on the features. The target variable was also modeled using non-linear and ensemble methods such as Random Forest and Boosting, which displayed solid results. While these models overfit the data to some extent, they generated predictions without memorizing the previous date's price as in the case of the lasso regression. A modified ensemble methodology was proposed to classify categories of data based on which model performs the best, and utilizing this to generate predictions, and the results compare favorably with an ARIMA benchmark. Nevertheless, there is plenty of room for improvement in this model before it can be utilized in any market strategy. We do not feel it is ready for production, as there are multiple aspects that can be improved as outlined above in this report. The complexity in oil price movements is difficult to be modeled using purely regression techniques and numerical data.

Appendix

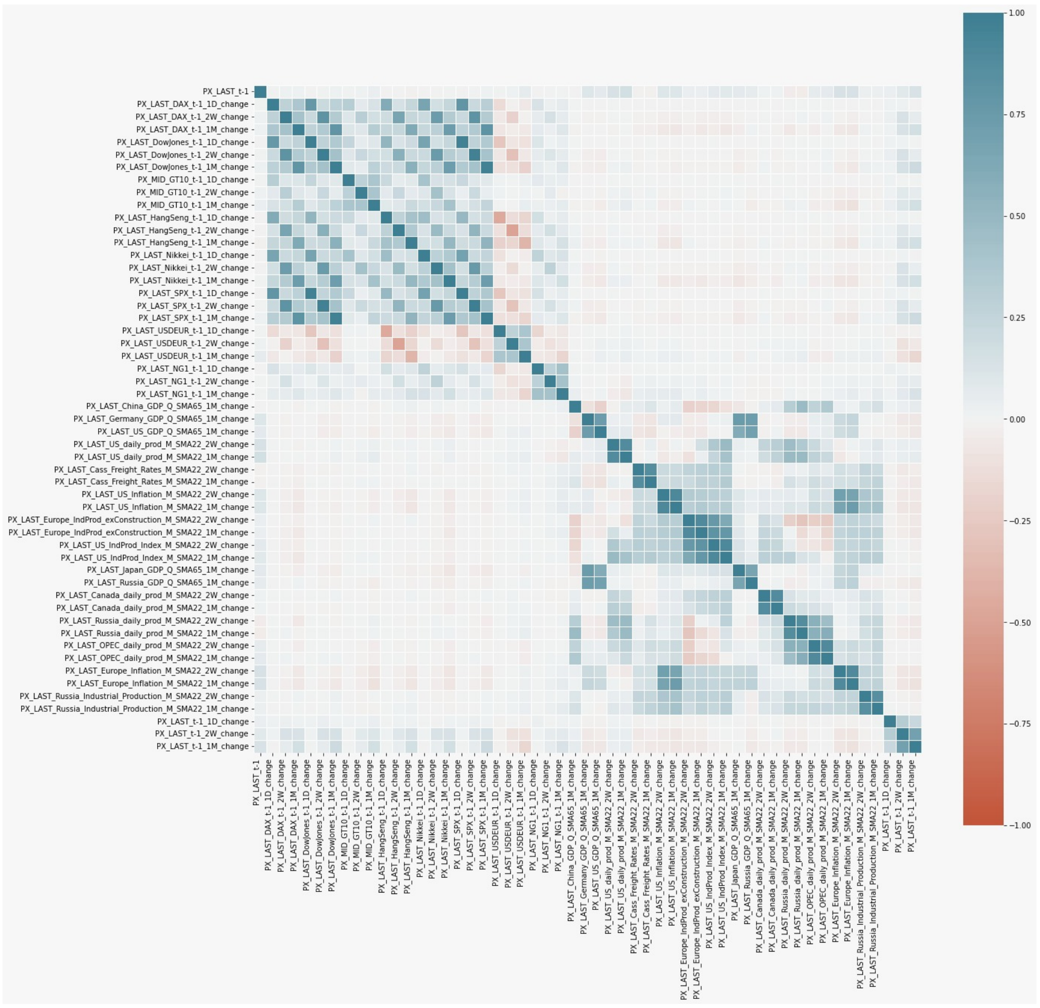
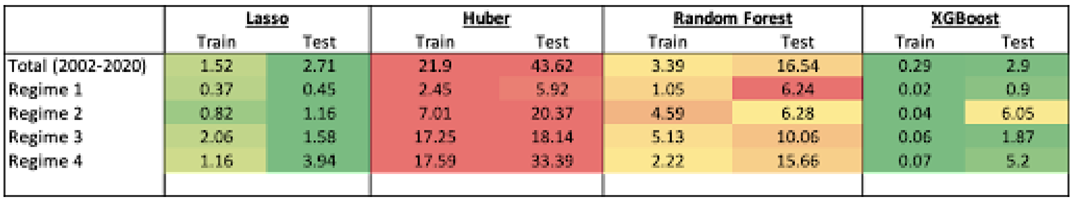
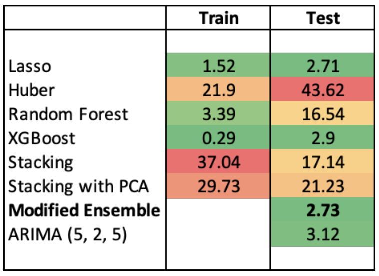


Figure 13: Full Feature Correlation Plot



	Lasso		Huber		Random Forest		XGBoost	
	Train	Test	Train	Test	Train	Test	Train	Test
Total (2002-2020)	1.52	2.71	21.9	43.62	3.39	16.54	0.29	2.9
Regime 1	0.37	0.45	2.45	5.92	1.05	6.24	0.02	0.9
Regime 2	0.82	1.16	7.01	20.37	4.59	6.28	0.04	6.05
Regime 3	2.06	1.58	17.25	18.14	5.13	10.06	0.06	1.87
Regime 4	1.16	3.94	17.59	33.39	2.22	15.66	0.07	5.2

Figure 14: Regression Results



	Train	Test
Lasso	1.52	2.71
Huber	21.9	43.62
Random Forest	3.39	16.54
XGBoost	0.29	2.9
Stacking	37.04	17.14
Stacking with PCA	29.73	21.23
Modified Ensemble		**2.73**
ARIMA (5, 2, 5)		3.12

Figure 15: Regression Results - Comparison

.

Bibliography

1. Madeleine Udell, Cornell University, ORIE 4741: Lectures Fall 2021, https://people.orie.cornell.edu/mru8/orie4741/lectures.html Last Accessed: 05-December-2021

2. Amol Mavuduru, A Practical Guide to Stacking Using Scikit-Learn, 15-Nov-2020. https://towardsdatascience.com/a-practical-guide-to-stacking-using-scikit-learn-91e8d021863d

3. Evan Lutins, Ensemble Methods in Machine Learning, 01-Aug-2017. https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

4. Jason Brownlee, Stacking Ensemble Machine Learning With Python, 10-April-2020. https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

5. Jason Brownlee, How to Develop Super Learner Ensembles in Python, 11-December-2019. https://machinelearningmastery.com/super-learner-ensemble-in-python/

6. Jason Brownlee, How to Use XGBoost for Time Series Forecasting, 05-August-2020. https://machinelearningmastery.com/xgboost-for-time-series-forecasting/

7. Jason Brownlee, How To Backtest Machine Learning Models for Time Series Forecasting, 19-December-2016. https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/