Predicting the Stock Market Trends Using Machine Learning Techniques

Updated: April 2018
PREDICTING THE STOCK MARKET TRENDS USING MACHINE LEARNING TECHNIQUES

Capstone Report

Student Statement

I, Asmae, have applied ethics to the design process and in the selection of the final proposed design. And I have held the safety of the public to be paramount and has addressed this in the presented design wherever may be applicable.

This project keeps its interaction with Personally identifiable data (PII) minimal. It also warns that its usage is solely for educational purpose and that it is not designed for general release.

Capstone Student: Asmae Boutata

Approved by the Supervisor(s)

Professor Nasser Assem, Capstone Advisor
Acknowledgements

I would like to express my gratitude to my advisor who helped me conduct this project to term. His advising and insight helped me understand the needs of the project and burn through the steps required to conduct the capstone project efficiently and finish it by the deadlines.

I would like to thank all the professors that taught me valuable skills that helped me in this project. This project has required all the skills I learned through the Computer Science curriculum at Al Akhawayn University in Ifrane, going from software design, to programming passing by machine learning.

I would also like to thank all the people that provided me with help while I was developing this project, from people who wrote libraries which I have used to the ones that helped me on forums and discussion boards whenever I was stuck, not forgetting the ones who wrote examples and tutorials over the Internet that helped inspiring my solution.

Finally, I would like to thank people that provided me with moral support during these four years, my friends, my sister and my parents. They have supported me through good and bad, highs and lows. I believe that I wouldn’t have made it without them.

I would not forget to thank anyone who spends time and effort reading this report. I hope it is informative and insightful to you.
# Contents

1 Introduction 1

2 Proposal 2

2.1 Objectives 2

2.1.1 Project Objectives 2

2.1.2 Learning Objectives 2

2.2 Project Timeline 3

2.3 Social and Ethical Implications of the Project 3

2.3.1 Ethical Implications 3

2.3.2 Social Implication 3

3 Feasibility Study 4

3.1 Technological side 4

3.1.1 Comparing Python and R 4

3.1.2 The Machine Learning Libraries 5

3.2 Economical Side 5

3.3 Legal Side 5

3.4 Project Methodology 6

4 Design 6

4.1 Assumptions and Axioms 6

4.2 Choosing Indexes for Our Prediction 7

4.3 Overall Architecture of the System 7

4.4 Component-wise design decisions 8

4.4.1 Data Store 8

4.4.2 Classifiers 9

4.4.3 Processing and Plotting the Data 10
24 Waiting Screen ................................................................. 49
25 Output Graph of Plot ......................................................... 50

List of Tables

1 Timeline of the Capstone Project ........................................... 3
2 Comparison: Python vs. R [1] ................................................ 5
Abstract

Stock markets are unpredictable as the fluctuations in the prices over time depend on several factors. This makes the novice investors full of doubts. However, if we look to the stock markets through a macroscopic lens, we can observe patterns that define the generic evolution of the stock.

Based on those trends, we postulate that if we use Predictive Machine Learning Classifiers (e.g. Regression, Decision Tree, Random Forest, RNN) with stock data that spans over a long-enough period of time, we can approximately predict the behavior of the stocks and the evolution of their prices over time.
1 Introduction

Stock markets are a fascinating piece of human civilization artifacts. As a matter of fact, in their basic definition they set the price on non tangible assets related to the equity of some enterprise or a market. They are very ubiquitous and can define how well a country’s economy is doing, set the value of the country’s currency, decide if it is suitable for investments or not.

From a simple economics perspective, the value of stocks is defined by the "demand/supply" paradigm. High demand and low supply will make the stock price skyrocket while the opposite will make the price plummet. However, there are several factors that affect the stock market either favorably or unfavorably (e.g. politics, companies’ ethics and behaviors).

The complication of stock markets sways the general public from investing in equity. They see that the market is volatile and that the risk outweighs the benefit. This sentiment is amplified by their lack of understanding of the micro and macroeconomics controlling the stock market and the fluctuations that stocks go through. Those factors make the stock market very hard to predict.

In order to encourage people to invest more in the stock market, we build software that can help predicting the trend of a stock market from its fluctuations over a long period of times. In fact, the macro economical changes are very static and do not get affected heavily unless an unexpected event happens. Thus we combine that theory with Machine Learning / Deep Learning classifiers in order to predict the trend of stock markets in the future. In this project, we verify the veracity of our proposal by building the software and testing it against known stock and comparing the efficiency of several classifiers. Finally, we build a Proof-of-Concept product that illustrates the way this project can be productized and presented to the general public.
2 Proposal

Stock markets are hard to predict; the fluctuations of the prices of stocks over a period of time depend on several internal / external factors that make a normal human unable to innately predict their evolution over the short and medium terms. However, if we look into the macroscopic evolution of the stocks, we can identify some trends that define a generic evolution of the stock.

Based on those trends, we postulate that if we use Predictive Machine Learning Classifiers (e.g. Regression, Decision Tree, Random Forest) with stock data that spans over a long-enough period of time, we can approximately predict the behavior of the stocks and the evolution of their prices over time.

2.1 Objectives

By doing this project we aim to achieve several project-oriented and learning objectives. They are as follows:

2.1.1 Project Objectives

- Providing a software suite that can model and predict the macro-fluctuations of the stock market.

- Comparing the accuracy of different ML Classifiers applied to stocks

- Helping beginners to understand the Stock Market Evolution

2.1.2 Learning Objectives

- Learning Supervised Machine Learning Techniques and applying them to a real life concept.

- Understanding the evolution of stock market values and the different variables that affect them.
• Getting Familiarized with programming languages and frameworks used for mathematical processing and machine learning (e.g. python, numpy, scipy, TensorFlow, R)

2.2 Project Timeline

We predict that this project is going to span over 12 weeks between January and May 2018.

We provide the following tentative schedule for this project:

<table>
<thead>
<tr>
<th>Week</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>Submitting the Project Proposal and Initial Review of Machine Learning</td>
</tr>
<tr>
<td>Week 2,3</td>
<td>Literature Review of Machine Learning Techniques</td>
</tr>
<tr>
<td>Week 4,5,6</td>
<td>Developing the Experimentation Software in Python/R</td>
</tr>
<tr>
<td>Week 7,8</td>
<td>Experimentation with Real Life Stock Data Information provided from public APIs</td>
</tr>
<tr>
<td>Week 9,10</td>
<td>Analyzing and Discussing the Results</td>
</tr>
<tr>
<td>Week 11,12</td>
<td>Writing the Capstone Report + Presentation</td>
</tr>
</tbody>
</table>

Table 1: Timeline of the Capstone Project

2.3 Social and Ethical Implications of the Project

In this section, we discuss the multiple implications of our project has. We discuss said implication from both social and ethical lenses, while the next chapter of this report illustrates the Engineering implications in the form of a feasibility study.

2.3.1 Ethical Implications

This Project does not have any Ethical Implications that we have to consider. The stock market data we are using for our classifiers is publicly available and we do not include any insider knowledge. The libraries, tools and frameworks we are using are also available under various Open-Source Licenses that guarantee their usability in non-commercial applications.

2.3.2 Social Implication

This project aims to simplify the concepts related to stock markets and allow more people to approximately predict the fluctuations that happen to stock prices. If productized and if the
machine learning classifiers are somewhat accurate, it may enable more people to invest in stocks and encourage more people to evaluate the risks and invest in stocks.

3 Feasibility Study

In order to evaluate if the project can be done in the given time frame, we are using the TEL-evaluation methods, where we cover the feasibility of the project from a technological, economical and legal perspective. Those perspectives would help us have a broad vision on the requirements and implications related to the project. We also discuss in this section the methodology used in conducting the project.

3.1 Technological side

This project would be developed using technologies and libraries pertinent to data analysis and machine learning. This puts a strain on the choice of the programming language and libraries chosen to conduct the project and build different experiments. In other words, the programming language should be able to easily parse and process large amounts of data that is organized in a specific format, it should also have well tested, commonly used number processing libraries and machine learning classifier implementations. Thus, we narrow our choices to python vs. R, as they seem to be the leading programming languages in the Data Processing community.

3.1.1 Comparing Python and R

In order to choose a Programming Language, we conduct a comparison of the features offered by both Python and R. Even if python does not support Data Processing off the shelf, python extensions that implement data processing and ML exist and are readily available on pypi. The following table illustrates the comparison between Python and R.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Python</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to Learn</td>
<td>Yes</td>
<td>Steep Learning Curve</td>
</tr>
<tr>
<td>Suitability for Data Manipulation</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Suitability for Repeated Tasks</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Suitability for Statistical Analysis</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support</td>
<td>Stackoverflow / Forums /tutorials</td>
<td>Mailing Lists</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Academic Publications</td>
</tr>
<tr>
<td>Debuggability</td>
<td>Easy</td>
<td>Hard (Learning curve)</td>
</tr>
</tbody>
</table>

Table 2: Comparison: Python vs. R [1]

3.1.2 The Machine Learning Libraries

In Python, there are several libraries that are used to implement Machine Learning Classifiers. This is mainly due the active scientific and data-analysis communities that develop libraries to help them with their research. We do think that we are going to evaluate different Machine Libraries (e.g. Tensorflow, scikit-learn, sklearn) during the implementation phase and rather mix and match different technologies to use their strong points.

3.2 Economical Side

This project will be based on Free and Open Source Technologies and Libraries that are readily available to developers and scientists, free of cost. This means that we don’t have to worry about costs related to licensing or reusing source code and that the only costs related to the project are related to the time and the effort spent into developing it.

This project is also developed for educational purposes and is not expected to be monetized in any manner. Our aim behind it is to evaluate the veracity of a postulate and understand better the intricacies of both the stock markets and machine learning. However, if the postulate is verified to be true, opportunities to monetize this project may be investigated more thoroughly.

3.3 Legal Side

Since we are using open source technologies to build this project, they grant us with the following freedoms as specified by the Free Software Foundation [2]:
• Freedom 0: The freedom to run the program for any purpose.

• Freedom 1: The freedom to study how the program works, and change it to make it do what you wish.

• Freedom 2: The freedom to redistribute copies so you can help your neighbor.

• Freedom 3: The freedom to improve the program, and release your improvements (and modified code)

3.4 Project Methodology

For this Project, we follow the same methodology used for academic research projects. We have already come up with a hypothesis/postulate. The next steps would be to conduct a literature review, where we explore previous efforts to solve similar problems, we build our experimental software, and we conduct the experiment, verify our postulate and then discuss the results and draw conclusions up.

4 Design

In the following we discuss the over architecture, design decisions and assumptions we have made to conduct this project.

4.1 Assumptions and Axioms

In order to simplify a problem as complex as predicting the stock market, where multiple variables decide the fluctuations of the prices and the volumes traded, we make the following assumptions:

• We assume that frenetic micro-changes (sudden spikes / valleys) of the stock price do not affect the overall evolution of the stock price. In other words, only the macroscopic changes matter.
• We assume that the stock market indexes are indicative of the evolution of the overall stock price and of the money traded in the market.

• We only consider major stock markets in the United States, as explained in the following section

4.2 Choosing Indexes for Our Prediction

There are several companies that are publicly traded in several stock markets in the world. As a matter of fact, every country/economic entity has a couple of stock markets where companies are traded (e.g. DAX in Germany, FTSE-100 in the UK, Nasdaq in the US, MASI in Morocco).

For our project, we choose to focus on the three major stock market indexes in the United-States: Dow Jones Industries, NASDAQ and Standard and Poor’s 500 Index. Those indexes have their Data quite accessible and represent over 40% of the capitalization of the markets globally. [3]

4.3 Overall Architecture of the System

From a high level perspective, the system is designed following a 3-tier Architecture: A User-facing Interface, a Business Logic Component and a Data Acquisition component. The user-facing interface allows the user to interact with the business logic, and we are designing it as two components: the command-line interface (shell) for input and the data-plotting interface (output). In order to make the implementation efficient, we have decided against having a more developed graphical user interface, as it is not permitted by the time span of the project. It may be designed as future work, however.

The Business logic interface consists of different Machine Learning Classifiers that we are comparing in this project. Each set of data goes over the four classifiers we have chosen (i.e. Linear Regression, Decision Trees, Random Forest, Recurrent Neural Networks) in order to have different predictions from the same data.

Finally, the data acquisition component in an API client that acquires the stock market data from a third party provider. We have chosen a provider that would return a dataset that
is current, very detailed and that we can easily process (in JSON format).

The following Figure 1 illustrates the general architecture of the system and explains how the components integrate with each other.

Figure 1: N-Tiered Architecture of the Predictor Software System

4.4 Component-wise design decisions

In this section, we take a deep dive into the design decisions we have taken to build this project. We demonstrate different assumptions and reason that pushed us to make certain decisions and explain the role of every component and how it fits in the general scheme of the project.

4.4.1 Data Store

We have specific criteria in mind while choosing the data store. It needed to be accessible freely or with a minimal charge, provide the data in an easily processable format and provide data that is both accurate, detailed and current. In order to have the classifiers as accurate as possible, the data should be very detailed and go back in time as much as possible. Thus, our choice of data provider was Alpha Vantage.
Alpha Vantage is a web service that provides real-time and historic Stock Market and Cryptocurrency Data. It only requires a Free, one-time registration in order to obtain an API key. It provides historic and current data for the stock market with up to 20 years of history. It has different granularities (Daily, Weekly, Monthly), and it can also provide real-time history, providing a resolution of 1 minute or higher of the different change in a stock’s price. They have a clean, documented, public API that returns JSON-encoded data.

4.4.2 Classifiers

In statistics and machine learning, classification is the problem of matching a new observation to a set of categories, using a large set of data that contains observations that are already known [4]. For example, a training set would consist of matching the gender (male/female) to a person’s profile picture on social media. In our case our known observation would be the price of a specific stock on a day $X$ and how it evolves on day $X + 1$ compared to day $X$ and to days $\leq X$. Our aim is to find patterns in the stock market fluctuations.

In the terminology of machine learning, classification is considered an instance of supervised learning. Supervised learning is when you have a set of tuples $\{(X, Y)\}$ where $X$ are the input variables and $Y$ are their corresponding output variables and you use an algorithm (Classifier) to approximate the function $f$ that links the input and the output such that:

$$Y = F(X)$$

Our approximation should be accurate enough that when we have new input data $X'$ we can predict new output for the data $Y'$. For this project we chose four widely used classifiers:

4.4.2.1 Linear Regression

Linear Regression [5] is a supervised machine learning algorithm that predicts the output on a continuous range and with a constant slope. It is used to find the line of best fit of a set of scattered data and can be used to predict values within a continuous range instead of categorizing inputs.
4.4.2.2 Decision Trees

Decision Trees [6] are a supervised learning method used for classification. It aims to build a model that predicts the value of a variable by learning simple decisions from the data features. Decision trees are usually simple to understand and interpret and have a logarithmic usage cost. They also do not require a lot of data processing beforehand. However, they tend to be overfitting (i.e. cannot generalize data well), unstable (i.e. a minimal change of the data would change the entire tree) and require constant tree balancing.

4.4.2.3 Random Forest

Random Forests [7] are a supervised learning method used for classification. They aim to address the shortcomings of decision trees. They operate by creating a multitude of decision trees during the learning phase and outputting their mode for classification. Random Forests are one of the most accurate classifiers, it also can run efficiently of large data sets. However, it tends to overfit some edge case datasets as well.

4.4.2.4 Recurrent Neural Networks

Recurrent neural networks [8] are a type of Neural Networks (interlinked computing systems that can learn from examples, inspired by the animals neurons/brains) that allow extrapolating the time behavior for a time series. Recurrent Neural Networks use their memory in order to process outputs, making them suitable for recognizing continuous actions like cursive handwriting.

4.4.3 Processing and Plotting the Data

For processing the data we use numpy. It offers algorithms for processing large amounts of data efficiently using techniques and algorithms inspired from the world of Computational Mathematics. The core library is written in C for high-efficiency and has python bindings using Cython [9].
For plotting the data, we use Bokeh, it is an interactive visualization library that outputs to a browser for presentation. Its aim is to provide an elegant, and precise representation of different graphics. It also manages to plot graphs with high performance over very large datasets. [10]

5 Implementation

In this section, we discuss the multiple techniques we have used to implement the project, the way we design and use the classifiers, the way we get and process the data and the way we plot the data. We provide a detailed insight on the implementation and inner workings of the software.

5.1 The Command Line Interface

The command line interface is the main way for a user to interact with our software. This interface is meant to be simple and intuitive. It starts by providing the users with contextual menus allowing them to choose a classifier, a prediction start date and a stock index.

Menus are provided through one function that changes what it prints to the screen based on what step the program is into. It starts by displaying the choices for the Index we want to predict, then moves on to picking a date for the prediction and then a Classifier. User input is read using the python’s own input() function that reads inputs, and we provide failsafe mechanisms for invalid input that terminate the program with an error.

In designing the command interface, we use the six library that help us bridge the incompatibilities between python 2 and python 3.

5.2 Data API

The data API module allows the communication with the external data provider. The API is only offered through an authenticated access through the use of an API key. The API we use provides realtime and historical stock data in 4 time settings: (1) intraday, (2) daily, (3) weekly,
and (4) monthly. Daily, weekly, and monthly time series contain up to 20 years of historical data. Thus, in our query we need to provide the granularity we want. The URL we use looks as follows:

http://alphavantage.co/query?[API Parameters]

where the API parameters are as follows:

• **apikey**: (required) the API key that allows you access the API

• **function**: (required) The time series of your choice, in our case: function=TIME_SERIES_DAILY

• **symbol**: (required) The ticker symbol we want to look for. In our case, we use IXIC, INX, DJI.

• **outputsize**: (optional, default=compact) the size of the data we want to get, in our case: outputsize=full

• **datatype**: (optional, default=json) the type of data we return (json or csv) we chose json as it is easily processable by python.

An example for a URL with full query parameters will look like the following

https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=MSFT&outputsize=full&apikey=demo

The API query returns a json object that contains the metadata related to a stock symbol (e.g. name, the information contained, the time zone of the stock market, the amount of data available), as well as different metrics used to measure a stock (e.g. the price on open, close, high, low, and total volume exchanged in the market). A typical output looks as follows:

```json
{
    "Meta Data": {
        "1. Information": "Daily Prices (open, high, low, close) and Volumes",
        "2. Symbol": "MSFT",
        "3. Last Refreshed": "2018-04-11",
        "4. Output Size": "Full size",
        "5. Time Zone": "US/Eastern"
    }
}
```
"Time Series (Daily)": {
    "2018-04-11": {
        "1. open": "92.0100",
        "2. high": "93.2900",
        "3. low": "91.4800",
        "4. close": "91.8600",
        "5. volume": "24860195"
    },
    "2018-04-10": {
        "1. open": "92.3900",
        "2. high": "93.2800",
        "3. low": "91.6400",
        "4. close": "92.8800",
        "5. volume": "26865981"
    },
    "2018-04-09": {
        "1. open": "91.0400",
        "2. high": "93.1700",
        "3. low": "90.6200",
        "4. close": "90.7700",
        "5. volume": "31533943"
    },
    "2018-04-06": {
        "1. open": "91.4900",
        "2. high": "92.4600",
        "3. low": "89.4800",
        "4. close": "90.2300",
        "5. volume": "38026000"
    }
}
In order to simplify the project and make it fit in the short timespan we have, we decide to isolate the close price as the only metric we use for our prediction. We believe that it’s the most important metric that a member of the general public would care about in order to make their decision if they should buy stock or not. We process the above JSON data in order to have a list of tuples that correspond to the following general schema

\[(\text{date}_1, \text{close}\_\text{value}_1), (\text{date}_2, \text{close}\_\text{value}_2)\ldots(\text{date}_N, \text{close}\_\text{value}_N)\]

where the list is sorted by date in an ascending order.

5.3 Classifiers

In this section, we discuss the general structure of the classifiers we have implemented. Most of the Machine Learning classifiers are provided by the Scikit Learn library in Python. The Deep Learning Classifier is provided by TensorFlow through Keras.

5.3.1 Machine Learning Classifiers

We have defined a python generic interface that defines the methods and variables of different classifiers, as those share the same code mostly, they only differ in what kind of classifier is used and the return type of the classifier’s predict function. The generic classifier implementation would look like the following:

```python
from utils import plotter
class Classifiers(object):
    def __init__(self, data, datelimit):
        self.raw_data = data
        self.datelimit = datelimit
```
Listing 2: Generic Representation of a Classifier

Where the Constructor __init__() takes the data provided by the Data API module and the date limit that represents the date on which to start the prediction.

The function predictnext() would take the data between the start of the data and the ‘nextdate’. It would then reshape the data into $n \times 1$ matrices, fits the data to the classifier and predicts the next point in the graph.

The function predict() repeats predictnext() for the duration between $[datelimit, today)$

The functions plot() and plotweb() prepare the data to be plotted either by Bokeh (see [5.4]) or The Web Interface (see [7.1]).

### 5.3.2 Deep Learning Classifier

The deep learning classifier also inherits from the Machine Learning classifier. However, it requires a specific way to prepare the training set data. It splits it into two sets that are as follow $X = \{data[0], data[1]...data[n - 1]\}, Y = \{data[1], data[2]...data[n]\}$. This alternating pattern allows the classifier to correlate different data points with their neighbors. The Deep Learning Classifier skeleton looks as follows:

```python
from utils import plotter
class Classifiers(object):
    def predictnext(self, nextdate):
        pass
    def predict(self):
        pass
    def plot(self, stock_index):
        pass
    def plotweb(self, classifier, stock_index):
        pass
```
```python
def __init__(self, data, datelimit):
    self.raw_data = data
    self.datelimit = datelimit

def create_dataset(self, dataset, look_back=1):
    pass

def predict(self):
    pass

def plot(self, stock_index):
    pass

def plotweb(self, classifier, stock_index):
    pass
```

Listing 3: Deep Learning Classifier

The remaining functions assure a similar role as in the normal classifiers. And predictnext is unused.

### 5.4 Plotter

The plotter is a utility tool that leverages the Bokeh library in order to plot and visualize the data into graphs that can be easily understood by humans. It operates by processing the real data and predicted data into a series of Xs and Ys that can be fed to a plot function, then it renders the graphs into HTML. It looks as follows:

```python
def plot(real_xs, real_ys, predicted_xs, predicted_ys, title, x_label='days', y_label='Close Value'):
    p = figure(plot_width=1400, plot_height=1400, title=title, x_axis_type='datetime')
    p.xaxis.axis_label = x_label
    p.yaxis.axis_label = y_label
    p.line(real_xs, real_ys, line_color="red", legend='real values')
```
The function takes the lists of Xs and Ys and renders the real values in a red line and the predicted values in a green line.

### 5.5 Workday Calculator

While developing this project, we have encountered an odd user input related issue. The date that the user may input may be an invalid Business Day, be it a weekend or a bank holiday. This is a problem as during those days, there are no publicly traded stocks in the market.

To overcome this difficulty, we have written a date normalizer that finds either the next or previous business day (based on where the normalizer is called).

The principle behind the date normalizer is very simple. Instead of using the operating system’s calendar – that is usually agnostic to business days, it uses a modified calendar modeled after the United States’ Business Calendar, and tries to find the closest business day, depending that if we want to go forwards or backwards in time.

### 6 Experimentation and Results

In this section, we discuss the experiments we have conducted in this project to verify the hypothesis that we can predict the behavior of the stock price time series using machine learning / deep learning classifiers.

#### 6.1 Experimental Setup

In this set up, we use the command line interface in order to compare the output for different classifiers (e.g. Linear Regression, Decision Trees, RandomForest, RNN) for 3 different indexes (Dow Jones (DJI), NASDAQ (IXIC), and S&P 500 (INX). We use January, 01 2018 as our
prediction start date. The experiments ran on an Intel i7 Machine clocked at 2.7 GHz with 8 GB of RAM, using Python3. The real values are represented in red and the predicted ones are in green.
6.2 Results and Discussion

6.2.1 Using Linear Regression

Figure 2: Data Evolution for DJI Predicting from 2018-01-01
Figure 3: Data Evolution for IXIC Predicting from 2018-01-01
From figs. 2 to 4 we observe that the linear regression classifier is quite good at predicting the general trend of growth for the stock market, it is not good at predicting the day-to-day variations in the graph, and thus it provides predicted values that are way off than what we expect. Thus, linear regression is useless for our application.
6.2.2 Using Decision Trees

Figure 5: Data Evolution for DJI Predicting from 2018-01-01
Figure 6: Data Evolution for IXIC Predicting from 2018-01-01
In our experiment, we have used a decision tree classifier. It is very slow and the loss is quite minimal. From figs. 5 to 7 we observe that the classifier can predict the time series with a stunning accuracy. However, each experiment would take 15-20 minutes to run on i7 machine at 2.7 GHz.
6.2.3 Using Random Forest

Figure 8: Data Evolution for DJI Predicting from 2018-01-01
Figure 9: Data Evolution for IXIC Predicting from 2018-01-01
In our experiment, we have used a Random Forest classifier with 15 jobs and the random state as 0. It is slow and the loss is quite minimal, but it’s faster than decision trees. From figs. 8 to 10 we observe that the classifier can predict the time series with a stunning accuracy. However, each experiment would take 5-7 minutes to run on i7 machine at 2.7 GHz.
6.2.4 Using a Recurrent Neural Network

Figure 11: Data Evolution for DJI Predicting from 2018-01-01
Figure 12: Data Evolution for IXIC Predicting from 2018-01-01
In our experiment, we have used a recurrent neural network with 5 epochs, a batch size of 10 and 2 LSTMs, as it was fast to execute with a minimal loss. From figs. 11 to 13, we observe that the classifier can predict the time series somewhat accurately, with a lot of loss for the INX predictor. In order to have better results while conserving high performance, the classifier
should be tuned better.

7 Productization

In order to target the general audience, a user-friendlier interface should be provided. As a matter of fact, not all users are able to run a command-line interface tool and interact with it correctly. Further, using machine-learning classifiers and deep learning is computationally heavy. Having users run their CPUs/GPUs at full thrust for prolonged periods may be a hinderer against using our solution.

In this section, we demonstrate the multiple steps we have gone through to transform our project into a Proof-of-Concept Product that can be expanded upon and then released to the general public once the shortcomings are fixed, and the different business logic processors are hardened. We have opted to design the PoC as a web service as it provides the most benefits we are looking for in this process.

7.1 Graphical User Interface

The User Interface is built using HTML5 /CSS3 and JavaScript. In its core, it is composed of a very simple user form that captures user input, a button that calls in the JavaScript that ensures the communication with the backend.

7.1.1 User Input Form

```html
<form class="pure-form pure-form-stacked" style="margin:1em;">
  <fieldset>
    <label for="ticker">Symbol</label>
    <select id="ticker" class='form-control'>
      <option value='DJI' selected>Dow Jones International</option>
      <option value='IXIC'>Nasdaq</option>
      <option value='INX'>S&amp;P 500</option>
    </select>
  </fieldset>
</form>
```
7.1.2 Querying the backend

To Query the Backend, we use JQuery as our primary mean to conduct HTTP requests. We write a modified AJAX POST function that can post the form in a json format. It looks as follows:

```javascript
function postJson(url, datum) {
    $.ajax(url, {
        data : JSON.stringify(datum),
        contentType : 'application/json',
        type : 'POST',
        success: function(data) {
            plot_the_graph(data)
        }
    });
}
```
Listing 6: JQuery Modified POST

if the function succeeds, it calls the code that plots the graph in the UI.

### 7.2 API Backend

The service’s backend is provided through a REST API that listens for HTTP-POST queries containing the information concerning the classifier, the date and the index and then sends it to the business logic processors. The REST API returns a json response containing all the data needed to plot the graphs.

#### 7.2.1 API Definition

In this section, we provide the definition of the API we are writing to productize the project in swagger/Open-API specification 2.0. Swagger/OAS is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful Web services [11]. This following swagger file defines a set of endpoints and settings required to describe our API. The swagger files can be used to generate HTML documentation of the API or as input to code generation tools in order to generate the API Clients / Server.

```json
{
  "swagger": "2.0",
  "info": {
    "title": "Stock Predictor API",
    "version": "v1"
  },
  "host": "localhost:8080",
  "schemes": [
    "http"
  ],
  "consumes": [
```
"application/json",

"produces": [
  "application/json"
],

"definitions": {
  "queryObject": {
    "properties": {
      "ticker": {
        "type": "string"
      },
      "date": {
        "type": "string"
      },
      "classifier": {
        "type": "string"
      }
    },
    "required": [
      "ticker",
      "date",
      "classifier"
    ],
    "type": "object"
  }
},

"paths": {
  "/predict": {
    "post": {

"description": "Request the Graph for a Predictor",
"operationId": "POST_predict",
"responses": {
  "200": {
    "description": ""
  },
  "500": {
    "description": ""
  }
},
"parameters": [
{
  "schema": {
    "$ref": "#/definitions/queryObject"
  },
  "in": "body",
  "name": "body",
  "required": true
}
]
}

Listing 7: API definition in Swagger (OAS 2.0) Format
7.2.2 Backend Code

The backend code was auto generated from the API definition defined in Swagger. The code generation tool used `swagger_py_codegen` converts a well defined API specification into python code. It helps ensure that the return codes and object correspond to the definition.

The backend uses Flask, which is a MVC framework for python, allowing the quick modeling and design of web services and APIs using python.

In the remainder of this section we discuss the security, and routing implications of the auto generated code.

7.2.2.1 Routing Queries to the right URL

In order to route queries to the right URL, the generated codebase defines a list of routes that link a flask resource to their API end point. This is generally defined as follows:

```python
routes = [
    dict(resource=Predict, urls=['/predict'], endpoint='predict'),
]
```

7.2.2.2 Security

With concern to security, the generated codebase ensures that several functions are happening.

First of all, the generated codebase sanitizes the input by verifying that every required key, value in the json object exists. In case this is wrong, the code returns an error 422 - Unprocessable entity, then it proceeds to ensure that the strings are correctly escaped. Then, when the input is valid, it would hand it to the Flask Resource for further processing.

Once the flask resource is done, then it would similarly revalidate the output before sending it back to the user. If the output is not valid, it will replace it with a dummy value. In further steps towards productization, this behavior should be replaced with an ERROR 500 - Internal Server Error.
7.3 Updated System Architecture

After the new user interface is finished, the system architecture will look as follows. Since we got rid of the CLI from the interface tier, we can easily distribute only the interface tier to users through a web browser while making the API endpoint publicly available.

7.4 Current Limitations

The current productized interface is limited and meant to be a Proof-of-Concept. It cannot be used as a final product in under any circumstances. As a matter of fact, it doesn’t implement any JavaScript based frontend input sanity-checks as it takes exponential times to ensure that the UI is well sanitized and the input is correct.

Furthermore, the choices for Ticker indexes are hardcoded in the html file as there is currently no service that provides a full list of ticker indexes that can be accessed through the data tier.
8 Conclusions and Future Work

Through this Capstone project, we were able to show that it is feasible to build a stock market predictor that use Machine Learning / Deep learning techniques. Building such a piece of software will make it easier for people to get interested in the stock market and understand their inner workings.

In order to achieve this aim, this project can be extended in several manners:


• Extend the API to dynamically return the list of supported classifiers and Index Tickers.

• Speed up the process by precalculating the result for popular index tickers and caching user queries.

• implementing the javascript sanitization methods.

• Adding Authentication to the Backend API.
A Appendix: Machine Readable Specification

BOUTATA Asmae

CSC

FORECASTING THE STOCK MARKET

ASSEM N

Spring 2018

This capstone project aims to explore ways machine learning can help predicting and forecasting the evolution of stock values in the stock market, helping novice investors make some gains and choose the best stock options to invest in.

This project will be designed in three steps. The first one would be a literature review that analyzes the different machine learning techniques that we can use to build the project, such as the use of supervised learning classifiers and regressions to predict future data points. In this phase, we will decide on the specific classifiers we want to use for this project. The second phase is the learning phase, where we implement programs that consume stock market data and apply them to different ML classifiers. This phase would require access to stock market data through APIs that are available freely online. Finally, the third phase would be where we produce predicted stock market data from the learning phase and according to different classifiers.

After completing the design, implementation and experimentation phases, a discussion and analysis of the result we obtain is necessary, we will compare those results with each other and also with real life data in order to draw conclusions on what classifiers work better in the case of the stock markets and have a better idea on how it can be productized.

The software and the technique and the process adopted are going to be built using open-source technologies that are freely available to programmers online, and it should be thoroughly tested. We do claim it has no social implications, as its main goal is to explore a new field and validate the postulate that machine learning can help predicting the evolution of the stock market.
B Screenshots of the Application

In this appendix, we show the screenshots of the command line interface and web interface we designed for our application.

B.1 Command Line Interface

![Figure 15: Launching the Command Line Interface](image)

Figure 15: Launching the Command Line Interface
Figure 16: Menu 1: Choosing a Ticker Index
Figure 17: Menu 2: Picking a Prediction Date
Figure 18: Menu 3: Picking a Classifier
Figure 19: Output: Graph Plotted to the browser using Bokeh

This section shows the command-line interface designed for our project in figs. 15 to 19
B.2 Web Interface

Figure 20: Welcome Screen of the Web Interface
Figure 21: Dropdown 1: Choosing an Index
Figure 22: Dropdown 2: Choosing a Classifier
Figure 23: Date picker 1: Choosing a split date
Figure 24: Waiting Screen

This section shows the command-line interface designed for our project in figs. 20 to 25.
Figure 25: Output Graph of Plot

References


