

NATIONAL RESEARCH UNIVERSITY

HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science

Bachelor's Programme 'HSE University and University of London Double
Degree Programme in Data Science and Business Analytics'

Software project report (Final)

on the topic

Providing Financial Information via Telegram bot

Student:

group БПАД192

Veips Filipp Artisovich

Sign Name, Surname, Patronymic

Supervisor:

Lukianchenko Peter Pavlovich

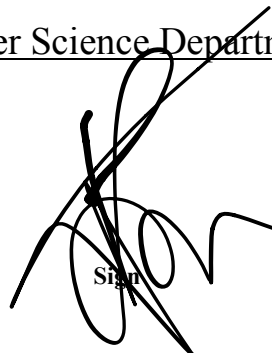
Name, Surname, Patronymic

Lecturer Position

Computer Science Department Company

Date 03.06.2021

10
Grade


Sign

Moscow 2020

Contents

Contents.....	2
1. Abstract.....	3
1.1 Object of research.....	3
1.2 Purpose of work	3
1.3 Methodology.....	3
1.4 Results.....	3
1.5 Key Words & Definitions	3
2. Introduction	4
2.1 Problem relevance.....	4
2.2 Goals and Tasks.....	4
3. Review and Contemporary Analysis of Sources and Analogues.....	5
3.1 Sources.....	5
3.2 Analogous utilities	5
3.3 Models and Algorithms.....	6
4. Theoretical part	7
5. Programming realization part.....	8
6. Conclusion	10
7. Bibliography	11
7.1 Websites	11
8. Appendix.....	12
8.1 Calendar plan of work	12

1. Abstract

1.1 Object of research: Application of statistical models to gain predictions in the securities market

1.2 Purpose of work: To make a programming unit which consists of both predictions' calculations using statistical models and handy interface via Telegram messenger.

1.3 Methodology: Technical analysis via programmed application.

1.4 Results: Successful complete of purposed task.

1.5 Key Words & Definitions:

- **Telegram Bot** (*further the Bot*) - specialized fully automated account in Telegram social network. It can be reached via simple chat in Telegram, in which by a special list of commands (provided by bot development team) it can give any information or service that were requested by the user and which the Bot is capable to perform.
- **Technical analysis** - means of examining and predicting price movements in the financial markets, by using historical price charts and market statistics
- **Stock market (Equity market / Share market)** – a huge abstract group of shareholders who can freely buy and sell stocks.
- **Stock Exchange** - special organization which has a license to provide service in connecting brokers' requests in operations with securities. In terms of the Bot usage, it will be considered as the main source of the data which will be provided to the user via API.
- **Application Programming Interface (API)** - commands and programming protocols which allow User to interact with the program. API will be viewed in particular as Stock Market API. It offers real-time or historical data on financial assets that are currently being traded in the markets.
- **Machine learning (ML)** - class of methods applicable to Artificial Intelligence (AI) which includes education of a machine by using Statistical methods to solve particular problems. ML will be used to analyze data obtained from API.
- **Recommendatory system** (*further the system/the System*)-program which will be integrated into the Bot and will provide recommendations and analysis on user's portfolio by user's requests using ML
- **Time series** - series of data points indexed in time order.
- **Time series forecasting** - a collection of models used to analyze time series and predict future values based on previously observed ones. Using Machine Learning the system will be able to predict stock quotes within different trading periods
- **Momentum indicators** - technical analysis tools used to determine how overbought (overheated) or oversold (undervalued) a particular share is.
- **Macroeconomic indicators** - statistics reflecting the output of an economy or a market. They consist of interest rates announcements, GDP, consumer price index, employment indicators, retail sales, monetary policy, and more. Broadly used to predict financial market volatility.
- **Intraday data** - data on securities trading on the markets during regular business hours. These include stocks and exchange-traded funds (ETFs). Intraday also provides the highs and lows that the asset crossed throughout the day

2. Introduction

Throughout the last few years people have been becoming more and more financially educated, interest in the stock market and public interest in investment is at its maximum level ever. Still, investments as an instrument do not provide people with 100% probability to gain profit. Analyzing the market is crucial for success in this field. However, not everyone has a lot of time to do it, that is why it is vital to have a useful tool which can analyze changing markets. The previous statements are proved by releases of investing applications by leading financial technology institutions. Russian companies: Tinkoff, Sber, Alpha Bank, VTB released such tools within several months. Leading media report about the biggest amount of investment made by Russian citizens ever. Combined with significant increase in amount of active Telegram users, the application which combine both popular tendencies is not only relevant, but even highly demanded by current audience of people who want to invest easily, gain information fast and raise their profits.

2.1 Problem relevance

- Make a system like a digital broker, which can forecast the price of shares and provide a holder with relevant information about current market situation.
- During the process of making a tool with the system get useful information on the process of developing a utility and on the behavior of stock markets which can be applicable in future.

2.2 Goals and Tasks

1. Gain information about financial analytical methods, get familiar with most common financial instruments and indicators.
2. Collect and store data from previous stock trading (throughout a few years) using API
3. Use statistical models to analyze data and make it forecast share prices
4. Develop a simple but effective UI for the project via Telegram bot.
5. Make extra features to simplify analyzing and presentation of the data
6. Successfully present the recommendation system with the Bot

3. Review and Contemporary Analysis of Sources and Analogues

3.1 Sources

The main utility for the project will be **Alpha Vantage API**, it provides constant updating, actual and trustworthy information on stocks, bonds, and currencies. This API is also characterized by its wide profile of companies including indexes S&P500 for 500 most valuable American companies, MOEX index for biggest Russian companies trading in Moscow stock market and many other indexes such as RTX, Nikkei 225 and others.

The additional source of the project is Yahoo Finance. From its API the bot will provide user with the information about the company, its financial statistics, financial indicators and multipliers and fundamental analysis that can be applied on requested data.

As for any additional information <https://ru.investing.com/> as the best website for any information connected to stocks, bonds and forex. It has actual and reliable information which is updating every second.

3.2 Analogous utilities

Facebook Prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well. However, experiment measurements have shown that Prophet is hardly accurate, and the model has an average of 30% error from the actual figures.

Bloomberg Terminal

Terminal is a solution for financial market analytics that brings together real-time data on market, breaking news, in-depth research, and powerful data coverage. Bloomberg has a broad range of research offerings including independent researches from different sources. Although Terminal provides a lot of important, relevant information it lacks forecasting functionality.

FinBrain

It is a system to provide stock predictions for users. It can provide rather highly accurate information based on advanced Machine learning and Deep learning algorithms. This system is quick responding and can provide also forex and cryptocurrencies information.

3.3 Models and Algorithms

There is a huge variety when it comes to choosing time series predicting models, suitable for the stock market. However, it is impossible not to highlight a group of Linear models as most common and useful in analyzing the stock market.

Two basic methods that will be used are **Autoregressive (AR) Estimation** and **Moving Average (MA) Estimation**:

Autoregressive (AR) Estimation - is a model that suggests variable to depend linearly, so the model can be presented as difference equation or recurrence relation. So, the following variable can be described as relation to the previous one. This relation can be presented through parameter, which is called lag operator(L) and can be described as $L^{-1}X_t = X_{t+1}$, for X to be the element of the series on which AR is applied. Consequently, AR(p), where p is order of AR, can be presented through the following formula: $X_t = c + \sum_{i=1}^p L^i \varphi_i X_t + \varepsilon_t$, where c-constant, φ_i – parameters of AR which can be found by Method of moments through Yule–Walker equations¹, ε_t -white noise.

Moving Average (MA) Estimation- it is a model applicable for univariate time series, where variables are dependent linearly. The formula for MA: $X_t = \mu + (1 + \theta_1 L + \dots + \theta_q L^q) \varepsilon_t$, where μ -mean of the series, X is element of series, $\theta_{1...q}$ – parameters of MA and ε_t -white noise, q is order of MA.

Autoregressive Integrated Moving Average (ARMA) model - it is a model that combines both AR and MA models to achieve better results. It depends on both p and q (orders of AR and MA respectively) and its formula: ARMA=AR+MA or $X_t = c + \sum_{i=1}^p L^i \varphi_i X_t + \varepsilon_t + \mu + (1 + \theta_1 L + \dots + \theta_q L^q) \varepsilon_t$. To find p and q Akaike information criterion (AIC) is used, however BIC (Bayesian information criterion) is more classic way to find them.

Autoregressive Integrated Moving Average (ARIMA) - One of the most popular linear models. Consists of Autoregressive (AR) and Moving-Average (MA) models. AR shows how variable are regressed with lagged variable determined by p, while MA shows that regression error is a linear combination of previous errors occurred with previous variables determined by q, I stand for integration and defines if values are replaced by differences between their values and previous values with lag = d. Order of ARIMA can be found by same methods like in ARMA.

Seasonal Autoregressive Integrated Moving Average (SARIMA) – is usually denoted ARIMA with additional orders (P, D, Q) which determine autoregressive, differencing and moving average parts for seasonal part of ARIMA. SARIMA analyses different parts of the samples called seasons and find correlation between and within them.

4. Theoretical part

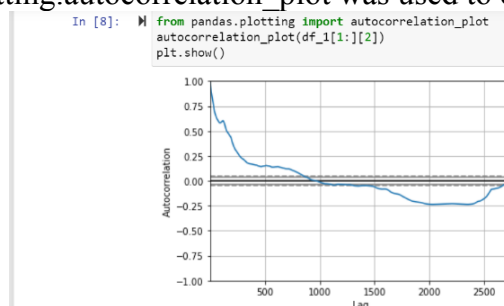
Statistical models application: AR, ARMA, ARIMA, SARIMA, the basic model for them is AR

Libraries used: As statistical models which are applied through many equations, there first issue was to solve is to get rid of any mistakes connected to calculations (or miscalculations), of the models to be applied on data. Using features of Python, library “statsmodels” is designed to apply statistical models on data, avoiding any miscalculations. Libraries Pandas, Numpy are used for data storing and analyzing, Matplotlib is for plotting data, Sklearn for applying metrics, for example MAE or MSE (Mean Absolute Error and Mean Squared Error respectively).

Orders: However, usage of statsmodels library does not providing user with needed models without any sense, the main dependency of AR like models on order is remains. To find order the following method is used: there is a closed, finite and most importantly given pool of estimated orders to be suited for analyzing data.

Order determination: To size order pool pandas.plotting.autocorrelation_plot was used to determine main tendencies in behavior of stocks for big data.

From Figure 1 follows that there is aa clear correlation in the first part of the graph, consequently p is limited to 50. After various test p is clearly have minimal error with minimal p.



Applying tests on d and q also determined optimal pools *Figure 1 TSLA stocks autocorrelation_plot* for d and q. P, D, Q pools were made for SARIMA in the same way, s in SARIMA is constant.

Determining model’s accuracy: From the barely limited list of the possible metrics applicable to determine accuracy, MAE metrics was selected. Mean Absolute Error (MAE): measure of errors between estimated value and observed value = sum of |estimated value - observed value|/number of values. As for stocks and bonds their price is equal to money put into securities the absolute difference for estimated values is far more important than MSE (mean squared error) or any similar metrics. To apply MAE the sample is divided with the ratio 2/10 on test and train parts. Model is fitted on train part and tested on test part with application of MAE metrics.

Prediction with applying models: To predict values in the future time series are extended, model trained on input data make predictions on extended time series without any possibility to test predictions due to the lack of observed data. Such predictions are not perfectly accurate; however, they show clear tendencies that are extrapolation of current data into the future.

5. Programming realization part

Frontend bot application: All the models are implemented into the bot as functions. Bot get calls from user through messages and special commands called with special symbol ‘/’ and returns data on companies, graphs or analysis dependent on users’ requests.

```
def evaluate_ar(df):
    ans=[]
    df_train= df[int(len(df)*0.2):]
    df_test= df[1:int(len(df)*0.2)]
    predictions = []
    ar = tsa.AR(df_train[2].values)
    ar_fit = ar.fit()
    predictions = ar_fit.predict(len(df_train), len(df_train)+len(df_test)-1, dynamic=False)
    ans.append(predictions)
    mae = mean_absolute_error(df_test[2], predictions)
    ans.append(mae)
    return ans
```

Figure 2 AR realization

Sample split on train/test parts with coefficient 2/10, model is fitted on train part, got predictions, tested on test part with MAE metrics. Other models the same action.

```
def evaluate_arima(df, model_order):
    ans=[]
    df_train= df[int(len(df)*0.2):]
    df_test= df[1:int(len(df)*0.2)]
    predictions = []
    arima = tsa.ARIMA(df_train[2].values, model_order)
    arima_fit = arima.fit(disp=0)
    predictions = arima_fit.predict(1, len(df_test), typ='levels')
    ans.append(predictions)
    mae = mean_absolute_error(df_test[2], predictions)
    ans.append(mae)
    return ans
```

Figure 5 ARIMA realization

```
def evaluate_arma(df, model_order):
    ans=[]
    df_train= df[int(len(df)*0.2):]
    df_test= df[1:int(len(df)*0.2)]
    predictions = []
    arma = tsa.ARMA(df_train[2].values, model_order)
    arma_fit = arma.fit(disp=0)
    predictions = arma_fit.predict(1, len(df_test), dynamic=False)
    ans.append(predictions)
    mae = mean_absolute_error(df_test[2], predictions)
    ans.append(mae)
    return ans
```

Figure 6 ARMA Realization

```
def evaluate_sarima(df, model_order):
    ans=[]
    df_train= df[int(len(df)*0.2):]
    df_test= df[1:int(len(df)*0.2)]
    predictions = []
    sarima=statsmodels.sarimax.SARIMAX(endog=(df_train[2].values.astype(float)),
                                         order=model_order,seasonal_order=(model_order[0],model_order[1],
                                         enforce_invertibility=False) model_order[2],12),trend='c',

    sarima_fit = sarima.fit(disp=0)
    predictions = sarima_fit.predict(1, len(df_test), typ='levels')
    ans.append(predictions)
    mae = mean_absolute_error(df_test[2], predictions)
    ans.append(mae)
    return ans
```

Figure 7 SARIMA Realization

```
import telegram
from telegram.ext import Updater, CommandHandler, CallbackQueryHandler
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from pandas_datareader import data as pdr
from io import BytesIO
import os
import string
from telegram import InlineKeyboardButton, InlineKeyboardMarkup
import json
import csv
import matplotlib.pyplot as plt
from alpha_vantage.timeseries import TimeSeries
import warnings
from pandas_vantage.fundamentaldata import FundamentalData
from pandas import read_csv, DataFrame
import statsmodels.api as sm
from statsmodels.iolib.table import SimpleTable
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
import statsmodels.tsa.api as tsa
import statsmodels.tsa.statespace as statespace
import FundamentalAnalysis as fa
```

Figure 3 List of imported libraries

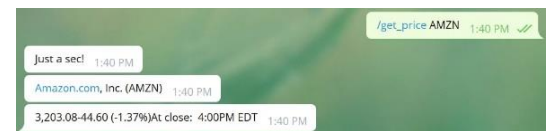


Figure 4 Chat with bot

```
def evaluate_models(df, p_values, d_values, q_values, choice):
    min_error=9999
    best_order=()
    best_prediction=[]
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                ans_list=[]
                index_dict = {}
                try:
                    if choice == 'AR':
                        ans_list = evaluate_ar(df)
                    elif choice == 'ARIMA':
                        ans_list = evaluate_arima(df, order)
                    elif choice == 'ARMA':
                        order = (p,q)
                        ans_list = evaluate_arma(df, order)
                    elif choice == 'SARIMA':
                        ans_list = evaluate_sarima(df,order)
                    elif choice == 'Best_Model':
                        ans_list_arima = evaluate_arima(df, order)
                        ans_list_arima.append('ARIMA')
                        index_dict[ans_list_arima[1]] = ans_list_arima

                        ans_list_arma = evaluate_arma(df, (p,q))
                        ans_list_arma.append('ARMA')
                        index_dict[ans_list_arma[1]] = ans_list_arma

                        ans_list_sarima = evaluate_sarima(df,order)
                        ans_list_sarima.append('SARIMA')
                        index_dict[ans_list_sarima[1]] = ans_list_sarima

                        ans_list_ar = evaluate_ar(df)
                        ans_list_ar.append('AR')
                        index_dict[ans_list_ar[1]] = ans_list_ar

                        ans_list.append(index_dict[min(index_dict.keys())][0])
                        ans_list.append([min(index_dict.keys())])
                        ans_list.append(index_dict[min(index_dict.keys())][2])
                        prediction = ans_list[0]
                        error = ans_list[1]
                        if error < min_error:
                            min_error, best_order,best_prediction = error, order, prediction
                            #print('ARIMA% MAE=%3f' % (order,error))
                except:
                    continue
    #print('Best ARIMA% MAE=%3f' % (best_order, min_error))
    if choice != 'Best_Model':
        return best_prediction
    elif choice == 'Best_Model':
        return ans_list
```

Figure 8 Best order/model searching



Figure 9 ARMA Results AMZN stocks

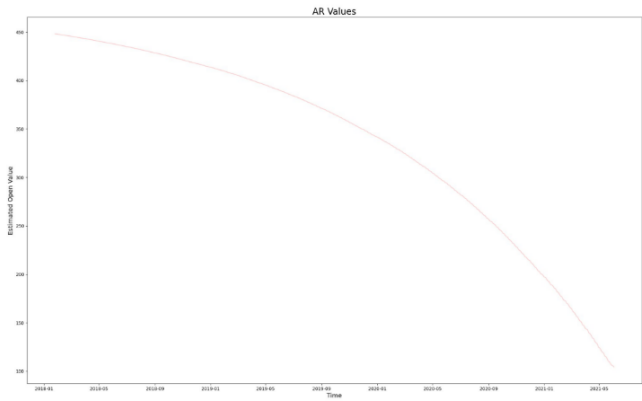


Figure 10 AR Results GOOGL stocks

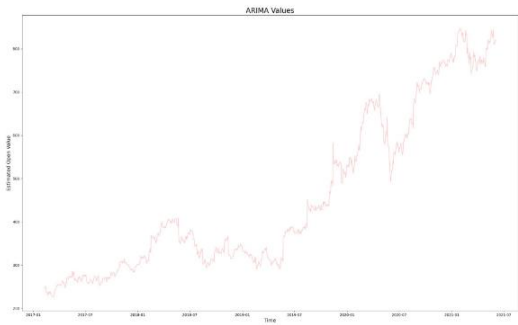


Figure 11 ARIMA Results AMZN

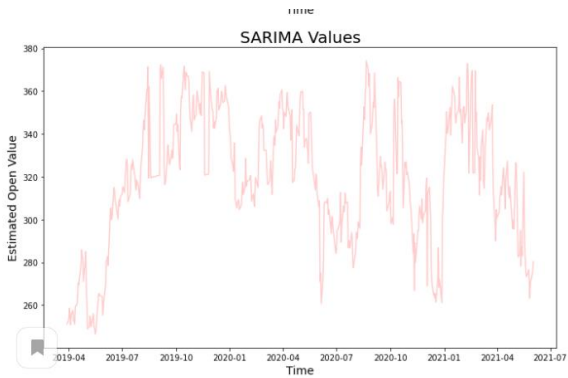


Figure 12 SARIMA Results TSLA

6. Conclusion

Overall, results are the following:

- Bot is successful in getting and responding to users' commands. It provides graphs, model results, stock info, stock data.
- Models evaluating results correctly. MAE is applicable to every model and evaluating results which can be considered trustworthy.
- There exists a predicting part that makes and posts through bot graphs of estimated future values.

So, though pure statistical models are not always applicable, their application is good in showing trends for stocks. Moreover, application of different models increases chances for data to be correct, as truth is always in the middle. The project goals are completed, and the bot is a good service for investor ready to serve. Though many areas of stock market stay out of bot, like technical indicators, statistically bot is capable to do what is made for.

7. Bibliography

- Irene Aldridge High Frequency Trading
- Ilija I.Zovko Topics in Market Microstructure
- Simon T Gray Liquidity Forecasting
- M. Ahmed, A. Chai, X.Ding, Y. Jiang, Y. Sun Statistical Arbitrage in High Frequency Trading Based on Limit Order Book Dynamics
- J. Yao, C. Lim Tan, H. Poh Neural Networks for Technical Analysis: a Study on KLCI
- Newbold, P., W.L. Carlson and B.M. Thorne Statistics for Business and Economics
- Begg, D., G. Vernasca, S. Fischer and R. Dornbusch Economics. (London: McGraw Hill, 2014) 11th edition [BVFD]
- George E. P. Box, Gwilym M. Jenkins Time Series Analysis: Forecasting and Control
- Theodoridis, Sergios (2015-04-10). "Chapter 1. Probability and Stochastic Processes". *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 2015. pp. 9–51. ISBN 978-0-12-801522-3.

7.1 Websites

- <https://www.hse.ru>
- <http://wiki.cs.hse.ru>
- <https://ru.wikipedia.org/wiki>
- <https://www.alphavantage.com>
- <https://www.kaggle.com>
- <https://habr.com/ru>
- <https://stackoverflow.com>
- <https://www.tinkoff.ru/invest>
- <https://rapidapi.com/apidojo/api/yahoo-finance1>
- <https://github.com/ranaroussi/yfinance>
- <https://iexcloud.io>
- <https://financialmodelingprep.com/developer>
- <https://tlgrm.ru/docs/bots>
- <https://marketstack.com>
- <https://jfin-swufe.springeropen.com/articles>

8. Appendix

8.1 Calendar plan of work

Pre-Alpha Period:	10.01.2021 - 21.02.2021
Checkpoint 1 Draft	by 08.02.2021
Checkpoint 1 Final Version	by 22.02.2021
Alpha Test Period:	22.02.2021 - 16.04.2021
Checkpoint 2 Draft	by 16.04.2021
Beta Test Period:	16.04.2021 - 30.04.2021
Checkpoint 2 Final Version	by 30.04.2021
Pre-Release Period:	1.05.2021 - 3.06.2021
Release:	3.06.2021

Github: <https://github.com/FilippVeips>

ⁱ Theodoridis, Sergios (2015-04-10). "Chapter 1. Probability and Stochastic Processes". *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 2015. pp. 9–51. [ISBN 978-0-12-801522-3](#).