

Project of an application based on machine learning for stock market prediction

Code

Vladislav Sorokin

student record book number 295462

Dzianis Harbatsenka

student record book number 295452

Nikita Zakharov

student record book number 295464

thesis supervisor

prof. PW dr hab. inż., Jerzy Balicki

Warsaw 2021

Version 1.0

Warsaw, 24.11.20

**DECLARATION**

We declare that this piece of work which is the basis for recognition of achieving learning outcomes in the Group Project course was completed on our own.

**Table of contents**

[1. SVR 4](#_Toc57201641)

[2. LSTM 5](#_Toc57201642)

[3. CNN 6](#_Toc57201643)

# SVR

Function to train SVR

function [dates, y, fxpr, Bestmse] = train\_svr(dtname, maxItr,file\_name,isAutoHyper,kernelFunctionName,MaxEvalNum,epsilon,loopMaxKernelScale, loopMaxBoxConstraint)

input data table name, number of max iterations, file name of the model, auto training, max number of evaluations, epsilon value, iteration number for my optimization of kernel scale, iteration number for my optimization of box constraint.

Return is required values to plot the graph.

Function to predict price using SVR

function [datespr, ypr, rst,errors,mse,Mape] = predict\_svr(dtname,model,daysToPredict)

Input data table name, model name, how many days to predict.

# LSTM

parameters.py

Below the global parameters are defined for the program. It includes the settings of LSTM network, testing procedure and other useful parameters.

stock\_prediction.py

The below script servers as a help for the main function. The first function in it

def load\_data(ticker, n\_steps=50, scale=True, shuffle=True, lookup\_step=1,  
 test\_size=0.2, feature\_columns=['High', 'Low', 'Open', 'Close', 'Volume']):  
The function load\_data downloads the dataset from WIG\_20 and transforms the data into the form comprehensive for the network. The result of the function and the way the data will look like depends on the parameters provided. The specific feature columns can be set. The dataset can be shuffled or not.

def create\_model(input\_length, units=256, cell=LSTM, n\_layers=2, dropout=0.3,  
 loss="mean\_absolute\_error", optimizer="rmsprop"):

The function create\_model defines and initializes the network with the predefined settings.

start.py

The main script is provided below. The function InitUT initializes the GUI of the application. Then two functions for each button click event are provided (Train and Test).

def button1\_clicked(self):  
Triggers the training procedure to start. The model is initialized, the data is loaded and then the network starts to get fitted.

def button2\_clicked(self):  
Triggers the prediction of the model after training process is finished. At this stage the network evaluates the predicted values.

def plot\_graph(self, model, data)

The function plots the graph to see the comparison between real and predicted values.

# CNN

Function to initialize CNN

model\_initialization <- function(timeseires\_length)

Function to train CNN

model\_training <- function(model, X, Y, cb)

Function to predict with CNN

model\_prediction <- function(model, X)

Function to save a model

model\_save <- function(model, name)

Function to load existing model

model\_load <- function(name)

Function to prepare data for train, test and prediction

prepare\_data <- function(data\_frame, timeseires\_length, t\_num, p\_num = 5)

Function to make prediction based on backpropagation

real\_prediction <- function(model,last\_ts,pnum = 5)

Function to form timeseries out of input data

form\_timeseriese <- function(open, close = NULL, steps)

Function to modify a timeseries by pop and push a value

append\_timeseries <- function(ts, value)

Function to divide data into train, test and prediction sets

divide\_data <- function(data, tnum, ts\_length, pnum = 5)

Function to load data from a chosen csv

choose\_data <- function(name)

Function to obtain existing models by name

get\_models\_names <- function(name)

Function to obtain names of data sets

get\_data\_names <- function()

Function to plot raw data set

plot\_raw\_data <- function(data\_frame)

Function to plot test data

plot\_test <- function(actual, test, ts\_length)

Function to plot predicted data

plot\_prediction <- function(actual, test)