Capstone Project

Predicting WTI spot prices

Machine Learning Engineer Nanodegree

Kirill Romanov

January 26th , 2017

I. Definition

Project Overview

1. Oil plays an increasingly significant role in the world economy since nearly two-thirds of the World's energy consumption comes from crude oil and natural gas.
2. Predict crude oil price is a challenging task because it depends various fundamental factors (supply and demand , stock, GDP growth, population growth etc.), political actions and speculative behavior on commodity exchange
3. Due to this complexity oil price prediction is an ideal task for the machine learning algorithms. There are many experiments for crude oil price prediction using SVM[1] as well as Artificial Neural Network-Quantitative (ANN-Q) model and Linguistic model[2]
4. The goal of this project is to build crude oil spot price predictor (for West Texas Intermediate) based on Support Vector Regression algorithm and simple neural network.
5. This algorithm takes as an input major fundamental indicators, produce additional technical indicators and give as output price prediction for the next month, for the third month and for the 6-th month from the date of prediction

Problem Statement

To build good prediction we realize the following tasks:

1. **Select features** dataset and collect historical data for these features
2. **Build additional features** based on historical oil spot price
3. **Normalize feature dataset** – exclude time-series without data and apply scaling and transformation
4. **Test feature transformation method** and select the best
5. **Split dataset** to training and testing dataset (using rule 80/20)

[1] W. Xie et al. "New Method for Crude Oil Price Forecasting Based on Support Vector Machines" ICCS 2006, Part IV, LNCS 3994, pp. 444 – 451, 2006

[2] Siti Norbaiti Abdullah "Machine learning Approach for crude oil price prediction" 2013

6. **Train** prediction models with baseline parameters using training dataset. We will use well recommended for this purposes Support Vector Regression model and Keras Sequential model with Theano backend
7. **Predict prices** on testing dataset, calculate performance metrics and analyze the results.
8. **Fitting hyperparameters of predictive models:**
    a. Use functionality of parameter search
    b. Calculate performance metrics
    c. Analyze the results and select best combination of hyperparameters for each model

Metrics

The common metrics to evaluate "goodness-of-fit" of regression model are:

1. Mean squared error (MSE). This metric is used as the optimization metric in some models (for example in Keras sequential model). The formulas for this metric is:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2$$

   Where A are actual prices for t period and are forecasted prices for the same period

2. Root mean squared error (RMSE) – it is just square root of MSE. We will use this metric for analysis

However, for practical reasons the model should also predict in which direction the price is moving (rise or falling). For this purposes we will use Mean Directional Accuracy (MDA) or Dstat:

$$D_{stat} = \frac{1}{N}\sum_{t} 1_{sign}(A_t - A_{t-1}) == sign(F_t - F_{t-1})$$

Where A(t) and A(t-1) are actual prices for t and t-1 period, and F(t), F(t-1) are forecasted prices for the corresponding periods.

Important note: as we solve regression problem, only first two metrics are used for optimization. Dstat we use only for information – to evaluate how good our model from this point of view is. In case if it is needed use this metric for model optimization, it is better to switch on classification problem and try to predict price direction movement (rise/fall/stable). This aspect is out of scope of our research.

II. Analysis

Data Exploration

1. For this project I manually collected monthly historical data for the period 1986-2016 that could explain spot price of crude oil
2. Below is the table with dataset description:

Feature Group	Feature name	Source of data	Units of measure	Feature Code
Target feature	Cushing, OK WTI Spot Price FOB	www.eia.gov	Dollars per Barrel	1_WTI_Spot
Prices of related assets	Europe Brent Spot Price FOB	www.eia.gov	Dollars per Barrel	2_Brent_Spot
	Cushing, OK Crude Oil Future Contract 1			2_WTI_Contr1
	Cushing, OK Crude Oil Future Contract 2			2_WTI_Contr2
	Cushing, OK Crude Oil Future Contract 3			2_WTI_Contr3
	Cushing, OK Crude Oil Future Contract 4			2_WTI_Contr4
Supply	World Crude Oil Production	https://ycharts.com	Millions Barrel per day	3_Total_CrOil_Prod
Demand	OECD Petroleum Consumption	https://ycharts.com	Millions Barrel per day	4_OECD_Consumpt
Inventory	OECD Petroleum Stocks	https://ycharts.com	Billions Barrels	5_OECD_Stock
	U.S. Ending Stocks of Crude Oil and Petroleum Products	www.eia.gov		5_US_Stock
	U.S. Imports of Crude Oil and Petroleum Products			5_US_CrOilImp
Economy	Inflation, consumer prices in high income countries	databank.worldbank.org	Monthly %	6_InflationHI
	Inflation, consumer prices in medium and low income countries			6_InflationLI
	GDP growth in high income countries			6_GDP_GrowthHI
	GDP growth in medium and low income countries			6_GDP_GrowthLI
	Inflation, consumer prices in high income countries		Billions people	6_Popul_HI
	Inflation, consumer prices in medium and low income countries			6_Popul_LI
	Average monthly exchange rate USD/GBP	http://fxtop.com/	GBP per 1USD	6_USD_GBP
	Average monthly exchange rate USD/GBP		JPY per 1USD	6_USD_JPY

To improve the quality prediction, additionally some technical indicators were generated additionally to this dataset:

Feature Group	Feature name	Formula	Feature Code
Technical indicators	Rolling mean price with 6-month time window	Mean price of last 6 periods of time	7_6m_ma
	Rolling mean price with 12-month time window	Mean price of last 12 periods of time	7_12m_ma
	Bollinger bands upper band	Rolling mean price with 6-month time window + 2*rolling standard deviation for 6 month	7_Bol_upper
	Bollinger bands lower band	Rolling mean price with 6-month time window - 2*rolling standard deviation for 6 month	7_Bol_lower
	Bollinger bands Bandwidth	(7_Bol_upper – 7_Bol_lower) / 7_6m_ma	7_Bol_BW
	Exponentially-weighted moving average with 6-month time window	pandas.ewma with default parameters	7_6m_exma
	Exponentially-weighted moving average with 6-month time window		7_12m_exma
	Monthly average return coefficient	(Price(t)/Price(t-1)	7_DailyReturn

Finally, I generated predicted label dataset. Our predictor will be built to forecast the following:

- Spot price for the next month
- Spot price for the third month from current
- Spot price for the sixth month from current

To make predicted label dataset we copy 1_WTI_Spot feature series and shift it from future to past for the corresponding month as it shown below:

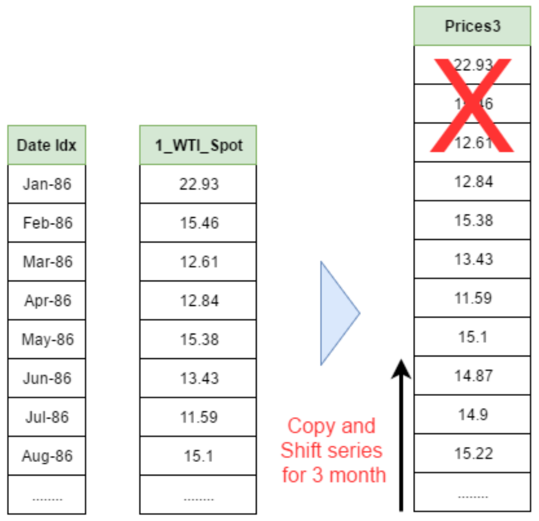


Figure 1 Algorithm to form lables for third month price predictor

Exploratory Visualization

We have deal with non-stationary 27-feature dataset and scale for each time series is different. To avoid multi-page plotting of graph with data just see two sets of diagram:

1. Histograms with data value distribution to demonstrate that each feature has different scale and not normally distributed
2. WTI Spot price dynamics to illustrate non-stationarity of data

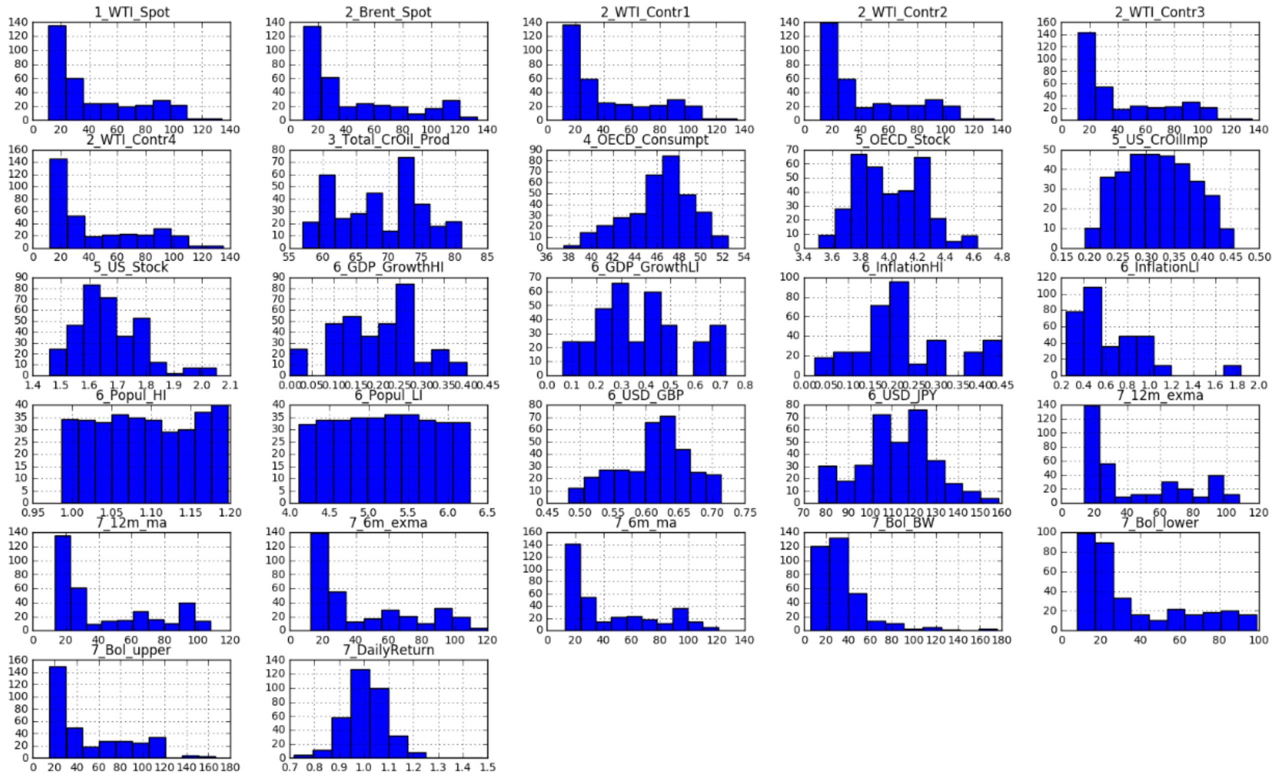


Figure 2 Features values distribution

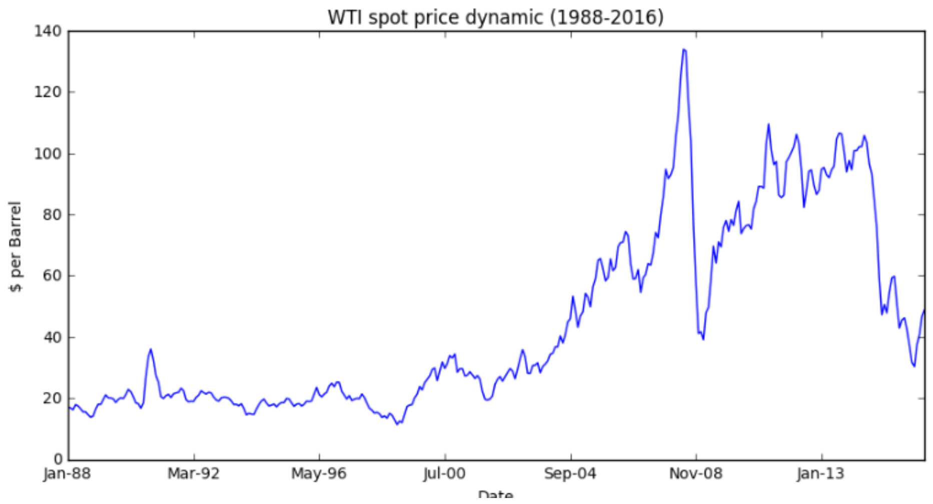


Figure 3 WTI spot price dynamic (non-stationary time series)

Algorithms and Techniques

To build predictive models we use to kind of models:

1. Support Vector Regression
2. Keras Sequental Neural Network model (with Theano backend)

The parameters for each model are below:

SVR

Parameter name	Parameter value for baseline value	Set of values for fine-tuning (if applicable)
Kernel	Default (Rbf)	-
C	10	[10, 20, 30, 50, 70, 80, 100, 200, 1000]
Gamma	Default (1/27)	[0.5, 0.1, 0.001, 0.0001, 0.00001]

Keras Sequental Neural Network (Theano backend)

Parameter name	Parameter value for baseline value	Set of values for fine-tuning (if applicable)
Model type	Sequental	-
Type of layers	Core	-
Number of layers	3	-
Input dimension of first layer (dim = number of features)	27	-
Output dimension of first layer	27	[27,54,108,216,532,1064]
Output dimension of second layer	13	[15, 27, 54, 80,160,320]
Output dimension of third layer (model output)	1	
Activation type	Default (relu)	
Loss function	mean_squared_error	
Optimization method	'adam' (Adaptive Moment Estimation)	
Epoch Nb	100	
Batch Size	4	

Benchmark

According to some academicals research[3], best results for oil price prediction (on testing dataset) are:

- RMSE<10
- Dstat>50%

We try to reach this benchmarks, however there are two major notes:

1. Some models were trained on dataset before 2004. As you can see on figure 3, from this period huge price volatility is observed, so our results could be worse.
2. Some research are used more extensive set of predictive features. To collect this feature it is necessary to purchase access to this data. Due to this factor we don't use these feature in our model

[3] These papers were mentioned in project overview section.

III. Methodology

Data Preprocessing

1. According to some studies[4], we do not need eliminate non-stationarity for SVR models, also due to extended capabilities to learn, we don't need to do it for NN model as well.
2. However, our features has different scales, so to avoid overfitting model to the features with big numbers, we should re-scale our features. Some techniques has an additional positive effect - non-stationarity reduction. To do this we use three techniques:
    a. **Z-score transformation**. Some researcher to normalize time series with outliers and pre-process financial series uses this kind of transformation. $Z = \frac{p_t - \mu_p}{\sigma_p}$ where p is the feature value at period t , p µ and p σ is the mean and standard deviation of the series respectively
    b. **Logarithmic Transformation**. Logarithmic transformations are often applied to time series data in order to achieve a more homogeneous variance in the residuals. Another advantage of logarithmic transformation is that taking natural logarithms of return series often produce a more stationary series. $Log = \ln(p_t)$ where Log = transformed data and p is the original value
    c. **Normalize from sklearn.** This algorithm scale individual samples to have unit norm

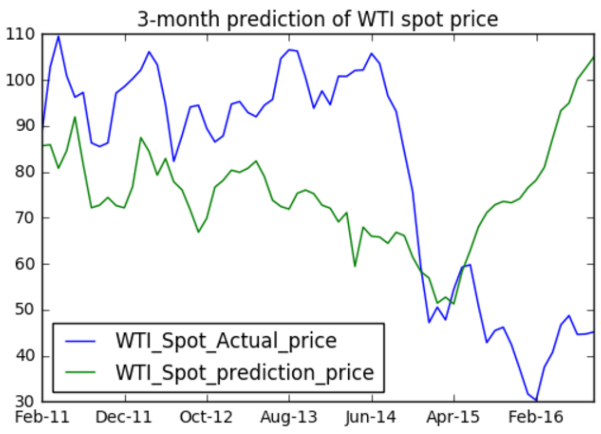
To practically asses the efficiency of each type of preprocessing algorithm, I run baseline SVR model (only for one month prediction) with each set of input processed data and compare performance. The results are below:

Baseline model	Transformation algorithm	RMSE value		Dstat value	
		Training	Testing	Training	Testing
SVR	Z-score	4.34	11.87	57.14	64.18
	Log	5.81	10.22	53.85	61.19
	Norm sklearn	26.22	59.41	49.08	47.76

Initially I choose data pre-processing algorithm only after testing it on 1-month SVR model. Based on it, there is obvious that Norm sklearn algorithm doesn't work well, but here is no difference between log and Z-score.

However, Z-score doesn't well for NN network:

[4] Ojemakinde, Bukola Titilayo, "Support Vector Regression for Non-Stationary Time Series. " Master's Thesis, University of Tennessee, 2006. http://trace.tennessee.edu/utk_gradthes/1756



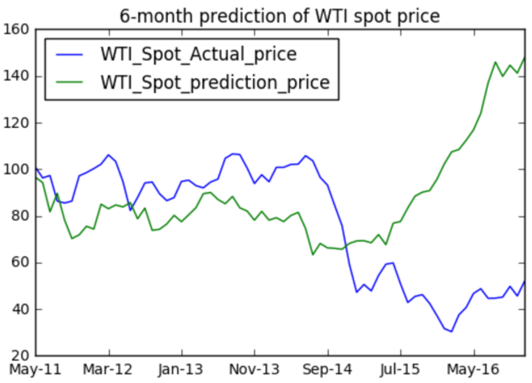


Figure 4 Bad prediction of NN model based on Z-score pre-processed data

And after manually testing Z-score input and Log method on NN model, I choose log normalization.

Regarding to NaN values for feature and label dataset:

1. Initial dataset contains historical data for 1986-2016 period. Data have collected manually from different sources.
2. Period 1986-1988 contains missing data for important features but we include this period to calculated technical indicators (e.g to have rolling mean for 12 month as of Jan-88, we have to have data for 1987). It seems be unreasonable use backfill propagation technique to fill missing date or calculate some means to fill it, so after calculate the technical indicators, we exclude period 1986-1988
3. Also, we initially have data till December 2016, but because we have predictors up to 6-month, after we form label dataset, we exclude period Jul-2016 – Dec-2016 from our dataset

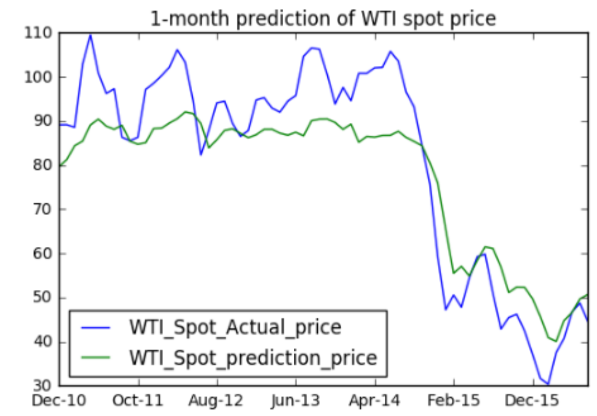
Finally, we split our dataset to training and testing data. We use classical proportion for split 80/20.

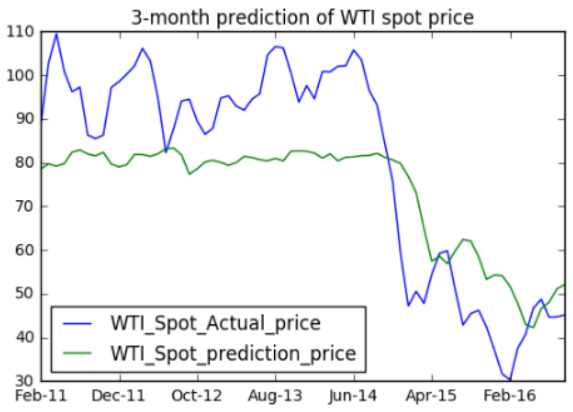
Implementation

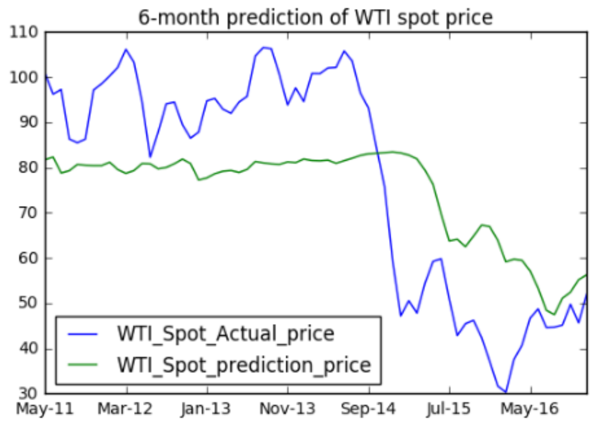
According to algorithms settings (see section "Algorithms and Techniques"), we train each baseline model for three predictors (next month prediction, third month prediction and sixth month prediction). The results are below.

SVR Baseline

As we can see, long-term predictors have lower performance results, comparing to short-term, to understand how we can improve models, let's see visual representation of the results:



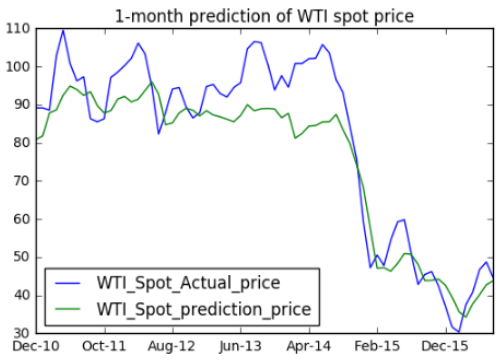


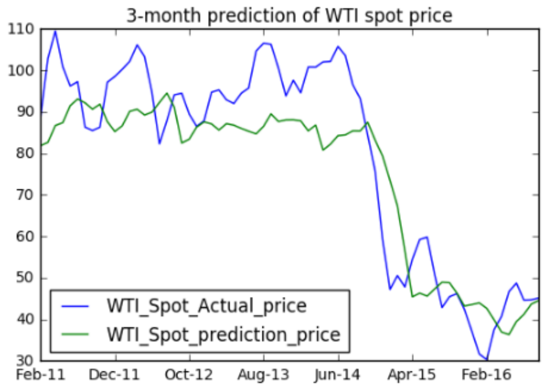


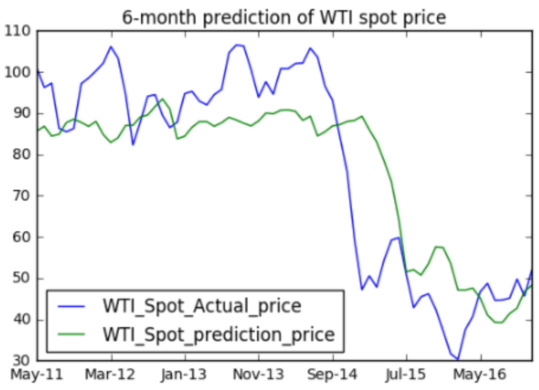
Predictor type	RMSE		Dstat	
	Train	Test	Train	Test
Next month predictor	5.81	10.22	53.85	61.19
3rd month predictor	8.76	16.04	51.28	40.3
6th month predictor	10.07	18.31	56.04	40.3

On these graphs, it is obvious that here is some underfitting of the models. We should try to raise C factor, also we try to choose right gamma value because we use rbf kernel

NN Baseline







Predictor type	RMSE		Dstat	
	Train	Test	Train	Test
Next month predictor	5.76	9.5	54.21	56.72
Third month predictor	8.3	11.74	54.21	44.78
Sixth month predictor	10.31	13.89	51.28	43.28

Despite similar performance metrics, figures for NN algorithms looks much more consistent comparing with SVR models. So we try to increase output neurons quantity for first and second layer of network to make model more precise.

Refinement, Model Evaluation and Validation

To find optimal parameters we will use parameters search instead of grid search. I do it for the following reasons:

1. When we use grid search we assume cross-validation mechanism. I already split the data on training and testing dataset, which are completely different (see figure below). Due to this fact I don't afraid an overfitting. Also my dataset is relatively small and when I train NN each additional observation is important, so two bins is quite enough for reliable experiments
2. I don't want some automatic selection of best model. I have more than one performance metrics and I want to see the relationship between hyper-parameters values and performance results

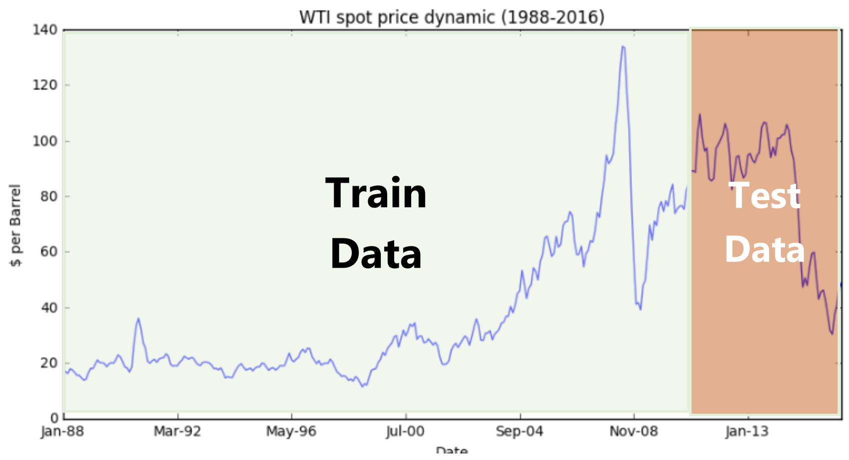


Figure 5 Different pattern of Train and Test dataset

So, to make model refinement I use parameter search mechanism from sklearn and save the results for each hyperparameters combination into special array.

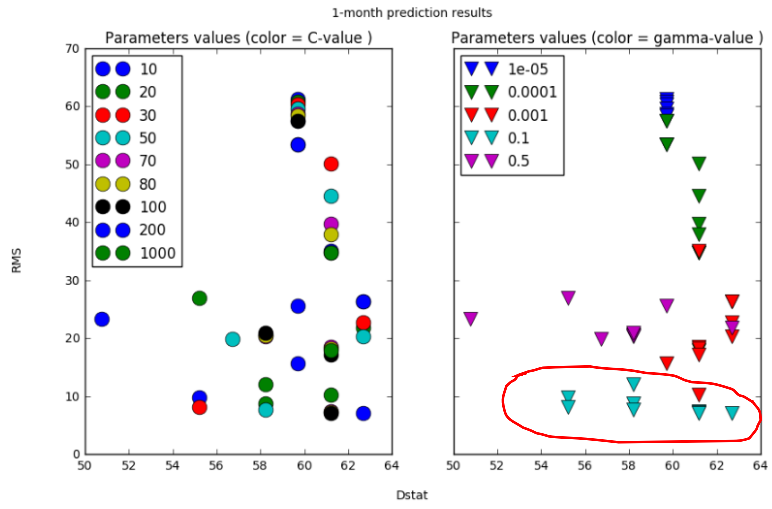
Full results of refinement, according to parameter search (the possible values of hyperparameters see in section "Algorithms and Techniques") are available in Python notebook file, below I demonstrate the logic how fine-tuning parameters were chosen based on the statistic.

Next-month predictor

Here are two models we can use to get the similar results. Let's begin with SVR
To make analysis easer I grouped results by hyperparameter values (calculate mean value and std). Here are results:

C-value	Dstat mean value	Dstat std	RMSE mean value	RMSE std
10	59.1	2.3	37.8	21.7
20	60.6	2	34.3	22
30	59.4	2.9	32.4	21.9
50	59.7	2.4	30.4	21.1
70	60.3	1.3	28.9	20.4
80	60.3	1.3	28.4	20
100	60.3	1.3	27.5	19.5
200	59.1	4.9	25.2	17.5
1000	**59.4**	**2.7**	**20.4**	**10.4**

C-value	Dstat mean value	Dstat std	RMSE mean value	RMSE std
0.00001	59.9	0.5	56.1	8.3
0.0001	61	0.9	40.3	12.8
0.001	61.5	1	20.4	7
0.1	**59**	**2.7**	**8.3**	**1.7**
0.5	57.5	3.3	22.2	2.5

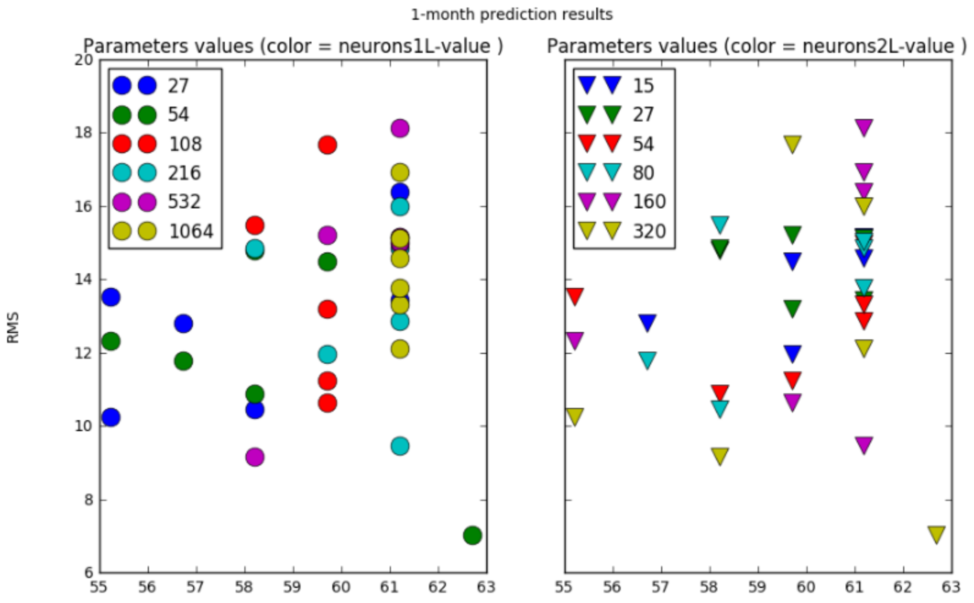


According to results above, for Next-month predictor we have to setup gamma value =0.1 and C with value >100. The best result in terms of RMSE and Dstat will be value =200. This will provide us RMSE about 8 and Dstat about 63%

NN results are following:

Neurons 1L Nb	Dstat mean value	Dstat std	RMSE mean value	RMSE std
27	58	2.7	12.8	2.3
54	58.4	2.6	11.9	2.8
108	59.7	0.95	13.9	2.7
216	60.4	1.24	13.3	2.4
532	60.4	1.24	14.6	2.9
1064	61.2	0	14.3	1.65

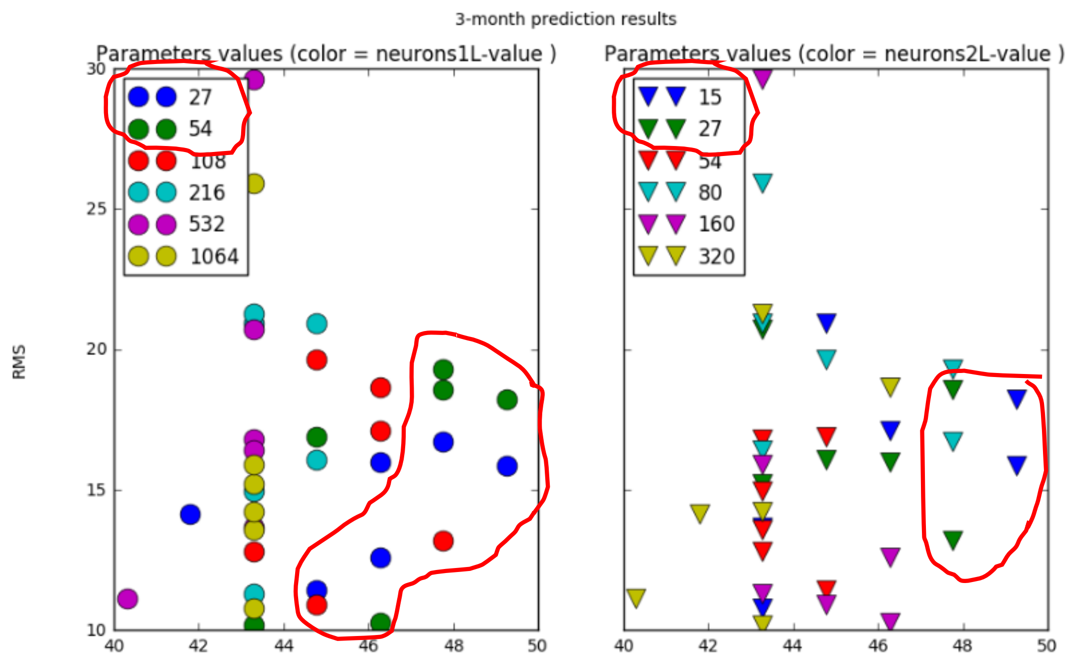
Neurons 2L Nb	Dstat mean value	Dstat std	RMSE mean value	RMSE std
15	59.9	1.7	14	1.3
27	59.7	1.3	14.4	0.9
54	59.5	2.4	12.8	1.5
80	59.4	2	13.6	2
160	60	2.4	14	3.6
320	59.7	2.7	12	4.1



Based on statistic above we can see that if we use1064-neurons first layer we can get stable results in terms of Dstat: more than 61%. The best combination of parameters is 54-320. With these values we get about 63% of Dstat and RMSE about 7.

Third-month predictor

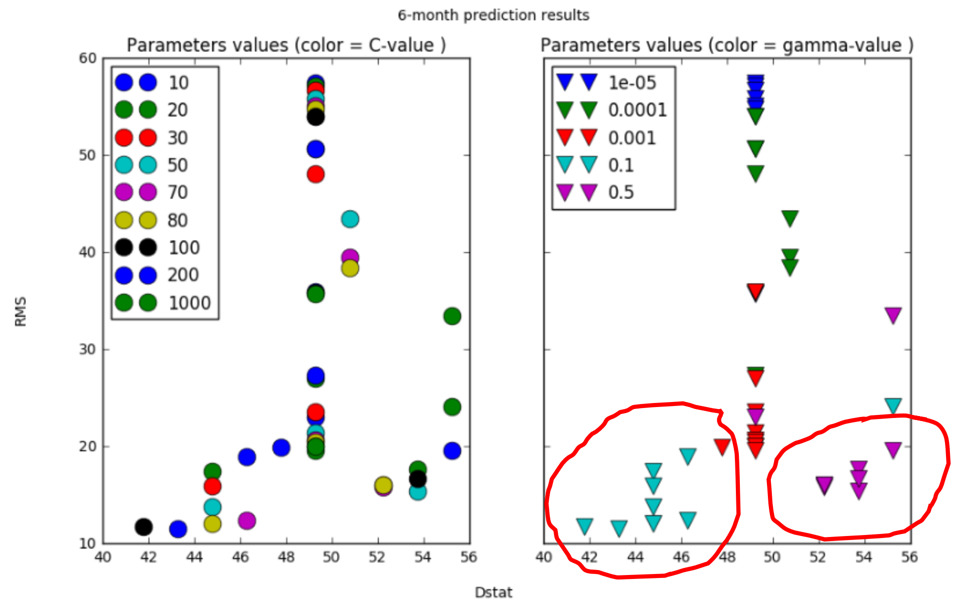
Here we get bad results for SVR models (both in terms of Dstat and RSME), so we can use only NN model. Because it is not clear trends in descriptive statistics, lets justify our choice of parameters on figure below:



As we can see on the figure above if we want to find some balance between RMSE and Dstat, we should chose minimal parameters of neurons Nb: 27/54 for the first layer and 15/27 for the second. For example 27-15. But if RMSE is more important, combo 54-160 is the best. In this case we get MSE about 10 , however Dstat is 46%.

Sixth-month predictor

For this predictor, as we can see in the notebook, SVR is best solution. Here is also figure is more clear to select best parameter:



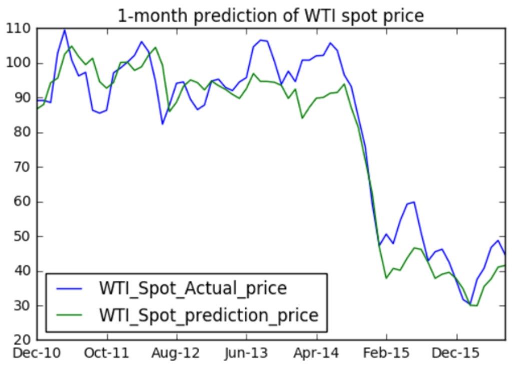
As we can see if RSME is more important we can choose combo gamma =0.1 and C=70-100 (70 is better). If Dstat is important, we should use gamma =0.5 and C=50-100 (50 is better).

V. Conclusion

Free form visualization

As a result of parameter selection made above, we identified some possible good combination of hyperparameters. Final selection depends on some preferences of user in terms of RMSE and Dstat. Here we state some examples of good models in terms prediction vs actuals.

Next-month predictor (NN, neurons values 27-54-320-1)



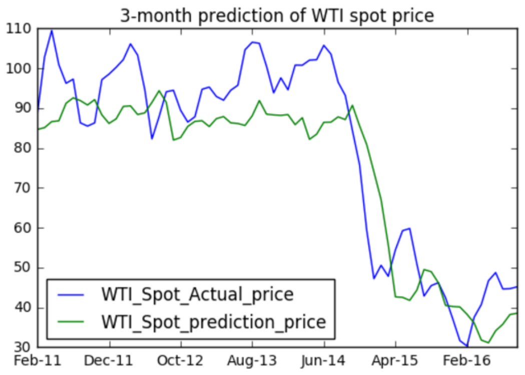
<u>RMSE</u>

Training	5.03
Testing	7.73

<u>D stat</u>

Training	52.8
Testing	59.7

Third-month predictor (NN, neurons values 27-54-27-1)



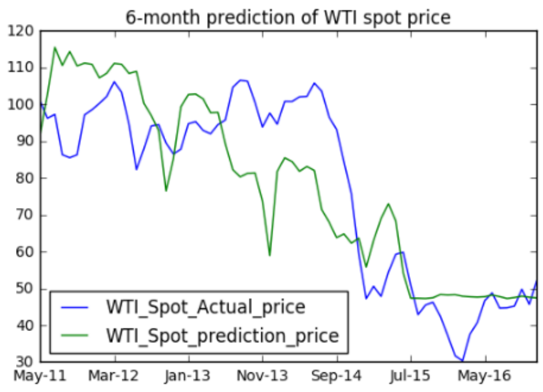
<u>RMSE</u>

Training	8.67
Testing	11.76

<u>D stat</u>

Training	53.11
Testing	44.78

Sixth-month predictor (SVR C=50, gamma =0.5)



RMSE

Training	4.6
Testing	15.4

D stat

Training	68.8
Testing	53.73

Reflection

1. During the implementation of this model, the main goal was to apply and test machine-learning techniques for the common problem – non-stationary time-series prediction
2. As a result, we can see that selected model can deal efficiently with complex time-series prediction:
    a. Despite price-oil shock in 2008 and sharp fluctuations after this period, our model demonstrate good prediction results during this period, as we can see on graphs above
    b. During fine-tune process we can see that if can identify some important hyperparameter (e.g gamma = 0.5 for SVR 6-month prediction model) this model will stable in terms of performance metrics.
3. According to benchmarks mentioned in the section “Benchmarks” our model meets the minimal specification. There are some objective factors (oil prices turmoil in 2008), but also there are some areas for improvement of our model. Some measures are described in the next section.

Improvement

During this project, we created script with clear workflow to do, to build and analyze prediction model. This script could be a base for any further extensions, which could be made to improve and extend this model:

1. **Feature engineering**. During some undocumented experiments with features (changing weight, produce additional features), I improve Dstat accuracy up to 80%. Also adding some additional features (we need to buy this) could also improve the

results of Dstat accuracy up to 90% while decrease RMSE to 2-5, according some research

2. **Build more complex NN model.** As we see above, some simple NN beat SVR algorithm in terms of RSME. It is obvious we could improve results by developing more complex ANN with additional parameters
3. **Use ensemble models.** We have relatively good models in terms of RSME but our results for Dstat are instable. We can train some models (SVM, NN) for classification problem to predict price direction movement and group it with our models
4. **Use additional types of input data to build different type of model.** Here we work with only quantitative data, but we can also add some categorical data, text data to build sentiment analysis model, decision trees. And then combine (not only as an ensemble model but as an AI system)