

# Oil Price Modeling with Machine Learning Methods

Guoliang Zhang

Ke Xuan

Wenwen Li

Northeastern University

360 Huntington Avenue, Boston, MA 02115

## Abstract

*Crude oil is the most important commodity in the world. Trillions of dollars of oil are being traded between countries every year and it is the underlying asset for thousands of stocks, futures, options and other financial vehicles. Machine learning has been proved to be an effective method to help predict oil price change, assisting traditional price forecasting approaches. This paper attempts to develop a concise and comprehensive machine learning architecture to predict the average daily return and volatility of crude oil for next 1 month, 2 months, 3 months and 6 months. The main method used is multi-layer neural network (NNW) and several linear models are used as benchmarks for the predictions. The performance of the proposed model is evaluated using the spot price data of WTI crude oil markets. The empirical data shows that the proposed model provides better predictions than tuned linear models with improved forecasting accuracy.*

## 1. Introduction

The oil industry is one of the most powerful branches in the world economy. More than four billion metric tons of oil is produced worldwide annually. Economists and experts are hard-pressed to predict the path of crude oil prices, which are volatile and depend on various situations. Currently the five models used most often are: oil future prices, regression based structural models, time-series analysis, Bayesian autoregressive models and dynamic stochastic general equilibrium graphs. On the other hand, the machine learning (ML) based methods have received increasing researching interests. ML approaches have shown a great advantage in dealing with analyzing assets impacted by vast factors. For crude oil, the main influence comes from oil spot and future prices, global economy, demand/supply relations, stock market performance, geopolitical status of big oil production countries, weather, as well as many other factors. Linear models and NNW models are used in this research as both are common ML methods to analyze time-series datasets.

The present paper is organized as follows: Section 2 is problem statement. Section 3 introduces the linear models and NNW models used in this research. In section 4 we report the results from the research and result analysis. Conclusion is presented in section 5.

## 2. Problem statement

The goal of the project is to develop machine learning models to predict spot oil price change in the future. More specifically, the average daily return and volatility. The hardest part of this part is finding the most important features from hundreds of thousands features that are related oil price and develop an appropriate model to learn data and perform predictions. Regression models are often constructed based on certain conditions that must be verified for the model to fit the data well, and to be able to predict accurately. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. It is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output. [1] Both models are widely used ML methods for prediction.

## 3. Technical approaches

### 3.1 Linear regression

Regression is a method of modeling a target value based on independent predictors. Three different regression models are used in the project: ordinary linear regression, Ridge regression and Lasso regression. Table1 gives a summary of each regression model.

Table 1: three kinds of linear

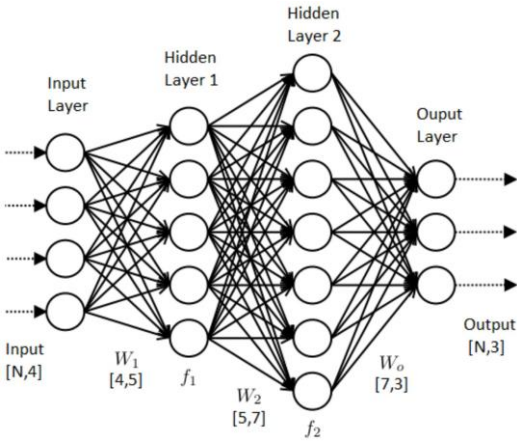
Name	Approach
Ordinary Linear Regression	$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^N l(h(x_i), y_i)$
Ridge Regression	$\min_{\theta} \ Y - X\theta\ _2^2 + \lambda \ \theta\ ^2$
Lasso Regression	$\min_{\theta} \ Y - X\theta\ _2^2 + \lambda \ \theta\ $

regression

### 3.2 Neural Network

A neural network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. [2]

Figure 1 shows us a multi-layer neural network. Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function  $f(\cdot) : R^m \rightarrow R^o$  by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. For a set of features  $X = x_1, x_2 \dots x_m$  and a target y, it can learn a non-linear function approximator for either classification or regression. Between the input and the output layer, there can be one or more non-linear layers, called hidden layers. [3]



## 4. Data Sets

### 4.1 Data Construction

Raw data include information on petroleum, energy consumption and economical indicators, provided by U.S. Energy Information Administration(EIA), downloaded from Quandl.com.

Data is stored in two json-style files: PET.txt and STEO.txt, where PET contains petroleum data and STEO is for information on energy and economy. Only time-series data is extracted to construct the raw datasets, which contain 180000+ columns(features) and 6000+ rows(samples), given only information from 1995 to 2018 is selected.

Notice that data is stored in sperate files based on the time frequencies.

PET_raw_day_all.pkl
PET_raw_week_all.pkl
PET_raw_month_all.pkl
PET_raw_year_all.pkl
STEO_raw_month_all.pkl
STEO_raw_qtr_all.pkl
STEO_raw_year_all.pkl
PET_id_dict.pkl
PET_id_dict.pkl

### 4.2 Feature Selection

Compared with the number of samples, the number of features is more than needed. Thus to select the ‘proper’ features, three aspects are considered here: domain knowledge, data quality, and correlation between features.

#### 4.2.1 Domain knowledge

Domain knowledge, represented by some keywords, is used to identify the most relevant features that may be helpful to predict oil price changes.

A python dictionary with K elemets, whose keys are keywords such as oil, reserve, import, export, and whose values are scores from 1 to 5 for the corresponding keywords. Higher scores mean higher relevance.

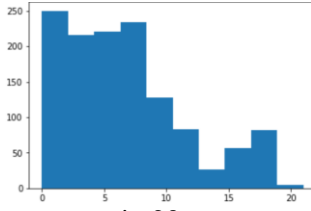
Given descriptions of each feature, the score of a feature,  $s_{feature}$ , is generated by the formula below:

$$s_{feature_j} = \sum_{i=1}^K I(key_i \text{ in } feature_j) * s_{key_i}$$

$$c = 0.5 * \max\{s_{feature_j}\}_{j=1}^{m=\# \text{ of features}}$$

For each raw dataset constructed in section 4.1, features whose scores are at least c would be remained, and the rest would be removed.

As an example, the following the histogram shows the number of features against the regarding score bin for PET\_raw\_week\_all.pkl.



At most a feature can gain 20 scores, and thus we assume features with score over 10(half of the most) relevant, and remove other features that do not meet this assumption.

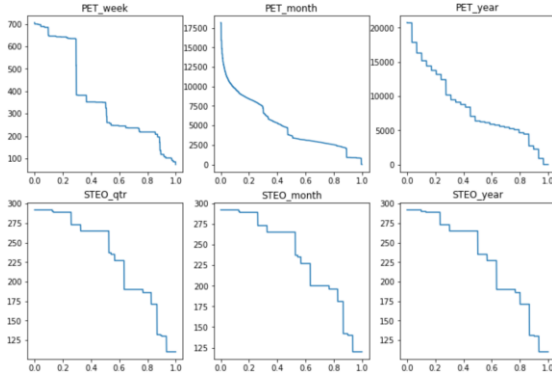
#### 4.2.2 Data Quality

Missing values appear frequently in the raw datasets, and we assume that features containing too many missing values are poor quality and ineffective to make accurate predictions.

The following formula defines the quality of a feature:

$$q(feature_j) = 1 - \frac{is\_NA(feature_j)}{length(feature_j)}$$

Only feature with  $q$  value bigger than the minimum requirement is to be kept. To properly set up the minimum requirement value, the following graphs are used.



*X Axis: min\_q: minimum q values from 0 to 1*  
*Y Axis: # of features whose q value >= min\_q*

Based on the graph above, minimum  $q$  values for each raw dataset are determined based on two principles:

- Keep fewer features given similar  $min\_q$
- Keep larger  $min\_q$  given similar number of features

Therefore, we choose  $min\_q$  values for each raw dataset shown as below:

Dataset	Min_q
PET_week	0.84
PET_month	0.85
PET_year	0.82
STEO_qtr	0.5
STEO_month	0.5
STEO_year	0.5

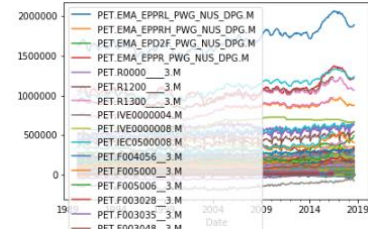
Notice that dataset for daily information is skipped in this section because there are totally 30 features there.

#### 4.2.3 Correlation

In nature, there could be high correlations among features since the dataset involve features that represent very similar information.

For example, feature for U.S. annual import for crude oil should be highly correlated to feature for New England Area import for crude oil. Since there are over 180000 features, it is reasonable to assume there should be many features highly correlated with each other.

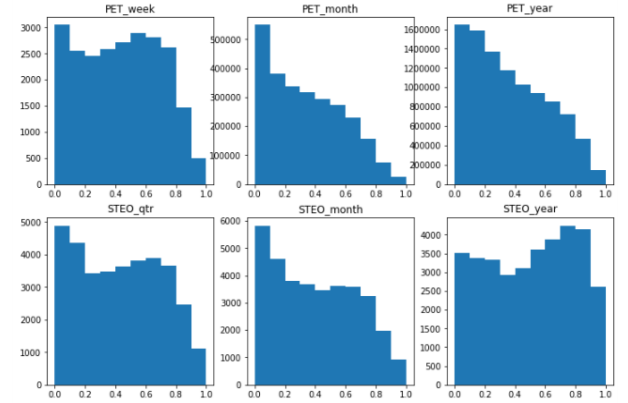
As an example, the following graph is the time-series plot for some features of PET\_week data.



In order to eliminate features with large correlations, the following procedures are implemented for each dataset.

- Setup the maximum correlation value allowable,  $max\_cor$
- Construct correlation matrix  $M_{K \times K}$
- for  $i = 1: K$ :
  - if  $M[:,i] > K$  not all false: remove feature  $i$
  - else: keep feature  $i$

To properly setup the  $max\_cor$  values, the following histogram is used:



*X Axis: bins of absolute value of correlation*  
*Y Axis: # of unique pairs of features inside each bin*

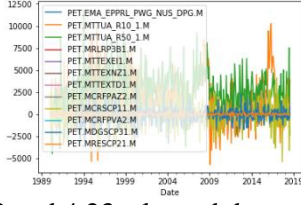
Based on the graph above,  $max\_cor$  values for each raw dataset are determined based on two principles:

- Keep as few features as possible
- Keep as large  $max\_cor$  as possible

Therefore, we choose  $max\_cor$  values for each raw dataset shown as below:

Dataset	Min_q
PET_day	0.95
PET_week	0.6
PET_month	0.25
PET_year	0.7
STEO_qtr	0.4
STEO_month	0.6
STEO_year	0.65

As an example, the following graph is the time-series plot for PET\_week data after removing highly-correlated features:



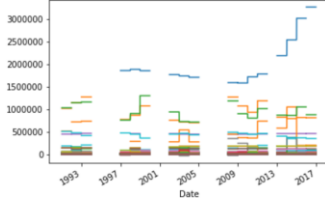
After 4.21 4.22 and 4.23, cleaned datasets are achieved with 452 features.

### 4.3 Time Alignment and Imputation

Since our dataset are based on different time frequencies, daily time index in the daily dataset should be used as the key while joining all the dataset together. This implies that a lot more missing values would be created, and thus proper forward fill methods should be adopted for data on differing time frequencies, shown as below.

Dataset	Forward fill limit
PET_day	0
PET_week	5
PET_month	21
PET_year	251
STEO_qtr	63
STEO_month	21
STEO_year	251

However, due to the source of data which does not provide all the data for free, some data points of features are intentionally omitted. Please refer to the following graph



time-aligned and forward-filled PET\_year

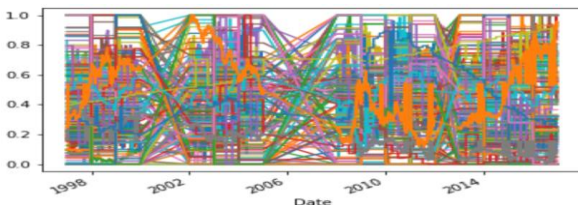
This intentionally-omitted issue occurs for all datasets, which would create situation that at some time point, most of features would be missing. To conquer this issue, method of removing records with large number of missing features are adopted. After removing those records, some features became poor quality, and thus we redo 4.22 again with min\_q = 0.9.

A dataset with 127 features and 3523 data points are reached.

### 4.4 Adding New Features and Normalizing

Alongside the 127 features finally selected, there are 13 features often considered important also added into the dataset, after steps presented in section 4.2 applied to them.

Besides, every feature is normalized to range of 0 to 1 to best address their importance in the training process. The graph below is the time-series for all features.



## 5. Experimental Setting

In this section, dynamic data training/predicting setting, responsive values generation, and evaluating metrics are introduced.

### 5.1 Dynamic Data

Dynamic data means at each date point  $t$ , prediction would be based on the information at  $t$  but trained on data in the certain length of history before  $t$ . The following formulas define training data, testing data, responsive data at each date point  $t$ :

$$\begin{aligned}
 TrainX_{(t)} &= X_{[t-tLength-tPeriod:t-tPeriod]} \\
 TrainY_{(t)} &= Y_{[t-tLength-tPeriod:t-tPeriod]} \\
 TestX_{(t)} &= X_{[t]} \\
 PredY_{(t)} &= model(TrainX_{(t)}, TrainY_{(t)}) * TestX_{(t)}
 \end{aligned}$$

$t$ : time point  $t$   
 $tLength$ : length of history (days)  
 $tPeriod$ : length of future to predict(days)  
 $model$ : machine learning method  
 $X$ : whole dataset of features  
 $Y$ : responsive value

### 5.2 Responsive Variables

There are totally four responsive variables:

$$\begin{aligned}
 Y1_{(t)} &= mean\left\{1 - \frac{P_t}{P_{t-1}}, t = t, t+1, \dots, t+tPeriod\right\} \\
 Y2_{(t)} &= std\left\{1 - \frac{P_t}{P_{t-1}}, t = t, t+1, \dots, t+tPeriod\right\} \\
 PredY3_{(t)} &= PredY1_{(t)} - PredY2_{(t)} \\
 PredY4_{(t)} &= PredY1_{(t)} + PredY2_{(t)}
 \end{aligned}$$

$P_t$ : oil price at time  $t$   
 $tPeriod$ : length of future to predict(days)

Goal of this project is to construct upper and lower bounds of average daily oil price changes. Assuming daily oil price changes following normal distribution, and Y3 and Y4 give us the bounds of 68% probability.

### 5.3 Evaluation Metrics

To evaluate the performance of upper and lower boundaries, 4 metrics are applied here:

$$\begin{aligned}
 E1 &= RMSE(PredY1, Y1) \\
 E2 &= RMSE(PredY2, Y2) \\
 E3 &= \frac{\sum_t I(PredY3_{(t)} \leq Y1_{(t)} \leq PredY4_{(t)})}{length(PredY1_{(t)})}
 \end{aligned}$$

$$E4 = \sum_t (Y1_t - PredY1_t)^2 + a * (PredY3_t - PredY4_t)^2 \quad \text{if } PredY3_t \leq Y1_t \leq PredY4_t$$

$$E4 = \sum_t (Y1_t - PredY1_t)^2 + a * (PredY3_t - PredY4_t)^2 + b * (Y1_t - PredY3_t)^2 \quad \text{if } Y1_t \leq PredY3_t$$

$$E4 = \sum_t (Y1_t - PredY1_t)^2 + a * (PredY3_t - PredY4_t)^2 + b * (Y1_t - PredY4_t)^2 \quad \text{if } Y1_t \geq PredY4_t$$

$a \gg b \gg 1$

Given other metrics similar to each other, model with lower E4 is considered better, assuming it gives a boundary that has two goodness:

- Real average daily return stays in it
- Boundary is narrow

## 6. Result

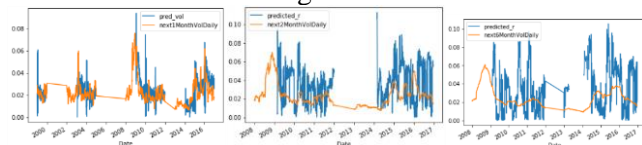
OLS, Ridge Regression, Lasso Regression and Multiple-Layer Neural Network are applied to predict average oil price daily changes.

For each method, we tried different combinations of different values of parameter, please refer to the coding for all details.

Model	Parameters
OLS	tLength, tPeriod
Ridge	tLength, tPeriod, alpha
Lasso	tLength, tPeriod, alpha
Neural Network	tLength, tPeriod, alpha, layers#, nodes#, activation function

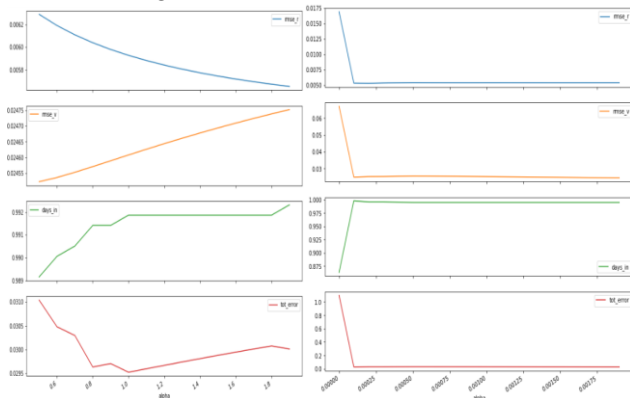
### 6.1 tPeriod

tPeriod with smaller values tends to have more reliable prediction. For example, the following graph is about prediction for the next 30,60 and 180 days of daily volatilities from left to right.



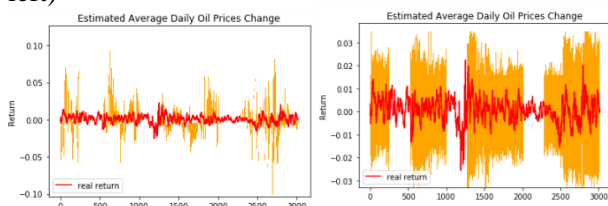
### 6.2 alpha

Alpha must be positive so as to gain a reasonable bound. For example, the following graph show the evaluation metric values for different alpha for Ridge and Lasso regression(Ridge left).



### 6.3 Activation Function

In NN, we tried 'tanh' and 'relu' activation function and find that 'relu' outperforms a lot better. ('tanh' left)



### 6.3 Number of Nodes and Number of Layers

Consistent with common sense, NN with larger number of layers and nodes would tend to give better prediction. In our report we tried 4 layers with 140

nodes in each layer at most due to computational power, and it gives us the best result.

Please refer to the table in next page to see the best results of every model at their best tuning.

## 7. Conclusion

After proper tuning, other than OLS, all other models perform comparably. It could mean that regularization is very important.

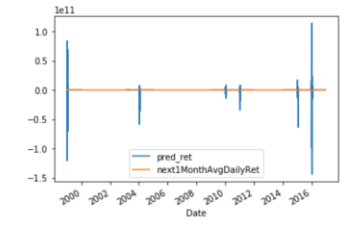
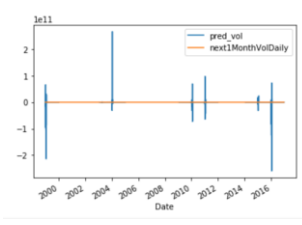
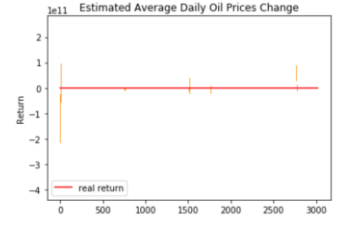
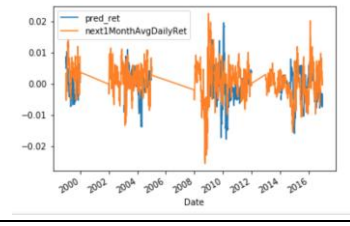
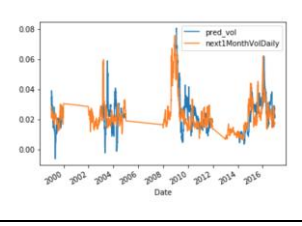
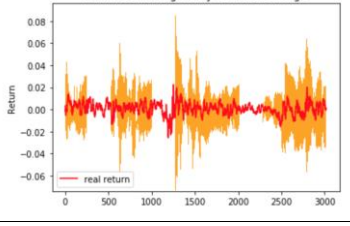
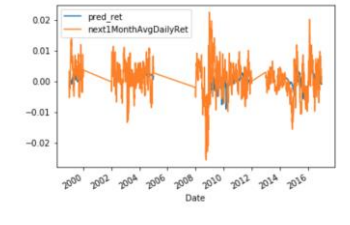
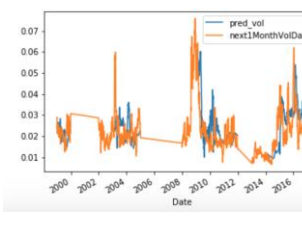
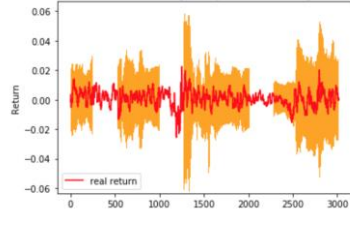
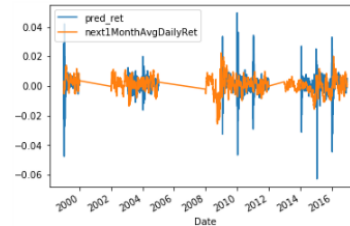
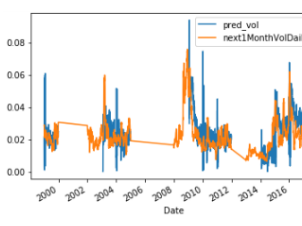
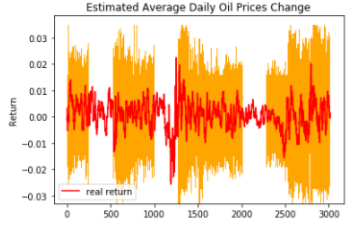
Secondly, the fact that smaller value of tPeriod is preferred means that time-series forecasting should use historical data that is close enough to the date of predicted variable.

Thirdly, it is important to notice that in NN model, different activation function would generate different results. This may be because the predicted variables are daily return and daily volatilities. In nature the range of them should be very small, even rarely reaching -.05 or .05. But tanh activation function produces results from -1 to 1, and thus its prediction would tend to have more extreme points leading to very wide and volatile boundaries at the end.

Looking at the predicted return and predicted volatilities by Neural Network in the next page, it is noticed that the model does pretty well when real data fluctuates a little but poor when real data fluctuates a lot. In other words, NN tends to predict extreme points when real data's fluctuation increases. If we assign certain values to predicted returns or volatilities who are over the, say 90%, percentile of all predicted values, and certain values to predicted values who is below, say 10%, percentile. It is promising to have a more accurate predicted boundary.

In all, machines are able to detect pattern between features and responsive variables. Due to limited number of training samples at each time point, regularization is necessary to avoid overfitting.



ML	Return	Volatility	Range	Parameter and Evaluation
OLS				E1: 7.011991e+09 E2: 1.154282e+10 E3 8.197018e-01 E4 4.401628e+23
Ridge				E1 0.005927 E2 0.024608 E3 0.991866 E4 0.029526
Lasso				E1 0.005266 E2 0.025242 E3 0.995933 E4 0.030042
NN				E1 0.005191 E2 0.022936 E3 1.000000 E4 0.021941

## References

[1] "Build with AI", DeepAI

<https://deepai.org/machine-learning-glossary-and-terms/neural-network>

[2] "Artificial Neural Networks as Models of Neural Information Processing"

<https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing>

[3] Neural Network Models

[https://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](https://scikit-learn.org/stable/modules/neural_networks_supervised.html)