

**UNIVERSIDAD AUTÓNOMA DE MADRID  
ESCUELA POLITÉCNICA SUPERIOR**



**Bachelor as Ingeniería Informática Bilingüe**

## **BACHELOR THESIS**

**Develop a web application to interact with predictive algorithms about the stock market.**

**Autor: Juan Llamazares Ruiz**

**Tutor: Francisco Saiz López**

**June 2023**

**All rights reserved.**

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución comunicación pública y transformación de esta obra sin contar con la autorización de los titulares de la propiedad intelectual.

La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y sgts. del Código Penal).

© 3 de Noviembre de 2017 por UNIVERSIDAD AUTÓNOMA DE MADRID Francisco Tomás y Valiente, nº 1  
Madrid, 28049  
Spain

**Juan Llamazares Ruiz**  
**Develop a web application to interact with predictive algorithms about the stock market.**

**Juan Llamazares Ruiz**

C\ Francisco Tomás y Valiente No 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

# Agradecimientos

---

# Abstract

---

The interest of modern technology has made it easier to stay informed about the stock market and make investments accordingly. With growing interest among individuals and investors to actively participate in the stock market, having access to tools that help them make better decisions has become increasingly desirable. To study in more detail the stock market is a complex and dynamic asset and can be difficult for these individuals to keep up with the latest trends, company results and world news. By developing a web application that can interact with predictive algorithms, investors can access real-time data and historical trends to make informed investment decisions.

The main purpose of this Bachelor Thesis will be to demonstrate explore the possibility of obtaining a predictive algorithm on the stock market that can be interacted with through a developed web application tool to help individuals and investors to optimize their time and effort to decide the best financial asset. We will consider the utilization of Recurrent Neuronal Networks (RNNs) to support a more accurate stock prediction, and the development of a forecasting model that could be easily implemented into the web application. Finally, we will illustrate the user the strategies to measure the accuracy of the algorithm and the performance of the developed web application.

Furthermore, we go into detail the many technical aspects of web application development such as user interface design and implementation, use of web services, front-end and back-end development.

# Keywords

---

Web application, predictive algorithms, stock market, stock prediction, machine learning, neuronal networks, LSTM, GRU, data analysis, market trends, historical data.



# Resumen

---

El interés de la tecnología moderna ha hecho más fácil mantenerse informado sobre el mercado de valores y realizar inversiones en consecuencia. Con el creciente interés de particulares e inversores por participar activamente en el mercado bursátil, cada vez es más deseable tener acceso a herramientas que les ayuden a tomar mejores decisiones. El mercado bursátil es un activo complejo y dinámico y puede resultar difícil para estos particulares mantenerse al día de las últimas tendencias, los resultados de las empresas y las noticias mundiales. Mediante el desarrollo de una aplicación web capaz de interactuar con algoritmos predictivos, los inversores pueden acceder a datos en tiempo real y a tendencias históricas para tomar decisiones de inversión con conocimiento de causa.

El objetivo principal del Trabajo de Fin de Grado será demostrar explorar la posibilidad de obtener un algoritmo predictivo sobre el mercado de valores con el que se pueda interactuar a través de una herramienta de aplicación web desarrollada para ayudar a particulares e inversores a optimizar su tiempo y esfuerzo para decidir el mejor activo financiero. Consideraremos la utilización de Redes Neuronales Recurrentes (RNNs) para apoyar una predicción bursátil más precisa, y el desarrollo de un modelo de predicción que pueda ser fácilmente implementado en la aplicación web. Por último, ilustraremos al usuario las estrategias para medir la precisión del algoritmo y el rendimiento de la aplicación web desarrollada.

Además, entraremos en detalle en los numerosos aspectos técnicos del desarrollo de aplicaciones web, como el diseño y la implementación de la interfaz de usuario, el uso de servicios web y el desarrollo front-end y back-end.

## Palabras clave

---

Aplicación web, algoritmos predictivos, bolsa, predicción bursátil, aprendizaje automático, redes neuronales, LSTM, GRU, análisis de datos, tendencias del mercado, datos históricos.



# Table of contents

---

<b>Introduction .....</b>	<b>1</b>
1. Motivation .....	1
2. Objectives .....	1
<b>State of the art .....</b>	<b>3</b>
1. Machine Learning .....	3
Recurrent neuronal networks .....	3
Long Short-Term Memory neuronal network .....	3
Gated Recurrent Units (GRU) .....	4
Long Short-Term Memory (LSTM) vs Gated Recurrent Units (GRU) .....	5
2. Python 3 .....	5
Framework Django .....	6
Python libraries .....	6
3. Javascript .....	7
4. ChartJS .....	7
5. Bootstrap .....	7
<b>Analysis and design .....</b>	<b>9</b>
1. Back-end development .....	9
Datasets .....	10
Neuronal networks .....	11
2. Front-end development .....	12
3. Methodology .....	12
<b>Implementation .....</b>	<b>13</b>
1. Back-end development .....	13
Process to develop neuronal network .....	11
2. Front-end development .....	15
Home view .....	15
Dashboard view .....	16
Frequently asked question's view .....	17
<b>Testing and results .....</b>	<b>18</b>
1. Introduction .....	18
2. Metrics .....	18
3. LSTM model tests .....	19
4. GRU model tests .....	20
5. Alternatives and improvements .....	21
<b>Conclusions and future work .....</b>	<b>23</b>



1. Conclusions .....	23
2. Future work .....	23
<b>References .....</b>	<b>24</b>
<b>Terminology .....</b>	<b>27</b>
1. Technical terms .....	27
2. Financial terms .....	27
<b>Appendice .....</b>	<b>25</b>
<b>Appendice .....</b>	<b>31</b>
<b>Dependencies and libraries .....</b>	<b>34</b>



# List of figures

---

<b>Figure 1:</b> LSTM architecture .....	4
<b>Figure 2:</b> GRU architecture .....	5
<b>Figure 3:</b> Django framework architecture .....	6
<b>Figure 4:</b> Distributed system architecture .....	9
<b>Figure 9:</b> Frontend template structure .....	12
<b>Figure 10:</b> Home view .....	15
<b>Figure 11:</b> Dashboard view .....	16
<b>Figure 12:</b> List of stocks available .....	16
<b>Figure 13:</b> Chart view .....	17
<b>Figure 14:</b> Frequently asked questions view .....	17
<b>Figure 15:</b> Model storage structure .....	21
<b>Figure 16:</b> Code MAPE function .....	20
<b>Figure 17:</b> Code RMSE fuction .....	20
<b>Figure 18:</b> Training loss LSTM .....	19
<b>Figure 19:</b> LSTM metrics .....	21
<b>Figure 20:</b> Pyplot chart with LSTM model .....	22
<b>Figure 21:</b> Training loss GRU .....	20
<b>Figure 22:</b> Split error .....	22

# List of tables

---

<b>Table 1:</b> Assets selected from the stock market .....	33
---	----

# List of codes

---

<b>Code 1:</b> <i>Alphavantage</i> API endpoint .....	10
<b>Code 2:</b> Data normalization .....	14
<b>Code 3:</b> Dataset split into train and test arrays .....	14
<b>Code 4:</b> JSON file model structure .....	21
<b>Code 5:</b> Method to calculate MAPE .....	18
<b>Code 6:</b> Method to calculate RMSE .....	18
<b>Code 7:</b> API JSON example response .....	10
<b>Code 8:</b> Imports from library Keras used for the neuronal networks .....	31
<b>Code 9:</b> Method to create LSTM neuronal network .....	31
<b>Code 10:</b> Method to create GRU neuronal network .....	31
<b>Code 11:</b> Method to store / load neuronal networks .....	32
<b>Code 12:</b> LSTM model layers .....	11
<b>Code 13:</b> GRU model layers .....	11



# Introduction

---

In recent years, machine learning has improved to the point that it is a very powerful tool to analyse and improve the performance and predictions about future events. It has been used by many institutions and individuals as it has plenty of use cases. For financial purposes it has heavily increased in the past years, therefore it will help investigate about the use cases of machine learning in the stock market.

The stock market is a huge financial market where publicly traded companies' stocks (shares) are bought and sold. It is one of the most important sources of capital for companies, and it allows many individual investors and institutions to buy and sell ownership stakes in publicly traded companies.

One of the main advantages of using machine learning for stock market predictions is its ability to process large amounts of data and identify patterns that may not be visible for the human eye.

Many people in the world invest in this type of asset. Most of them do it based on the financial company results or sometimes based on technical analysis.

## 1. Motivation

Stock market is laborious asset that requires time a deep understanding of the financial markets. More and more people are entering this sector and because of the vast amount of data available its learning curve is increasing constantly. A web-based interactive tool could help investors and non-experience individuals to learn about the effectiveness of prediction models and make more informed decisions. Additionally, the interactive web-based tool will provide an intuitive and user-friendly platform for users to interact with the predictions making it accessible for a large range of users.

## 2. Objectives

The goal of this work is to develop a web-based application tool to provide and interactive platform that allows users to learn about stock market predictions and make informed decisions using different machine learning prediction algorithms.

---

The main objectives will now be described:

- Obtain the minimum possible error compared to the actual closing price; it will be visualized through the web-based tool. Historical data will be used for these predictions.
- Provide different techniques to predict the stock market to improve the user decision-making with visual representations.
- Facilitate data analysis. The tool will allow users to filter, search and analyse different prediction models and compare their effectiveness based on various parameters. It will help users to understand the performance of different models and identify patterns and trends in the data that may not be immediately apparent to the human eye.
- The tool will be up to date as it uses data from financial APIs, as well as providing accurate up-to-date predictions.

# State of the art

---

Machine learning techniques, like Long Short-term Memory (LSTM) networks, have been widely used in recent years for stock market predictions. LSTM networks are a type of recurrent neural network that is used mainly for times series data, like stock predictions. It can learn and remember past patterns in the data, making them very useful for predicting future trends.

Apart from LSTM networks, other studies have also used other machine learning techniques such as Random Forest and Gated Recurrent Unit (GRU). In this project, we will focus on the use of the two gated and long short-term memory recurrent networks.

In the following section, we will see the explanation about each of the technologies involved in the development of the project.

## 1. Machine Learning

The technology used for predictions is called “Machine Learning” is a division of artificial intelligence and computer science that uses data and algorithms to generate an imitation to an analysis improving its accuracy.

When using machine learning for neuronal networks such as LSTM and GRU, two types of time-series forecasting models. They can anticipate future values by analysing previously observed values. This forecasting method is commonly applied to non-stationary data, a term used to describe data with statistical characteristics, like mean and standard deviation, that change over time instead of remaining constant. Apart from time-series data, these models facilitate the use for more complex sequence processing tasks like natural language processing.

### Recurrent neuronal networks

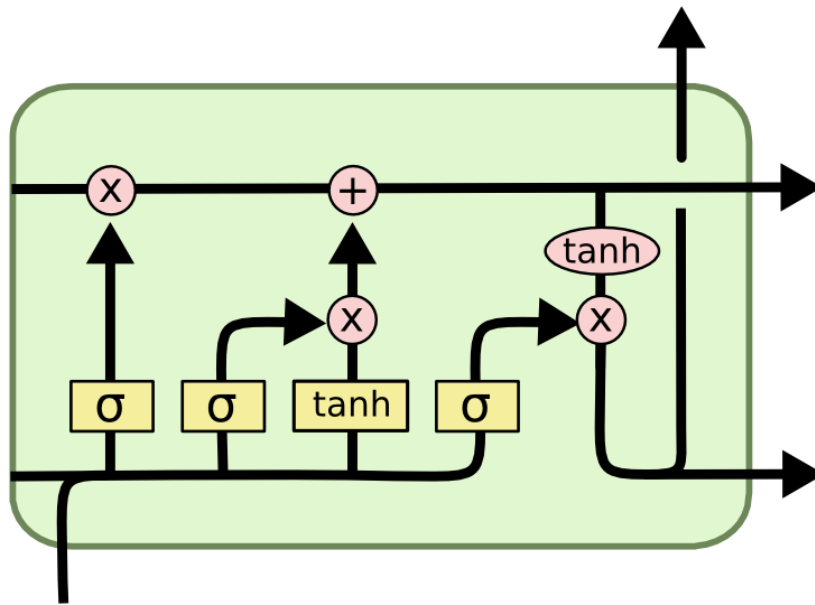
We will now go into detail with the recurrent neuronal networks. LSTM and GRU are two powerful neuronal network models capable of learning complex long-term dependencies between input and output sequences and make predictions based on past data.

### Long Short-Term Memory neuronal network

Long Short-term Memory (LSTM) networks is a type of neural network (RNN) used for a variety of tasks, including stock market predictions, machine translation, speech recognition, and

more. It is particularly effective for tasks that involve long-term dependencies, where the past inputs have a significant impact on future outputs,

The LSTM architecture consists of memory cells and gates that regulate the flow of the information through the network. The memory cells are used to store information over time, allowing the network to selectively remember and forget information as needed. The gates are made up of sigmoid and tanh activation functions, where they control the flow of the information into and out of the memory cells. A visual example of the LSTM architecture is shown in **Figure 1**.



**Figure 1:** LSTM architecture. Source [6]

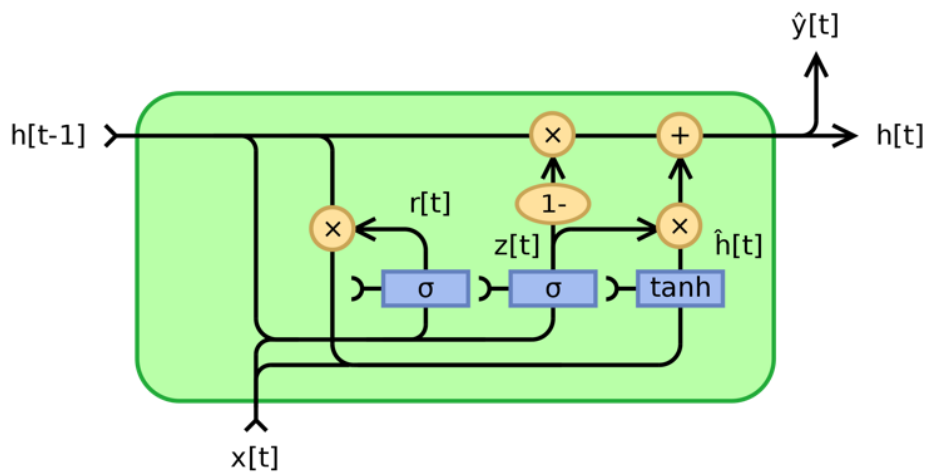
## Gated Recurrent Unit (GRU)

A Gated Recurrent Unit (GRU) is a type of recurrent neural network architecture for sequential data processing. They have been designed to handle sequential data, such as speech signals, time series data and text.

The architecture of the GRU neuronal network consists of an input layer, a hidden layer, and an output layer. The hidden layer contains a set of GRU units, each of which maintains a hidden state vector that is updated based on the input data and the previous hidden state.

The GRU network is computationally efficient and can be trained using standard backpropagation techniques. A visual example of the GRU architecture is shown in **Figure 2**.





**Figure 2:** GRU architecture. Source [7]

## Long Short-Term Memory (LSTM) vs Gated Recurrent Units (GRU)

The LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) networks are both types of recurrent neural networks that have been widely used for sequential data processing. While both networks are designed to handle long-term dependencies in sequential data, there are some key differences between the two.

One of the main differences between these two neuronal networks is the number of control gates to control the flow of the information. While GRU uses two gates (an update gate and a reset gate), the LSTM networks uses three gates (an input gate, an output gate and a forget gate). This last gate allows to selectively discard information that is no longer relevant, which is particularly useful for tasks such as language modelling and performance.

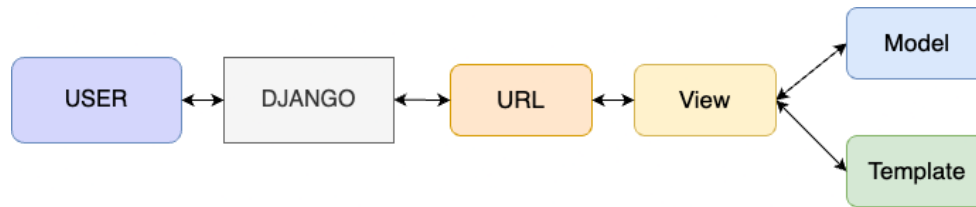
LSTM is more powerful and flexible than GRU and it is more accurate in larger datasets, but it is also more complicated and susceptible to over-fitting. In fact, GRU uses less training parameters and therefore uses less memory and executes faster than LSTM.

## 2. Python 3

Python is a popular, high-level programming language used in program modern web applications. It is the most recent version of the Python programming language and has the strongest emphasis on code readability. Python3 offers a comprehensive and extensive library of pre-built modules and packages and allows developers to create highly functional and dynamic applications. It is an easy to learn language and can be used for both small- and large-scale projects.

## Framework Django

The framework used to connect the back-end with the front-end is Django, it is one of the most used frameworks in python 3. This framework provides a platform for robust web applications with rapid development and scalability. The main components of the framework are its models, views, and templates.



**Figure 3:** Django framework architecture

- The model component: responsible for maintaining the data for an application. This includes defining the structure of the data, the relationships and performing operations on data.
- The view component: responsible for processing user requests and returning responses. Views can contain both the logic and templates needed to generate a response for a request.
- The template component: responsible for defining the visual structure and appearance of a web application. This includes the use of HTML, defining CSS styles and using JavaScript.

## Python libraries

As part of the development, different python libraries have been used to accomplish the scope of the project, some of the most important libraries are:

- **Tensorflow:** is an open-source library developed by Google for machine learning and artificial intelligence. It is used to train and deploy machine learning models.
- **Keras:** is an open-source software library that provides a Python interface of artificial neural networks.
- **Scikit-learn:** an open-source Python [1] library for machine learning. It provides implementations of many machine learning algorithms and convenient APIs for efficient model training and prediction.
- **Numpy:** is a Python library that supports a large collection of high-level mathematical functions to operate multi-dimensional arrays and matrices.
- **Matplot:** Matplot is a plotting library for Python. It provides many customizable plotting options, allowing to visualize data in 2D and 3D. It is well used with other Python libraries such as Numpy and Pandas.

- 
- **CSV File reading and writing:** Python library used to read and write the financial data used in the model prediction.

### 3. Javascript

The use of JavaScript in web development has grown exponentially in recent years due to its capability and relative ease of use. Together with HTML and CSS, JavaScript is now essential to create interactive, dynamic websites. More and more applications are being developed with it.

### 4. ChartJS

ChartJs is an open-source JavaScript library for creating charts on web pages. It allows to customize and modify various types of charts, including bar charts, pie charts and scatter plots. In this project it will be used to display the results of the stock prediction algorithms.

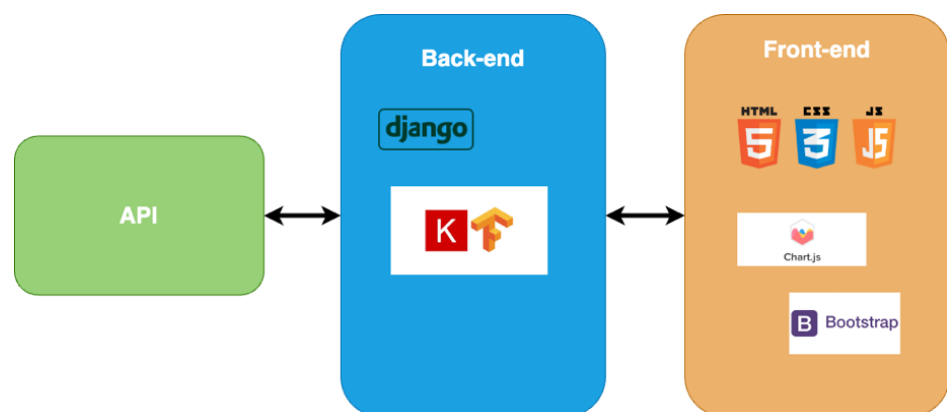
### 5. Bootstrap

Bootstrap is an open-source CSS framework for developing responsive and mobile-first websites. It is compatible with all major web browsers, containing HTML and CSS-based design templates. It also provides several helpful components and conventions that can help speed up the frontend development process.



# Analysis and design

This chapter will describe the design realised and the procedures for achieving the project goals. It will be divided in two main layers, the backend, responsible for processing and managing data, logic, and communication between the server and client-side of the web application, and the frontend which is responsible for displaying and interacting with the user interface of the web application, including layout, design, and user experience. In **Figure 4** it is described the distributed system architecture.



**Figure 4:** Distributed system architecture

Each part of the distributed system is essential for the proper functioning of the application. The idea to separate every part in different layers provides some benefits like:

- Modularity and scalability, it allows independent development and scalability of each component.
- Flexibility and reusability, by separating the backend from the frontend you are allowed to use different technologies and even programming languages for each layer. Additionally, the backend services can be reused across multiple frontend applications.

## 1. Back-end development

The backend or “server-side” is the portion of the website that you don’t see. In a distributed system it refers to the components that make up the system’s infrastructure. It also communicates with the frontend, sending and receiving information.

We will break down comprehensively the design of the back end of our web application.

---

## Datasets

The data supplied for the evolution of the project as well as the data used to train the models to predict the future values of the company stock comes from an API. In this case we will use the free tier version of the Alphavantage API.

As we have just mentioned, the information will be provided by Alphavantage API. It provides financial market data from traditional asset classes such as stocks, ETFs, or mutual funds. It further offers real-time stock data, charts, and analytics tools. It is designed as a RESTful interface allowing users to easily access data from API endpoints. From its many features, we will use the free version tier for the evolution of the project.

The API endpoint used to get historic closing prices of each stock is called *TIME\_SERIES\_DAILY\_ADJUSTED* and provides daily open/high/low/close/volume rate values, end-of-day adjusted closing prices, and records of any past stock splits or dividend payments for a chosen global equity, with data spanning over 20 years. The API endpoint call is described in 1.

```
1.         url = 'https://www.alphavantage.co/query'
2.         params = {"function": "TIME_SERIES_DAILY_ADJUSTED", "symbol": symbol, "apikey": API_KEY,
3.         "outputsize": "full"}
4.         try:
5.             r = requests.get(url, params=params)
6.         except requests.exceptions.HTTPError as err:
7.             raise SystemExit(err)
```

**Code 1.** It has few parameters such as symbol, API key and the output size.

```
1.         url = 'https://www.alphavantage.co/query'
2.         params = {"function": "TIME_SERIES_DAILY_ADJUSTED", "symbol": symbol, "apikey": API_KEY,
3.         "outputsize": "full"}
4.         try:
5.             r = requests.get(url, params=params)
6.         except requests.exceptions.HTTPError as err:
7.             raise SystemExit(err)
```

**Code 1:** *Alphavantage API endpoint*

The response is in JSON format, and contains different metadata including the symbol, the date and the close price used to train the neuronal networks.

```
1. {
2.     "Meta Data": {
3.         "1. Information": "Daily Time Series with Splits and Dividend Events",
4.         "2. Symbol": "AAPL",
5.         "3. Last Refreshed": "2023-05-02",
6.         "4. Output Size": "Full size",
7.         "5. Time Zone": "US/Eastern",
8.     },
9.     "Time Series (Daily)": {
10.         "2023-05-02": {
11.             "1. open": "181.03",
12.             "2. high": "181.78",
13.             "3. low": "179.26",
14.             "4. close": "180.95",
15.             "5. adjusted close": "180.95",
```

```

16.         "6. volume": "61996913",
17.         "7. dividend amount": "0.0000",
18.         "8. split coefficient": "1.0",
19.     },
20. },
21. }
22.

```

**Code 2:** API JSON example response

## Neuronal networks

As previously highlighted, the implementation of neuronal networks represents a fundamental aspect of backend development. There are numerous approaches to predict the value of an asset. In this case, deep algorithms are used for the estimation. Inside of deep learning we find a variety of recurrent neural networks that are algorithms used for learning long term dependencies. Two of the most used recurrent neural networks are the Long Short-Term Memory Network (LSTM) and the Gated Recurrent Unit (GRU). The design of layers in these two architectures may have minor differences, but they play a critical role in the overall structure.

The Long Short-Term Memory Network (LSTM) architecture consists of multiple layers, including two LSTM layers with 50 units each, followed by a dense layer containing 1 unit.

```

1. Model: "sequential"
2.
3. Layer (type)           Output Shape           Param #
4. =====
5. lstm (LSTM)             (None, 1, 50)          10400
6.
7. lstm_1 (LSTM)           (None, 50)             20200
8.
9. dense (Dense)           (None, 1)              51
10.
11. =====
12. Total params: 30,651
13. Trainable params: 30,651
14. Non-trainable params: 0
15.

```

**Code 3:** LSTM model layers

The next neural network utilized is the Gated Recurrent Unit (GRU), which includes two GRU layers. In these layers, the first two layers contains 50 units, followed by the last layer of type dense and with single value.

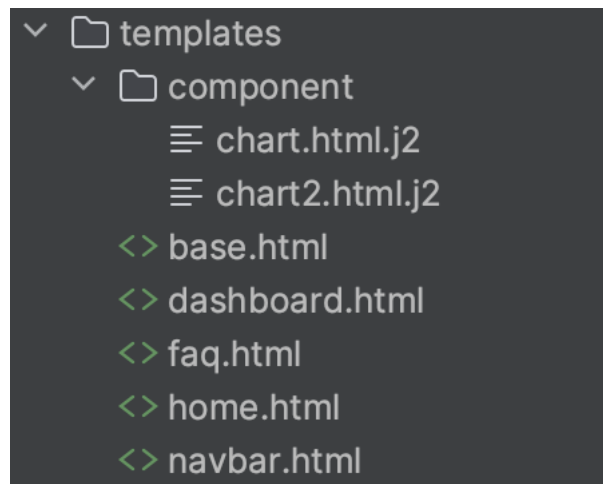
```

1. Model: "sequential_1"
2.
3. Layer (type)           Output Shape           Param #
4. =====
5. gru (GRU)               (None, 1, 50)          7950
6.
7. gru_1 (GRU)             (None, 50)             15300
8.
9. dense_1 (Dense)         (None, 1)              51
10.
11. =====
12. Total params: 23,301
13. Trainable params: 23,301
14. Non-trainable params: 0
15.

```

## 2. Front-end development

Another main part and the most visual is the front-end. All the views, components and functions used will be described. In **Figure 5** you will see the front-end structure divided in different templates.



**Figure 5:** Frontend template structure

## 3. Methodology

The project has been developed following a lean process. A lean process refers to adopting an approach that focuses on efficiency and flexibility by minimizing the time used for each task. Each part of the project has been first created with a simple functional version. Starting from a first web version that worked with all the components of the project, connected with the financial API, and then focusing on improving each part of the system. Following the lean process also helps to finish the scope of the project with a very simple version of it.



# Implementation

---

Through this section, the development of the project will be described. The scope of the project consists of two parts, backend development and frontend development.

## 1. Back-end development

The backend of the project is divided into two main parts. First one is the neuronal network which is the most important part since it is used to make the stock predictions. The second main part is the development of the views in the Django framework.

### Developing advanced neuronal networks

One of the easiest methods to develop neuronal networks is with the use of two famous libraries. Tensorflow and Keras, both open-sourced and designed for fast experimentation with deep neuronal networks.

To begin with the implementation, we will use the use methods from Keras (**Code 11**). The process of creating the neuronal network is very systematic for most neural networks and the structure is used in many other projects. We will go into detail for each step in the following paragraphs.

Begin the process **importing the essential data**, in our case from the API earlier described and the use of reading and math libraries, such as pandas and numpy. The use the of the csv library is to read in the data from a CSV file. If the data is not available locally, make use of a financial API to acquire the entire database of stock information. The only metadata necessary for time series prediction is the date and the closing price and date of the corresponding asset.

Once the relevant data has been collected, the next step is **data normalization**. This step is crucial in ensuring that all input features have similar scales, which is instrumental in finding the optimal performance and successful convergence. There are multiple methods to normalize the data, we will use Min-Max scaling technique, is a type of normalization that will replace each value with an adapter smaller number between zero and one.

```

1. # Normalize data
2. scaler = MinMaxScaler(feature_range=(0, 1))
3. scaled_data = scaler.fit_transform(price_list.reshape(-1, 1))

```

**Code 5:** Data normalization

The next step involves **dividing the dataset into training and testing arrays**. This process of splitting the dataset into training and testing arrays is a crucial step in the machine learning and predictive modelling process. The split is necessary to properly measure how well the model works on new information, and to get a better idea of how well it can be applied to different situations. The typically used ratio is 80:20, with 80% of the dataset being allocated to the training array and the remaining 20% reserved for the testing array.

```

1. # Train and test split
2. test_ratio = 0.2
3. training_ratio = 1 - test_ratio
4. train_size = int(training_ratio * n_days)
5. test_size = int(test_ratio * n_days)
6.
7. # Create training and test data
8. train = scaled_data[0:train_size]
9. test = scaled_data[train_size - test_size:train_size]
10.

```

**Code 6:** Dataset split into train and test arrays

The following step consist of **building and training the models**. It involves utilizing deep learning libraries described such as TensorFlow and Keras. The process for training the LSTM network and GRU network is similar and will now be discussed.

First, define the number of LSTM layers, hidden units, and activation functions by the specified requirements. Afterwards, the model can be trained using the normalized data that has been divided into training and validation sets. To improve the model's performance, parameters like learning rate, batch size, and epochs need to be further optimized. The same steps are repeated for the GRU (Gated Recurrent Unit) model, which is a distinct form of a recurrent neural network. The code to generate both models is described in **Code 12** and 1. `def`

```
rmse(y_true, y_pred):
```

```

2.     return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))
3.

```

**Code 9.** ds

Once the training phase is completed, the next stage involves **testing and making predictions**. The model generates predictions using the test dataset, which are then evaluated by comparing them with the actual values. To measure the accuracy and precision of the models, several metrics are used, such as Mean Squared Error (MSE), Root Mean Square Error (RMSE) and training loss. We will go more into detail of the results in section 5.

```

1. #####
2. # Make predictions and test #
3. #####
4. train_predict_lstm = lstm_model.predict(train)
5. test_predict_lstm = lstm_model.predict(test)

```

```
6.  
7. train_predict_gru = gru_model.predict(train)  
8. test_predict_gru = gru_model.predict(test)
```

**Code 7:** Testing and making predictions

Before accurately evaluate the performance of the models, it is important to **calculate the metrics**: additional evaluation metrics should be calculated such as root mean absolute error (RMSE), mean absolute percentage error (MAPE). These metrics will provide an exhaustive insight into the model's predictive power and accuracy.

Before accurately assessing the performance of the models, it is crucial to determine and **calculate the metrics**, such as root mean square error (RMSE) and mean absolute percentage error (MAPE), should be computed. These metrics will offer a comprehensive understanding of the model's predictive strength and precision.

Finally, **visualizing the results** by graphing the predicted values against the actual values, as it is essential to understand the performance of the model. Line plots or candlestick charts are used to visualize the results. Through these visualizations, patterns and trends in the stock market data can be identified.

This entire process has been repeated multiple times until we found the optimal model performance. There has been many areas and distinct parameters of improvement which include different layer architectures, activation functions, regularization techniques, or hyperparameters. In section 5, we will explore in-depth the adjustments made to the parameters for determining the optimal model.

## 2. Front-end development

As we have mention in the analysis and design section. In this section, we will talk about all the Front-end development of our web application. As we mention with the lean methodology used. Initially, the templates were coded in plain HTML, and as the project progressed, CSS and JavaScript styles were incorporated. The primary design comes from the CSS framework Bootstrap, while the charts are created using the JavaScript library ChartJs. The details of the main templates will now be described.

### Home view

The first template is the home view, the first view the user will access on the website. it contains a short description about the project and some contact information.

### App to predict stock performance using machine learning algorithms

The goal of this work is to develop a web application tool to visualize different stock market prediction algorithms using machine learning.

The idea is to obtain the minimum possible error compared to the actual closing price of the stock. This will be achieved by means of users interacting through the web, as well as allowing them to filter, search and analyze different prediction algorithm models and compare them according to its effectiveness based on different parameters.

The data used will be extracted from different financial APIs. The project will be developed in a Python environment using Django as the main framework.

Author: Juan Llamazares Ruiz

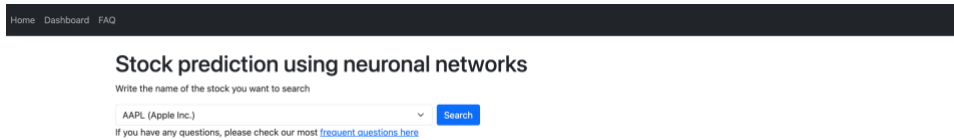
Contact: [juan.llaamazares@estudiante.uam.es](mailto:juan.llaamazares@estudiante.uam.es)

[Access now](#)

**Figure 6:** Home view

## Dashboard view

The dashboard serves as the primary interface, it will include all the information the user needs to study and analyse the predictions. The view will receive two types of requests, plain get request will only show the search input box and some extra information.



Home Dashboard FAQ

### Stock prediction using neuronal networks

Write the name of the stock you want to search

AAPL (Apple Inc.)

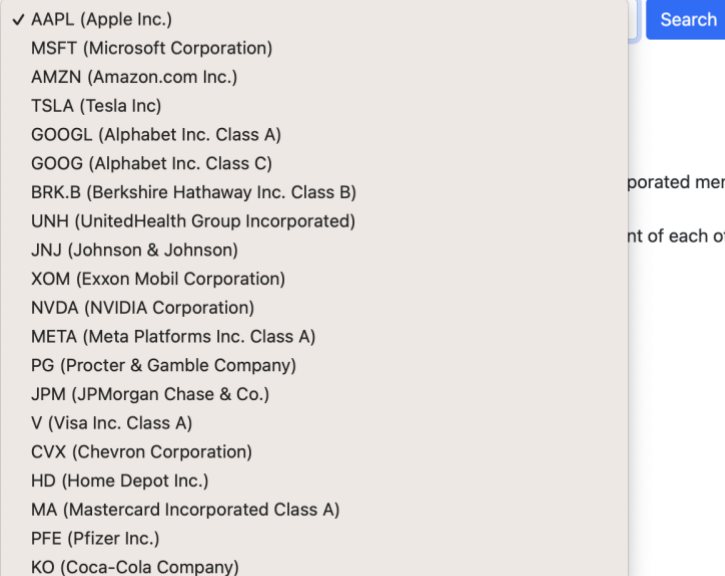
If you have any questions, please check our most [frequent questions here](#)

**Figure 7:** Dashboard view

The search input box will have a list of all the possible assets, in section 5 we will go into detail of the decision to list the assets instead of allowing it all. Once the user decides what is the asset it is going to predict, it will select the search button to submit the request.

## Stock prediction using neuronal networks

Write the name of the stock you want to search



Search

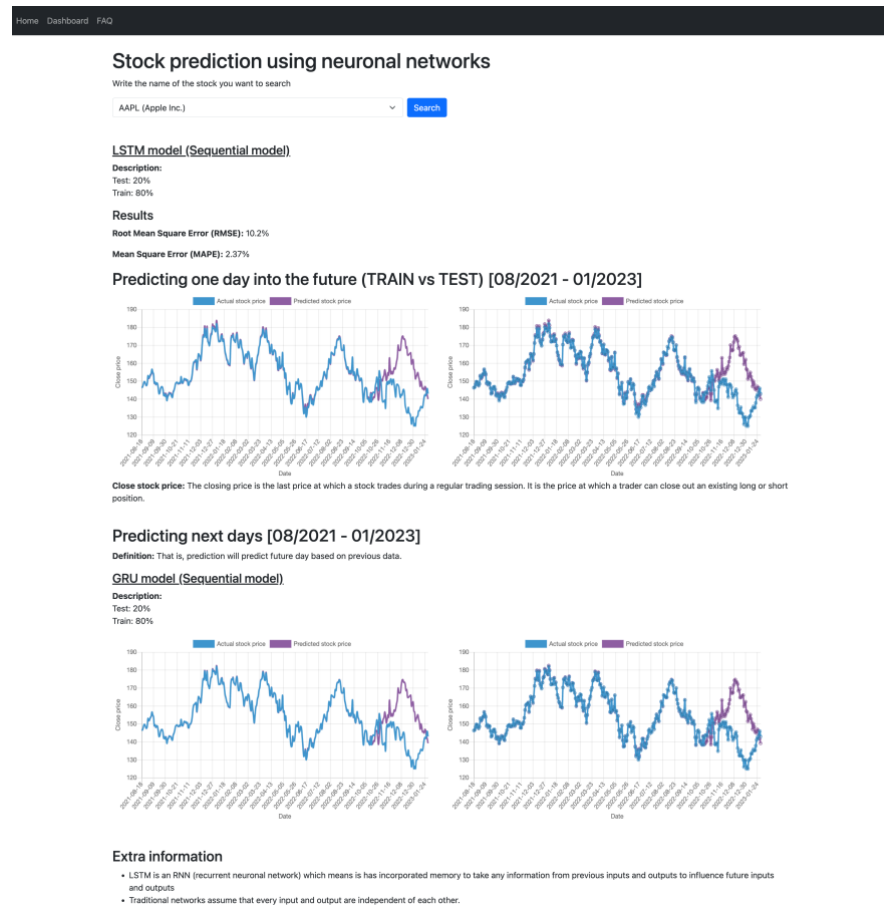
- ✓ AAPL (Apple Inc.)
- MSFT (Microsoft Corporation)
- AMZN (Amazon.com Inc.)
- TSLA (Tesla Inc)
- GOOGL (Alphabet Inc. Class A)
- GOOG (Alphabet Inc. Class C)
- BRK.B (Berkshire Hathaway Inc. Class B)
- UNH (UnitedHealth Group Incorporated)
- JNJ (Johnson & Johnson)
- XOM (Exxon Mobil Corporation)
- NVDA (NVIDIA Corporation)
- META (Meta Platforms Inc. Class A)
- PG (Procter & Gamble Company)
- JPM (JPMorgan Chase & Co.)
- V (Visa Inc. Class A)
- CVX (Chevron Corporation)
- HD (Home Depot Inc.)
- MA (Mastercard Incorporated Class A)
- PFE (Pfizer Inc.)
- KO (Coca-Cola Company)

**Figure 8:** List of stocks available

Once the POST request calls the back end, and if the stock model is generated, it will return an array of data. Including the symbol, the number of days, and the stock price data. The code

then forms a list of the keys in the data variable (date list) and the values (close price list). The date list and close price list are then both reversed, and the close price list is limited to the number of days specified. The stock prediction LSTM and GRU results are then stored in variables with the metrics obtained.

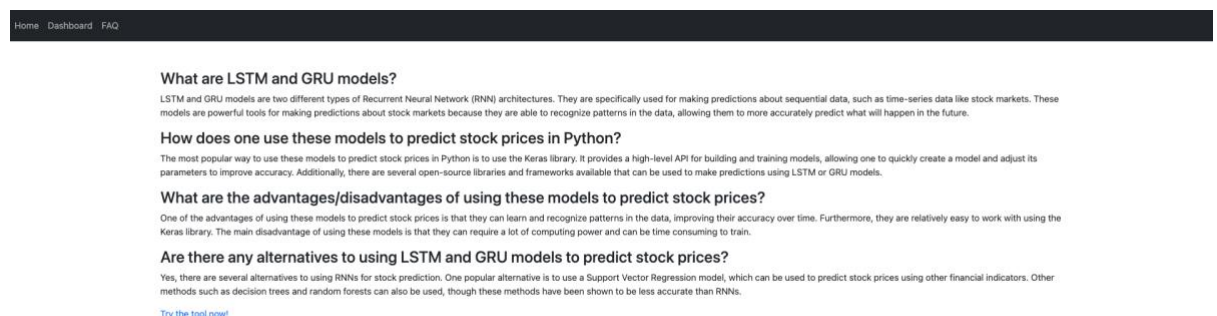
The view then changes with the data obtained as shown in **Figure 9**.



**Figure 9:** Chart view

## Frequently asked question's view

Developing a user experience FAQ (Frequently Asked Questions) as part of the project is essential for understanding technical and financial terms, as it makes it easier for users to understand the use case of the tool. The reason for it, is that it will decrease the customer service requests and support needs. The view is shown in **Figure 10**.





# Testing and results

---

## 1. Introduction

Since one of the objectives of this work is to evaluate the prediction of the models developed, we will go into the detail on all the tests carried out to evaluate the efficacy of our web application tool. The approach to find the model has been adjusted to the time and effort of the project. Therefore, the results will probably not be as accurate as if the project was fully dedicated into finding the best model. Later, some other possible improvements and alternatives will be explained.

As we previously stated, the significance of testing and its outcomes has been crucial in the project's development. Given that the project involves designing a web application, the testing will involve in the improvement of the technology used in the backend. Without testing different parameters and machine learning techniques, the results itself would not be as accurate as the obtained.

## 2. Metrics

The evaluation of the metrics is divided into two parts. First, we evaluate the quality provided by the two models LSTM and GRU. The most popular metrics used for it are Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). MAPE metric measures the size of the error in percentage terms. A MAPE value between 0 and 10 indicates accurate predictions, anything above 20 indicates a problem with the model.

**Error! Reference source not found.**

```
1. def mape(y_test, y_pred):  
2.     y_test, y_pred = np.array(y_test), np.array(y_pred)  
3.     return np.mean(np.abs((y_test - y_pred) / y_test)) * 100  
4.
```

**Code 8:** Method to calculate MAPE

RMSE (Root Mean Squared Error) is typically used if the errors are normally distributed and have a constant variance. A good value for RMSE depends on the specific application and the model being used. Generally, a value under 0,5 is a good result.

**Error! Reference source not found.**

```
1. def rmse(y_true, y_pred):
```

```
2. return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))
3.
```

**Code 9:** Method to calculate RMSE

### 3. Recurrent neuronal network model tests

The aim is to determine the optimal architecture for a Recurrent Neuronal Network (RNN) model to predict future stock prices.

To evaluate the testing of the neuronal network, a visual library for testing, specifically the Python library Matplotlib, has been implemented. It illustrates the performance of the different models evaluated and represents the effectiveness of the stock prediction process through graphical plots.

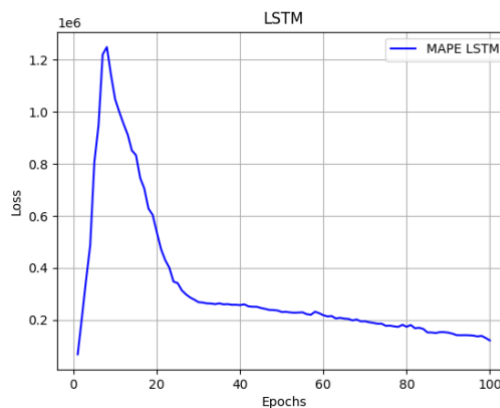
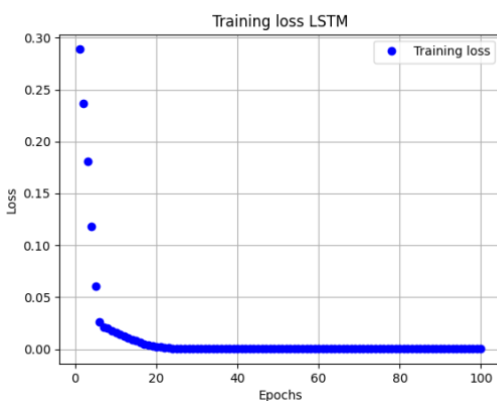
#### Long Short-Term Memory

For training and testing of the LSTM model, an Adam optimizer was used. Additionally, 100 epochs and a batch size of 32. The details of the layer structure are described in 1.

```
Model: "sequential"
2.
3. Layer (type)                Output Shape                Param #
4. =====
5. lstm (LSTM)                 (None, 1, 50)              10400
6.
7. lstm_1 (LSTM)               (None, 50)                  20200
8.
9. dense (Dense)               (None, 1)                   51
10.
11. =====
12. Total params: 30,651
13. Trainable params: 30,651
14. Non-trainable params: 0
15.
```

After different parameter and function changes, we found the best model to predict future stock prices values. This layer architecture and parameters will be used with all the assets used for the project. It is not the best approach, but it will work to the intention of the web tool.

For most of the models created the training loss and MAPE metric has decreased drastically from the first 0-30 epochs, after that number of epochs, it has slowly decreased.



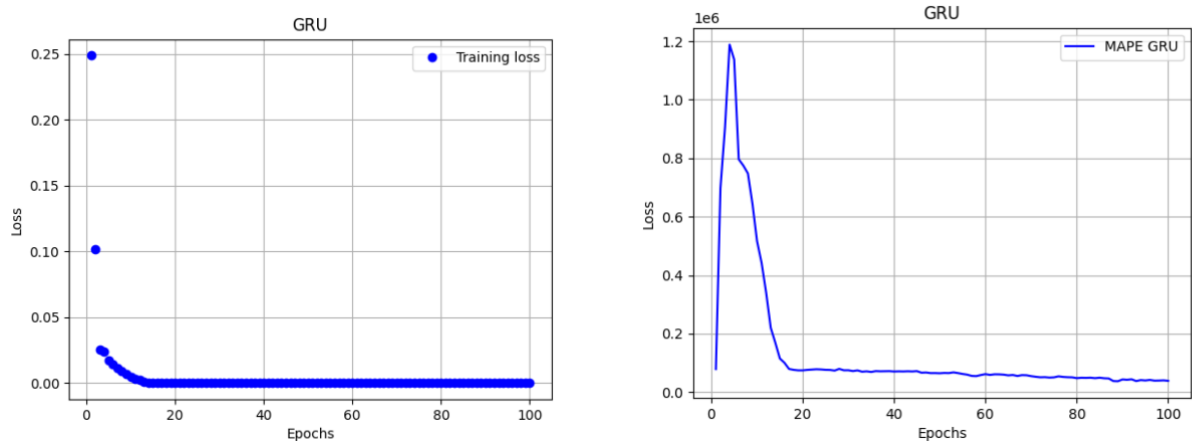


**Figure 11:** Training loss and MAPE from Long Short-Term Memory

## Gated Recurrent Unit

The GRU model has been compiled using the Adam optimizer with 100 epochs and a batch size of 32.

In other hand, the results from training the Gated Recurrent Unit model. From initial observations, the GRU as expected, it appears to be trained more rapidly in comparison to the Long Short-term Memory. In under 10-15 epochs, there is a significant decrease in training loss, and the MAPE falls below 20 within 20 epochs. Following this, both metrics follow a linear decreasing trend.



**Figure 12:** Training loss and MAPE from Gated Recurrent Unit

## Results LSTM vs GRU

Comparing the outcomes of both models can be a difficult task, as several aspects can impact their performance. The financial market has experienced numerous fluctuations in recent years, primary caused by market trends and conditions. These changes can potentially affect the outcomes of time series networks that rely on historical closing prices. When it comes to technical factors, both networks have similar outcomes in terms of the time taken for training and the accuracy of the predictions made.

## 4. Challenges encountered

During the development of the project, some challenged have been found while trying to unify the financial part with the technical part.

The first and biggest found difficulty is the creation of the neuronal network models, as it takes an enormous consideration of time, and letting the user wait that long to generate a model it not a proper solution since we want to keep their attention. Therefore, we found a solution, the models are pre-generated. The models will be saved as “.h5” files, which is the typical file format to store and distribute trained models in the HDF5. The “.h5” file allows efficient storage and retrieve of large models, making it very usable for sharing and deploying trained neural networks. A “.json” file will also be used to store and load the metrics used for testing.

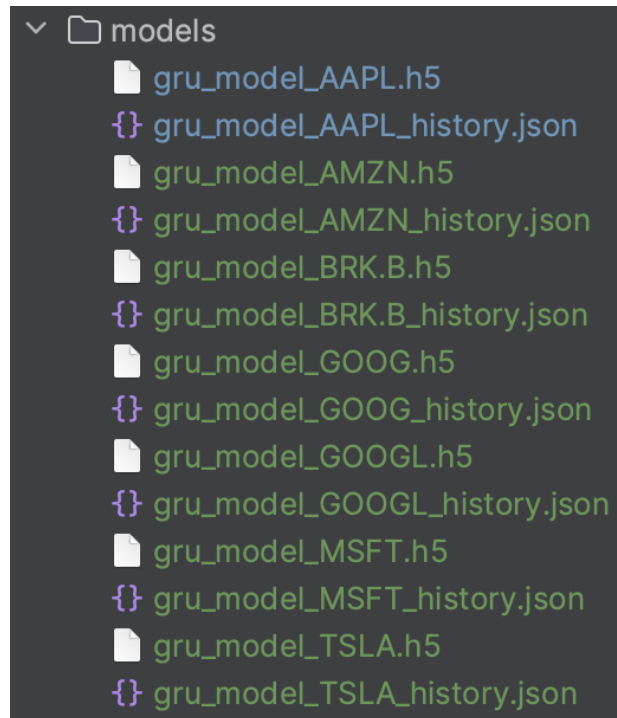


Figure 13: Model storage structure

```
1. {  
2.   "loss": [  
3.     7.640528565389104e-06,  
4.     7.6318056017044e-06,  
5.     ...  
6.   ],  
7.   "root_mean_squared_error": [  
8.     0.0027625723741948605,  
9.     ...  
10.  ],  
11.   "mean_absolute_percentage_error": [  
12.     35901.578125,  
13.     ...  
14.  ]  
15. }  
16.
```

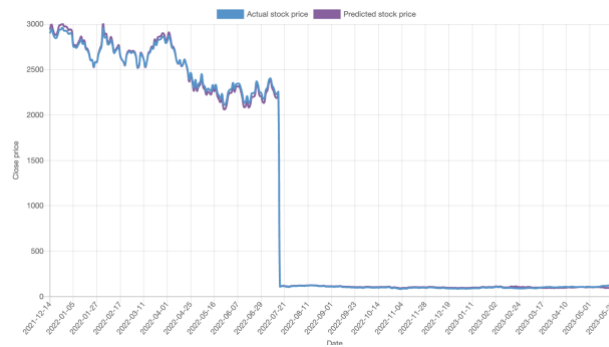
Code 10: JSON file model structure

Another problem found is the stock market has thousands of different assets, as we have a limit storage space for the models, only the biggest 20 stocks in the US market will be added listed in **Table 1**.

## 5. Alternatives and improvements

We will now comment very few errors found on the project and how can they be solved.

To begin with, the stock split error occurs due to the limitations of the free tier API being used. It could be solved by using a more robust API, or by switching to a paid version of the same API.



**Figure 14:** Split error

Some potential solutions for boosting the performance of a neural network include:

- Improve the financial API, the free tier API used for the project did not consider splits and other data, which lead to have limited results. By incorporating a better source of data, it will improve the accuracy of the prediction
- Improve neuronal network using hyperparameter tuning techniques to optimize the overall architecture
- Using an alternative deep learning framework like Pytorch, in contrast to Keras, is ideal for developers looking to rapidly build, train, and assess their models. Pytorch offers a faster option and some debugging capabilities.

---

# Conclusions and future work

---

## 1. Conclusions

To conclude with the project, we have successfully created a web application to allow users and other individuals interact and experience with a tool to predict some financial assets, in this case stocks with recurrent neuronal networks. It will also teach investors another approach of understanding the future market trends using machine learning technology.

Talking about the neuronal networks used on the project, we can conclude that stock prediction based only on the historical price's movement will not be the only factor that affects the future price movement. However, using this recurrent neuronal network like LSTM or GRU will provide in a way the stock trend for the future days.

The conclusion of this technical bachelor thesis is that stock prediction using neural networks should not solely be relied upon. While neural networks can be used to analyse the past data and trends to generate predictions, these predictions should be considered in combination with an analysis of other market trends and financial data. Using a combination of both methodologies will increase the accuracy of the predictions being made. Instead of relying solely on the output generated from a web application tool, a comprehensive analysis of the markets and stocks should be conducted. This will help to provide a deeper insight into the stocks and the trends of the market which should then be used to generate the most accurate predictions.

## 2. Future work

There are different alternatives to improve this project, we will go into the details of the most effective ways to keep the development of the web application.



# Bibliography

---

- [1] "Alphavantage API," [Online]. Available: <https://www.alphavantage.co/>
- [2] "Python," [Online]. Available: <https://www.python.org/> . [Accessed 2023].
- [3] "Techtarget" [Online]. Available <https://www.techtarget.com/searchapparchitecture/definition/RESTful-API> [Accessed 2023]
- [4] "Geeks for geeks", "stock price prediction" [Online]. Available: <https://www.geeksforgeeks.org/stock-price-prediction-using-machine-learning-in-python/>
- [5] "LSTM and GRU neuronal networks" [Online]. Available: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00512-z> . [Accessed 2023]
- [6] "Long short-term memory (LSTM) architecture" [Online]. Available: [https://en.wikipedia.org/wiki/Long\\_short-term\\_memory](https://en.wikipedia.org/wiki/Long_short-term_memory). [Accessed 2023]
- [7] "Gated recurrent unit (GRU) architecture" [Online]. Available: [https://en.wikipedia.org/wiki/Gated\\_recurrent\\_unit](https://en.wikipedia.org/wiki/Gated_recurrent_unit) . [Accessed 2023]
- [8] "Tensorflow LSTM" [Online]. Available: [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/LSTM](https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM). [Accessed 2023]
- [9] "Keras vs Tensorflow vs Pytorch" [Online]. Available: [https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article#:~:text=TensorFlow%20is%20an%20open%2Dsourced,because%20it's%20built%2Din%20Python](https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article#:~:text=TensorFlow%20is%20an%20open%2Dsourced,because%20it's%20built%2Din%20Python.). [Accessed 2023]
- [10] "Stock price forecasting" [Online]. Available: <https://www.mdpi.com/2227-7390/8/9/1441>. [Accessed 2023]
- [11] "Towardsdatascience lstm time series forecasting" [Online]. Available: <https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f>. [Accessed 2023]
- [12] "Medium, stock price prediction" [Online]. Available: <https://medium.com/the-handbook-of-coding-in-finance/stock-prices-prediction-using-long-short-term-memory-lstm-model-in-python-734dd1ed6827>. [Accessed 2023]



# Terminology

---

## 1. Technical terms

- **API:** Application Programming Interface is a set of functionalities and protocols for facilitating communication between two or more computer systems.
- **RESTful interface:** is an API designed with the principles of Representational State Transfer, utilizing standard HTTP methods for communication between clients and servers in a stateless, scalable, and simple structure.
- **POST request:** is an HTTP request method used to send data to a server for processing. It is typically used when creating or updating a resource, such as submitting a form on a website.
- **GET request:** is an HTTP request method used to send data to a server for processing. It is typically used when creating or updating a resource, such as submitting a form on a website.
- **Machine Learning (ML):** is a subfield of artificial intelligence. It examines the design and construction of algorithms capable of learning from data, even without specific programming. These ML algorithms can autonomously recognize hidden patterns and tendencies, generate predictions, and form conclusions, all without outside input.
- **Hierarchical Data Format version 5 (HDF5):** is a file format designed to store and organize large amounts of numerical data.
- **JSON (JavaScript Object Notation):** is a user-friendly, light data exchange format commonly used for data transmission between servers and web applications, as well as data storage and organization.

## 2. Financial terms

- **Stock:** also known as an equity, is a type of security that signifies ownership in a company and represents a claim on the company's assets and earnings.
- **Stock market:** is a form of exchange where buyers and sellers trade ownership stakes in publicly traded companies, as well as derivatives and other financial instruments, such as options and futures.
- **Stock close price:** is the final price at which a stock trades on any given day of trading.
- **Stock split:** action that occurs when a stock's existing shares are split into multiple smaller shares to reduce their market price. It usually happens when a company's stock has become too expensive for potential investors to buy.





# **Appendices**



# Appendice

---

This appendix contains blocks of code in the back-end development:

```
1. from keras.models import Sequential
2. from keras.layers import Dense, GRU
3. from keras.layers import LSTM
```

**Code 11:** Imports from library Keras used for the neuronal networks

```
1. def create_lstm_model(train, metrics):
2.     lstm_model = Sequential()
3.     # First layer
4.     lstm_model.add(LSTM(units=50, return_sequences=True, input_shape=(train.shape[1], 1)))
5.     # Second layer
6.     lstm_model.add(LSTM(units=50))
7.     lstm_model.add(Dense(1))
8.     # Compiling the RNN (Recurrent Neuronal Network)
9.     lstm_model.compile(optimizer="adam", loss='mse', metrics=metrics)
10.    # Fitting the RNN to the Training set
11.    history = lstm_model.fit(train, train, epochs=100, batch_size=32, verbose=2)
12.    return lstm_model, history
13.
```

**Code 12:** Method to create LSTM neuronal network

```
1. def create_gru_model(train, metrics):
2.     gru_model = Sequential()
3.     # First layer
4.     gru_model.add(GRU(units=50, return_sequences=True, input_shape=(train.shape[1], 1)))
5.     # Second layer
6.     gru_model.add(GRU(units=50))
7.     gru_model.add(Dense(1))
8.     # Compiling the RNN (Recurrent Neuronal Network)
9.     gru_model.compile(optimizer="adam", loss='mse', metrics=metrics)
10.    # Fitting the RNN to the Training set
11.    history = gru_model.fit(train, train, epochs=100, batch_size=32, verbose=2)
12.    return gru_model, history
13.
```

**Code 13:** Method to create GRU neuronal network

```

1.  try:
2.      if new_model:
3.          lstm_model, history_lstm = create_lstm_model(train, metrics)
4.          lstm_model.save("models/lstm_model_" + symbol + ".h5")
5.
6.          json.dump(history_dict_lstm, open("models/lstm_model_" + symbol + "_history.json",
"w"))
7.
8.          gru_model, history_gru = create_gru_model(train, metrics)
9.          gru_model.save("models/gru_model_" + symbol + ".h5")
10.         history_dict_gru = history_gru.history
11.         json.dump(history_dict_gru, open("models/gru_model_" + symbol + "_history.json",
"w"))
12.     else:
13.         try:
14.             lstm_model = keras.models.load_model("models/lstm_model_" + symbol + ".h5")
15.             history_dict_lstm = json.load(open("models/lstm_model_" + symbol +
"_history.json", "r"))
16.         except OSError:
17.             lstm_model, history = create_lstm_model(train, metrics)
18.             lstm_model.save("models/lstm_model_" + symbol + ".h5")
19.             history_dict_lstm = history.history
20.             json.dump(history_dict_lstm, open("models/lstm_model_" + symbol +
"_history.json", "w"))
21.
22.     except:
23.         print("[ERROR] Creating/Reading model")
24.

```

**Code 14:** Method to store / load neuronal networks

List of chosen assets from the stock market

Company name	Symbol
Apple Inc	AAPL
Microsoft Corporation	MSFT
Amazon.com Inc	AMZN
Tesla Inc	TSLA
Alphabet Inc. Class A	GOOGL
Alphabet Inc. Class C	GOOG
Berkshire Hathaway Inc. Class B	BRK.B
UnitedHealth Group Incorporated	UNH
Johnson & Johnson	JNJ
Exxon Mobil Corporation	XOM
NVIDIA Corporation	NVDA
Meta Platforms Inc. Class A	META
Procter & Gamble Company	PG
JPMorgan Chase & Co	JPM
Visa Inc. Class A	V
Chevron Corporation	CVX
Home Depot Inc	HD
Mastercard Incorporated Class A	MA
Pfizer Inc	PFE
Coca-Cola Company	KO

**Table 1:** Assets selected from the stock market

## Dependencies and libraries