Report on learning practice # 4

Stationarity of the processes

Performed by:

Performed by:

Grandilevskii Aleksei

J4133c

Dubinin Ivan

Sorokin Mikhail

J4132c

Saint-Petersburg

2021

Table of contents:

1. Substantiation of chosen sampling.

This Lab we used to have timestamps in our dataset. Dataset we used in the previous works hasn't them. But the specialized dataset from labs 1-3 is a processed squeeze from the raw data obtained using the Riot.API. (https://www.kaggle.com/gyejr95/league-of-legendslol-ranked-games-2020-ver1#challenger_match_V2.csv). The original dataset contains 7Gb of practically raw data from the API. We wrote our own script for parsing timestamps from raw data and already processed dataset.

```python
# Dataset initialization
path_to_file = '../Datasets/Challenger_Ranked_Games.csv'
df1 = pd.read_csv(path_to_file, engine='python')
df1.head(3)
```

Python

```python
# The original dataset is to big to download on GIT, 3,5Gb,
# the link is here: https://www.kaggle.com/gyejr95/league-of-legendslol-ranked-games-2020-ver1
# You can download and test the code by your own if there are any DS updated

path_to_file = '/Users/zer0deck/Documents/Документы/Магистратура/archive/match_data_version1.csv'
df2 = pd.read_csv(path_to_file, engine='python')
df2['gameId'].astype(int)
df2.head(3)
```

Python

```python
df = df1.merge(df2, on = ['gameId'])
df.head(7)
```

Python

```python
# Here we delete unused columns and prepare timestamps for Lab4:
# Column gameCreation has 13 digit value, which is the 10 digit epoch time format + 3 digit miliseconds value
# This is written in official Riot API Documentation https://riot-api-libraries.readthedocs.io/en/latest/
stamp = []
for i in range(0, len(df['gameId'])):
        ts = int((df['gameCreation'][i])/1000)
        stamp.append(datetime.utcfromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S'))
df['timeStamp'] = stamp
df = df[['gameId', 'gameCreation', 'timeStamp', 'gameDuraton', 'blueWins', 'blueFirstBlood',
        'blueFirstTower', 'blueFirstBaron', 'blueFirstDragon',
        'blueFirstInhibitor', 'blueDragonKills', 'blueBaronKills',
        'blueTowerKills', 'blueInhibitorKills', 'blueWardPlaced',
        'blueWardkills', 'blueKills', 'blueDeath', 'blueAssist',
        'blueChampionDamageDealt', 'blueTotalGold', 'blueTotalMinionKills',
        'blueTotalLevel', 'blueAvgLevel', 'blueJungleMinionKills',
        'blueKillingSpree', 'blueTotalHeal', 'blueObjectDamageDealt', 'redWins',
        'redFirstBlood', 'redFirstTower', 'redFirstBaron', 'redFirstDragon',
        'redFirstInhibitor', 'redDragonKills', 'redBaronKills', 'redTowerKills',
        'redInhibitorKills', 'redWardPlaced', 'redWardkills', 'redKills',
        'redDeath', 'redAssist', 'redChampionDamageDealt', 'redTotalGold',
        'redTotalMinionKills', 'redTotalLevel', 'redAvgLevel',
        'redJungleMinionKills', 'redKillingSpree', 'redTotalHeal',
        'redObjectDamageDealt']]
df.to_csv('timeStamped_Challenger_Ranked_Games.csv')
```

Python

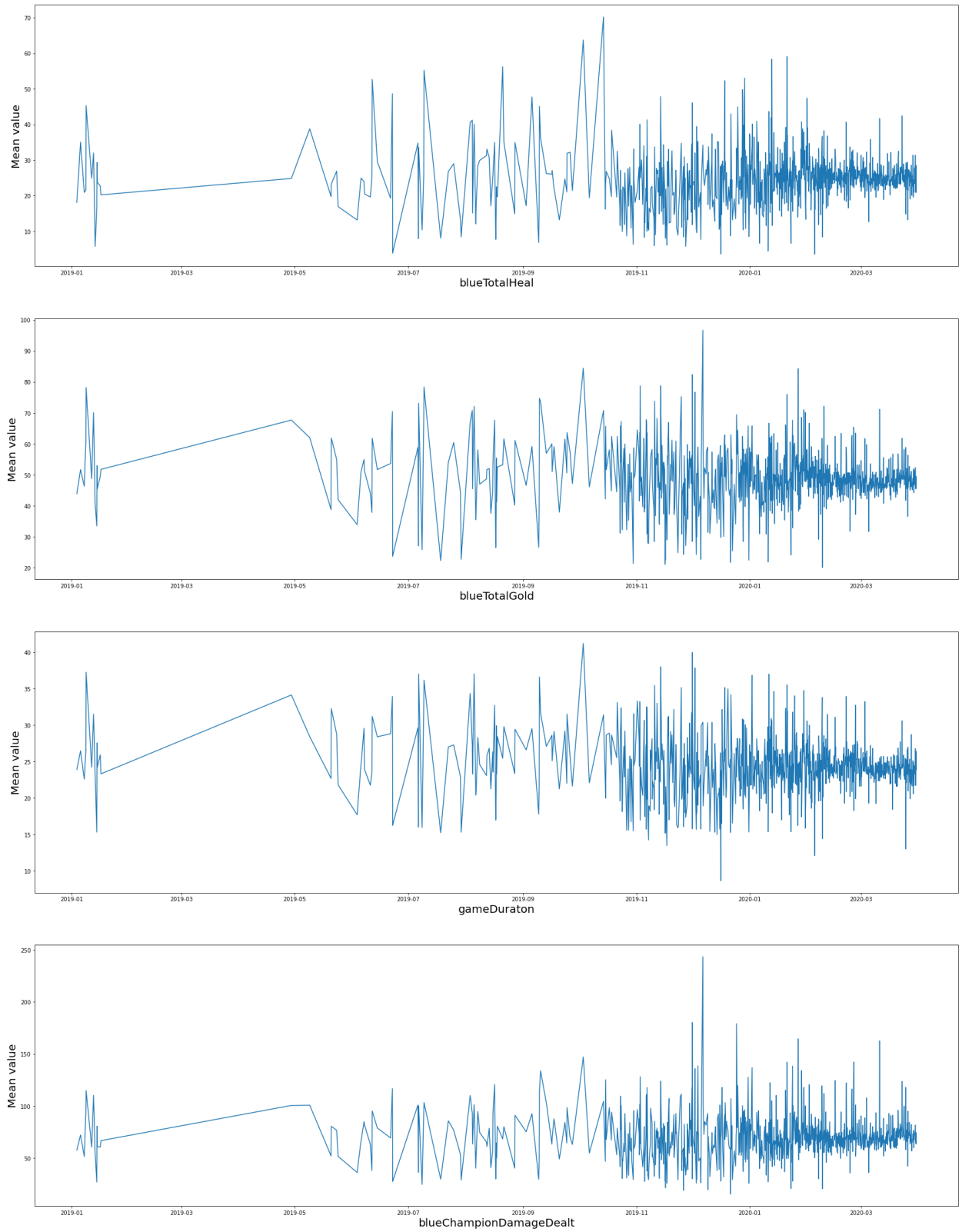
Pic.1. Timestamps parsing.

As You can see in the code timestamps are timestamps are represented as Unix-ts, so our script also converts them into default Python 3 timestamps.

	timeStamp	blueTotalHeal	blueTotalGold	gameDuraton	blueChampionDamageDealt
0	2019-01-03 18:00:00	18.092000	43.865000	23.900000	57.547000
1	2019-01-05 18:00:00	35.028667	51.676000	26.477778	72.211333
2	2019-01-07 18:00:00	20.913333	46.317333	22.594444	51.522333
3	2019-01-08 15:00:00	21.567000	60.554000	26.416667	94.014000
4	2019-01-08 18:00:00	45.280000	78.178000	37.250000	114.909000
5	2019-01-11 18:00:00	24.877500	48.762000	24.225000	60.525500
6	2019-01-12 18:00:00	31.983000	70.050000	31.500000	110.272000
7	2019-01-13 15:00:00	5.726000	40.619000	23.233333	54.175000
8	2019-01-14 12:00:00	16.576000	33.484000	15.333333	26.938000
9	2019-01-14 15:00:00	29.339000	52.979000	27.550000	80.685000

Pic.2. Working dataset for lab 4 visualization.

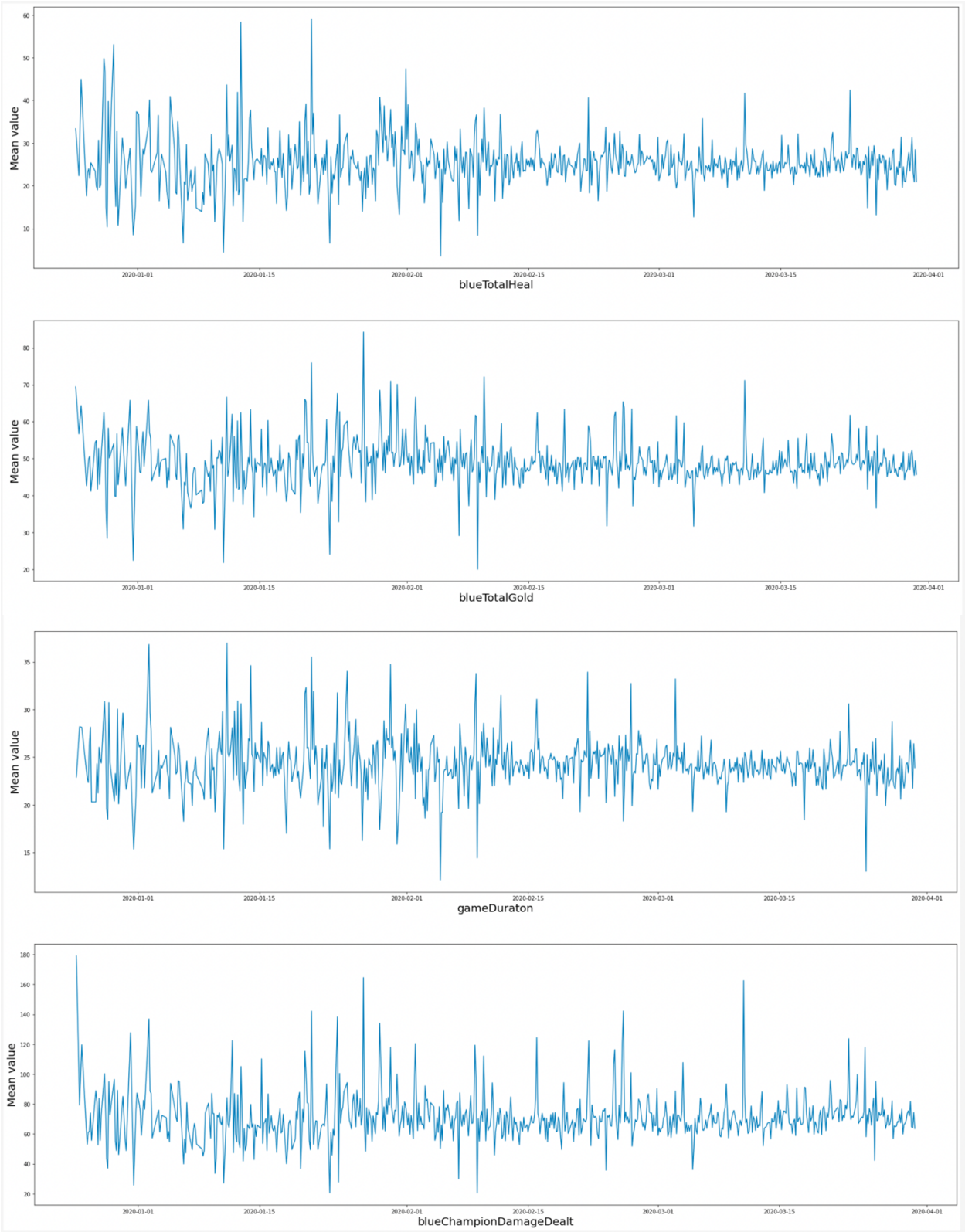
For this task were chosen these target variables: *blueTotalHeal, blueTotalGold* and following predictors: *gameDuraton, blueChampionDamageDealt. timeStamp* variable contains timestamps for our data. Time series for all chosen variables aggregated by 3 hours with mean value are shown on the picture below.



Pic.2 – Initial time series for all variables

2. Stationary analysis.

As one can easily see gained time series represent non-stationary processes. To make them more stationary space part till 2020-01-01 was removed. The gained time series are presented on picture 3. All this time series passed Augmented Dickey-Fuller Test and can be considered stationary. The Results of Dickey-Fuller Test are presented for each variable on the table below.



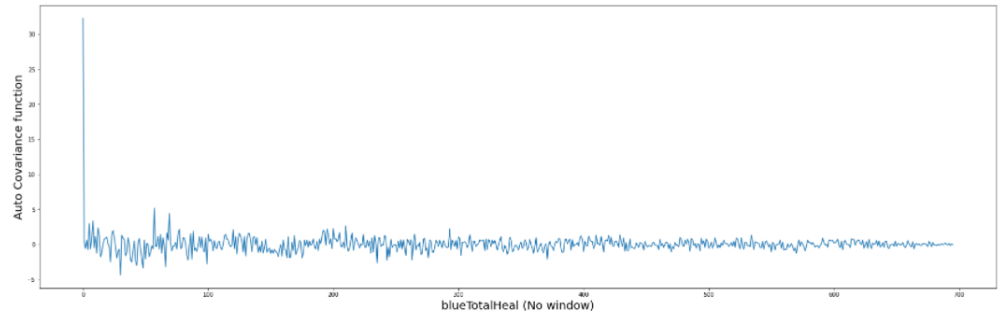
Pic.3. Gained stationary process.

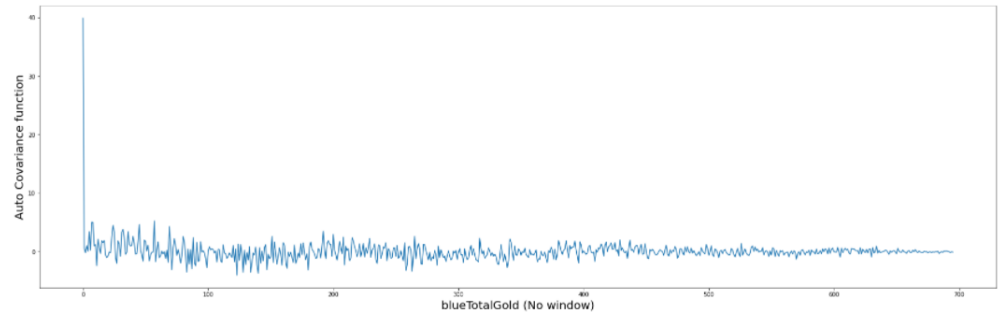
column name	statistic value	p-value	stat/non-stat
blueTotalHeal	-7.8409336531706515	1.6662415874867163e-10	STATIONARY
blueTotalGold	-6.731542011878182	5.6679759422940115e-08	STATIONARY
gameDuraton	-8.429682382616507	7.280465385711972e-12	STATIONARY
blueChampionDamageDealt	-7.442436027619189	1.3822419845723706e-09	STATIONARY

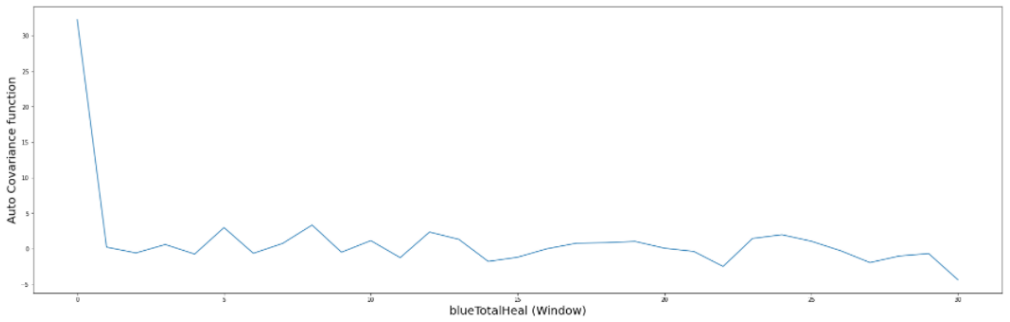
Pic.4. ACF

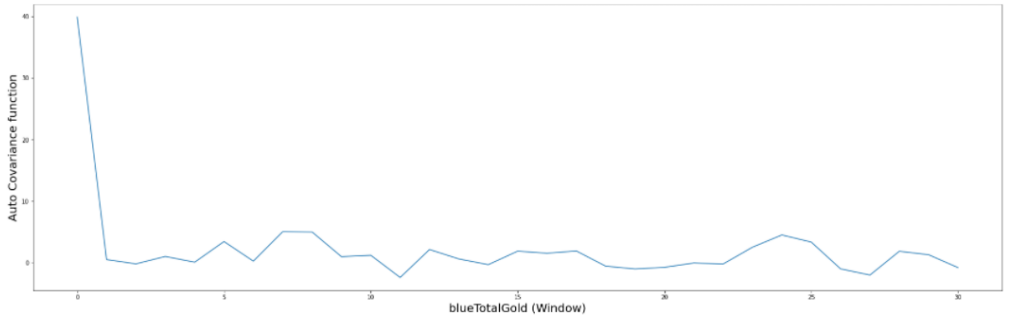
3. Covariance or correlation function analysis.

Picture 4 represents Auto-covariance function for target variables. On picture 5 Auto-covariance function gained using window of 30 items is shown. These functions look like their processes are stationary.

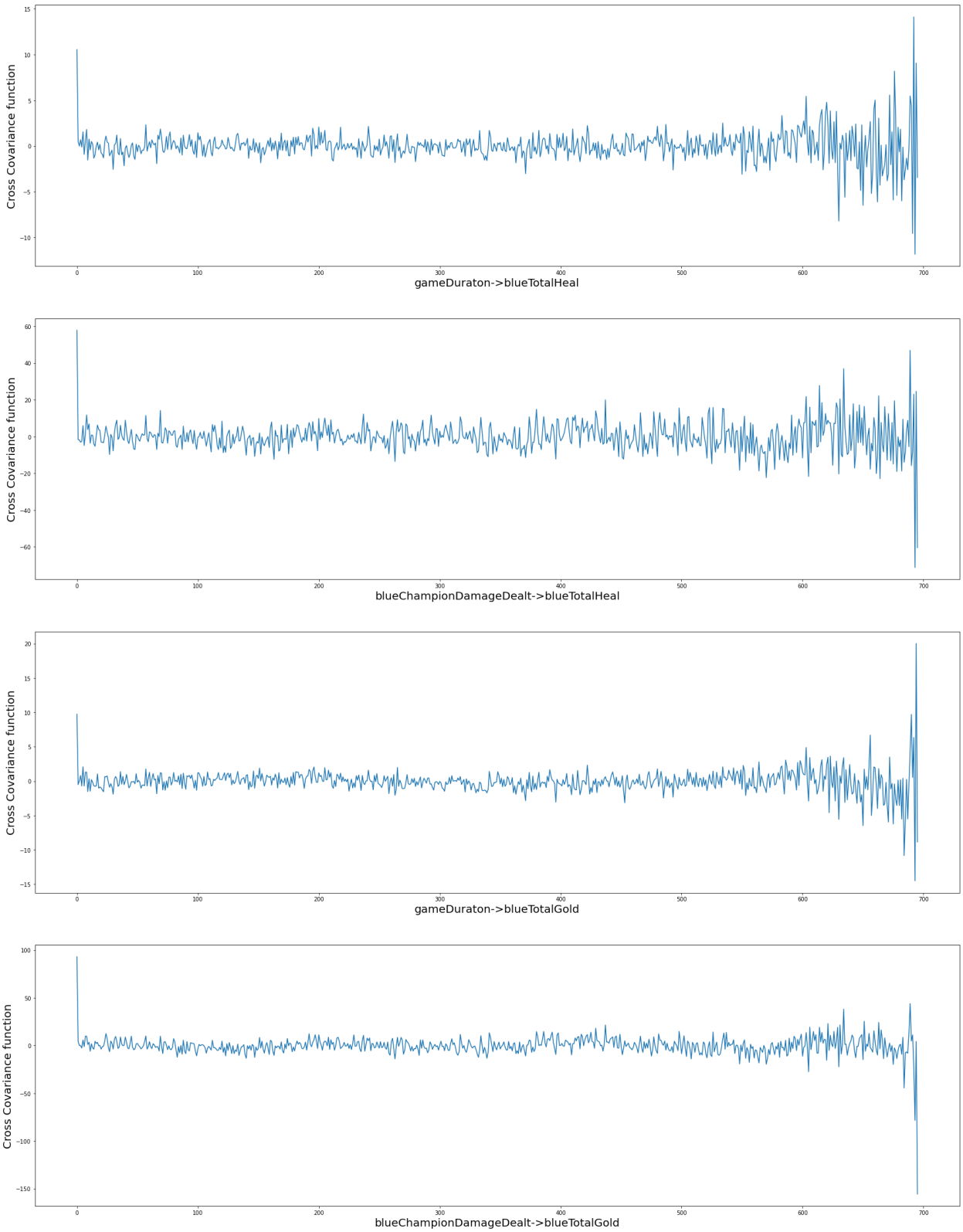






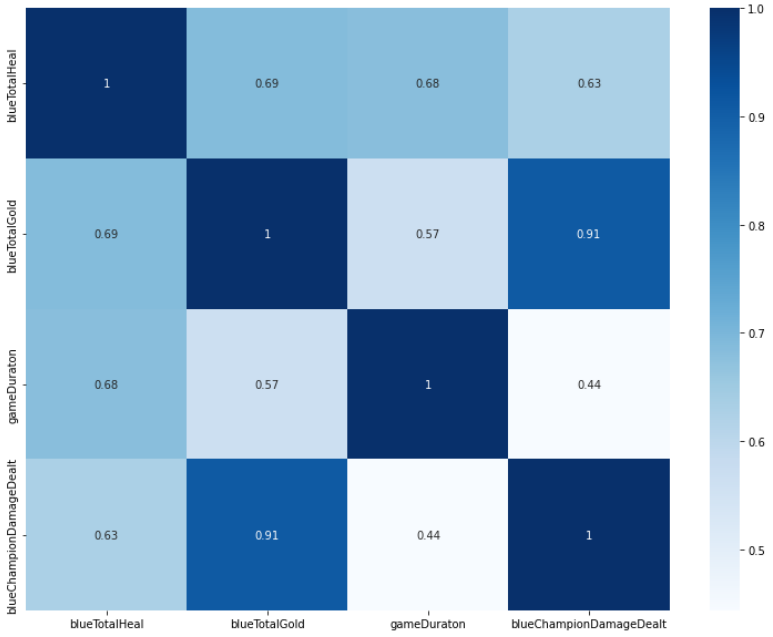


Picture 6 represents cross-covariance (mutual correlation) function between each target and predictor variable



Pic.6. Cross-covariance functions.

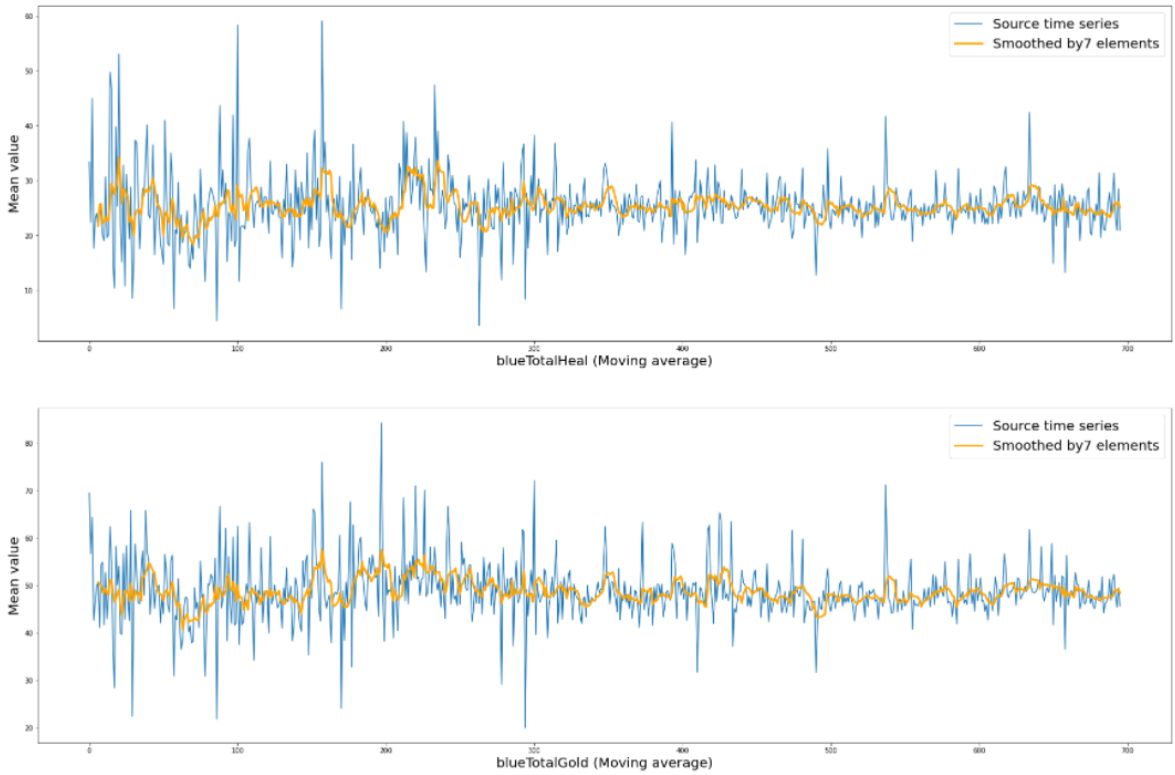
Cross correlation matrix for variables is presented on picture 7. As one can see, target and predictor variables are highly correlated, especially BlueTotalGold is correlated with blueChampionDamageDealt.



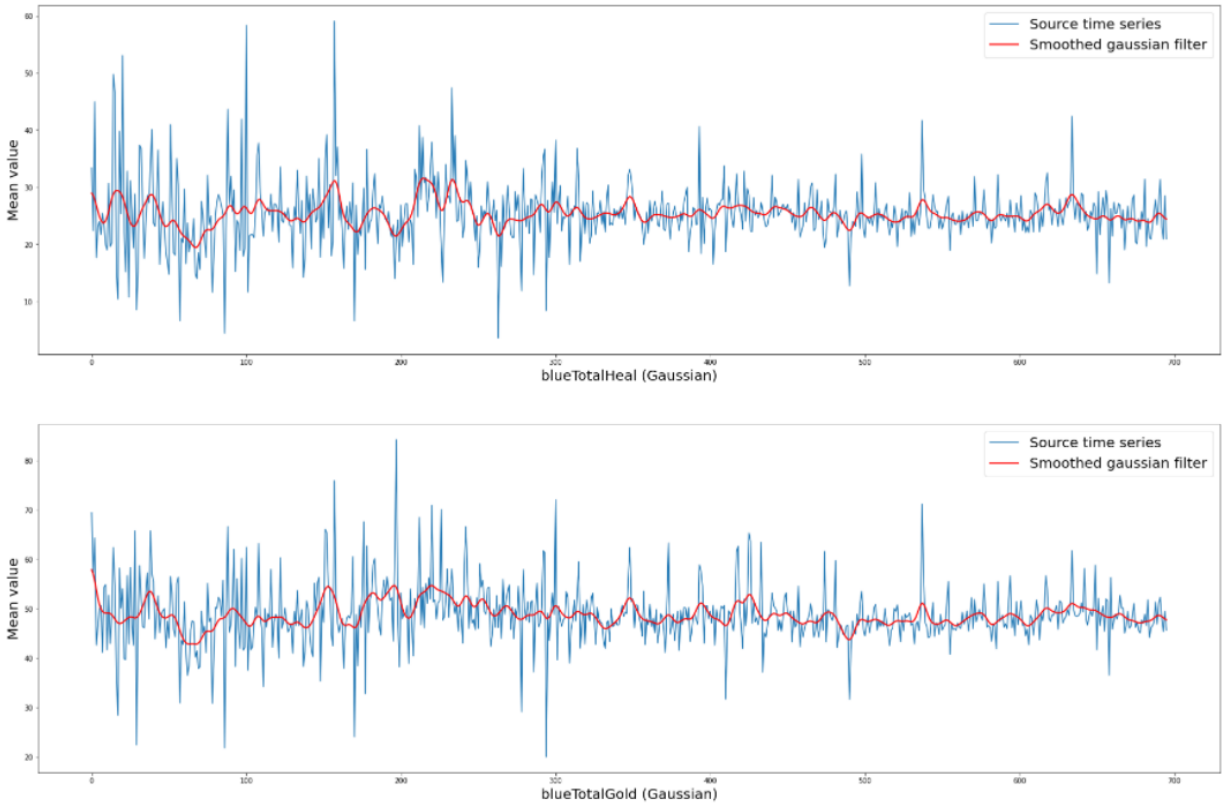
Pic.7. Cross correlation matrix.

4. Noise filtration.

For filtering high frequencies Moving average filter (Pic.8) and Gaussian filter (Pic.9) from FEDOT framework were used.



Pic.8. Using Moving average filter.

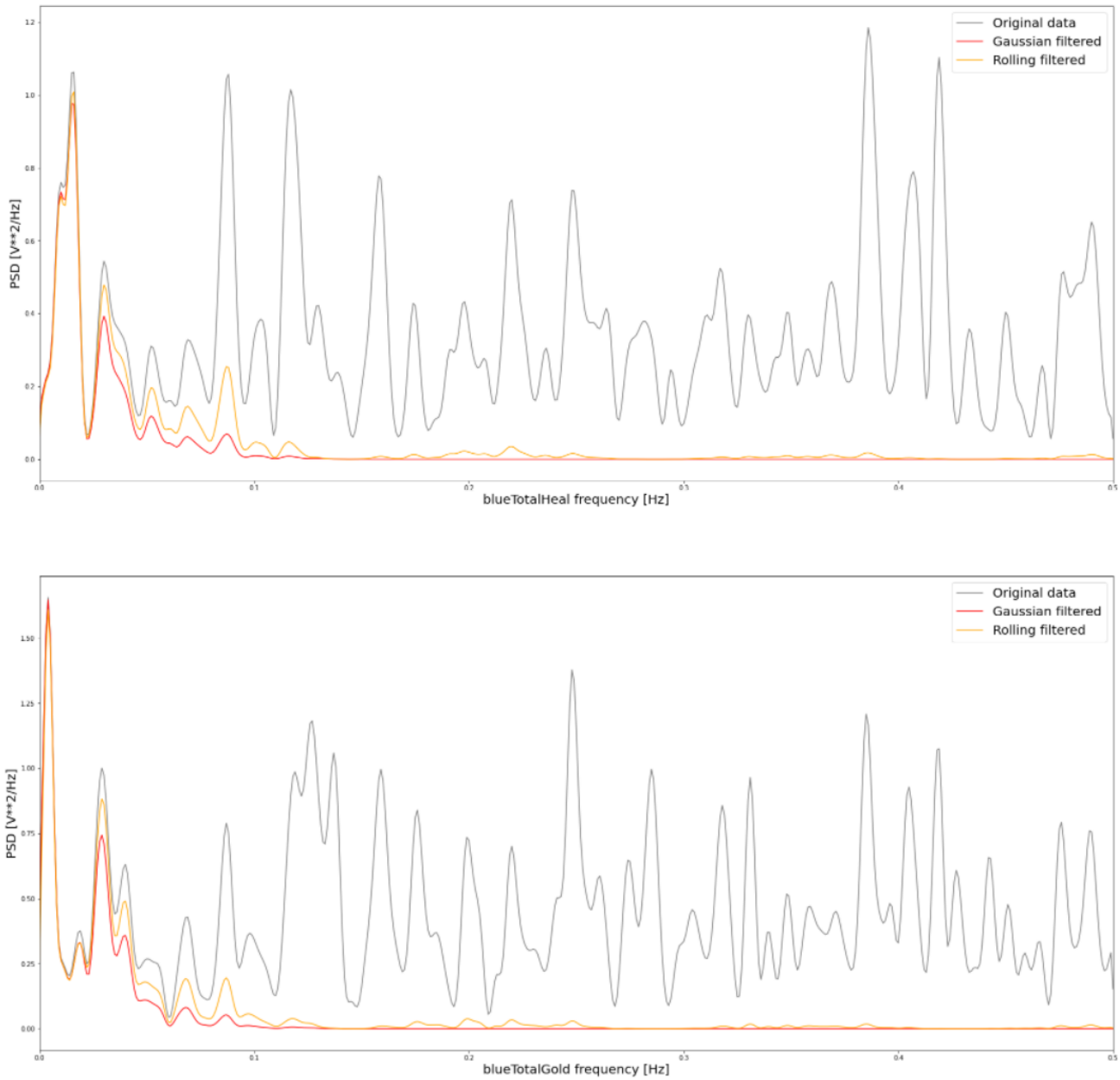


Pic.9. Using Gaussian filter.

5. Estimation of spectral density function.

Spectral density using was estimated Welch's method from scipy.signal module. The results for both non-filtered and filtered time series are presented on picture 10. There are two plots for each target variable. Two different filters are shown using different colors: orange for Moving average and red for Gaussian filter.

As one can see both filters removed high frequencies from spectral density function.

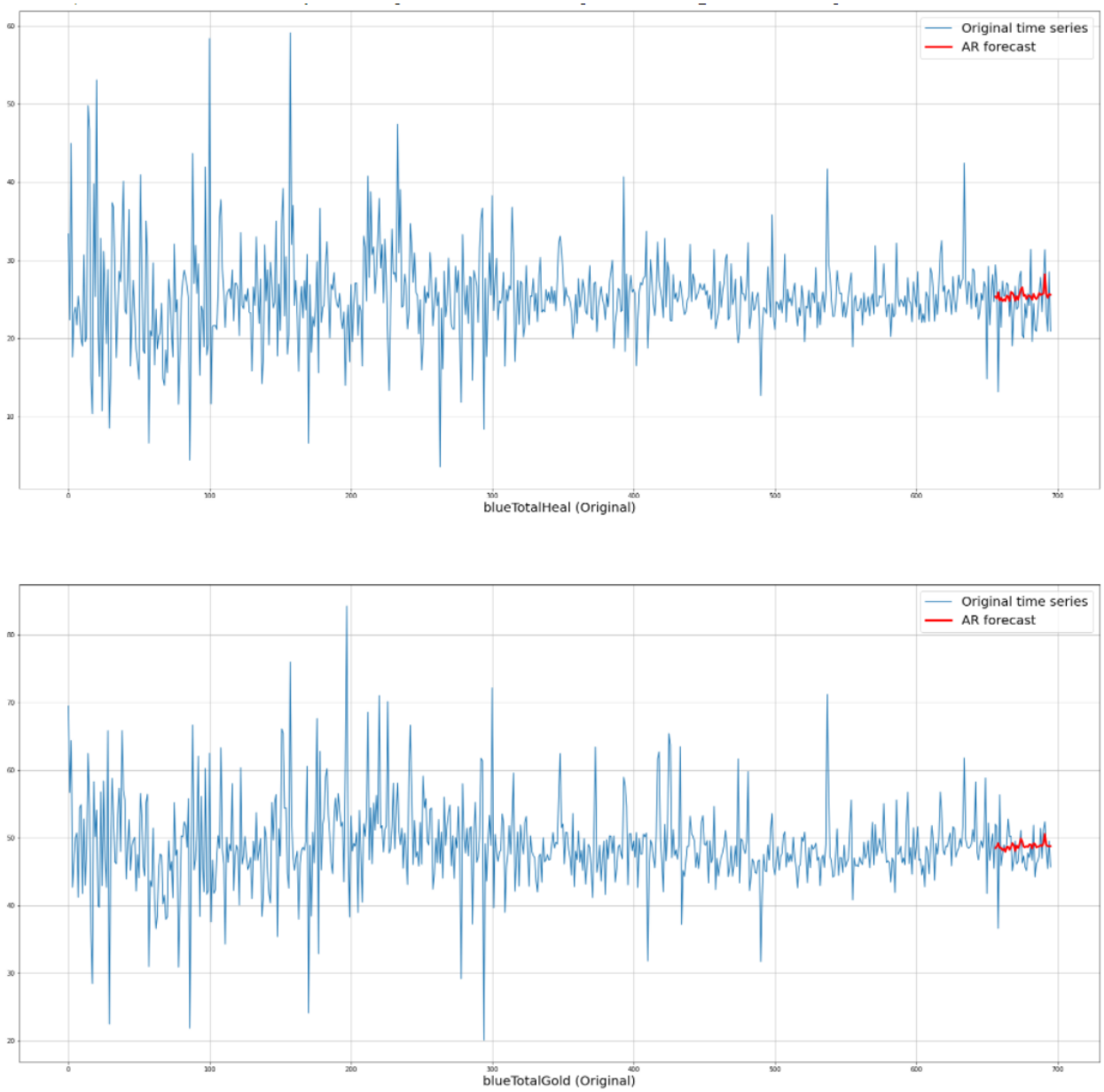


Pic.10. Spectral density functions.

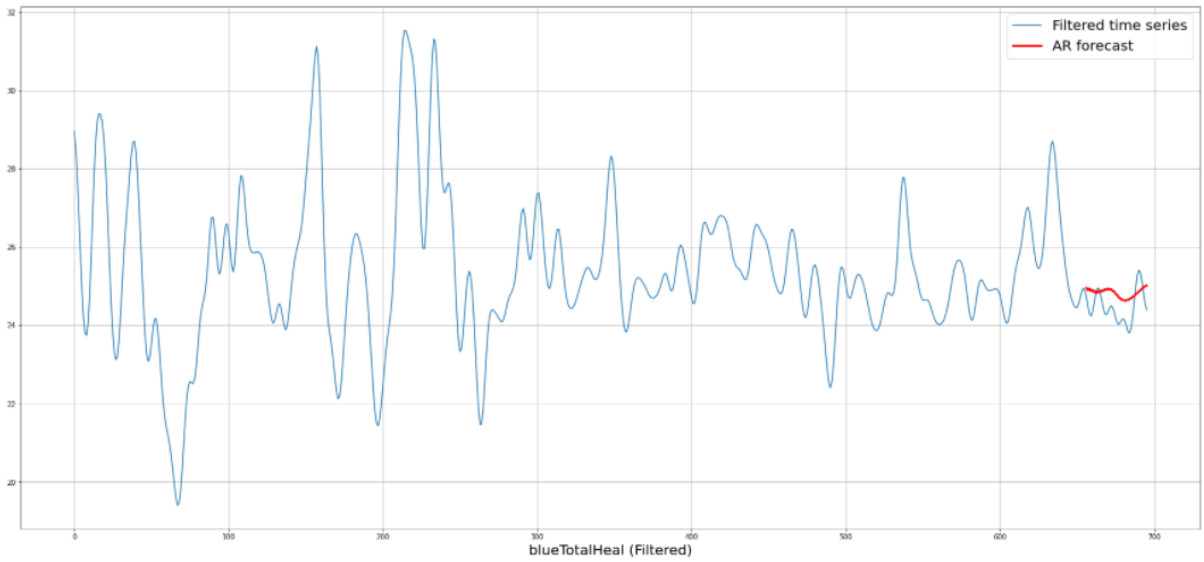
6. Auto-regression model.

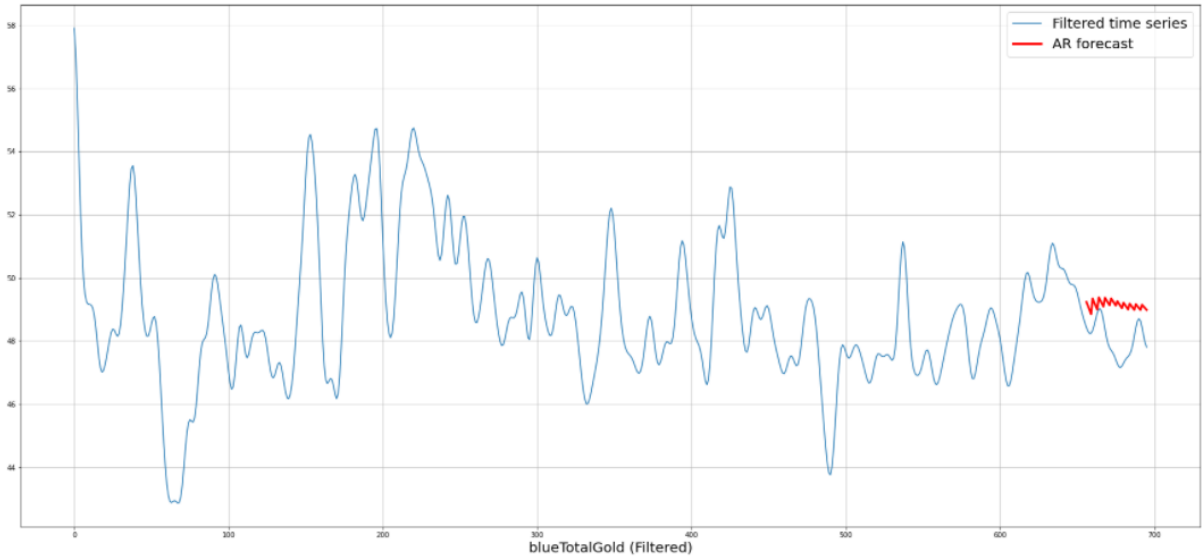
For building Auto-regression model AR Pipeline from FEDOT framework was used. The generated auto-regression model was used to forecast the future values of time series. All in all there were built four AR models – for both filtered and non-filtered time series of both target variables. The result of forecasting for non-filtered time series are shown on picture 11, for filtered data – on picture 12.

As one can see the generated forecasts are not accurate.



Pic.11. AR model forecasting non-filtered time series.





Pic.12. AR model forecasting filtered time series.

Sourcecode:

- The full repository with all the labs: https://github.com/vandosik/M-M-MSA
- The repo with Datasets and additional used Data info: https://github.com/vandosik/M-M-MSA/tree/master/Datasets
- The Lab 4 ipynb file: https://github.com/vandosik/M-M-MSA/blob/master/Lab_4/lab_4.ipynb

We recommend to use the first link because our GitHub project has README file with similar links and instructions which is really easy to use.

Instruction

This is M&M MSA group 19 repo.

Choose Lab 1 - 4 folder to get access to relevant materials

1. Lab_1. Analysis of univariate random variables
2. Lab_2.
3. Lab_3.
4. Lab_4.

> Inside each folder you can find the list of files include lab_XXX.ipynb file, lab_XXX_task.txt and lab_XXX_task.pptx, report.docx and **README file which is a copy of the Markdown github report**

Dataset files

All used datasets are publicated in Datasets folder

Participants of the project

- Ivan Dubinin: vandosik
- Alexey Grandilevskii: zer0deck
- Mikhail Sorokin: Mikha1lSorokin