NYC Yellow taxi demand predictions

**1. Task setting**

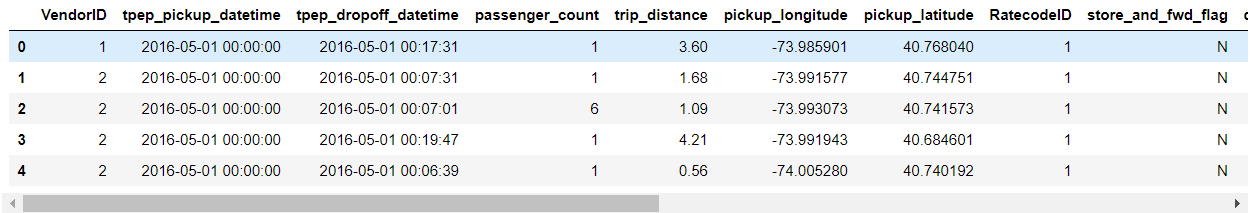
One of the most famous things in the NYC is yellow taxi. They’ve become a symbol of the city and the demand on taxi services is extremely high. In this project, we will try to predict the demand on yellow taxi services in different geographic areas on 6 hours in the future.

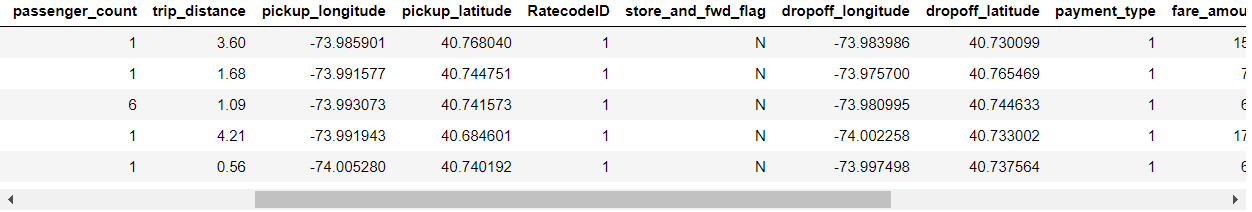


*Yellow taxi on Broadway*

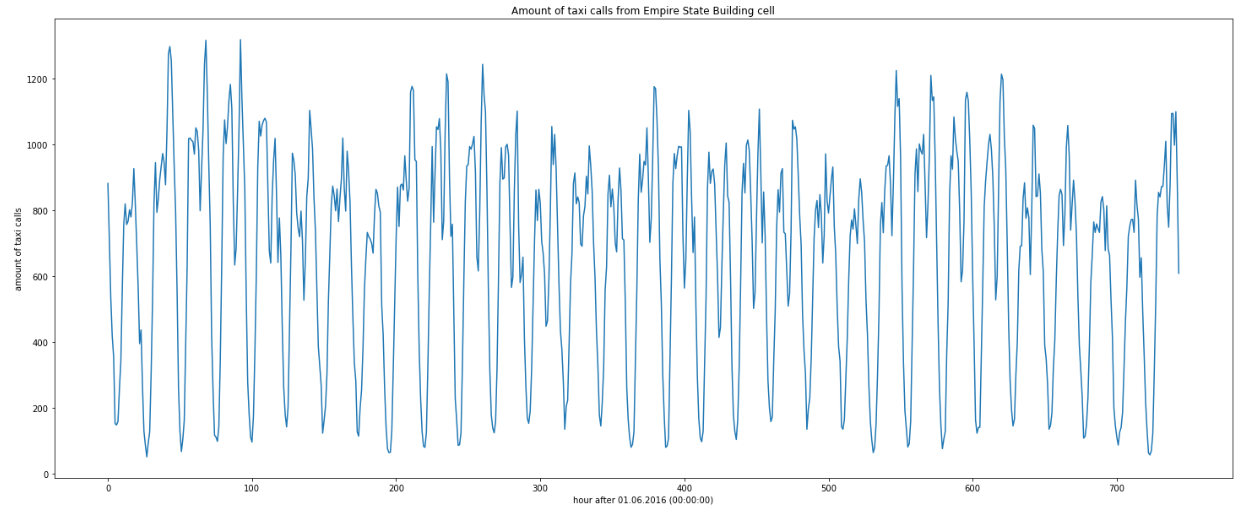
We will use data from the official NYC government info (<http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml>) from January of 2016 until May of 2016. June of 2016 will be the data to check our predictions (holdout dataset). So, let’s look on our dataset in scale. For start, we will take data of May 2016. For detailed description look at [data\_dictionary\_trip\_records\_yellow.pdf](http://localhost:8888/files/Desktop/coursera_final_project/data%20loading%20and%20visualization%20(part%201)/preprocessing/data_dictionary_trip_records_yellow.pdf) file.

That’s how the first 3 columns of our dataset look like:

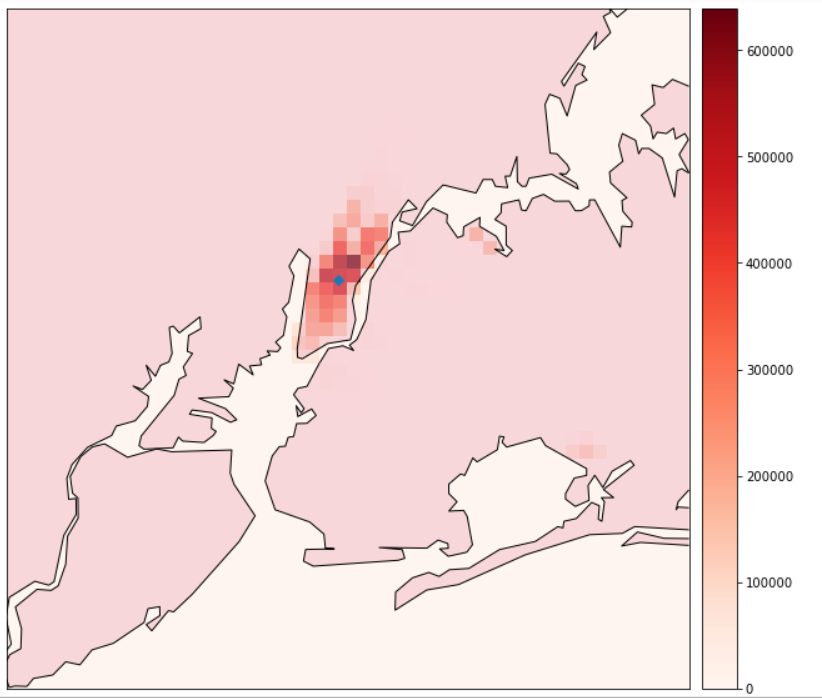




There are lots of features, but for now, let’s concentrate on geographic splitting. We will dispose of seconds in *“tpep\_pickup\_datetime”* column and split all the taxi calls on 2500 regions (50x50), which will form a square of the NYC. For example, that’s how a time series of region with Empire State Building look like (in May 2016).

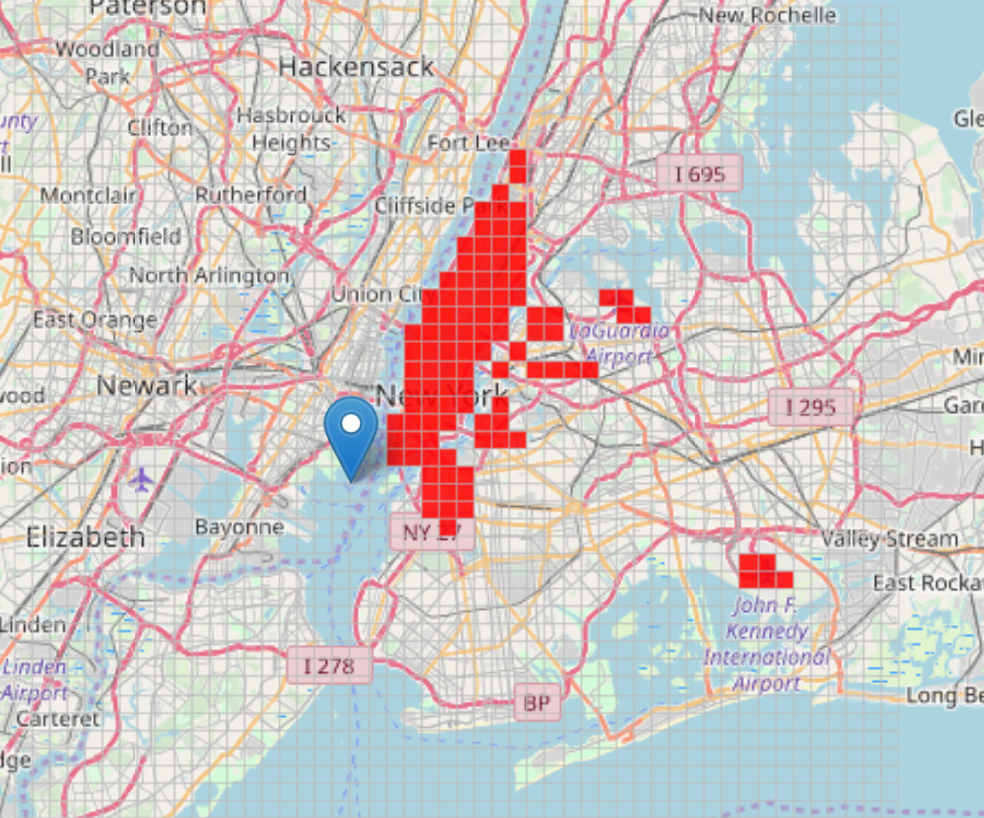


That’s how heatmap of amount of all the taxi calls in May 2016 look like (blue rectangle is Empire State Building):



As we can see, all the principal demand is in Manhattan and in airports.

Some of our regions are unattainable or the demand is pretty low, so let’s count only cells with a mean amount of taxi calls per hour more than 5. After that, we find out, that we need only 102 cells to make predictions for. Let’s concentrate on them. That’s how they look on interactive map (blue pointer is Statue of Liberty):



More interactive maps are in *maps\_1.html* file.

**2. First predictions**

**2.1 Linear model fitting**

Now, let’s build our first model and make predictions for one separate cell. Let’s stick to the cell with Empire State building.

At first, we will build a linear regression model with some regressive features built from our time series.

*(Linear regression – one of the simplest predictive models in statistics and ML, for more – see wiki:* [*https://en.wikipedia.org/wiki/Linear\_regression*](https://en.wikipedia.org/wiki/Linear_regression)*).*

For week’s seasoning calculating we will use Fourier series:



With *K =* 5, where

T – length of time series

168 – length of week (in hours).

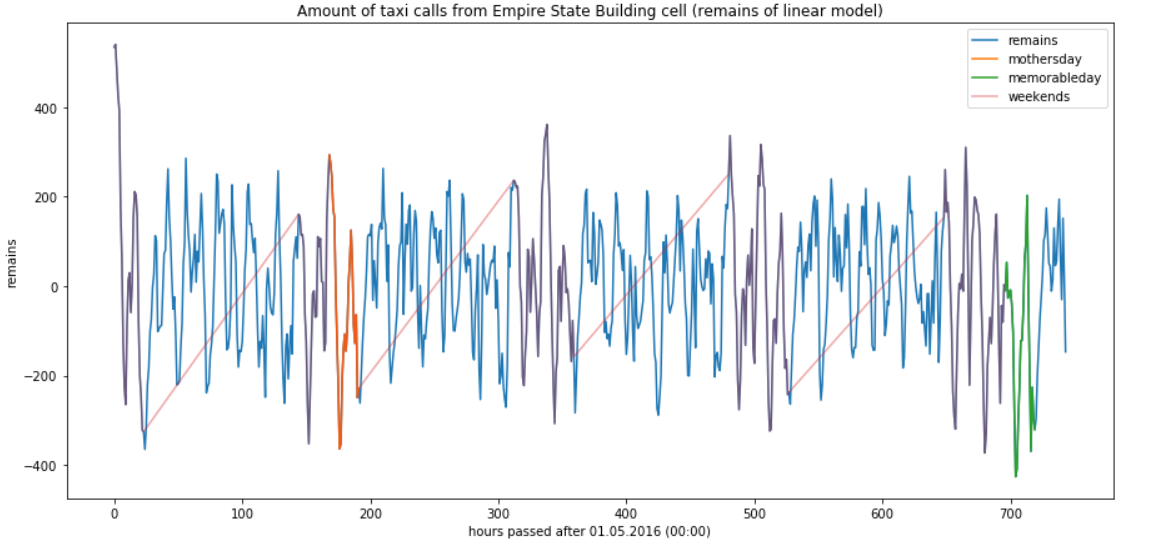
Also, we will count previous values and its moving averages and categorical features of weekends.

That’s how the predictions of linear regression of the time series look like with different regularizes (L1 – Lasso, L2 – Ridge):

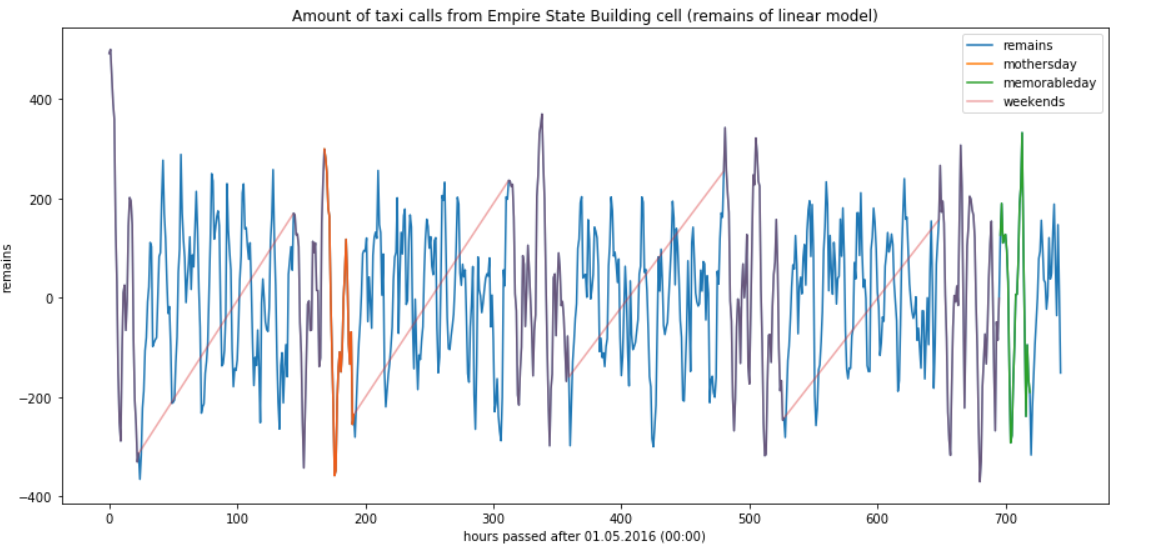


From the graph, we can see that there is no big difference in using models with different regularizes, so we will stick to the linear regression without any regularizes.

Now, let’s look at the graph of residuals of predictions:



We know that there are 2 national holidays in the USA during May of 2016: Mother’s Day and Memorable Day (marked on graph above). Mother’s Day matches the weekend day, so there is no big difference, but Memorable Day is on Monday and we can see an abrupt descent. Let’s count that holidays as categorical features too and fit our linear regression again. That’s how the graph will look like:



After the addition of new categorical features, we can see that our linear model considered time series’ behavior on holidays. Fine.

**2.2 ARIMA model fitting**

Now, let’s fit an ARIMA model’s modification with season counting (SARIMAX or SARIMAX) on the residuals.

*(For more info about ARIMA (autoregressive integrated moving average) see its wiki page:* [*https://en.wikipedia.org/wiki/Autoregressive\_integrated\_moving\_average*](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average)*)*

Let’s see a decomposition of residuals time series and check Dickey – Fuller criteria to see if the time series static or not:

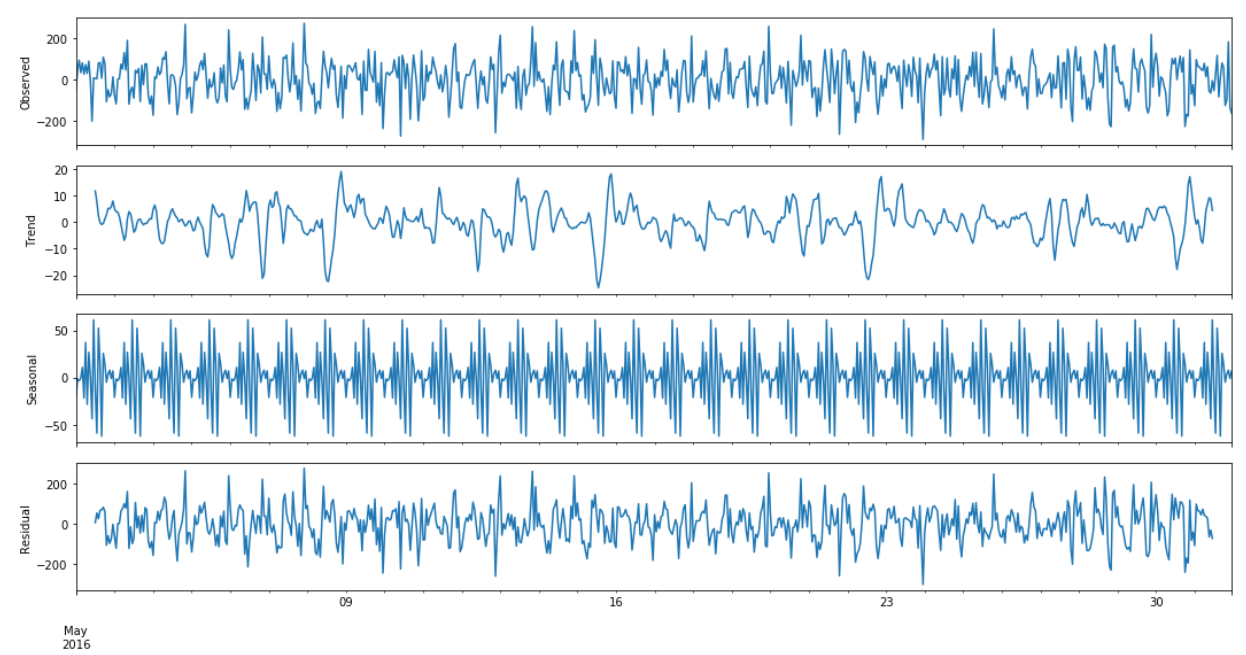
*H0: time series is not static*

*H1: H0 is incorrect*



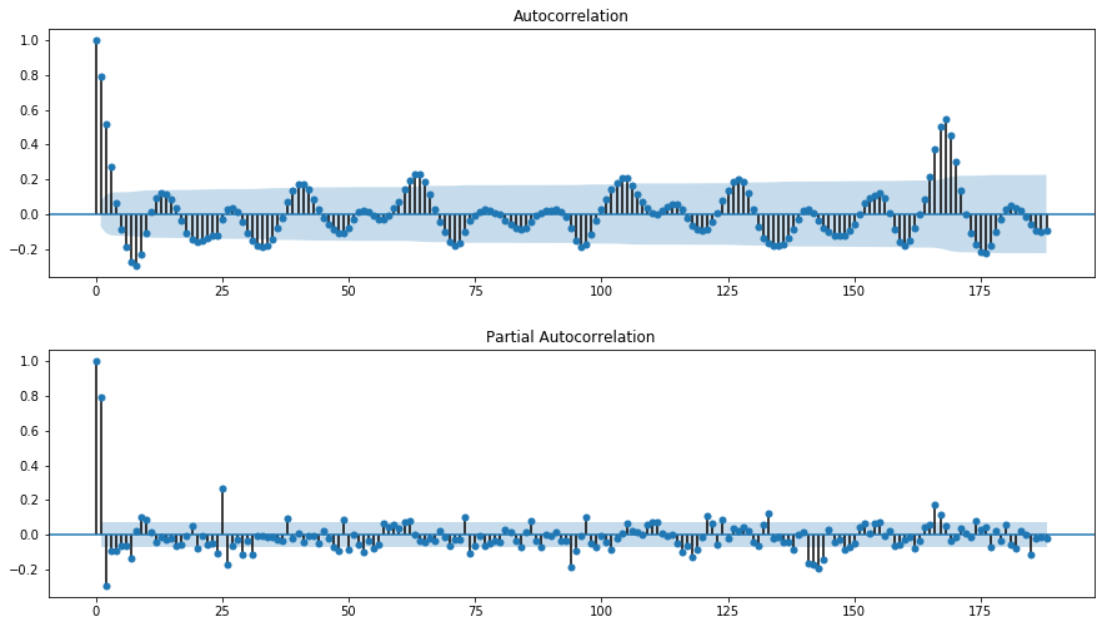
We can see that Dickey – Fuller criteria’s value is very low, but let’s use one seasoning differentiation and one straight differentiation to make graphs of autoregressions more convenient for hyper-parameters tuning.

That’s how the graph of STL-decomposition looks like after the differentiations:

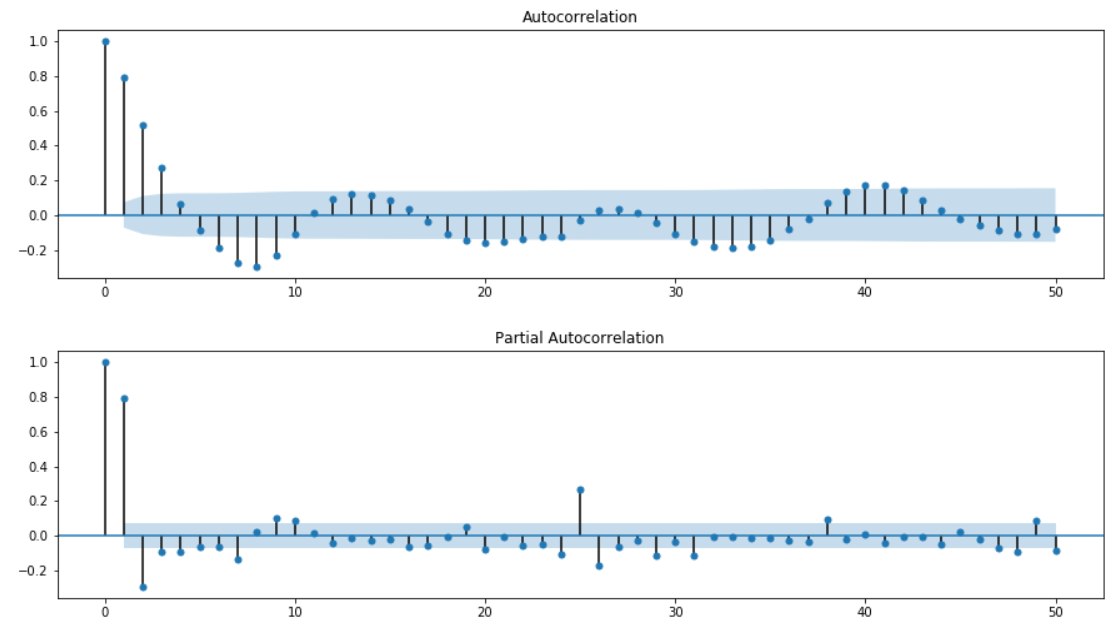


From graph, we can see that the time series is still static, but trend became closer to zero and observed and residual values became a bit harsher.

Now, let’s look at the autocorrelation and partial autocorrelation graphs:



The values of lags are still very unstable. Counting facts, that differentiation hadn’t made the autocorrelation graphs with appropriate small values hyper parameters and computational resources are limited, let’s see only first 50 lags and choose parameters from them:



The values are not as small as we expected, so let’s stick take the smaller values of initial approximation:

*Q = 1; q = 10; P = 1; p = 7;*

*D = 1; d = 1 (1 season and 1 straight differentiations)*

*(for more info of parameters’ initial approximation count see* [*https://www.quantstart.com/articles/Autoregressive-Integrated-Moving-Average-ARIMA-p-d-q-Models-for-Time-Series-Analysis*](https://www.quantstart.com/articles/Autoregressive-Integrated-Moving-Average-ARIMA-p-d-q-Models-for-Time-Series-Analysis)*).*

For comparing different ARIMA models we will use AIC criteria:



*Where*

*k – amount of params in statistical model,*

*L – maximum value of credibility (we consider that errors are normally-distributed and independent).*

Finally, we have to fit 2 \* 11 \* 2 \* 8 = 352 different models. Let’s do that.

That’s how the list of top-4 best models by AIC criteria looks like:

|  |  |
| --- | --- |
| parameters | AIC |
| (2, 7, 1, 1) | 8362.713843 |
| (7, 5, 1, 1) | 8365.734286 |
| (4, 6, 1, 1) | 8366.369248 |
| (6, 7, 1, 1) | 8367.465301 |

And the best model’s short summary:

Statespace Model Results

==========================================================================

Dep. Variable: value Dep. Variable: value

No. Observations: 744

Model: SARIMAX(2, 1, 7)x(1, 1, 1, 24) Log Likelihood -4149.357

Date: Fri, 05 Oct 2018 AIC 8362.714

Time: 08:06:00 BIC 8509.205

Sample: 05-01-2016 HQIC 8419.27

- 05-31-2016

Covariance Type: opg

==========================================================================

Ljung-Box (Q): 60.44 Jarque-Bera (JB): 16.49

Prob(Q): 0.02 Prob(JB): 0.00

Heteroskedasticity (H): 0.71 Skew: -0.03

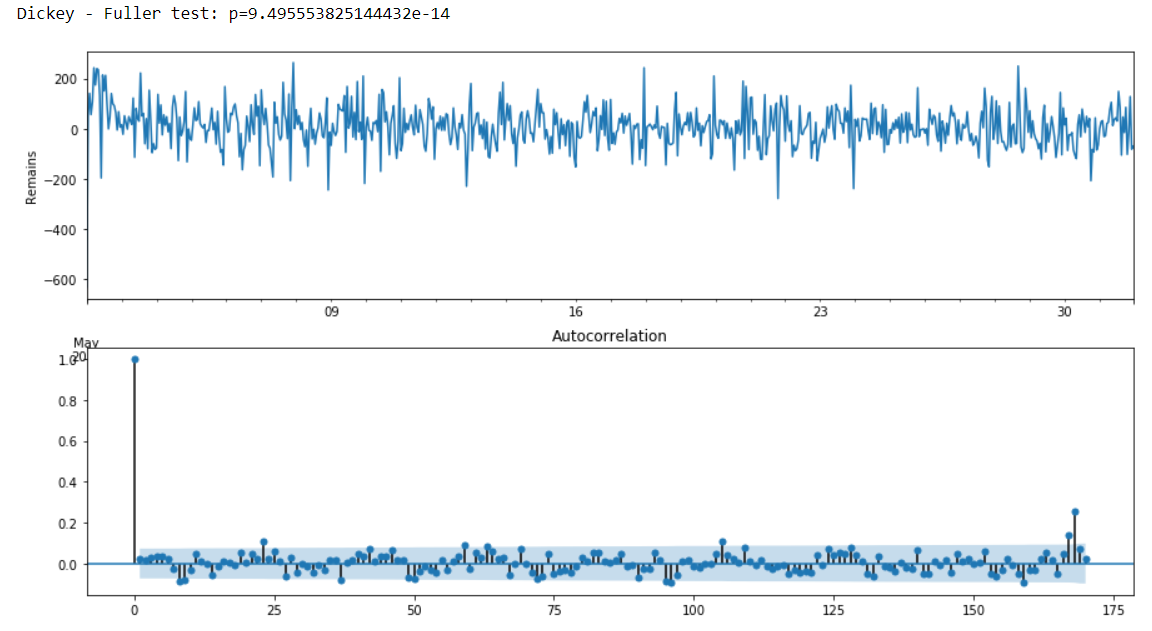
Prob(H) (two-sided): 0.01 Kurtosis: 3.74

==========================================================================

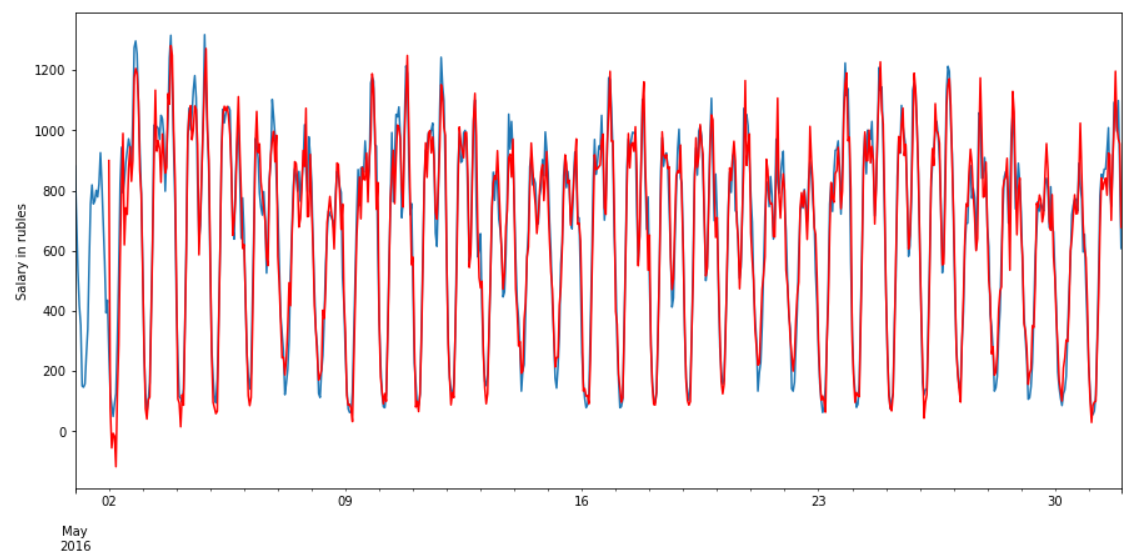
Now, let’s see a residuals plot and its autocorrelation and check Dickey – Fuller criteria:

*H0: time series is not static*

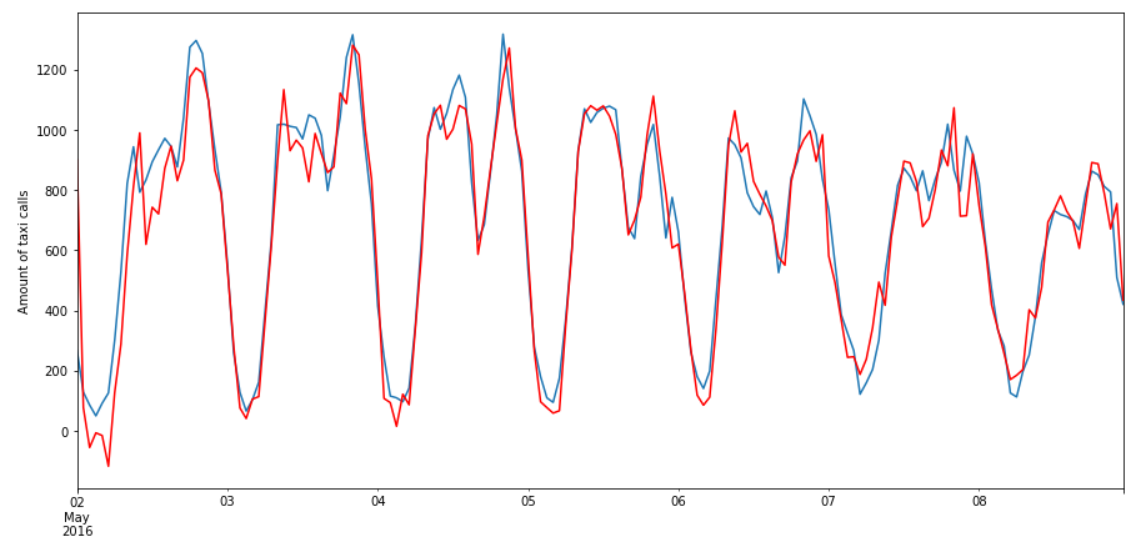
*H1: H0 is incorrect*



Considering criteria’s value and on the graphs, we can see that our residuals’ plot is static and unbiased. That means that ARIMA model describes our residuals’ data well. Now, let’s compare our linear regression model with ARIMA model fitted on its residuals with real data:

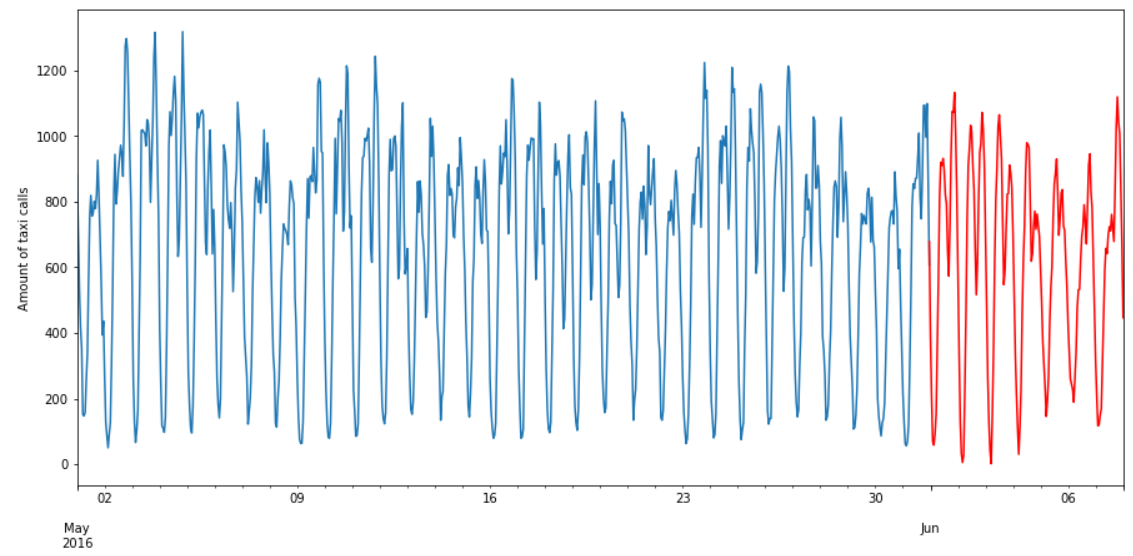


And some random section in scale:



We can see that our model predicts generally well, but the series is a bit sharper than real data.

Now, let’s build a prediction on 1 week in the future:



Generally, we made a good model that is able to predict our time series for a separated cell, but it’s hard to generalize that approach because of big amount of data (5 times more from January 2016 and 102 time more because of amount of taken regions). It took about 7 hours to fit simplified ARIMA and if we will stick that approach it will be about of 7 \* 5 \* 102 = 3520 hours of computation. In the next chapter we will try to change the approach to make it more general.

**3 Generalization of ARIMA’s approach**

**3.1 Splitting regions on clusters**

We can split all our regions of *n* clusters to tune our ARIMA’s parameters only on an exponential region and generalize that parameters on every region in a cluster. Let’s do splitting.

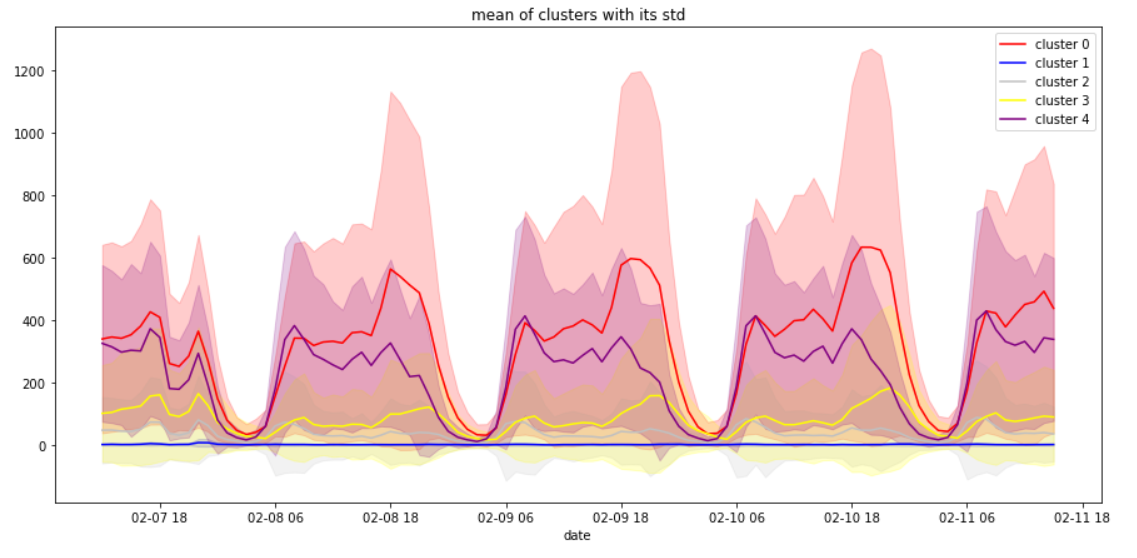
For clustering we will use k-means algorithm with *k=5* and then we will estimate the quality of splitting graphically.

*(for more info about K-mean see wiki:* [*https://en.wikipedia.org/wiki/K-means\_clustering*](https://en.wikipedia.org/wiki/K-means_clustering)*).*

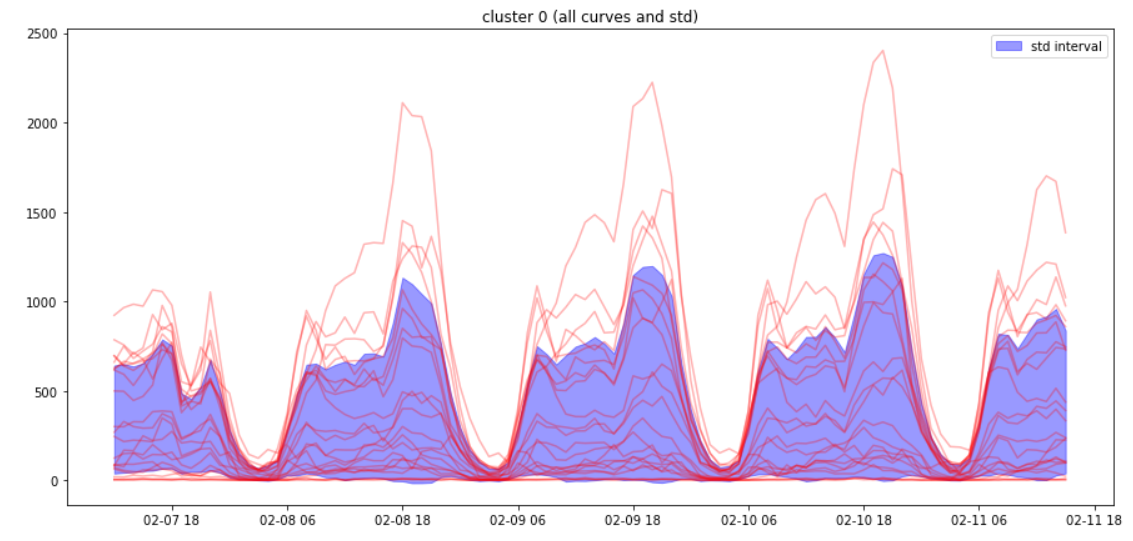
That’s how the time series for every region looks like on a random section after splitting on clusters (different color = different cluster):

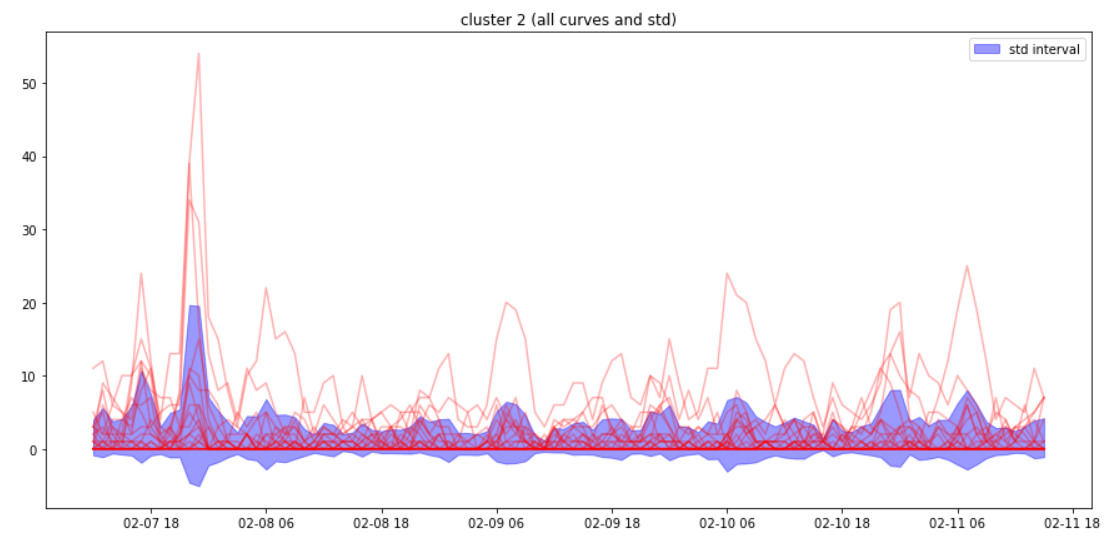


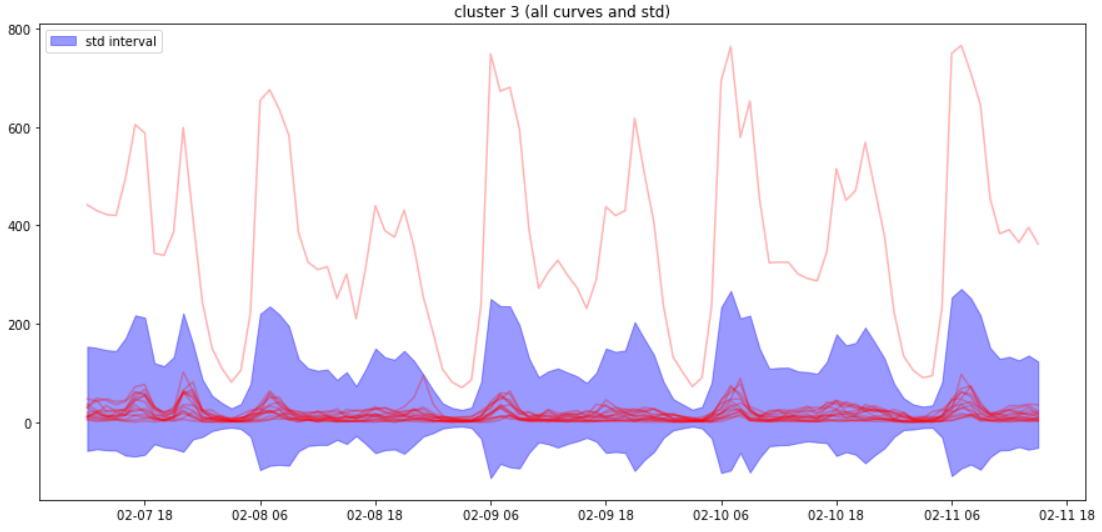
Now, let’s look at the mean time series of every cluster and its standard deviation:

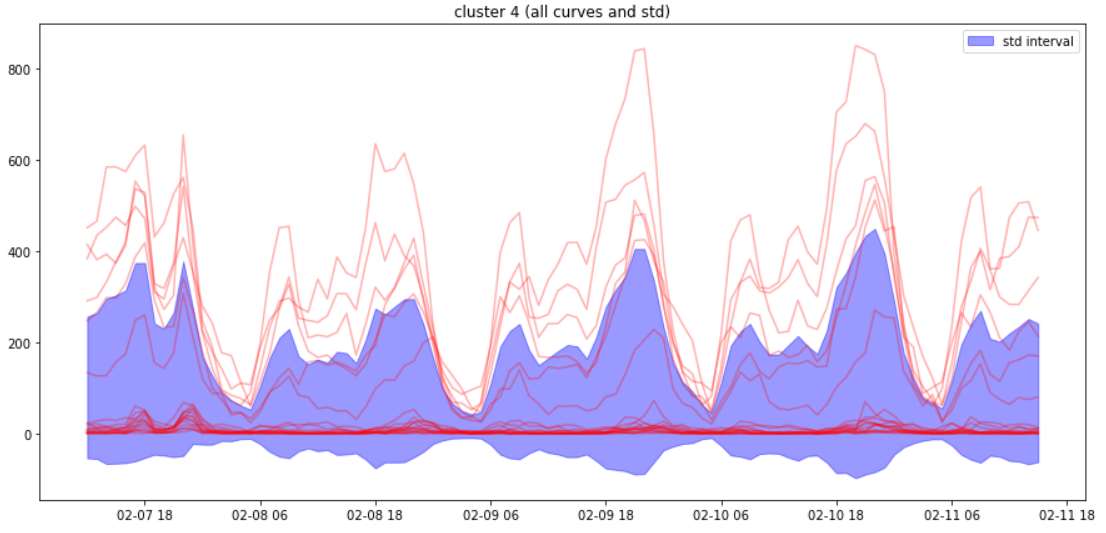


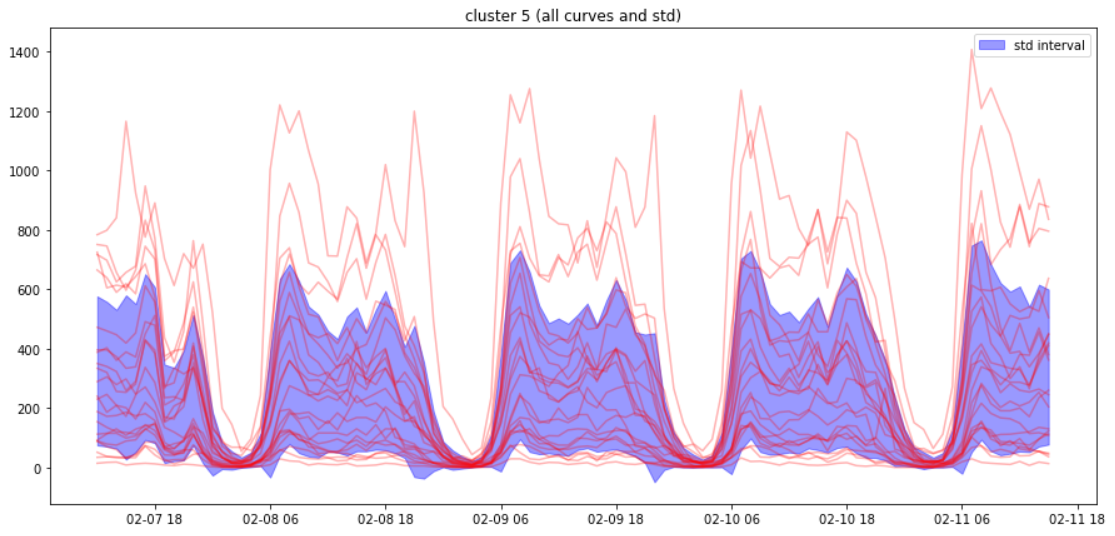
From graph, we can see that generally mean time series of every cluster behave differently. Now, let’s estimate standard deviation and a time series of every region to see how many regions in cluster are out of its standard deviation.









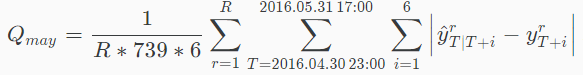


As we can see from graphs above, some clusters contain anomalies regions. To increase the accuracy of predictions we can use different number of clusters, different algorithms (hierarchical clustering etc.), but now, let’s stick to the clusters we’ve got. To avoid anomaly in parameters’ tuning, we will use scaled to [0, -1] section mean value of all the series in a cluster. That approach is comparable to errors of 1st and 2nd type in statistics.

*(for more info about errors of 1st and 2nd type see wiki:* [*https://en.wikipedia.org/wiki/Type\_I\_and\_type\_II\_errors*](https://en.wikipedia.org/wiki/Type_I_and_type_II_errors)*).*

**3.2 ARIMA’s parameters tuning on every cluster**

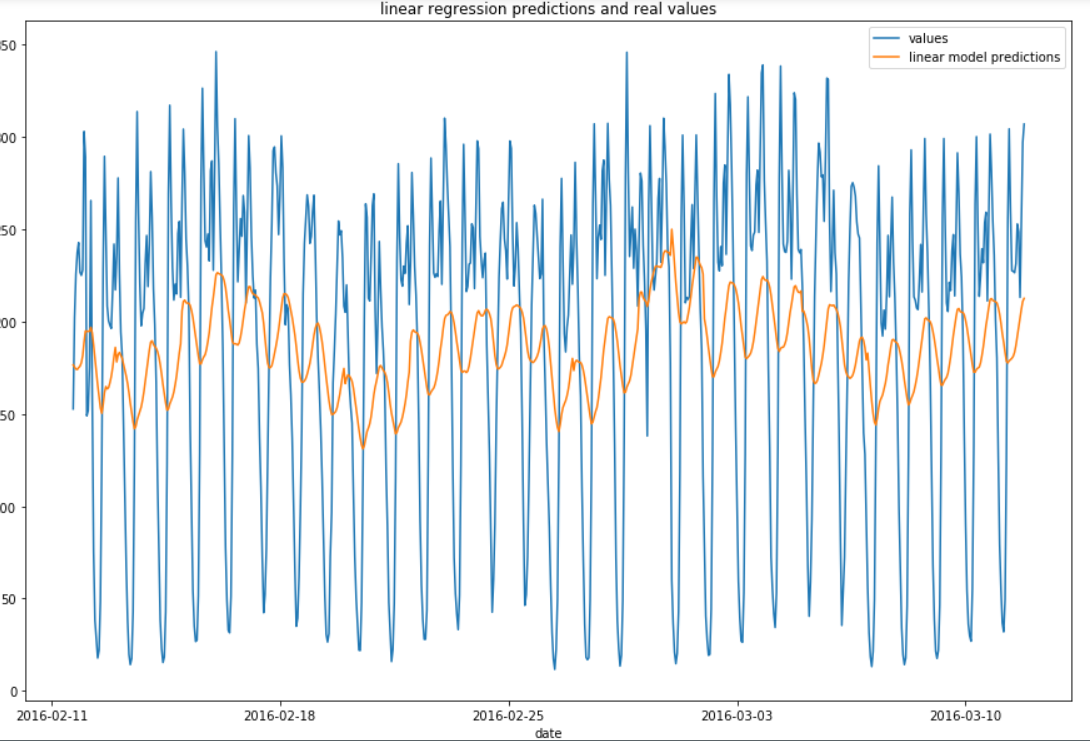
Now, let’s take all the data from January of 2016 and until April of 2016 and fit our linear regression and ARIMA models on this data and count accuracy of predictions on May 2016 using formula below:



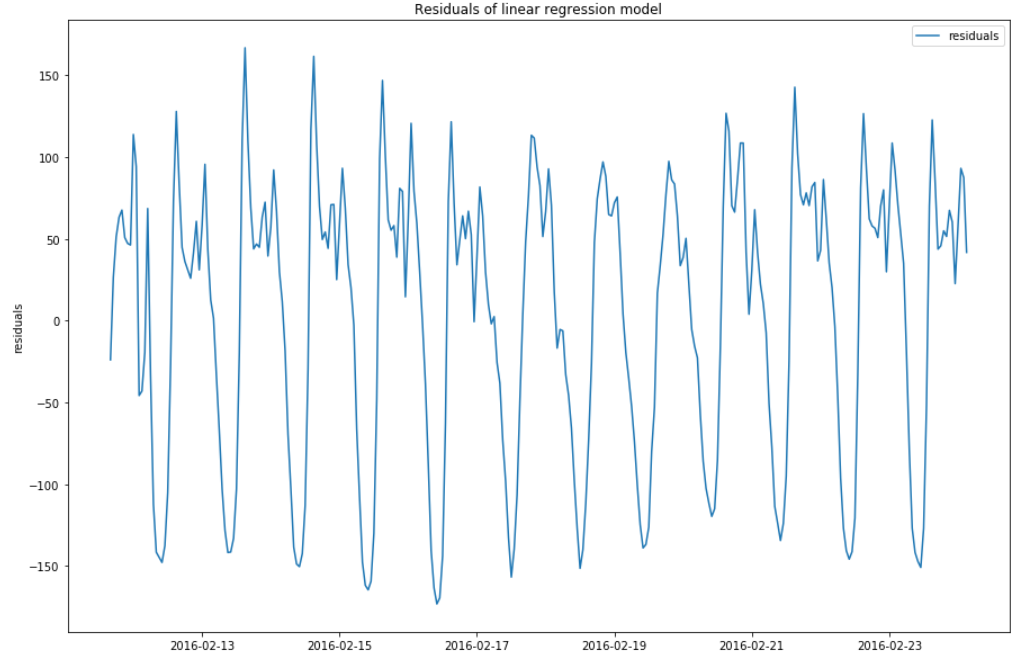
(equals mean absolute error for every moments’ prediction for 6 hours in the future).

In these linear regression’ features we will use the same formula for Fourier series, but with a month seasoning (replace 168 with 720 – number of hours in 1 month (30 days)). We hope that ARIMA will predict week’s seasoning because of the same approach in its algorithm.

That’s how the graph of only linear regression fitted on our data looks like for a mean time series in the first cluster:



And its residuals:



Now all we need to do is the same algorithm as in the previous chapter of ARIMA’s tunings and find out the best values of *p, q, P, Q* to use them in further predictions. See the results below.

The table of best parameters:

|  |  |
| --- | --- |
| Cluster | Best parameters (q, Q, p, P) |
| Cluster 1 | (2, 7, 2, 1) |
| Cluster 2 | (2, 2, 2, 2) |
| Cluster 3 | (2, 2, 2, 0) |
| Cluster 4 | (2, 2, 2, 2) |
| Cluster 5 | (3, 1, 0, 2) |

So, now we can make predictions on May and count *Qmay*.

After building models on every region (22 hours of calculating time) we’ve got the next value May:

Q = 20.65832

And on June:

Q = 31.11283

In this chapter we generalized our approach and made our time of calculation faster in ~170 times in compare with tuning ARIMA model on each region calculation.

**4 Regression approach**

In this chapter, let’s try to generalize our approach more by using only regression models. For that we can take not only linear regression model, but something more complex, for example random forest, boosting, neural networks etc. In this project, we will use boosting on trees algorithm implementation by Yandex – CatBoost framework (for more info visit <https://github.com/catboost/catboost>).

**4.1 New features generation**

Before we start fitting our model, let’s generate more features for regression:

* Geographic features like amount of taxi calls from the neighbors’ regions, amount of taxi rides in the neighbors’ cells
* Time features like all the holidays from January to June of 2016 (like we did in the 2nd chapter)
* Fourier series for month, week, day seasonings
* Moving average
* Hour, day, month
* Sum and mean of taxi calls for different time intervals
* Cluster
* ARIMA model predictions

We can generate more features (like time of plane arrivals for cells with airports) or timetable of concerts on Broadway and many others to increase quality of predictions, but now, let’s stick to the ones we have.

**4.2 Regression with boosting**

We will use one of the most efficient algorithms in machine learning – boosting trees *(for more info visit* [*https://en.wikipedia.org/wiki/Gradient\_boosting*](https://en.wikipedia.org/wiki/Gradient_boosting)*).*

We will use 6 different models fitted on 1, 2, 3, 4, 5, 6 hours predictions in the future from current moment.

Now, let’s fit our regression model on all the data until May and see *Q* value:

prediction on 1 hour MAE: 14.777137312962692

prediction on 2 hour MAE: 17.226430893009695

prediction on 3 hour MAE: 18.10609282176826

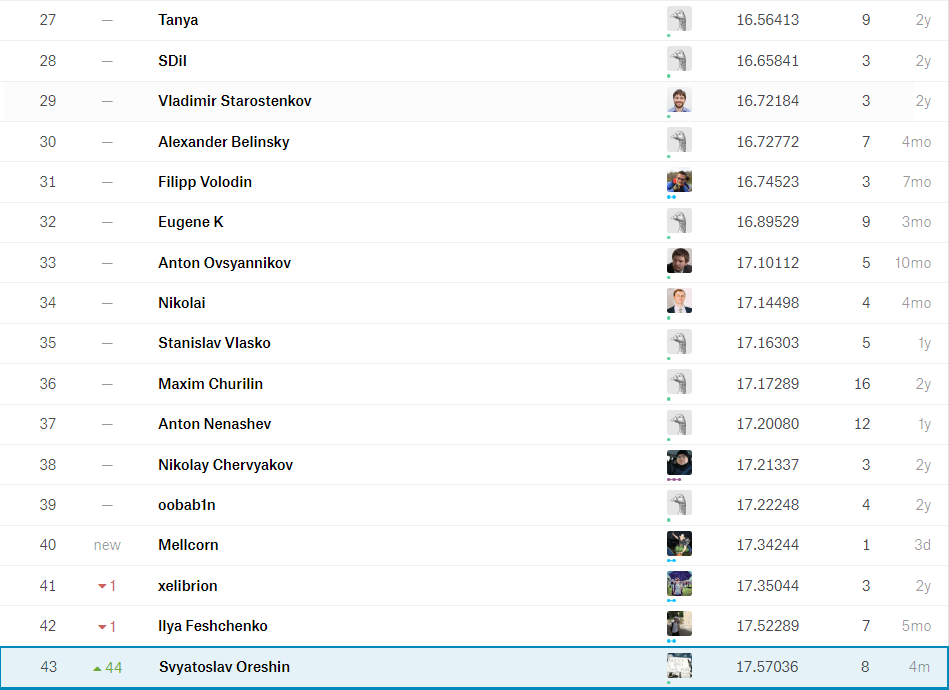
prediction on 4 hour MAE: 18.476919211104367

prediction on 5 hour MAE: 18.876197261852454

prediction on 6 hour MAE: 18.984902355928117

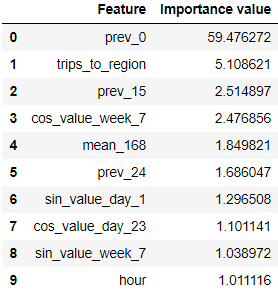
Q = 17.7412

We can see that the accuracy of predictions has increased, so let’s see how our model predicts on June 2016:



We can see that the result is pretty good, and the computational time is much better (fitting time: 4.5 hours). We still can increase quality by tuning hyper parameters of CatBoostRegressor and generating more features.

Now, let’s look at top-10 feature importance values:



From table, we can see that the most important features for our models are previous values and number of trips to the region.

In the next chapter we will analyze the results we’ve got and make conclusions.

**5 Results and conclusions**

For result visualization of June 2016 visit:

<http://nbviewer.jupyter.org/github/Aqice/NYC-taxi-time-series-predictions/blob/master/results%20and%20conclusions%20%28part%206%29/results.ipynb>

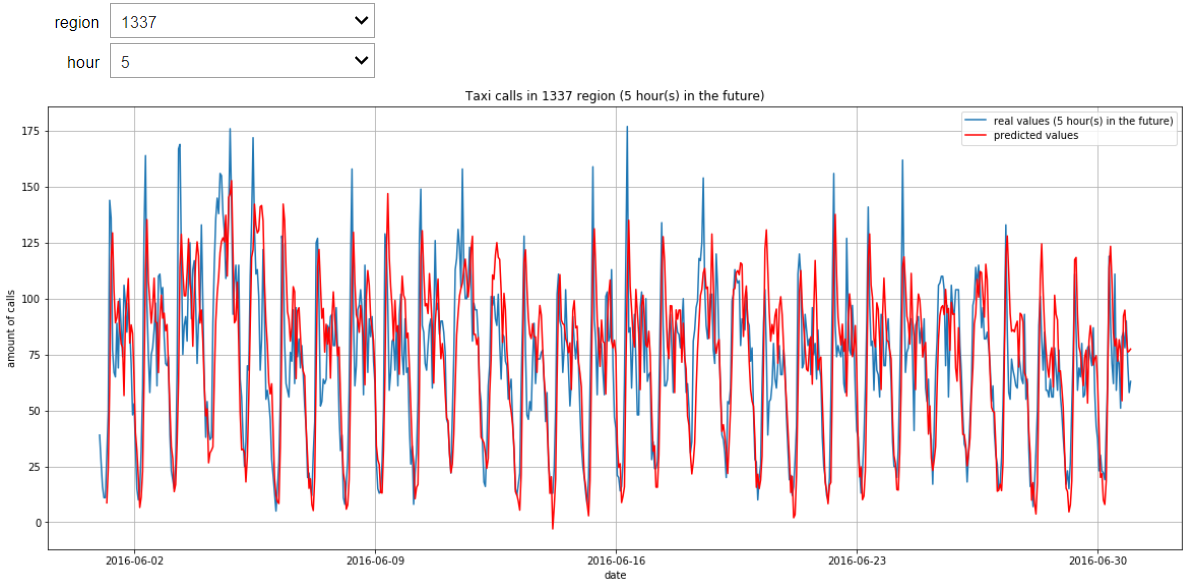
If you want to interact with results, you should download 6th part from github, install all libraries and run the Jupiter notebook.

**5.1 Widget description:**

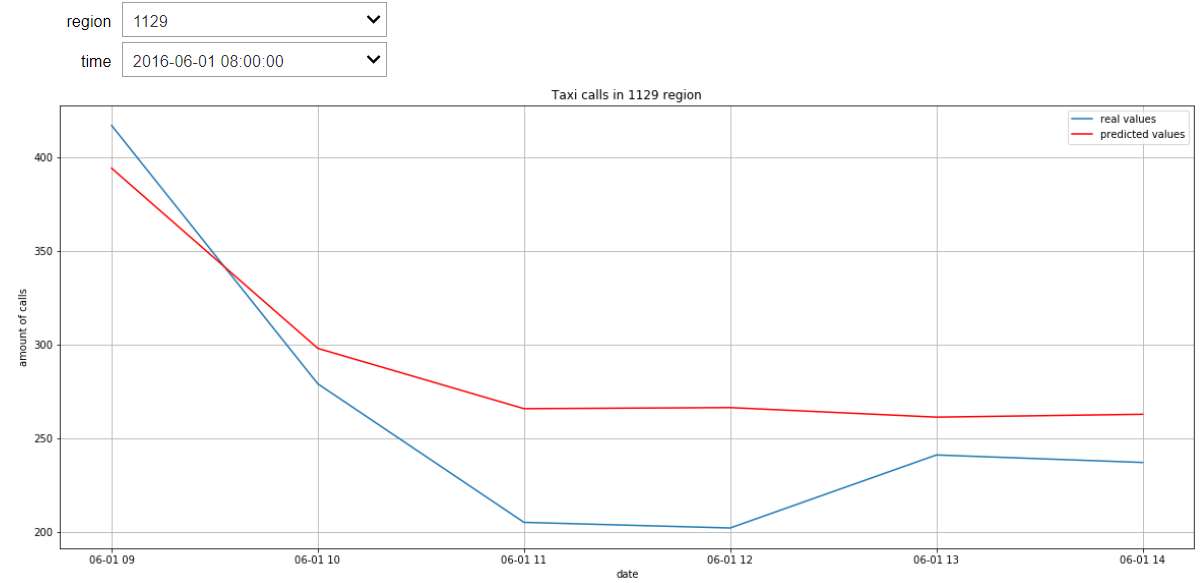
Predictions on static maps on 1 hours in the future for every moment in June 2016:



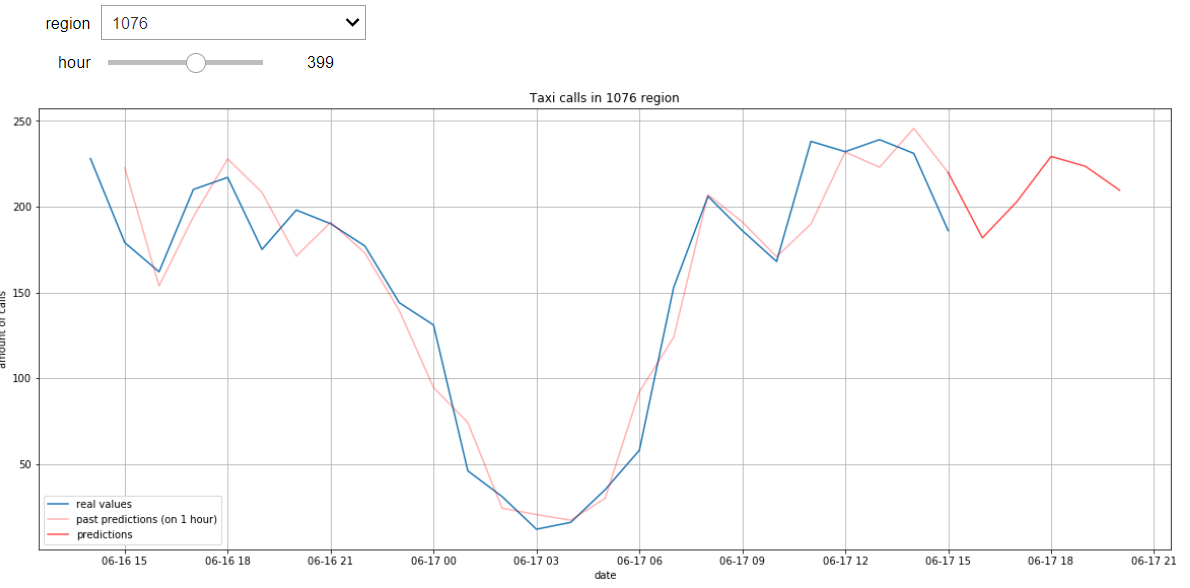
Predictions for every region on selected number of hours in the future:



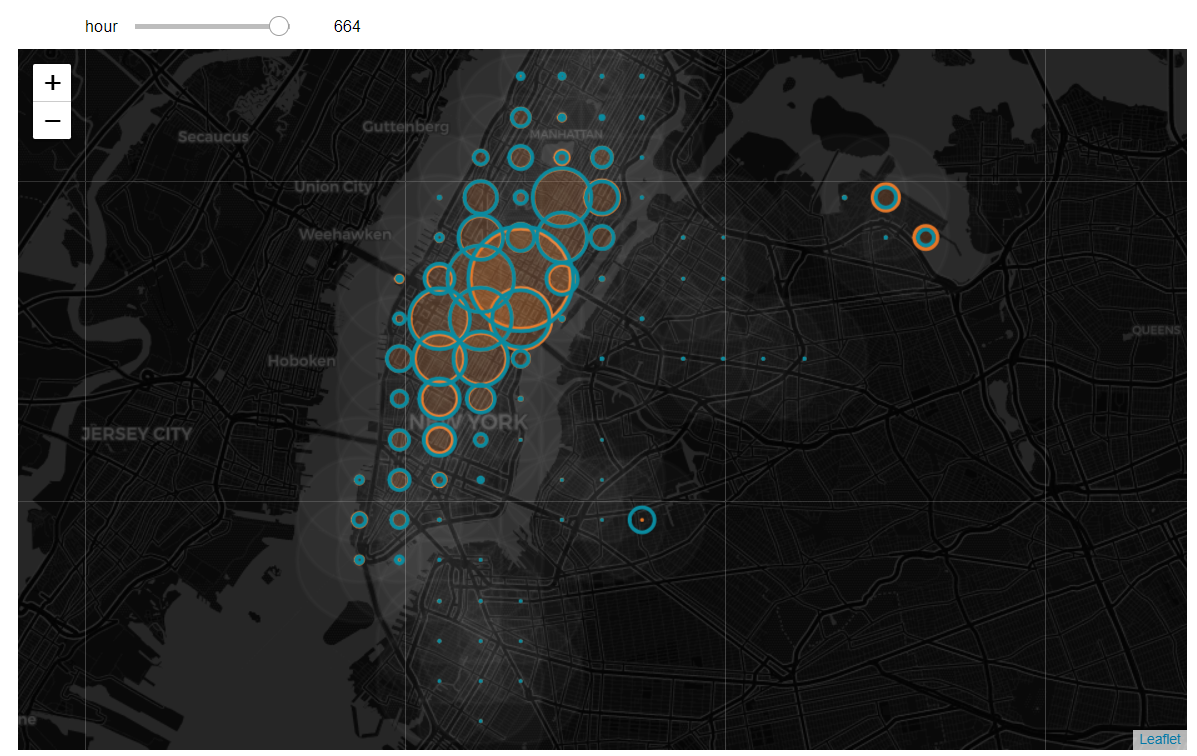
Predictions on 6 hours in the future from every hour in every region:



Previous predictions on 1 hour and real data with prediction on 6 hours in the future for every region:



Interactive map with blobs of real values and prediction for every hour:



For interactive map of fixed time see *map2.html*

**5.2 Conclusions**

In this project we’ve tried several approaches for multiple time series predictions: using ARIMA and complex regressive models. The approach with using regression is faster and shows better results, and it’s easier to make this approach general for bigger amount of data and regions and it takes less time to fit models.

The accuracy of predictions could be increased by:

* Using grid-search for hyper parameters’ tuning in our regression model
* Using other regressive models and (or) stacking them together
* Make hierarchical clustering and trying other clustering algorithms
* Generate more features
* Use more historical data

Further project development could be done in making online fitting for real-time prediction