Predicting Brazilian stock market index using other global market indexes – overnight market oscillation

Capstone Project for Advance Data Science IBM

Stakeholders Introduction

Capstone Project for Advance Data Science IBM

The use case

The main idea

After Brazilian stock market negotiations are closed, NY stock market is still running, and Asian stock market will operate during all night in Brazil, while is day in Asia.

So, there is overnight information from abroad that could help to predict the Brazilian close stock market index for the following day, that have not already started. And previous the opening session in Brazil, models can make such one-step ahead estimation, or even during the day, until it is closed (online data).

The same structure can be applied to other market indexes and assets, taking advantage of overnight circumstances - information available and fast prediction algorithm.

The data set

Data is free and can be obtained online through an Yahoo Finance API - so, no special data architecture system is required, since it is not a huge amount of data (for high frequency data it might change. However, the business problem is overnight oscillations, so daily data is a good start).

All the data is nominal (there is no deflation process, and currencies movements must be captured by indexes itself)

The dependent variable:

- ^BVSP = Ibovespa Composite Index (Brazil)

The independent variables chosen are relevant financial markets:

- ^BVSP = Ibovespa Composite Index (Brazil)
- ^DJI = Dow Jones Composite Index (USA)
- 000001.SS = SSE Composite Index (China)
- ^N225 = Nikey Composite Index (Japan)
- '^IRX'= 13 Week USA Treasury Bill

Data were cleaned (to obtain only the necessary information and NAs due to holidays were filled in with the last observation) and transformed (logarithmic, lagged, scaled and with difference - this last case is the Part II of this code), and inverse transformation at the end of the process.

The solution to the use case

For this demo…

An OLS Model (non-deep learning algorithm) and a LSTM Model (deep learning) - following Advanced Data Science with Capstone instructions - were built in order to capture such relations from Brazilian stock market index and its counterparts: USA, Japan, China. An interest rate (USA short-run) was tested (Monetary Policies movements).

But it has shown high volatility, poor correlation with target variable, and models performed better without it

The product is almost costless, and besides it can be applied to other financial assets (use cases), this one-step ahead prediction using online data can used with other kind of machine learning models.

The solution is a Python notebook. And, since data is online and not too big (for this project) no other systems are required.

Data science peers introduction

Capstone Project for Advance Data Science IBM

Main Findings

Data lose value quickly (Financial Market), long lags (and timesteps) are not important (do not bring better results)

Simpler structures perform better than complex structures (More hidden layers did not outperform simpler structures - MAPE and RMSE metrics)

OLS outperforms LSTM, in level, almost the double of LSTM RSME in the test data set

But, there is a problem with stationarity in LSTM residuals (time series data)

Studies with series in difference (Part II) were performed and, in this case, the LSTM outperform the OLS (lower RMSE), but estimations do no present the same variance, due to log return of a financial asset remains around zero (market efficiency theory - arbitrage price opportunities do not persist)

Architecture

For this job, we used free data which comes from an API that connects to Yahoo Finance and get financial data for the indexes that are chosen.

For this use case, daily close values are the data that we need, since some financial markets are closed when other are operating, and this gap is what the models tries to capture and create a prediction for the close value of that day (even before it started or while it is running – is online data)

So, no robust system is required (is around 6500 samples). Moreover, financial data lose value quickly, they are very volatile and too old information is not relevant – patterns change a lot

This solution uses only a Python notebook

ETL: data quality assessment, data pre-processing and feature engineering

Data passes through several transformations

- extract only the close values
- indexes are changed to datetime
- NA's are filled with last observation/sample (holidays), since market did not run during that date (Cleaning)
- log transformation is made
- change data from Dataframe to numpy arrays
- MinMax scaling is applied
- Finally, the lags of the variables are created – since it is a time series model estimation

After training, predicting and evaluating, the inverse of the transformations is applied in other to the de one-step ahead forecast

Another transformation is tested (since stationary of the residuals of the models is desired): log difference. But, the results of those models are not good, as you may see ahead.

Visualization

Data: how it comes from the API

```
2 Fulldata.head()
```

]:

Attributes	High				Low				Open		...	Close
Symbols	000001.SS	^BVSP	^DJI	^N225	000001.SS	^BVSP	^DJI	^N225	000001.SS	^BVSP	...	^DJI
Date												
1995-01-02	647.859985	4397.600098	NaN	NaN	647.859985	4300.100098	NaN	NaN	647.859985	4353.899902	...	
1995-01-03	639.880005	4385.899902	3845.199951	NaN	639.880005	4093.699951	3827.709961	NaN	639.880005	4369.899902	...	3838.47
1995-01-04	653.809998	4098.000000	3857.989990	19724.759766	653.809998	3860.899902	3831.070068	19641.220703	653.809998	4098.000000	...	3857.64
1995-01-05	646.890015	4040.300049	3860.679932	19717.759766	646.890015	3944.500000	3843.189941	19518.160156	646.890015	3967.899902	...	3850.91
1995-01-06	640.760010	4105.700195	3887.260010	19586.490234	640.760010	3813.000000	3841.840088	19417.210938	640.760010	4036.699951	...	3867.40

5 rows × 24 columns

Data: how it is used to analysis, visualization (in log)

:

Symbols	^BVSP	000001.SS	^DJI	^N225
Date				
1995-01-04	8.285992	6.482817	8.257813	9.887563
1995-01-05	8.303183	6.472176	8.256067	9.884106
1995-01-06	8.249941	6.462655	8.260340	9.879167
1995-01-09	8.196382	6.439350	8.258772	9.875341
1995-01-10	8.092545	6.413951	8.260167	9.878244

Visualization

Close Indexes

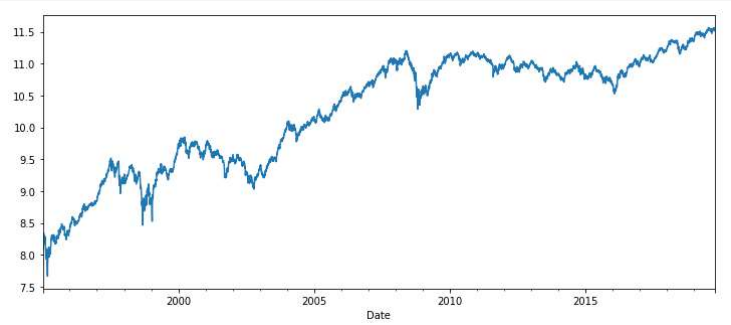
Log transformed

Descriptive Statistics

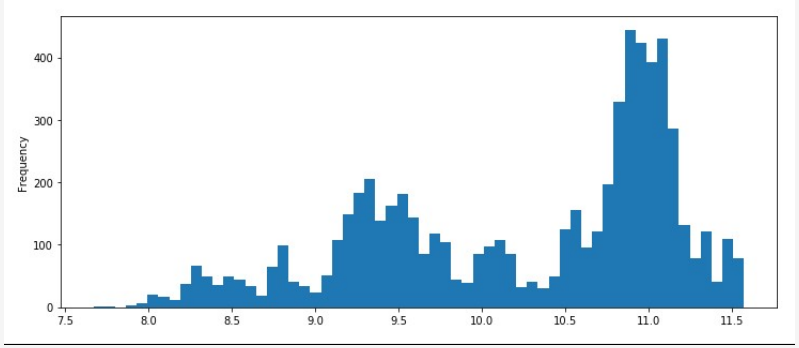
Symbols	^BVSP	000001.SS	^DJI	^N225
count	6461.000000	6461.000000	6461.000000	6461.000000
mean	10.252896	7.569805	9.356093	9.563569
std	0.899614	0.489136	0.407885	0.301402
min	7.667766	6.246998	8.251163	8.861489
25%	9.474165	7.233050	9.141320	9.286304
50%	10.622497	7.633234	9.297814	9.632786
75%	10.992403	7.971866	9.632212	9.804153
max	11.569466	8.714741	10.216807	10.097022

Visualization: target variable (Brazilian stock market index)

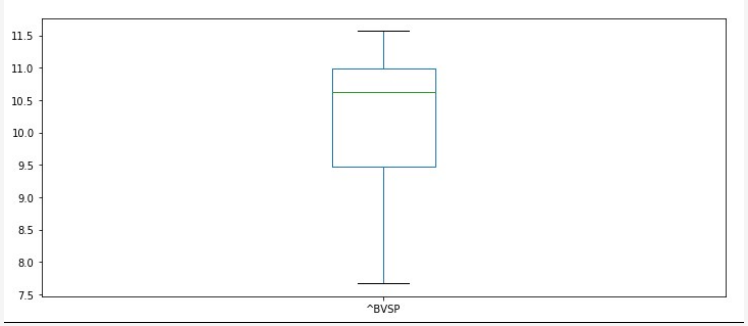
In log: time series evolution



Histogram



Box-plot



Correlation Matrix

Symbols	^BVSP	000001.SS	^DJI	^N225
Symbols				
^BVSP	1.000000	0.857977	0.830463	-0.053551
000001.SS	0.857977	1.000000	0.774681	-0.016857
^DJI	0.830463	0.774681	1.000000	0.280915
^N225	-0.053551	-0.016857	0.280915	1.000000

Model Decision and Training

OSL: non-deep learning model, simple, fast, multivariable problem (can be compared with LSTM). The risk was about the residuals, but they are stationary (evidence of cointegration of such time series).

LSTM: deep learning model, many applications with time series data. One hidden layer outperformed models with more layers

Finding: simpler structures outperformed complex structures (OLS presents better results)

batch_size = 180 (LSTM Stateful performed better with such number. Stateless presented worse results)

timesteps = 10 (to many lags, more complexed, presented worse results – financial data becomes old quickly, very dynamic)

4 variables in the dataset and 40 variables to the models (due to lags)

6120 samples for training and only 180 for test (financial data: most recent patterns are very different from the beginning of the time series)

Model Decision, Training and Evalutation

In the case of the LSTM, since there are several hyperparaments to decide, we created a "For" nested structure to keep the results from the possible combinations of such parameters (tested even with 2 hidden layers LSTM, but results in such case were poor).
Than, we chose the model with the best evaluation metric (MAPE) in the test data set.

Hyperparameters tested and selected

```
## Hyperparamters tests: remaining only the best combination
batch_size = 180#  64, 32, 16, 8
epochs =600#[160,250,400,600,1000]
ltest_percent=.05
listan=[44]#[30,55,44,100]
listafuncao=['linear']   #[linear''relu','elu']
listalraprend = [0.01]#,0.01]
listaotm = ['adam'] #['adam']
listadrop=[0.3] #[0.2,0.3,0.4]
listafuncaop=['mse'] #['mae','mse']
tam=len(listan)*len(listafuncao)*len(listalraprend)*len(listaotm)*len(listadrop)*len(listafuncaop)
print(tam)
```

```
1
```

Sequence of for's and keeping results

```
#Modeling: for's in order to get results from hyperparameters combinations
resultadosn=[]
resultadosfuncaoact=[]
resultadoslraprend=[]
resultadosotm=[]
resultadosdrop=[]
resultadosfp=[]
resultadosmapetrain=[]
resultadosmapetest=[]
resultadosrmsetrain=[]
resultadosrmsetest=[]
dfresultados_ypredtrain= pd.DataFrame(np.zeros((length,tam)))
dfresultados_ypredtest= pd.DataFrame(np.zeros((testset_length,tam)))
i=0
for n1 in listan:
    for funcaoact in listafuncao:
        for lraprend in listalraprend:
            for otm in listaotm:
                for drop in listadrop:
                    for fp in listafuncaop:
                                np.random.seed(123)

                                ## Scaling data for LSTM
                                dataset = df
```

We used 2 metrics for evaluation of the models during training and decision about hyperparmeters and to compare with OLS model (using the same variables):

RMSE
MAPE (Mean Absolute Percentage Error)

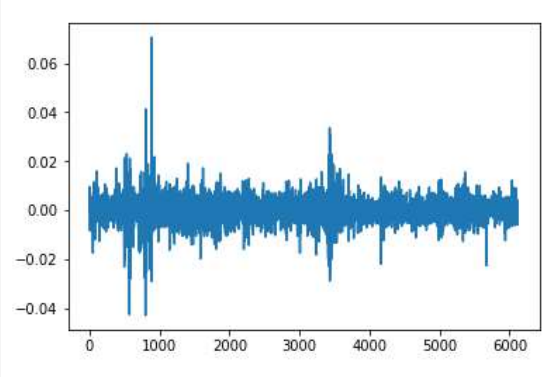
OlS Model: Evaluation and Results

We used 2 metrics for evaluation of the models during training and decision about hyperparmeters:
RMSE and MAPE (Mean Absolute Percentage Error)

MAPE and RMSE for evaluation

```
MAPE Train: 0.6428
MAPE Test: 0.2406
RMSE Train: 0.0050
RMSE Test: 0.0031
```

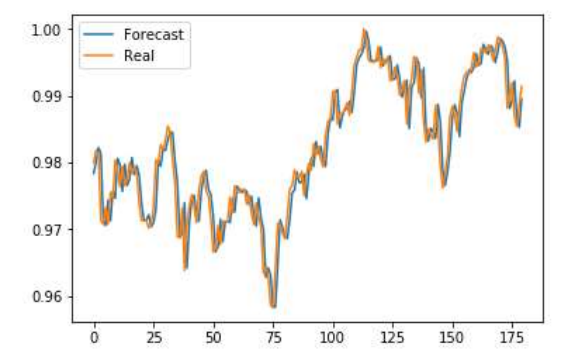
Residuals



Augmented Dick-Fuller tests

```
ADF Statistic: -15.409664
p-value: 0.000000
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

Evolution of Forecast



Prediction of today value

```
Prediction call OLS for today 101568.98
```

LSTM Model: Evaluation and Results

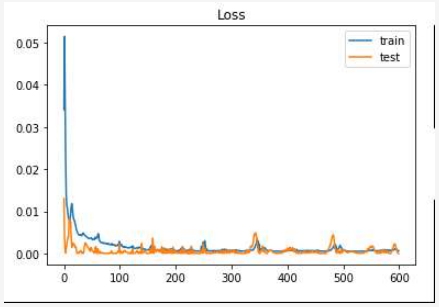
Final model description:

Kind: Stateful
batch_size: 180
Architecture: 1 hidden layer with 44 neurons
Epochs: 600
Activation Function: Linear
Drop out rate: 0.30
Optimizer: adam
Learning rate: 0.01
Loss function: mse

Hyperparameters tested

```
## Hyperparamters tests: remaining only the best combination
batch_size = 180#  64, 32, 16, 8
epochs =600#[160,250,400,600,1000]
ltest_percent=.05
listan=[44]#[30,55,44,100]
listafuncao=['linear']   #[linear''relu','elu']
listalraprend = [0.01]#,0.01]
listaotm = ['adam'] #['adam']
listadrop=[0.3] #[0.2,0.3,0.4]
listafuncaop=['mse'] #['mae','mse']
```

Loss Function



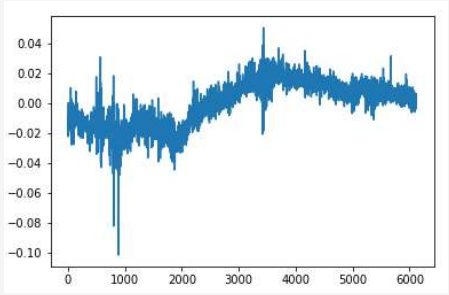
MAPE and RMSE for evaluation

```
Best MAPE Train: 13.7028
Best MAPE Test: 1.1243
Best RMSE Train: 0.0152
Best RMSE Test: 0.0039
```

Prediction of today value

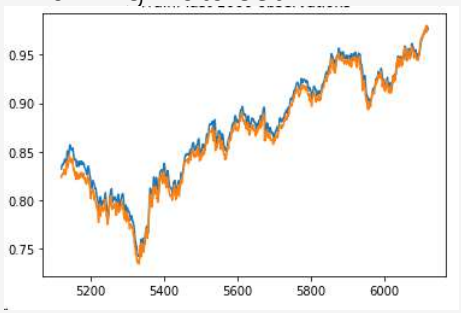
```
Prediction call for today 101769.48
```

Residuals

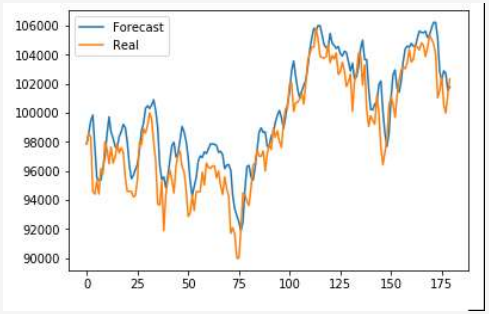



```
ADF Statistic: -1.496761
p-value: 0.535113
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

Training Data set



Test Data set



Comparison

Prediction of today value

```
Prediction OLS call  for today 101568.98
Prediction LSTM call for today 101769.48
Actual-Online 102325.94
```

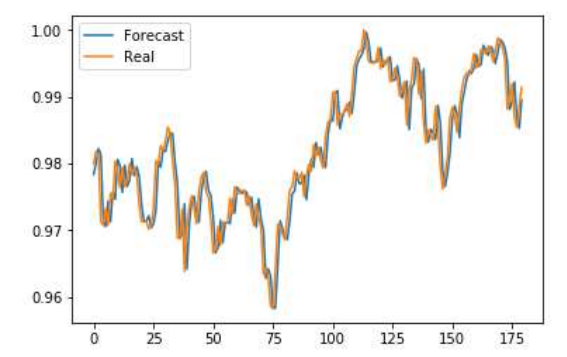
OLS

```
MAPE Train: 0.6428
MAPE Test: 0.2406
RMSE Train: 0.0050
RMSE Test: 0.0031
```

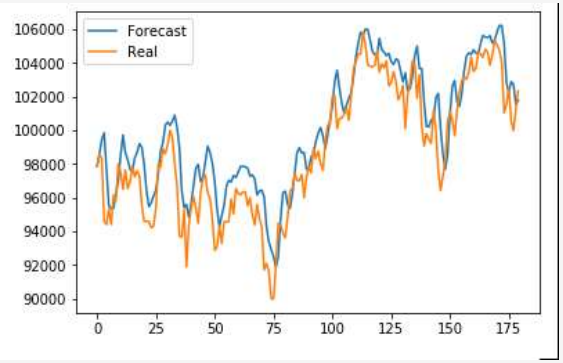
LSTM

```
Best MAPE Train: 13.7028
Best MAPE Test: 1.1243
Best RMSE Train: 0.0152
Best RMSE Test: 0.0039
```

Evolution of Forecast OLS



Evolution of Forecast LSTM



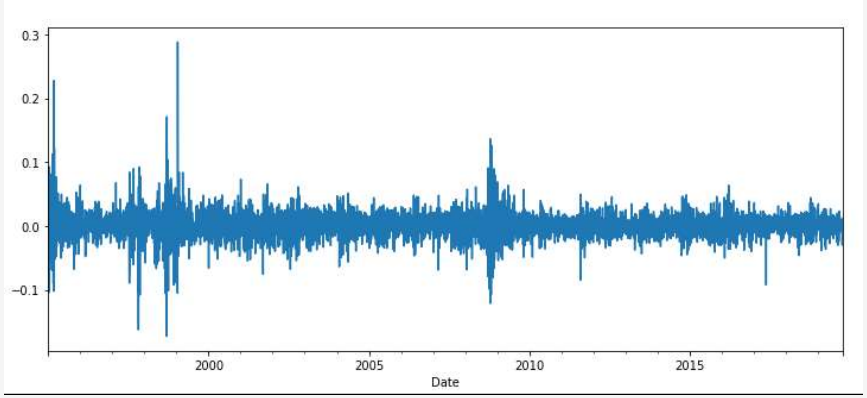
RMSE for test data set
after transformations were inverted:

OLS: 87.27

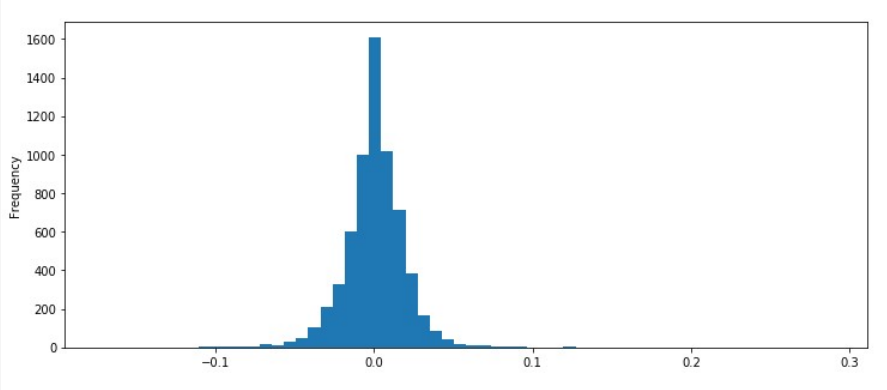
LSTM: 135.33

Part II: Data in difference

Evolution target variable in difference



Target variable in difference histogram

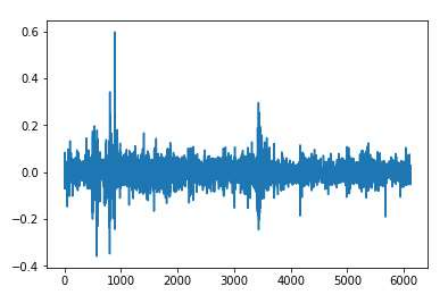


Part II: Data in difference

OLS: Evaluation Metrics

```
MAPE Train: 6.9265
MAPE Test: 5.4762
RMSE Train: 0.0420
RMSE Test: 0.0262
```

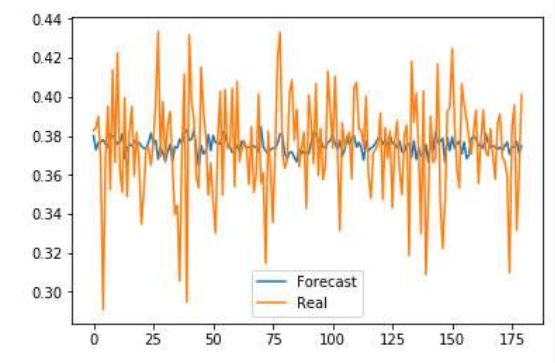
OLS: Residuals



OLS: Adf Test

```
ADF Statistic: -78.308710
p-value: 0.000000
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

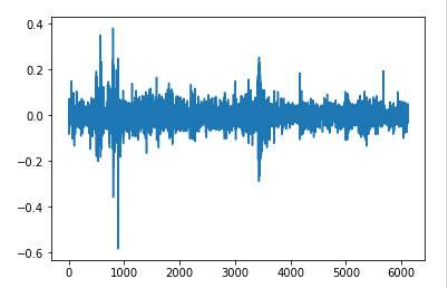
OLS: Forecast vs Real



LSTM: Evaluation Metrics

```
Best MAPE Train: 7.1355
Best MAPE Test: 5.4119
Best RMSE Train: 0.0421
Best RMSE Test: 0.0259
```

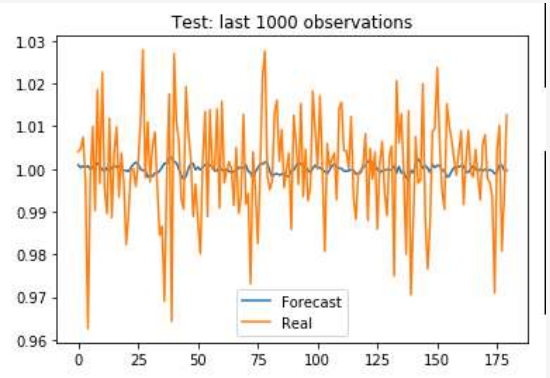
LSTM: Residuals



LSTM: Adf Test

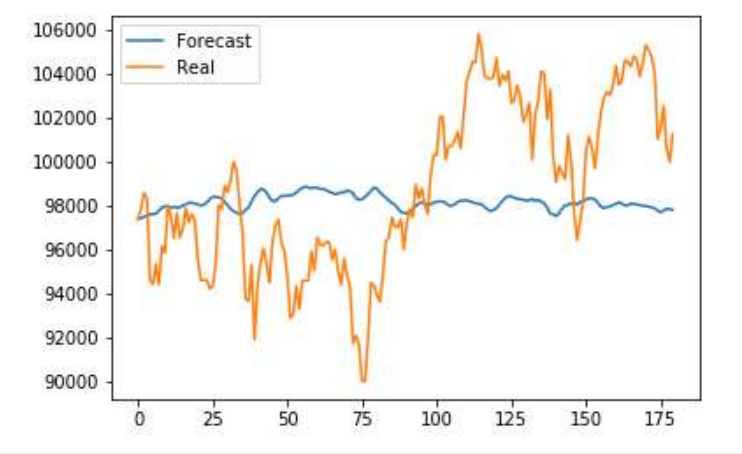
```
ADF Statistic: -23.866252
p-value: 0.000000
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

LSTM: Forecast vs Real

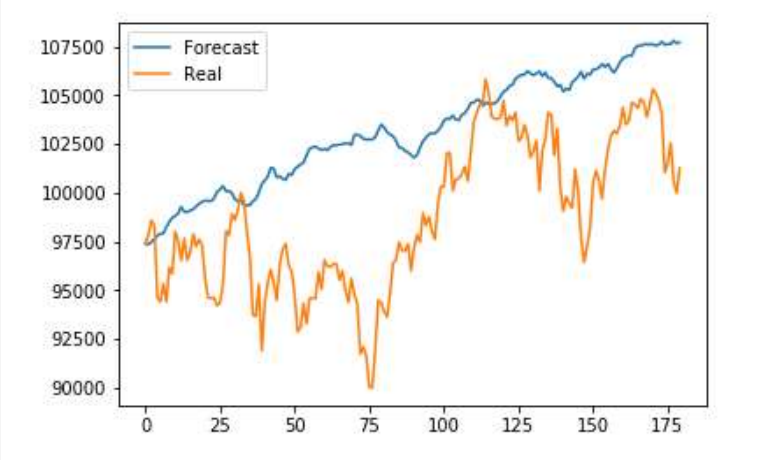


Part II: Data in difference

Evolution of Forecast LSTM: in level



Evolution of Forecast OLS: in level




```
Prediction LSTM in diff call for today 97800.17
Prediction OLS in diff call for today 107706.53
Actual-Online 101248.78
```

RMSE for test data set
after transformations were inverted:

OLS: 390.11

LSTM: 296.87