Silesian
University
of Technology

FINAL PROJECT

Stock price prediction

Przemysław BACA
Student identification number: 294686

Programme: Control, Electronic, and Information Engineering
Specialisation: Informatics

SUPERVISOR
DSc Anna Gorawska
DEPARTMENT Katedra Informatyki Stosowanej
Faculty of Automatic Control, Electronics and Computer Science

CONSULTANT
⟨title first name surname⟩

Gliwice 2023

Thesis title

Stock price prediction

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Keywords

(2-5 keywords, separated with commas)

Tytuł pracy

Predykcja kursów walut

Streszczenie

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in Polish.)

Słowa kluczowe

(2-5 keywords, separated by commas, in Polish)

Contents

Chapter 1

Introduction

Recently, there has been significant progress in terms of Artificial Intelligence (later called AI) and Machine Learning (later called ML). Projects that in the past could barely recognize some simple objects or try to recognize some patterns are now surpassed by programs like advanced chat bots that can talk about literally anything or AI's that can create very realistic art. Many famous personas like Elon Musk or Bill Gates start to recognize the potential of AI. As this field is rapidly growing it appears to be both reasonable and demanded to perform research in this area.

This work aims to create a target application to learn about the possibilities of neural networks in the context of predicting the stock exchange prices. The presented project is educational and designed to enable experimentation with the use of parameters of a given neural network - such as input data size, range, epoxy quantity, et cetera. In order to allow the end user to interact with the presented application easily, the project requires creating a simple graphical interface, so the end user can easily interact with the application and quickly view the results. Experimentation and the student's interaction with the assigned field is the best way to learn. For this purpose, this work focused on simple but logical dependencies on this particular type of neural network. The aim of this project was not to present an application capable of predicting the actual stock market but rather to create an application that could improve learning and showcase the possibilities of Artificial Intelligence. The goal was not only to educate but also to inspire to act and improve the current knowledge.

Chapter 2

[Problem analysis]

What is AI and what can it be used for

As the Cambridge Dictionary states, Artificial Intelligence is “the study of how to produce machines that have some of the qualities that the human mind has, such as the ability to understand language, recognize pictures, solve problems, and learn”. It can be concluded that Artificial Intelligence is a field of science whose goal is to replace the work of humans with machines. AI is used for a wide variety of tasks and applications. Some examples include:

- Image and speech recognition: AI algorithms can be trained to identify objects, people, or words in images or audio recordings.

- Language understanding: AI can be used to interpret and respond to natural language inputs, such as in virtual assistants or chatbots.

- Predictive analytics: AI can be used to analyze large data sets and make predictions about future events or trends.

- Robotics: AI controls and navigates robots, allowing them to perform tasks such as assembly line work or search-and-rescue operations.

- Self-driving cars: AI is used to interpret sensor data and decide how to control the car.

- Healthcare: AI can be used to analyze medical images, identify potential health risks, and assist doctors in making diagnoses.

- Fraud detection: AI can be used to identify suspicious patterns of behavior in financial transactions.

These are just a few examples of the many ways AI is being used today, and as the technology continues to advance, it is likely to be used in an even wider range of applications in the future.

AI in stock prediction

Since the beginning of Artificial Intelligence, people have wanted to use it for profit. The concept of a program that would be able to predict the future of the stock market is an idea that inspires many engineers to act. This is an unachievable goal, but AI is constantly evolving thanks to this challenge.

Predicting stock prices is a challenging task that has been the subject of much research in AI and machine learning. Various neural network architectures have been used for this task, and the choice of architecture will depend on the specific characteristics of the data and the problem at hand. Some popular architectures that have been used for stock price prediction include:

- Recurrent Neural Networks (RNNs): RNNs are particularly well-suited to sequential data, such as time series data, and have been used to predict stock prices by analyzing historical price and trading volume data.
- Long Short-Term Memory (LSTM) networks: LSTMs are a type of RNN that are able to maintain a long-term memory of the inputs it has seen and has been used to predict stock prices by analyzing historical price data.
- Convolutional Neural Networks (CNNs): CNNs are a type of neural network that are particularly well-suited to image and video data and have been used to predict stock prices by analyzing images of stock charts.
- Gated Recurrent Units (GRUs): Similar to LSTMs but with less computational complexity.

It is worth noting that stock price prediction is a complex problem, and only some models would work best for some scenarios. Many other variables, such as news, announcements, market sentiments, and trends, also need to be taken into account and can be incorporated into the model using techniques like Sentiment Analysis and Named Entity Recognition. It is also important to mention that stock prices are highly dynamic and non-linear, making it difficult to predict with high accuracy, and multiple models with different architectures may need to be combined to produce an accurate prediction.

For this project, LSTM model have been chosen to predict the stock prices. One of the main reasons why LSTMs may be well-suited for stock price prediction is their ability to maintain a long-term memory of the inputs they have seen. Stock prices are a type of time series data, which means that the current price of a stock is likely to be influenced by a long sequence of previous prices and other related financial data, such as trading volume and market sentiment. Because LSTMs can maintain a long-term memory of the inputs they have seen, they can understand patterns or dependencies in the data that may span multiple time steps, making them well-suited to analyze historical price data and make predictions about future price movements. Another reason LSTMs may be well-suited for stock price prediction is that they can handle input data with high dimensionality and noise. Stock prices are influenced by various factors, such as economic indicators, company news, and global events, which can be challenging to quantify or predict. LSTMs are able to handle input data with high dimensionality and high noise, making them well-suited to analyze a wide range of financial data and make predictions about future price movements. Stock price prediction is a complicated problem and no single model would work best for every scenario. LSTM might work well on one dataset and not on another. Also, LSTMs alone might not be enough to accurately predict stock prices, and other techniques such as Sentiment Analysis and Named Entity Recognition, as well as other variables such as news, announcements, market sentiments and trends, may need to be taken into account and can be incorporated into the model.

LSTM stands for Long Short-Term Memory. It is a type of Recurrent Neural Network (RNN) architecture that is used to process sequential data, such as time series or natural language. An RNN processes input data one step at a time, maintaining an internal "memory" of the inputs it has seen. This allows the network to understand patterns or dependencies in the data that may span multiple time steps. However, the simple RNN architecture has a problem known as the "vanishing gradients" problem, where the gradients of the error concerning the parameters of the network become very small as they are backpropagated through multiple time steps. By introducing a number of "gates" that regulate the movement of data into and out of the network's memory, LSTM finds a solution to this issue. These gates can be thought of as "switches" that turn the flow of information on and off, allowing the network to maintain a much longer-term memory of the inputs it has seen. The LSTM architecture is used in various sequential data processing tasks, such as natural language processing, speech recognition, and time series prediction. LSTM is particularly well-suited to tasks where the output depends on a long sequence of prior inputs, such as in natural language processing or stock price prediction, as the output depends on many data

Prediction of the stock market

Stock prices can be predicted using a combination of methods, including technical analysis, fundamental analysis, quantitative analysis, and news analysis.

- Technical analysis: This method involves analyzing charts and historical data to identify patterns and trends that can indicate future stock price movements. Technical analysts look at factors such as support and resistance levels, moving averages, and trading volume to make predictions about future price movements.
- Fundamental analysis: This method involves analyzing a company's financial and economic fundamentals, such as earnings, revenue, and debt, to determine its intrinsic value and potential for future growth. Fundamental analysts look at a company's financial statements and other financial metrics to predict its future performance and stock price.
- Quantitative analysis: This method involves using mathematical models and algorithms to analyze financial data and make predictions about future stock prices. Quantitative analysts use statistical methods and machine learning techniques to analyze large amounts of data and make predictions about future stock prices.
- News analysis: This method involves analyzing news and events that may impact a company's stock price, such as changes in management, mergers, and acquisitions, or regulatory developments.
- It is important to note that no method is perfect, and past performance does not guarantee future results. Additionally, stock prices can be affected by many factors, including macroeconomic conditions, geopolitical events, and investor sentiment, which can be challenging to predict.

Mathematical formulae

$$y = \frac{\partial x}{\partial t} \tag{2.1}$$

and single math symbols x and y are typeset in the mathematical mode.

Chapter 3

Requirements and tools

- functional and nonfunctional requirements
- use cases (UML diagrams)
- description of tools
- methodology of design and implementation

Functional requirements

Functional Requirements for a Stock Price Prediction System

- Data Input: The system must be able to accept and process historical stock prices; depending on the choice of the user, it should be able to read open price, close price, highest price, lowest price, and average price.
- Data Preprocessing: The system must be able to clean, format, and prepare the data for analysis. It should scale the data appropriately to make it easier to fetch it into the neural network.
- Model Training: The system must be able to train an LSTM neural network model using historical data.
- Model Evaluation: The system must be able to evaluate the performance of the trained models.
- Stock Price Prediction: The system must be able to predict future stock prices based on the trained model. It should predict the prices based on the chosen parameters (company name).
- Data Visualization: The system must provide clear and intuitive visualizations of the predicted stock prices and historical data, including line charts and other relevant charts.

- Scalability: The system must be able to handle large amounts of data and be easily scalable to accommodate future growth in data and user base.

Non-functional requirements

Non-Functional Requirements for a Stock Price Prediction System

- Performance: The system must be able to predict stock prices with a response time of fewer than 10 seconds.
- Scalability: The system must be able to handle increasing amounts of data and user traffic without a significant decrease in performance.
- Reliability: The system must have a high level of reliability.
- Maintainability: The system must be easy to maintain, with regular updates and patches to fix bugs and improve performance.
- Compliance: The system must comply with relevant laws and regulations, such as data privacy laws and financial regulations.
- Monitorability: The system must provide monitoring and logging capabilities to track system usage and performance.
- Customizability: The system must be customizable according to user preferences and requirements.

Chosen technology

Python has been chosen for this project as it is a highly recommended language for developing and researching neural networks and AI of all kinds. It is both intuitive to use and offers powerful features when it comes to working with data. Many data scientists use python as it offers a wide range of libraries that make working with data very efficient. Python is also very popular and has a great community supporting this language.

Keras has been chosen for deep learning for its many great features. Firstly, Keras is widely recommended for its simplicity. It helps with working with neural networks and is intuitive to use. Even though it is easy to use, it also offers a high level of abstraction so that the user can write their own methods and classes and experiment with different system architectures. What is more, Keras also has a large community, so it is easy to seek help and discuss possible solutions with other users. And lastly, even though Python has been chosen for this project, it is worth noting that Keras works with many languages and can be used by developers from many different fields.

Yahoo Finance is a financial news and information website that provides a variety of services and tools for individuals and businesses. It offers real-time stock quotes, financial news, market data, and investment research and analysis. It also provides API for fast and easy communication between the site and the program that can be written. That is why it was chosen to ensure constant access to recent data that is fast at the same time. There are few python libraries that help with the reading data directly from Yahoo Finance. For this project, yahoofinancials has to be chosen as it provides easy access to raw data. Other libraries (e.g. yfinance or pandas_datareader) either offer too much processing of the data or need to be faster.

NumPy is a Python library that stands for 'Numerical Python'. It is used for working with arrays of numerical data and provides functions for performing mathematical operations on these arrays. It is a fundamental library for scientific computing in Python.

Pyplot is a sublibrary within the Matplotlib library for Python. It provides a convenient interface for creating a variety of different types of plots and charts, such as line plots, scatter plots, histograms, and more. Pyplot is based on the MATLAB plotting functions and provides a similar interface to the MATLAB plotting functions. It provides a simple way to create plots and charts with just a few lines of code.

Scikit-learn (also known as sklearn) is a machine learning library for the Python programming language. It provides a range of supervised and unsupervised learning algorithms in Python for data mining and data analysis. It provides several preprocessing tools that can be used to prepare data for machine learning algorithms.

PyQt is a framework dedicated to building applications with a graphical interface using Python programming language. PyQt provides a range of modules and classes that developers can use to create GUI elements, such as buttons, dialogs, menus, and more. It also provides support for handling events, managing layouts, and working with various data types and formats.

```
from PyQt6 import QtGui
from PyQt6.QtGui import QPainter, QAction, QColor, QPixmap
from PyQt6.QtWidgets import (
    QApplication,
    QMainWindow,
    QPushButton,
    QMenuBar,
    QMenu,
    QFrame,
    QGridLayout,
    QVBoxLayout,
    QHBoxLayout,
    QWidget,
    QSlider,
    QLabel,
    QComboBox,
)

from yahoofinancials import YahooFinancials

from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, LSTM

from sys import argv

import matplotlib.pyplot as plt

import numpy as np
from sklearn.preprocessing import minmax_scale
```

Figure 3.1: Libraries used for this project.

Chapter 4

External specification

- hardware and software requirements
- installation procedure
- activation procedure
- types of users
- user manual
- system administration
- security issues
- example of usage
- working scenarios (with screenshots or output files)

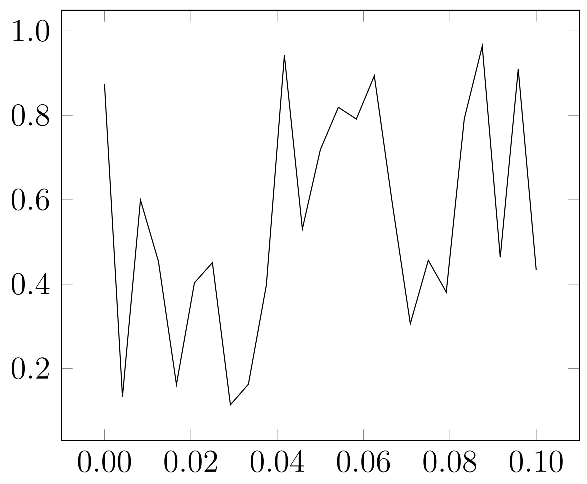

Figure 4.1: Figure caption (below the figure).

Chapter 5

Internal specification

- concept of the system
- system architecture
- description of data structures (and data bases)
- components, modules, libraries, resume of important classes (if used)
- resume of important algorithms (if used)
- details of implementation of selected parts
- applied design patterns
- UML diagrams

Documentation

This section consists of variables that are used later in the project. By changing these values, modification of the application is possible. All variables that matter are stored here. company_name and price_type are used in reading data from Yahoo Finance. In company_name, the NASDAQ (National Association of Securities Dealers Automated Quotations) ticker symbol should be imputed so that the appropriate company data could be read. The ticker symbol is an abbreviation of the company name used in NASDAQ to associate a particular company with its data. In price_type, the type of price that should be read should be specified from open price (open), closing price (close), the highest price of the given period of time (high), lowest price of the given period of time (low) or the volume traded (volume).

```
# Variables
company_name = 'GOOGL' # company NASDAQ name
price_type = 'open' # open, close, high, low, volume


# Time interval
startDate = '2021-06-01'
endDate = '2022-06-01'
interval = 'daily' # daily, weekly or monthly


# Shape of input data
chunkSize = 20
```

Figure 5.1: Variables.

YahooFinancials is easy to use and outputs raw data that can be conveniently used later. data_price_raw stores all data of a given company in the time period between startDate and endDate in chosen intervals. data_prices stores only the prices without the index of date, name, et cetera. Two 'for loops' append the data to the arrays created earlier. The data is assigned so that the neural network's input data is the chosen number of days, and the test data is the next day so that the next day can be predicted on the chosen number of the days before it. For example, for chosen 30 days, the input data are 29 days, and the output should be the one next day (29:1).

```
yahoo_financials = YahooFinancials(company_name)
data_prices_raw = yahoo_financials.get_historical_price_data(
    startDate, endDate, interval)
data_prices = data_prices_raw[company_name]['prices']


for i in range(len(data_prices)):
 data.append(data_prices[i][price_type])


for i in range(chunkSize, len(data_prices) - 1):
 data_prediction.append(data[i-chunkSize:i])
 data_result.append(data[i])
```

Figure 5.2: Working with the stock market data.

In order to train the model, the gathered data should be reshaped so that they have three dimensions. This operation can be done with a few arithmetic operations, but it is more convenient to use reliable methods to do so. Firstly, the given data is transformed with minmax_scale() to normalize the data between 0 and 0.999. Then using the numpy reshape() function, the data is reshaped into three dimensions.

```
# Reshape the data
# into 3 dimensions
data_result, data_prediction = np.array(data_result), np.array
    (data_prediction)


data_prediction = minmax_scale(data_prediction, feature_range
    =(0,.999))
data_prediction = np.reshape(data_prediction, (data_prediction
    .shape[0], data_prediction.shape[1], 1))
data_result = minmax_scale(data_result, feature_range=(0,.999)
```

Figure 5.3: Reshaping data.

As it was stated before in this paper, the chosen type of model for this project is the Sequential model, as it was proven to work best for such solutions in different papers. In this project, several combinations of different layers have been considered, and finally, the layers presented below have been added to the model.

LSTM -> Dropout -> LSTM -> Dropout -> Dense

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(
    data_prediction.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))
```

Figure 5.4: Neural network model.

The model has been compiled using the ‘adam’ optimizer, and for loss, ‘mean squared error’ has been used.

```
model.compile(optimizer='adam',
loss='mean_squared_error')
```

Figure 5.5: Adam compiler.

PyPlot has been used for plotting the result for testing purposes as it provides very clear and easy-to-read graphs. In order to verify how this model can predict future prices, the two graphs have been combined - the one with the actual prices (blue) and the one with the predicted prices (red). Before printing the results, the predicted future prices are put through the min-max scaler to transform them back into real values.

```
# print(data_result)
data_result = minmax_scale(data_result, feature_range=(0,
    highest_price))
future_price = minmax_scale(future_price, feature_range=(0,
    highest_price))
plt.plot(future_price, 'r', data_result, 'b')
# plt.title(("Real and predicted price of " + company_name + "
    stock"))
plt.title(("Real and predicted price of "+Variables.
    company_name[company_enum]+" stock"))
plt.xlabel('Time [days]')
plt.ylabel('Price [{\$}]')
plt.legend(['Predicted price', 'Real price'])
# plt.figure(dpi=300)
# plt.plot(data_result)
# plt.show()
plt.savefig('plots/'+Variables.company_name[company_enum])
plt.clf()
```

Figure 5.6: Presenting obtained data on the graphs.

The layouts have been used to manage the graphical interface efficiently and allow future developers to upgrade the application easily. There are easier and probably even more convenient ways of setting up a GUI for quite a small application, but this approach is recommended and provides a more transparent code. The main window is divided into two areas - upper and lower panels. The upper panel will consist of a graph of the stock prices. The lower panel will have a list of companies whose stocks could be presented on the graph.

```
layout = QGridLayout()
upper_panel = QHBoxLayout()
lower_panel = QHBoxLayout()
layout.addLayout(upper_panel, 0, 0)
layout.addLayout(lower_panel, 1, 0)
```

Figure 5.7: Presenting obtained data on the graphs.

In order to easily show and change pictures, a QPixmap with QLabel has been used. This QLabel has been set to be scalable. To change pictures, a list of companies has been provided using QComboBox. All the companies assigned to variable company_name have been added to the list. Then the event of changing the company in the list has been assigned to a method named update_image.

```
# ******** Image *********


self.image = QPixmap("plots/AAPL.png")
self.label_plot = QLabel()
self.label_plot.setPixmap(self.image)
self.label_plot.setScaledContents(True)


# ****** Menu ******


self.select_mode = QComboBox()
for v in Variables.company_name:
    self.select_mode.addItem(v)
self.select_mode.currentIndexChanged.connect(self.update_image
    )
```

Figure 5.8: Presenting obtained data on the graphs.

This method checks which company is chosen and changes what QPixmap displays accordingly.

```
 def update_image(self, index):
# get the text of the selected item
 selected_item_text = self.select_mode.currentText()


# load the corresponding image and set it to the QLabel
 if selected_item_text == Variables.company_name[0]:
     pixmap = QPixmap('plots/' + Variables.company_name[0] + '.png')
     self.label_plot.setPixmap(pixmap)
 elif selected_item_text == Variables.company_name[1]:
     pixmap = QPixmap('plots/' + Variables.company_name[1] + '.png')
     self.label_plot.setPixmap(pixmap)
 elif selected_item_text == Variables.company_name[2]:
     pixmap = QPixmap('plots/' + Variables.company_name[2] + '.png')
     self.label_plot.setPixmap(pixmap)
 elif selected_item_text == Variables.company_name[3]:
     pixmap = QPixmap('plots/' + Variables.company_name[3] + '.png')
     self.label_plot.setPixmap(pixmap)
 elif selected_item_text == Variables.company_name[4]:
     pixmap = QPixmap('plots/' + Variables.company_name[4] + '.png')
     self.label_plot.setPixmap(pixmap)
 elif selected_item_text == Variables.company_name[5]:
     pixmap = QPixmap('plots/' + Variables.company_name[5] + '.png')
     self.label_plot.setPixmap(pixmap)
```

Figure 5.9: Method responsible for changing the displayed graphs.

UML diagram

This UML diagram has been generated automatically and represents the classes used in the presented project. These classes are not associated with one another. The MainWindow class inherits from QMainWindow class, which is part of a PyQt framework. +————————————————+ | Variables | +————————————————————+ | - price_type: str | | - epochs: int | | - startDate: str | | - endDate: str | | - interval: str | | - chunkSize: int | | - company_name: list[str] | +————————————————————+

+————————————————————+ | NeuralNetwork | +————————————————————+ | - model: Sequential | +————————————————————+ | + build_model(data_prediction) | | + evaluate_model(data_prediction, | | data_result) | | + save_model(name) | | + load_model(name) | | + predict_prices(data_prediction) | +————————————————————

—+

+——————————————+ | MainWindow | +——————————————+ | - predict_button: QPushButton | | - image: QPixmap | | - label_plot: QLabel | | - select_mode: QComboBox | +——————————————+ | + start() | | + setup_ui() | | + predict_button_click() | | _createMenuBar() | +——————————————+

Chapter 6

Verification and validation

- testing paradigm (eg V model)
- test cases, testing scope (full / partial)
- detected and fixed bugs
- results of experiments (optional)

Table 6.1: A caption of a table is **above** it.

	method						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Chapter 7

Conclusions

- achieved results with regard to objectives of the thesis and requirements
- path of further development (eg functional extension . . .)
- encountered difficulties and problems

Appendices

Index of abbreviations and symbols

DNA	deoxyribonucleic acid
MVC	model–view–controller
N	cardinality of data set
μ	membership function of a fuzzy set
$\mathbb{E}$	set of edges of a graph
$\mathcal{L}$	Laplace transformation

Listings

(Put long listings here.)

```
if (_nClusters < 1)
    throw std::string ("unknown number of clusters");
if (_nIterations < 1 and _epsilon < 0)
    throw std::string ("You should set a maximal number of iteration or minimal difference -- epsilon.");
if (_nIterations > 0 and _epsilon > 0)
    throw std::string ("Both number of iterations and minimal epsilon set -- you should set either number of iterations or minimal epsilon.");
```

List of additional files in electronic submission (if applicable)

Additional files uploaded to the system include:

- source code of the application,
- test data,
- a video file showing how software or hardware developed for thesis is used,
- etc.

List of Figures

List of Tables