BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

PRICE - An approach to predict currency exchange rates

Scientific supervisor
Prof. dr. Czibula Gabriela

Student
Bodea Adonis Sebastian

2020

UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

PRICE - O abordare pentru predicţia ratelor de schimb valutar

Coordonator ştiinţific
Prof. dr. Czibula Gabriela

Absolvent
Bodea Adonis Sebastian

2020

Abstract

The present thesis tackles the problem of forecasting currency exchange rates and tries to make use of the artificial neural network method in order to create reliable prediction models for forecasting one day into the future the following exchange rates: the European Union currency against the British pound (EUR\GBP), the European Union currency against the US dollar (EUR\USD) and the British pound against the US dollar (GBP\USD). Such an endeavour would be considered useful by potential investors or even everyday people who are interested in the currency exchange market. In the interest of delivering reliable predictions models, this thesis investigates multiple architectures (for example Long Short Term Memory Neural Networks or Convolutional Neural Networks) which are combined with external indicators to the exchange rates (gold and oil prices) and varying lengths of past observations as input, and compares the results obtained with existing ones from literature. The models constructed using the aforementioned combinations which are assessed to be the most performant are then further incorporated in a mobile application whose main aim is to deliver relevant information about currency exchange rates. To better capture this idea of meaningful information, the mobile application will offer also the functionality of converting sums of money between different currencies at the current day's rates, the possibility to see past values of certain exchange rates and the option to analyze past predictions made by the exposed models, besides the already mentioned feature of forecasting the currency exchanges introduced earlier one day into the future. The originality of this thesis therefore consists in a performance assessment of multiple neural networks architectures for tackling the problem of currency exchange forecasting, the comparison of the most performant models found with existing results in literature and the integration of the best prototypes in a mobile application. This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

***Keywords*—** Currency exchange rates prediction/forecasting, Artificial Neural Networks, Feed Forward Neural Networks, Long Short Term Memory, Convolutional Neural Networks, ConvLSTM, Bidirectional Long Short Term Memory, Keras, Tensorflow, Flask, Flutter, DigitalOcean

Contents

List of Figures

List of Tables

Introduction

Most people are afraid of investing in the currency exchange rates market. It is volatile, it calls for a lot of patience and, probably most important, it requires a lot of accurate and relevant information to actually make a good decision. Given these drawbacks, it is no wonder not many people are thrilled to take the risk. But what if an application existed that could remove some of these disadvantages, let you visualize existing currency rates and make an accurate prediction for you of how EUR will change tomorrow for example?

Predictions are something each and every one of us do on a daily basis. Whether we think about changing our job, choosing a life partner or making a new investment, we are all making a forecast about what the future holds [TG15]. Some of the predictions are easy enough for us to make using our reasoning and available information but, unfortunately, there are also some events which are extremely difficult to predict, so we turn to people or, more recently, to machines which have a better expertise than us.

Financial predictions, and namely exchange rates predictions, fall into the latter category. Their unpredictable nature rises mostly from the socioeconomic and political factors which influence them at every step, especially after the breakdown of the Bretton Woods system in March 1973, which introduced the floating exchange rates era. This characteristic was discussed at length by Meese and Rogoff in their paper [MR83], which states that "there is no better economic model for exchange rates forecasting during floating exchange rates than the simple random walk" [Gal16]. This result remained widely accepted in literature and became one of the main reasons researchers turned their interest from economic models to ones from other domains to try to solve this particular forecasting problem.

Particularly, in computer science, a model which seemed to yield good prospects of getting to grips with this problem is artificial neural networks. The fact that this model is data driven and can capture the non-linearly nature of this time-series data [KB10] attracted the attention of researchers, which subsequently gave rise to multiple papers and articles. Nevertheless, an issue even for this model appears to be the identification of reliable indicators (besides past data) which contribute to the fluctuation of currency exchange rates.

Thus, the experimental study in this thesis aims to make use of the artificial neural networks and to investigate the usage of external indicators in attempting to deliver reliable prediction models for forecasting one day into the future the following exchange rates: the European Union currency against the British pound (EUR\GBP), the European Union currency against the US dollar (EUR\USD) and the British pound against the US dollar (GBP\USD).

Multiple architectures (such as Feed Forward Neural Networks, Long Short Term Memory Neural Networks or Convolutional Neural Networks), different time series external to the aforementioned exchange rates (like gold and oil prices) and varying lengths of past observations as input are considered in this analysis which also compares the results obtained with existing ones from literature. The present thesis also incorporates the best performant models found in the research phase into a mobile application, named PRICE (*Predicting and Retrieving International Currency Exchange*), whose goal is to deliver reliable information about currency exchange rates. With this idea in mind, the functionalities which PRICE offers to have are the following: forecast the currency exchanges introduced earlier one day into the future, converting money between different currencies at the current day's rates, the possibility to see past values of certain exchange rates and the option to analyze past predictions made by the exposed models. The originality of this thesis is therefore threefold: a performance evaluation of multiple neural networks architectures for tackling the problem of currency exchange forecasting, a comparison of the best models found in this study with existing results in literature and the integration of the best prototypes in a mobile application.

The rest of the thesis is structured as follows. Chapter 1 presents the theoretical background of the intelligent systems put forward by PRICE. Chapter 2 examines the available results in literature regarding currency exchange rates prediction. Chapter 3 discusses the practical aspects of this application, with focus on the application development and the experimental results. The conclusions of this thesis and directions to further continue and improve this research are given in Chapter 4.

Chapter 1

Machine Learning and Prediction for Time Series Data

The purpose of this chapter is to introduce the main theoretical background of which the present thesis made use. The concepts explained include: machine learning, classic types of learning, neural networks based learning, predictive analytics and time series data.

1.1 A brief definition of machine learning

Machine learning (ML) is the "science (and art) of programming computers to learn from data" [Gér19]. From this definition we can extract two keywords: data, which refers to any information made available to the machine and learning, which in the context of ML means the ability of the machine to draw inferences or gather information in order to reach an optimal solution to a specific problem. Nevertheless, to think of learning as a simple ability does not comprise its full significance. Learning can also be characterised as being a complex activity, even for humans, which requires time and can manifest in various ways (for both humans and computers).

1.2 Types of learning

1.2.1 Supervised Learning

Supervised learning works with a dataset which includes the input parameters alongside the correct output and aims to find an optimal function for predicting the output for a new set of inputs. Mathematically, for a set of the form (x, y), this type of learning aims to find a mapping, denoted by f(x), such that the difference between f(x) and y is minimum. Since this type of learning can be used to solve different problems (classification or regression for example), the aforementioned difference is not defined in the algebraic sense, but it rather depends on the context.

1.2.2 Unsupervised Learning

In contrast to supervised learning, this technique receives a dataset composed of only the inputs and focuses on finding patterns, clusters or certain structures between them without interference from humans in the process. For this reason, the scope of unsupervised learning is not always thoroughly defined and further actions are usually decided based on the output. Some examples of problems for which this type of learning is well-suited are clustering and anomaly detection.

1.2.3 Reinforcement Learning

Reinforcement learning is concerned with the kind of learning which occurs when a software agent interacts with an environment. The agent make decisions in this environment based on its past actions (memory) and the availability of new moves at the moment (exploration). For each decision, it is rewarded or punished with a numerical value and it seeks "to learn to select actions that maximize the accumulated reward over time" [WP08]. One of the most important class of problems for which reinforcement learning is well-suited is developing game tactics.

1.3 Neural Networks based Learning

Neural Networks (NN) or Artificial Neural Networks (ANN) is a computing method which resembles the biological neural networks of animals. Essentially, an ANN is a collection of units of computation called artificial neurons (or nodes), organized in layers, which transmit and receive signals (real values) between them. The nodes are connected between them with links (mimicking the synapses and dendrites of biological neurons) which have a certain weight. The weight of a link denotes the influence the first neuron has on the second one. Based on the way the nodes are organized in layers, multiple types of ANNs emerged (Deep Feed Forward, Long/Short Term Memory or Convolutional to name a few). Nonetheless, at the heart of most architectures are the layers presented in the Figure 1.1.

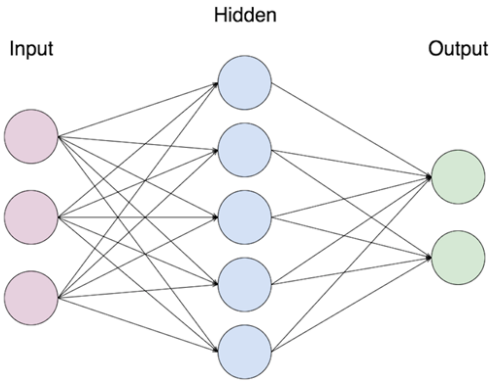


Figure 1.1: A feed forward artificial neural network [Put18]

- *Input Layer*

  Irrespective of the architecture, each ANN has exactly one input layer whose scope is to receive the data (the number of neurons in this layer is expected to be equal to the size of an input vector).

- *Hidden Layer*

  The role of a hidden layer is to capture more complexity between the inputs and to pass this information to other hidden layers or to the output layer. The number on hidden layers can very based on the architecture and the problem that needs to be solved.

- *Output Layer*

  As in the case of the input layer, each ANN has exactly one output layer and its scope is to transmit the result of the computations. This result is a vector of real values with a certain meaning.

1.3.1 Feed Forward Neural Networks

The first type of networks to appear, feed forward neural networks (FFNNs) allow the information to flow only in one direction (from the input to the output nodes and through hidden nodes, if present). They are mostly trained using backpropagation, with the error often being some variation of the difference between the input and the output, thus making them suitable for supervised learning [VL19].

1.3.2 Convolutional Neural Networks

The most notable observation one could make while analysing the architecture of a convolutional neural network (depicted in Figure 1.2) is the appearance of two new types of layers: the convolutional layer and the pooling layer. These two layers are the building block of a CNN since they allow dimensionality reduction while at the same time extracting key features of the input. The features extracted by these layers are then feed to an FFNN or multilayered perceptron (denoted by the Fully Connected Layer and the Output Layer in Figure 1.2) which ultimately returns the output.

- *Convolutional Layer*

  The role of a convolutional layer is to identify where and to what extent a certain feature is contained inside an input data item. The feature in cause is described using a multidimensional array which is convolved with a block, from the input data item, of the same size. This operation outputs a number which denotes the extent to which the feature is present in that block. This procedure is repeated until all data from the input is parsed, with the blocks from the item being obtained by sliding a multidimensional array, with sizes equal to those of the feature described, throughout the input data item.

The final output of this layer is a multidimensional array, reduced in dimensionality by the sizes of the feature, which highlights the blocks that contain the feature.

- *Pooling Layer*

  The main scope of the pooling layer is to reduce the dimensionality of the input to be analaysed, thus reducing the computational power needed by the neural network and avoiding overfitting. The procedure employed here is again blocks sliding, which each block in the input being replaced by either the maximum value in it (max pooling) or the average of the values (average pooling). The intuition behind this layer is that the exact location of a feature is not as important as its relative location to other features.

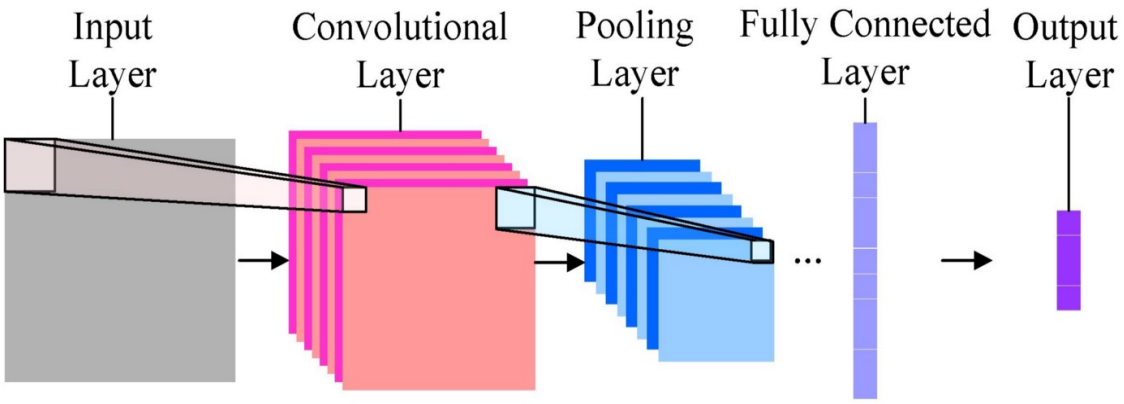


Figure 1.2: Architecture of a CNN used for classification [Cha19]

In practice, CNNs proved to be highly efficient in image recognition. This is largely owed to its ability to mimic the visual cortex functionality in animal brains. Other areas where CNNs produced good results are financial time series and natural language processing through their ability to discover local patterns in data. Another reason for which they are preferred to other architectures (like RNN) in tackling problems with time series data is that they are more computationally efficient.

1.3.3 Long/Short Term Memory Neural Networks

Long/Short Term Memory (LSTM) neural networks are a subtype of the Recurrent Neural Networks (RNN). An RNN can be seen as a "FFNN with a time twist, since neurons are fed information not just from the previous layer but also from themselves from the previous pass" [VL19]. This is achieved through the introduction of loops in the networks, such that information still persists inside it (depicted in the Figure 1.3 through the arcs between the neurons of the same hidden layer). This particular aspect makes the order in which the input is given relevant, since now each input item its related to the others and influences directly the previous and the next one. For that reason, the input can not be seen as a collection of distinct items but rather as a series where each piece shapes and is shaped by the others.

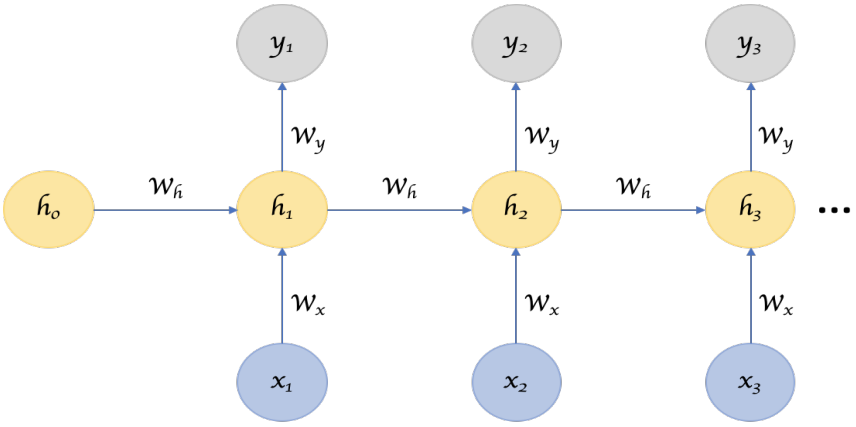


Figure 1.3: A RNN with a single hidden layer [Ven19]

What a LSTM neural network goes and does different from the a classic RNN is that it also changes the structure of the neuron (called memory cell) to retain information from previous passes. As depicted in the Figure 1.4, the memory cell has three gates (input, forget, output) which control how the information flows inside the node.

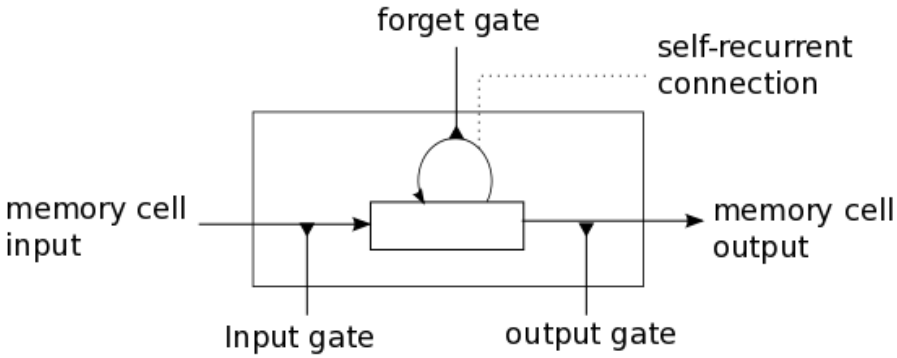


Figure 1.4: Memory cell for a LSTM Neural Network [CC18]

The input gate controls how incoming information alters the state of the memory cell, while the output gate decides the effect the memory state of the cell has on other neurons. The forget gate modifies the memory cell's self-recurrent connection, determining how much is remembered from previous state [CC18]. The reason behind this design was mainly to tackle the exploding and vanishing gradient problems inherent to RNNs and FFNNs [Wan19], [PMB12].

Thus, one class of problems for which LSTM neural networks is well suited is making predictions based on time series data, since, as mentioned before, the input is seen a data series, and lags between important events can be overcame through the remembrance of past states.

Bidirectional LSTM Neural Networks

A special kind of long-short term memory neural networks are the bidirectional ones. The main idea put forth by these is to learn not only from the past, but also from the future. In other words, this means to analyze the input sequence both in the order that it was given and identify how previous input observations influence the following ones, and in reverse order to discover any effect that future values may have on past ones. This is achieved by creating another layer of LSTM cells which has the same structure and works in the same way as the classic LSTM layer does. It also contributes to the next layer of the model as much as the ”normal” layer, with the only difference being the order in which the input sequence is read (as it can be seen from Figure 1.5, where the ”inverse” layer is denoted by *A'*).

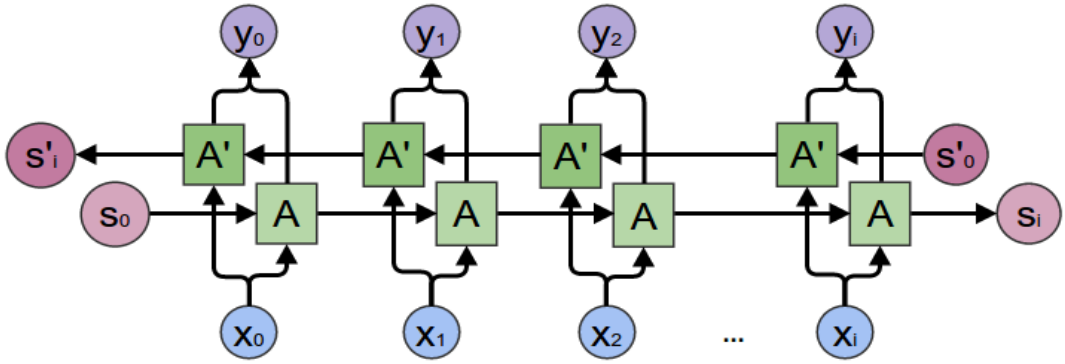


Figure 1.5: Bidirectional LSTM layer [Agg19]

Besides having another alternative for dealing with time-series forecasting problems, this extension brought to classical LSTM layers has proven to be also effective in tackling natural language processing problems.

ConvLSTM Neural Networks

ConvLSTM neural networks can be seen as an architecture which tries to combine the ability of CNNs to reduce dimensionality while at the same time extracting key features of the input with the capacity of LSTM neural networks to identify a temporal dependency between the input sequence. This is achieved by replacing the matrix multiplication operation inside the normal LSTM cell at the level of each gate with a convolution operation (as it can be seen in the Figure 1.6).

This union between these two architectures created a LSTM model which is suitable for 3D input data, thus making it fit for problems with spatial sequence data such as video surveillance or CT scans analysis.

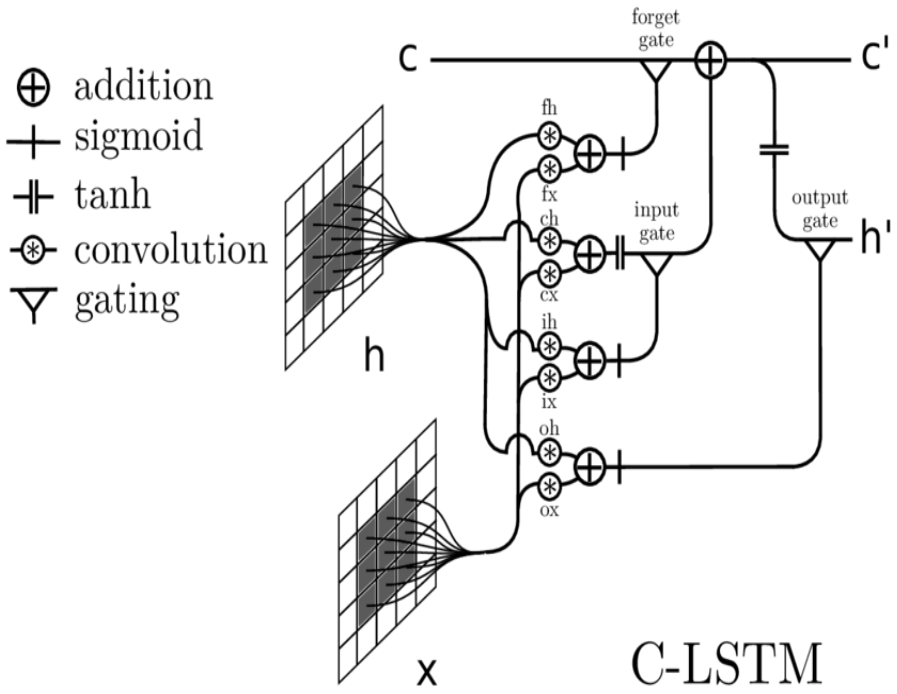


Figure 1.6: ConvLSTM cell [Sto16]

1.4 Predictive Analytics

"Predictive analytics is a category of data analytics aimed at making predictions about future outcomes based on historical data and analytics techniques such as statistical modeling and machine learning" [Edw19]. Until recent times thought as a field only for mathematicians and statisticians, predictive analytics became an important tool for companies and people, with the global market projected to reach approximately 10.95 billion dollars by 2022 [Edw19]. This sudden interest in this field manifested mostly due to its potential to offer insights in regards to investment opportunities, resources and risk management, as well as detection of anomalies. The availability of faster and cheaper computers, of software that is easy to use and of enormous and increasing amount of data also contributed to the spike of interest in this field. Predictive analytics is used nowadays for solving problems like stock prices forecasting, currency exchange rates forecasting, weather prediction, business planning or resources allocation in domains like:

- marketing
- insurance
- telecommunications
- retail
- travel
- capacity planning
- financial services
- manufacturing

The usual steps of a predictive analytics process are described below [Eck07] [Ima]:

1. *Project Definition*

   Define the objectives, desired outcomes and deliverables for the project, and convert them into predictive analytic items (objectives and tasks).

2. *Data Collection*

   Gather data from multiple sources and analyze it to determine the most appropriate and accurate one.

3. *Data Preparation*

   "Select, extract, and transform data upon which to create models" [Eck07].

4. *Modeling*

   "Create, test, and validate models, and evaluate whether they will meet project metrics and goals" [Eck07].

5. *Deployment*

   Insert the analytical model into the decision making process of the business, by sharing the results with business users or by automating certain aspects.

6. *Model Monitoring*

   Review the model performance to ensure that the expected results are provided and improve where possible (minimize redundant activities for example).

1.5 Time Series Data

Time series data refers to data which can be seen as a series of items ordered in time. They came rather naturally in real life, some examples being: weather temperature, currency exchange rates, stock market prices or the quality of air. When analysing a time series, the degree to which the following characteristics are present is important to be identified: autocorrelation, which is "the similarity between observations as a function of the time lag between them" [Pei19], seasonality, which is interested in recurring fluctuations, and stationarity, which means that its "statistical properties do not change over time" [Pei19] (the mean and variance are constant). The primary goal behind analysing time series data is forecasting. Depending on how many time-series are included in the prediction, the problem can be classified as univariate (when only one time-series and its past observations are used for predicting future values) or multivariate (multiple time-series are used for predicting future values of one or multiple time-series).

1.6 Performance Metrics

In the case of the regression problems (like predicting currency exchange rates), suitable metrics for assessing the performance of a model are the following:

- *Mean Squared Error (MSE)* - sensitive to outliers and can be used only for models which have the same unit measure

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (p_i - a_i)^2 \tag{1.1}$$

- *Mean Absolute Error (MAE)* - not sensitive to outliers and can be used only for models which have the same unit measure

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} |p_i - a_i| \tag{1.2}$$

- *Normalized Root Mean Squared Error (NRMSE)* - can be used for models which have different unit measures

$$NRMSE = \frac{\sqrt{\frac{1}{n} * \sum_{i=1}^{n} (p_i - a_i)^2}}{a_{max} - a_{min}} \tag{1.3}$$

where *p* represents the vector of expected output, *a* represents the vector of the actual output, a_{max} and a_{min} represent the maximum and minimum values in the vector *a* and *n* is the size of the vectors.

Chapter 2

Related work on predicting currency exchange rates

The purpose of this chapter is to define the problem tackled by the proposed model and application, whilst presenting different approaches already tried in the fields of econometrics & statistics and intelligent systems.

2.1 Problem Definition

Informally, as the name suggests, currency exchange rate prediction/forecast is a problem regarding estimation. Given that at the moment you can buy a United States dollar with 0.91 euro (European Union currency) for example, it is asked of you to speculate how and with how much this value is going to change over a specified period in the future, for instance a week. Formally, you are required to solve a predictive analytics problem by giving a formula or a mathematical model which make use of economic indicators like inflation rates, interest rates, money supply or the prices of other assets in order for the value of the future exchange rate returned by your prediction system to be as close as possible to the actual value.

The necessity of having a solution as good as possible for this problem is strongly related to the business climate. Since "international transactions are usually settled in the near future, exchange rate forecasting is used to evaluate the benefits and risks attached to the international business environment" [Sus]. Another area where currency exchange rate prediction is used is for observing the stability and growth of certain countries, especially in comparison with others.

The following sections present how this problem was and is still tackled by different fields. Methods and models derived from the studies in the sphere of *Econometrics and Statics* are the focus in the the succeeding section, while the last one examines how artificial intelligence, through the use of *intelligent systems*, attempted to address this issue.

2.2 Econometrics and Statistics

Up until the breakdown of the Bretton Woods system, which pegged all the currencies to the US dollar, with the dollar being pegged itself to gold, currency exchange rates could be predicted to some degree through simple models like Purchasing Power Parity (PPP), Uncovered Interest rate Parity (UIP) or the Relative Economic Strength. Unfortunately, nowadays, these models seem too naive to offer a reliable solution to the problem at hand, mainly due to their structure which is dependent on other indicators which need to be predicted as well, like interest rate or inflation rate. For this reason, this section is concentrated on other methods like Monetary and ARIMA models and the results obtained by using them.

2.2.1 Monetary Models

Monetary models for exchange rate forecasting were developed after the collapse of the Button Woods system. Mainly there are 3 types of models, each with its corresponding exponents: the flexible price monetary model, introduced by the works of Frenkel (1976) and Bilson (1978), the sticky price of Dornbusch (1976) and Frankel (1979), and the sticky price-asset monetary model proposed by Hooper and Morton (1982) [GR92].

The two key assumptions of any monetary model are that purchasing power parity holds continuously over time and the money demands is stable in both the domestic and the foreign countries [ZL07]. With these in place, the basic monetary model can be represented in the following way, as shown by Boyko [Boy02]:

$$s = (m - m*) + \alpha_1(y - y*) + \alpha_2(i - i*) \tag{2.1}$$

where all small letters denote logarithms, asterisk denotes a foreign country, *s* denotes exchange rate, *m* denotes the money supply, *y* denotes real income (or industrial production, or real output), *i* denotes the interest rate, α_1 denotes the domestic income elasticity and α_2 denotes the interest rate semi-elasticity of the demand for money (last 2 assumed to be equal for the domestic and foreign countries). In the formula above, the difference in inflation (π-π*) and difference in accumulated trade balances (tb-tb*) can also be included [Boy02].

2.2.2 ARIMA Models

ARIMA (Auto-Regressive Integrated Moving Average) is a class of models whose objective is to forecast a given time series based on its own past values. This is achieved by making use of its own lags and the lagged forecast errors [Pra19]. Structurally, an ARIMA model is the extension for non-stationary time-series of another model, namely ARMA (Auto-Regressive Moving Average), which itself is the composition of other two modules: AR (Auto-Regressive), whose origin dates back in 1920s from the work of Udny Yule and Eugen Slutsky [Las20], and MA (Moving Average).

AR model

The Auto-Regressive model tries to forecast the values of a time series using a linear function of the past data and white noise error terms. The latter component is used to account for errors which can not be expressed through the linear function alone, but which respect the behaviour of a stationary time series. The number of past values used in the function is called the order, the model usually being denoted *AR(p)*, where p has the aforementioned meaning. Mathematically, the formula is given by the following equation:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_i + \epsilon_t \tag{2.2}$$

where ϕ_1,...,ϕ_p are the parameters of the model, c is a constant and ϵ_t is white noise.

MA model

The Moving-Average model is similar to the AR model in the sense that it also uses past data. However, the time-series in this case has to be stationary since an MA makes use of the mean and, as in the previous case, of white noise error terms. The order of the model is defined again as previously, this time being denoted by *q*. The formula for the MA model is given below:

$$x_t = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \tag{2.3}$$

where θ_1,...,θ_q are the parameters of the model, μ is the mean of the series and ϵ_1,..,ϵ_t are the white noise error terms.

Integration

Besides just combining the aforementioned models, an ARIMA model goes further. It introduces a new component, *Integration*, which is used to ensure that the time series which will be used by the AR and the MA models is stationary. This process is done through differencing, which substitutes the observed values of the series with the difference of consecutive values. Mathematically, instead of applying the model for the series x_t, the series $y_t = x_t - x_{t-1}$ is considered. However, sometimes applying this operation just once is not enough to obtain a stationary time-series. The number of times this process is repeated is called the order of differencing and is denoted by *d*. Thus, the formal notation of this model is *ARIMA(p,d,q)*, where each parameter retains its meaning, and its formula is:

$$y_t^d = c + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \sum_{i=1}^{p} \phi_i y_i^d + \epsilon_t \tag{2.4}$$

where θ_1,...,θ_q are the parameters for the MA part, ϕ_1,...,ϕ_p are the parameters for the AR part, ϵ_1,..,ϵ_t are the white noise error terms and c is a constant [Nau].

2.2.3 Experimental results

Meese and Rogoff exposed the shortcomings of the monetary models in their paper [MR83], where they concluded that **"the random walk performed as well as any estimated model"** [ZL07]. The models included for comparisons were the flexible price model, the sticky price monetary model, and the sticky price model that incorporated the current account).

Făt and Dezsi made a **comparison between the ARIMA method and Exponential Smoothing Techniques (EST)** for forecasting the following exchange rates: the European Union currency against the Romanian leu (**EUR\RON**), the United State dollar against the Romanian leu (**USD\RON**), the British pound against the Romanian leu (**GBP\RON**), the Japanese yen against the Romanian leu (**JPY\RON**), the Chinese renminbi against the Romanian leu (**CNY\RON**) and the Russian rouble against the Romanian leu (**RUB\RON**). They made use of **80 daily observations between 3 January 2011 and 22 April 2011** and found that **the EST method outperformed the ARIMA method** [FD11].

Babu and Reddy **compared the methods of ARIMA, ANN and fuzzy systems**. The currencies rate they tried to forecast were the United State dollar against the Indian rupee (**USD\INR**), the British pound against the Indian rupee (**GBP\INR**), the European Union currency against the Indian rupee (**EUR\INR**), and the Japanese yen against the Indian rupee (**JPY\INR**) by making use of 1,284 daily observations between the dates of January 2010 and April 2015. They concluded that **the ARIMA method has the best perfomance out the 3 methods** for predicting the aforementioned rates [RB15].

Yıldıran and Fettahoğlu found that **a model of the form ARIMA (2,1,0) is suitable for the short term forecasting** of the United States dollar against the Turkish lira rate (**USD\TRY**), while **a model of the form ARIMA (0,1,1) is better for the long term prediction**, by making use of **3,069 daily observations between 3 January 2005 and 8 March 2017**. They also concluded that the **ARIMA method is more effective in short-term prediction** [YF17].

2.3 Intelligent Systems

Until recent times, the problem of forecasting currency exchange rates was seen as a problem for the fields of econometrics and statistics. However, the advance in technology invited fields like computer science, with the sub-field of artificial intelligence in particular, at the table too. This gave rise to intelligent systems which could, on the one hand, mimic human reasoning, but, on the other, also process larger quantities of data available in a way humans could never do (at least for now). One approach which became rather alluring to the scientific community through its ability to capture non-linearity in the data is artificial neural networks. Besides being used as a standalone method, it was also coupled with ARIMA models or genetic programming to generate hybrid systems.

Experimental Results

Galeshchuk [Gal16] used a **multilayer perceptron with the architecture 5-10-1 for forecasting the daily, monthly and quarterly** currency exchange rates between the European Union currency and the US dollar (**EUR\USD**), the Great Britain currency and the US dollar (**GBP\USD**), and between the US dollar and Japanese yen (**USD\JPY**). The observations used were **daily between 01 Jan 2014 until 25 Apr 2014**, **monthly between May 2009 till May 2014** and **quarterly between May 1999 until May 2014**. The **average and maximum relative prediction errors** were **0.2–0.4%** for the exchange rate **EUR\USD**, **0.2–0.9%** for the exchange rate **GBP\USD** and **0.3–1.3%** for the exchange rate **USD\JPY** in the case of the **daily prediction**, **1.3–3.3%** for the exchange rate **EUR\USD**, **2.2–4.5%** for the exchange rate **GBP\USD** and **0.3–1.3%** for the exchange rate **USD\JPY** in the case of the **monthly forecasting**, and **2.3–5.1%** for the exchange rate **EUR\USD**, **1.9–5.0%** for the exchange rate **GBP\USD** and **3.5–10.2%** for the exchange rate **USD\JPY** in the case of the **quarterly prediction**. [Gal16].

Sermpinis et al. [SSD14] studied the forecasting of the exchange rates of the European Union currency against the US dollar (**EUR\USD**), the European Union currency against the Great Britain currency (**EUR\GBP**) and the European Union currency against the Swiss franch (**EUR\CHF**), using **Multi-Layer Perceptron (MLP)**, **Recurrent Neural Network (RNN)** and **Psi-Sigma Network (PSN)** architectures in terms of **neural networks based approaches** and "**Kalman Filter**, **Genetic Programming (GP)** and **Support Vector Regression (SVR)** algorithms as **forecasting combination techniques**" [SSD14]. They used **daily observations which spanned the period 1999-2012** and which were divided **"in three rolling forecasting exercises (F1, F2 and F3)" [SSD14], with F1 focusing on the 1998-2008 decade, F2 on the 2001-2010 period and F3 on the observations between 2003-2012**. Their findings concluded that **the best model was the one that used the SVR method** out of all tried, with **MAE measures of 0.0039, 0.0048, 0.0037** and **RMSE measures of 0.0053, 0.0058, 0.0051** for the **F1, F2 and F3 periods respectively** in the case of the **EUR\USD** exchange rate, **MAE measures of 0.0036, 0.0049, 0.0035** and **RMSE measures of 0.0071, 0.0081, 0.0068** for the **F1, F2 and F3 periods respectively** in the case of the **EUR\GBP** exchange rate, and **MAE measures of 0.0051, 0.0057, 0.0048** and **RMSE measures of 0.0056, 0.0059, 0.0052** for the **F1, F2 and F3 periods respectively** in the case of the **EUR\CHF** exchange rate. From the neural networks architectures the one that fared the best was the PSN, with MAE measures of 0.0053, 0.0059, 0.0048 and RMSE measures of 0.0069, 0.007, 0.0064 for the F1, F2 and F3 periods respectively in the case of the EUR\USD exchange rate, MAE measures of 0.0052, 0.0066, 0.0051 and RMSE measures of 0.0085, 0.0092, 0.0081 for the F1, F2 and F3 periods respectively in the case of the EUR\GBP exchange rate, and MAE measures of 0.0067, 0.0068, 0.0059 and RMSE measures of 0.0071, 0.0075, 0.0064 for the F1, F2 and F3 periods respectively in the case of the EUR\CHF exchange rate. Further in their study, Sermpinis et al. also explored the models' performances when used for trading.

Yasir et al. [YDA+19] made use of **an convolutional neural network architecture** and **explored the impact sentiments play in forecasting** the following currency exchange rates: the British pound against the United States dollar (**GBP\USD**), the Hong Kong dollar against the United States Dollar (**HKD\USD**) and the Pakistani rupee against the United States dollar (**PKR\USD**). Their observations covered the period from **April 2008 to January 2019 daily** and were **coupled with crude oil prices and gold price index as indicators**. For the **sentiment analysis they used tweets regarding local and global events**. The results they found suggest **the architecture proposed performed better in the case when the sentiment analysis was included** [YDA+19].

Zhang created **a hybrid model between ARIMA and ANN**. They forecasted the United States dollar against the British pound rate (**USD\GBP**) using **731 weekly observations between 1980 and 1983** and made **a comparison between the hybrid model, an ANN model and an ARIMA model**. The study concluded that **the hybrid model outperforms the others**. [Zha03].

Rehman et al. [RKM14] proposed a model which makes use of ANNs, but also of genetic programming. They used **a Recurrent Cartesian Genetic Programming evolved Artificial Neural Network (RCGPANN)** which was feed **historical data sets of 1000 days** for five different currencies (**Japanese Yen, New Zealand Dollars, Canadian Dollars, Korean Won and Indonesian Rupia all pegged against the US dollar), starting from the 1st of February, 2003**. The results they obtained were a **98.872% accuracy, a mean absolute percentage error value of 1.1280%** and **a better perfomance when compared with an ARMA model and a multilayer perceptron** [RKM14].

Chapter 3

PRICE: A software solution for predicting currency exchange rates

The aim of this chapter is to present the development process of the application named PRICE and to describe the experimental results obtained during the creation and choice of the models used by it.

3.1 Application development

3.1.1 Functional specifications

The main objective of PRICE is to enable people to find information in regards to currency exchange rates. The focus of the information is a prediction concerning how certain rates (for example the European Union currency against the British pound) are going to change on a daily basis, but it is also going to offer past and current figures regarding specified rates.

With this in mind, PRICE proposes to offer the following functionalities (which can also be visualised in the Figure 3.1): *predict certain currency exchange rates one day into the future*, *display past predictions of certain currency exchange rates*, *convert a sum of money between different currencies* and *show past values of specified currency exchange rates*. The predicting component is realized through an intelligent system which makes use of the artificial neural network model of computation. The architecture of the model was decided based on experimental results between several architectures, namely feed-forward ANNs, long/short term memory ANNs, convolutional ANNs and combinations of the three. It is worth mentioning that a single model was chosen as representative for all the currency exchange rates for which PRICE offers the prediction functionality and which best fits the EUR\GBP rate, as revealed by the comparisons (the chosen model performed rather poorly on the GBP\USD rate, so a little effort was also spend in adapting it for this rate as well). A more detailed review of each of the functionalities of PRICE is shown below the Figure 3.1, while an elaborate analysis of the model is presented in the section *Experimental Results*.

The aforementioned functionalities come in the form of a mobile application. This decision was made in order to ease the use PRICE for people and with the knowledge that people nowadays tend to prefer spending time on applications rather than on browser when they are on their mobile devices [Spe14].

Users who might benefit from such an application are mainly potential investors. Information in regards with the currency of a certain country and how it may change (in comparison with stable economies) can be the decisive factor in the decision of making a certain investment or not. Nevertheless, ordinary people may find this application valuable too since it can provide them with information regarding the currency of the country they may visit in the immediate future.

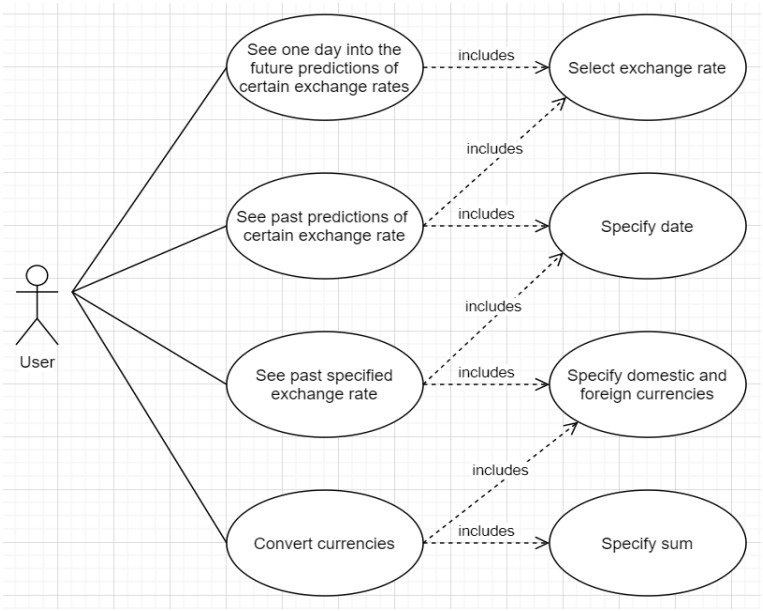


Figure 3.1: Use Case Diagram

Use Case 1

1. *Use Case Description Information*

   1.1 Name

   Convert currencies

   1.2 Goal

   Convert a sum of money from one currency to another.

   1.3 Pre-condition

   Stable internet connection is required.

   1.4 Post-condition

   None.

   1.5 Constraints/Issues/Risks

   The server from which the exchange rates are retrieved might be unavailable.

1.6 Trigger Event(s)

In order to convert the sum of money specified from the local currency to the foreign currency, the user must press the "Convert" button.

1.7 Primary Actor

Any person who has the PRICE application on their mobile device.

2. Use Case Pathway Names

2.1 Primary Path (Convert currencies)

Firstly, the user completes the following steps in any order: select the local currency, by tapping on the first drop down list (from top to bottom), then scrolling through the options and finally pressing on the desired currency; select the foreign currency, by tapping on the second drop down list (from top to bottom), then scrolling through the options and finally pressing on the desired currency; specify the sum of money to be converted, by tapping on the only editable text field and then typing the desired sum. Secondly, the user presses the "Convert" button and they finally see the converted sum in the text field below the button.

2.2 Exception Path(s)

2.2.1 Local and/or foreign and/or sum missing

If the user does not select the local currency and/or does not select the foreign one and/or does not type a sum, and presses the "Convert" button, an error dialog will be displayed explaining what has happened (more exactly, which input was not provided).

2.2.2 Sum is not a number

If the sum provided by the user is not a proper number, an error dialog will be displayed explaining that.

Use Case 2

1. Use Case Description Information

1.1 Name

See past exchange rates

1.2 Goal

Retrieve the exchange rate between two specified currencies at a past date.

1.3 Pre-condition

Stable internet connection is required.

1.4 Post-condition

None.

1.5 Constraints/Issues/Risks

The server from which the exchange rates are retrieved might be unavailable.

1.6 Trigger Event(s)

In order to retrieve the exchange rate specified by the local and the foreign currencies

and the chosen date, the user must press the "Get rate" button.

1.7 Primary Actor

Any person who has the PRICE application on their mobile device.

2. Use Case Pathway Names

2.1 Primary Path (See past exchange rates)

Firstly, the user completes the following steps in any order: select the local currency, by tapping on the first drop down list (from top to bottom), then scrolling through the options and finally pressing on the desired currency; select the foreign currency, by tapping on the second drop down list (from top to bottom), then scrolling through the options and finally pressing on the desired currency; specify the date, by tapping on the "Select a date" button and then selecting a date from a calendar. Secondly, the user presses the "Get a rate" button and they finally see the specified exchange rate in the text field below the button.

2.2 Exception Path(s)

2.2.1 Local and/or foreign missing

If the user does not select the local currency and/or does not select the foreign one, and presses the "Get rate" button, an error dialog will be displayed explaining what has happened (more exactly, which input was not provided).

2.2.2 Date missing

If the user does not select a date, the implicit date will be considered the current one.

Use Case 3

1. Use Case Description Information

1.1 Name

Predict certain exchange rates

1.2 Goal

Predict a chosen exchange rate from the list available one day into the future.

1.3 Pre-condition

Stable internet connection is required.

1.4 Post-condition

None.

1.5 Constraints/Issues/Risks

The server from which the predictions are retrieved might be unavailable.

1.6 Trigger Event(s)

In order to retrieve the prediction for the exchange rate chosen, the user has to press the "Get prediction" button.

1.7 Primary Actor

Any person who has the PRICE application on their mobile device.

2. Use Case Pathway Names

2.1 Primary Path (Get predictions)

Firstly, the user has to select the exchange rate, by tapping on the drop down list. Then, the user presses the "Get prediction" button and sees the prediction in the text field below the button.

2.2 Exception Path(s)

2.2.1 Exchange rate missing

If the user does not select an exchange rate, and presses the "Get prediction" button, an error dialog will be displayed explaining what has happened (more exactly, that no exchange rate was chosen).

Use Case 4

1. Use Case Description Information

1.1 Name

See past predictions of exchange rates

1.2 Goal

See past predictions made by PRICE for a chosen exchange rate and date.

1.3 Pre-condition

Stable internet connection is required.

1.4 Post-condition

None.

1.5 Constraints/Issues/Risks

The server from which the predictions are retrieved might be unavailable.

1.6 Trigger Event(s)

In order to retrieve the prediction for the exchange rate and date chosen, the user has to press the "Get prediction" button.

1.7 Primary Actor

Any person who has the PRICE application on their mobile device.

2. Use Case Pathway Names

2.1 Primary Path (Get predictions)

Firstly, the user has to select the exchange rate, by tapping on the drop down list. Secondly, they have to specify the date, by tapping on the "Select a date" button and then selecting a date from a calendar. Finally, the user presses the "Get prediction" button and sees the prediction in the text field below the button.

2.2 Exception Path(s)

2.2.1 Exchange rate missing

If the user does not select an exchange rate, and presses the "Get prediction" button, an error dialog will be displayed explaining what has happened (more exactly, that no ex-

change rate was chosen).

2.2.2 Date missing

If the user does not select a date, the implicit date will be considered the current one.

3.1.2 Technologies

The technologies used in the development of PRICE are presented below and are mainly divided in the following categories: *programming languages* and *APIs, cloud services, frameworks, libraries, SDKs and toolkits*. These categories are further divided into subcategories based on where this technology exactly fits in the PRICE application: mobile application, server-side or model design.

Programming Languages

Model and Server

Python is a high-level programming language first released in 1991. Designed by Guido von Rossum, Python can be classified as an interpreted general-purpose programming languages which promotes code readability by adhering to the off-side rule (where blocks of code are characterised by their identation). The main programming paradigms that can be identified inside Python are functional, imperative, object-oriented, structured and reflective, while the main operating systems on which it can be run are Linux, macOS and Windows [pyt]. It was chosen as the language for the creation of the model and the implementation of the server for the multitude of machine learning libraries and frameworks available as well as the ease with which a RESTful Web service can be set up.

Mobile application

Dart is a high-level programming language developed by Google and designed by Lars Bak and Kasper Lund. A multi-paradigm programming language (functional, imperative, object-oriented, reflective), Dart was firstly released in 2011, has a C-style syntax and is a cross-platform programming language heavily used in creating desktop, mobile or web applications. Being the language in which applications developed using the Flutter SDK are written in, the choice of Dart as a programming language for the mobile application was constrained by the choice of Flutter as the SDK.

APIs, Cloud Services, Frameworks, Libraries, SDKs, Toolkits

Model

Keras is an open-source library written in the Python programming language whose aim is to facilitate the fast creation and design of deep neural networks. Available to run on top TensorFlow, CNTK, or Theano, Keras centres on being human-friendly (“designed for human beings, not machines” [ker]), modular and extensible, which are also the reasons for being chosen for the development of the model.

Tensorflow is a free and open-source software library created for enabling symbolic computation and development of machine learning applications. Written in Python, C++ and CUDA, TensorFlow was developed by the Google Brain team (and released in 2015) and guarantees stable Python and C APIs. It is available on multiple platforms (Linux, macOS, Windows, Android) and was chosen in the implementation of PRICE for its close connection with Keras (as backend).

Numpy is an open-source software library whose aim is to facilitate the use and manipulation of large, multi-dimensional arrays and matrices in the Python programming language. It also offers high-level mathematical functions which can be applied on these arrays and matrices. Numpy was a prerequisite for using Keras as the deep learning library for developing the model.

pandas is free software library whose main purpose is to help in manipulating and analysing data. Written for the Python programming language, it provides data structures and operations for manipulating numerical tables and time series. In the development of the PRICE application, it was used for reading and refining data from .csv files.

H5py is a Python package which enables manipulation of HDF5 files. Its usage in the PRICE application was for saving and retrieving the chosen model's best run's weights since this is the file format chosen by Keras.

Matplotlib is a plotting library written in and for the Python programming language. In designing the model, it was used for plotting graphs in the training and testing phases of the model in order to get a better understanding of the model's accuracy.

Server

Flask is micro web framework written by Armin Ronacher in Python. Initially released in 2010, Flask is considered to be a microframework due to its lack of a database abstraction layer or form validation which are provided through third-parties libraries. The choice of Flask in combination with Flask-RESTful (an extension for building REST APIs) and Flask-SQLAlchemy (an extension which adds support for SQLAlchemy) as the web framework for PRICE was made due to the low complexity required by the server.

SQLAlchemy is an open-source SQL toolkit and object-relational mapper designed for the Python programming language by Michael Bayer and initially released in 2006. At the level of the server, SQLAlchemy was used for caching already computed predictions.

Requests is an HTTP library written for Python which aims to make HTTP requests easier and more human accessible. Its main purpose in the PRICE application was to retrieve the necessary input data for the prediction.

DigitalOcean is cloud infrastructure provider whose aim is to enable developers to deploy and scale their applications. Used by businesses like Docker, Slack or GitLab, PRICE made use of their cloud service for hosting the Prediction Server (the student benefits they offer was an important decision factor).

Mobile application

Flutter is an open-source UI software development kit (SDK) developed by Google and released in 2017. It is mainly used for developing applications on Android and iOS. The main advantage of Flutter over similar SDKs and the key reason for implementing PRICE using it is that even if it is a platform for developing non-native applications, the code is natively compiled by making use of Dart's native compilers.

Exchange rates API is a free service from where current and past exchange rates published by the European Central Bank can be retrieved through HTTP GET requests and whose domain is `https://exchangeratesapi.io/`. PRICE makes use of this service directly in implementing the "Convert currencies" and "See past exchange rates" functionalities as well as indirectly for the "Predict certain exchange rates" and "See past prediction of exchange rates" features for fetching the input data for the prediction model.

3.1.3 Design and Implementation

Architectural Design

The architectural design of PRICE greatly reflects the nature of a mobile application. As it can be seen in the Figure 3.2, the heavy computation and the storage of data is placed in the hands of external servers and services, while the mobile application just requests and display the information. More exactly, the majority of data is retrieved from the Exchange rates API service, while the predictions are calculated by the Prediction Server. Additionally, already computed predictions are saved in a database from which the Prediction Server can retrieve them such that there is no need to compute them for every request. For calls which ask for forecasts which have not been computed yet, the Prediction server first retrieves the necessary data for the computation from the Exchange rates API and then inserts the newly calculated prediction into the database.

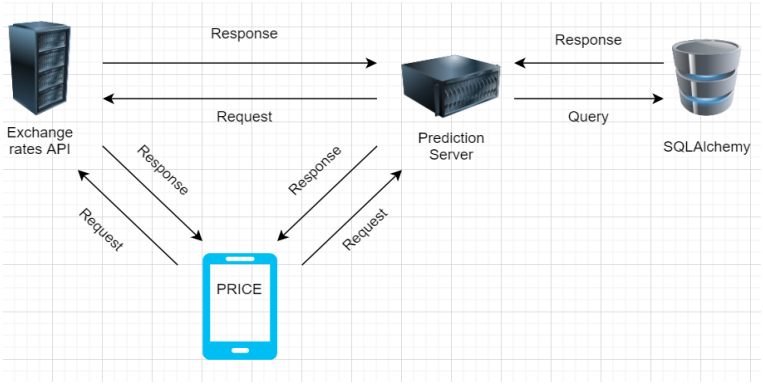


Figure 3.2: PRICE's Architecture Diagram

A more detailed description of the interactions between the mobile application, the Prediction Server, the database and the Exchange rates API can be seen in the Figures 3.3, 3.4, 3.5 and 3.6, which depict the sequence diagrams for the functionalities of PRICE.

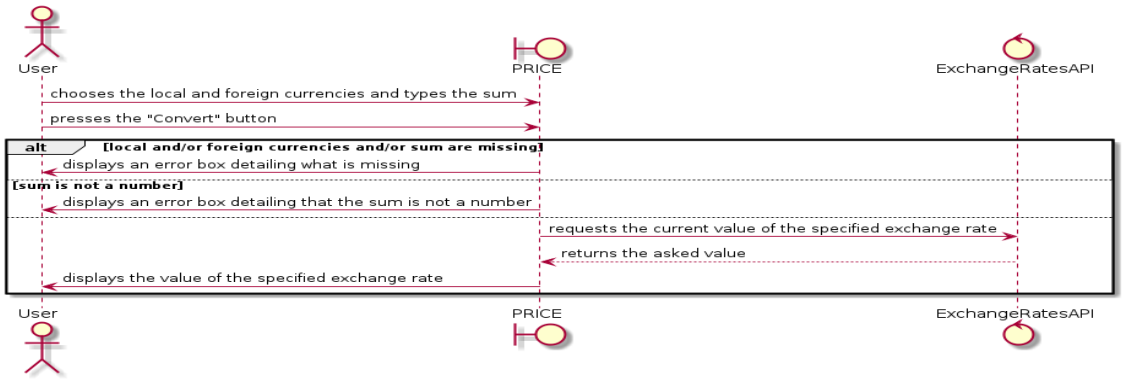


Figure 3.3: Convert Currencies Sequence Diagram

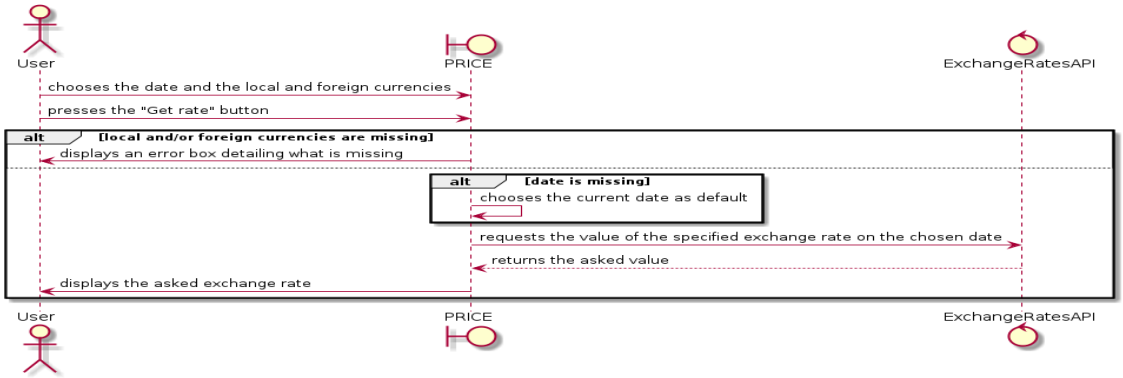


Figure 3.4: See Past Exchange Rates Sequence Diagram

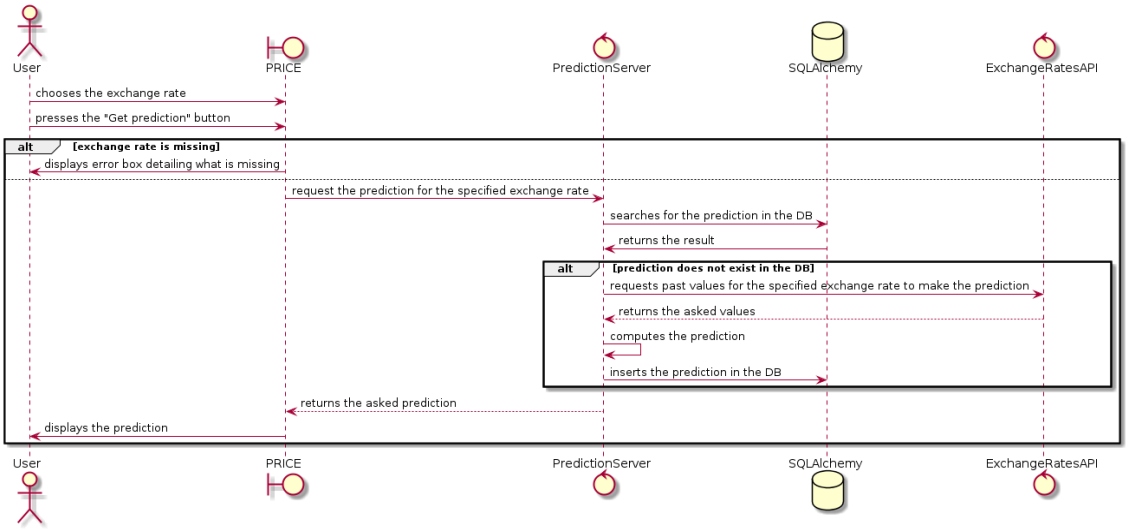


Figure 3.5: Predict Exchange Rates Sequence Diagram

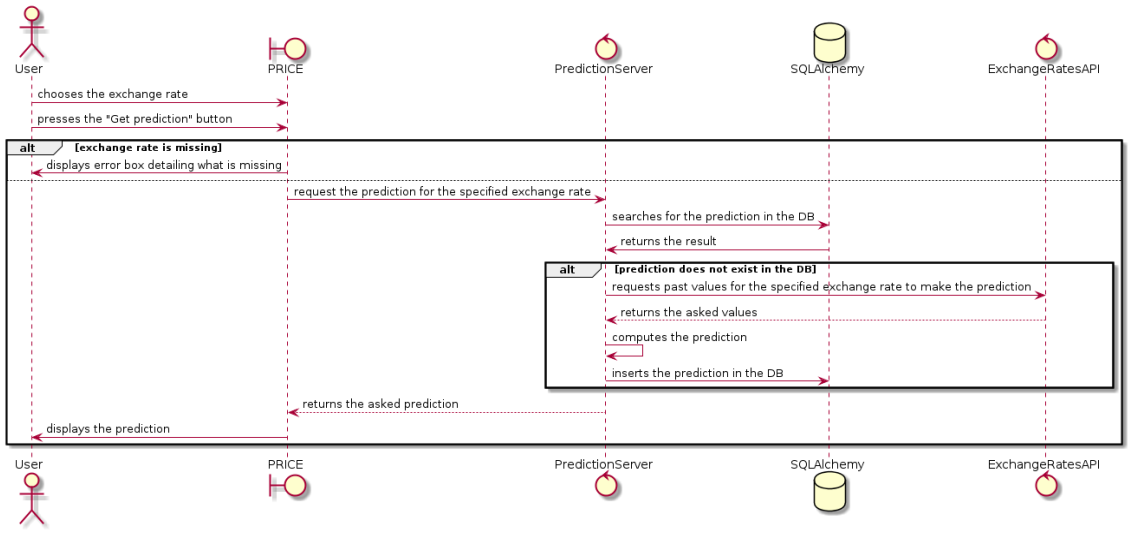


Figure 3.6: See Past Predictions of Exchange Rates Sequence Diagram

Components Design and Implementation

The mobile application was mostly broke down into pieces based on the functionalities which PRICE promised to have. As illustrated in the class diagram from the Figure 3.7, there exists 4 main groups of classes separated by files (*main.dart*, *predictionPage.dart*, *pastPage.dart*, *pastPredictionPage.dart*), each implementing an use case of PRICE. The principal classes in each file are *_CurrentWidgetState*, *_PredictionWidgetState*, *_PastWidgetState* and *_PastPredictionWidgetState.dart* respectively. They control the interaction with the user by retrieving, and validating if necessary, their input and by attaching certain functionalities to certain components (for example, at the press of some buttons, calls to the Exchange Rates API or the Prediction Server can be triggered). The implementation of such a functionality is depicted in the Figure 3.8 where the button *_getPredictionButton* is tasked with initiating a call to the Prediction Server for retrieving the forecast of the specified exchange rate and the displaying in the text field. Alongside the aforementioned classes there are also helper classes like *ServerHelper* (which takes care of the calls between the application and the Prediction Server and the Exchange rates API), *TimezoneHelper* (which handles the conversion between different time zones) and *DrawerOnly* (which implements the navigation part of the app), as well as built-in classes from Flutter which are mostly used for defining and controlling the layout (*TextField*, *RaisedButton*, *TextEditingController*) and from which the main ones need to inherit as part of the structure of an application developed in Flutter (*StatelessWidget*, *State<T>*, *StatefulWidget*).

The Prediction Server has a rather low complexity. It only exposes 3 main routes, one for each of the exchange rates that PRICE offers to predict (EUR\USD, EUR\GBP, GBP\USD). The routes are of the form *http://134.122.66.105:5001/prediction/rate/<day>*, where the rate is specified as a concatenation of the local and foreign currencies' ISO 4217 code and the day is a date specified in the format *YYYY-MM-DD* (for example, for retrieving the prediction of the EUR\GBP rate on 21st of March 2020, one would call the route

http://134.122.66.105:5001/prediction/EURGBP/2020-03-21). The routes are called through an HTTP GET request and return a JSON object of the form '{"prediction": value}', where the value is a number represented as a string (this is due to the inability of Flask to jsonify numpy float numbers). The Figures 3.9 and 3.10 illustrates the implementation of such a route alongside the function which exposes the model used for the forecast. As already mentioned, each prediction computed is stored in a SQLAlchemy database. Only one table is necessary for storing them and its schema is the following: column *ID* - Integer, primary key, autoincrementing, used for unique identification of the prediction; column *day* - Date, not nullable, used for storing the date of the prediction; column *rate* - String, not nullable, used for storing the rate this prediction is referring to; column *value* - Float, not nullable, used for storing the value of the prediction. The Prediction Server is deployed in the internet through the use of the DigitalOcean cloud service, as indicated earlier, with the actual machine on which it runs being located in Frankfurt, Germany. The main specifications of the machine are 1GB of memory and 25GB of storage data, and, for a direct access to it, one would need to establish an SSH connection using the developer's credentials.

3.1.4 Testing

From a software engineering point of view, the testing levels used in the development of PRICE were unit testing through adequate frameworks (pytest more explicitly) and integration and system testing by means of manual testing (with more emphasis on the last two). Due to the reliance on external services like ExchangeRatesAPI and DigitalOcean, only small components independent of them could be unit tested, like the auxiliary functions in the Prediction Server. The strategy adopted in the implementation of these tests was the black-box method since it enabled the developer to detach themselves from the written code and have a more critical look at the behaviour of the functions for boundary values (a snippet of these tests can visualized in the Figure 3.11). In the case of the integration testing, the methodology used was bottom-up. Firstly, the functionalities of the Prediction Server were tested gradually until it was ensured that they work properly (in this case, the integration was explicitly with the ExchangeRatesAPI and the database). Then, the mobile application functionalities which were independent of the Prediction Server were guaranteed to work (namely the ones which interact directly with the ExchangeRatesAPI). The system testing (which can also be seen as the final step of the integration testing) finally occurred when the features of the mobile dependent on the Prediction Server needed to be verified that work. The choice of manual testing in the case of these levels was made due to the scale of the application which, from the developer's point of view, was a small one.

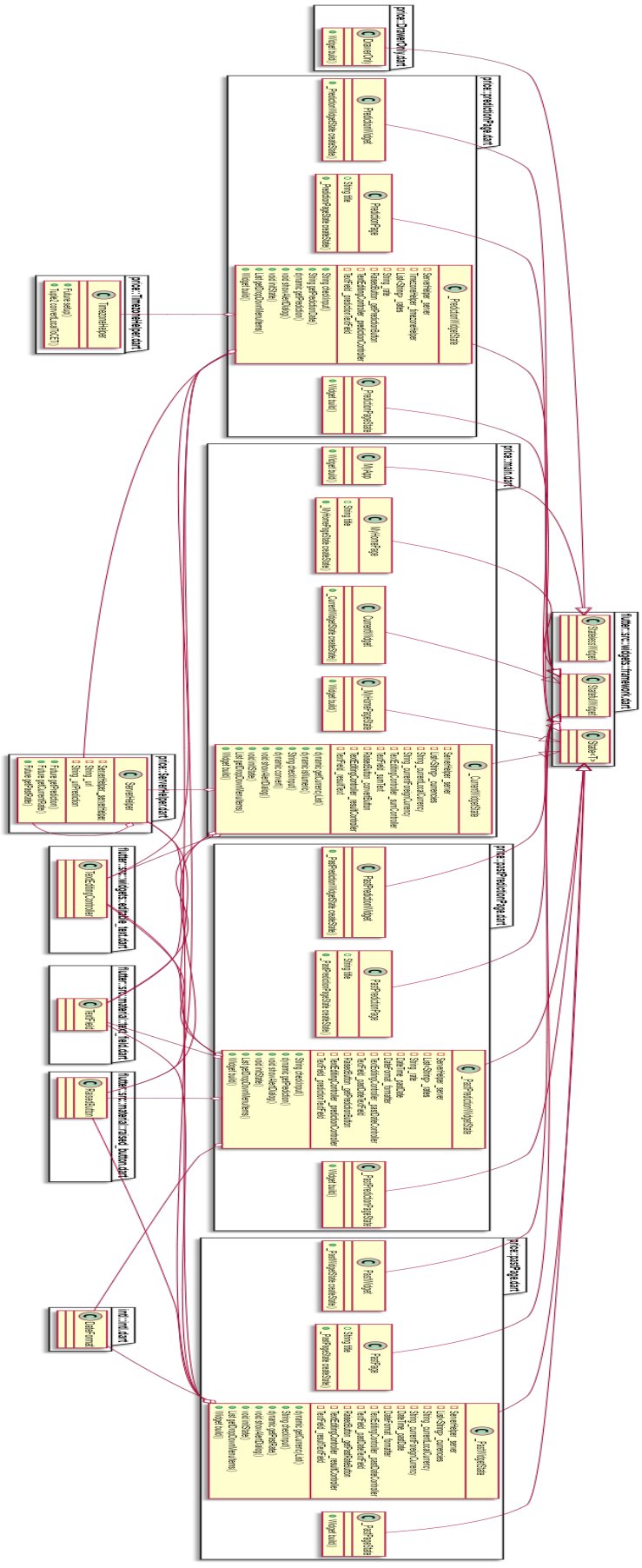


Figure 3.7: Class Diagram for the mobile application

```
/// The function which collects the input from the user and retrieves the prediction for the specified exchange rate
/// This function is bound to the getPredictionButton button
getPrediction() async{
  //_timezoneHelper.convertLocalToDetroit();
  var connectivityResult = await (Connectivity().checkConnectivity());
  // check for network connectivity
  if (!(connectivityResult == ConnectivityResult.mobile || connectivityResult == ConnectivityResult.wifi)){
    // if no connection is available, explain it to the user
    showAlertDialog("Oops, you did something wrong", "Please connect the device to Internet!");
  }
  else{
    // check the input from the user
    String message = checkInput();
    if(message.isEmpty){
      // if there is no problem with the input, make the call to the server
      Tuple2<int, int> time = _timezoneHelper.convertLocalToCET();
      double result;
      if(time.item1 >= 16 && time.item2 >= 30) {
        result = await _server.getPrediction(_rate.substring(0, 3), _rate.substring(4, 7),
            DateTime.now().add(Duration(days: 1)));
      }
      else{
        result = await _server.getPrediction(_rate.substring(0, 3), _rate.substring(4, 7),
            DateTime.now());
      }
      _predictionController.text = result.toStringAsPrecision(6);}
    else{
      // display error message to the user explaining where the mistakes were
      showAlertDialog("Oops, you did something wrong", message);
    }
  }
}
```

Figure 3.8: Code example from the mobile application

```
def construct_model_eur_gbp():
    """
    Function which constructs the prediction model for the EUR/GBP exchange rate
    :return: an instance of the Keras Sequential Model
    """
    model = keras.models.Sequential()
    model.add(keras.layers.LSTM(100, activation='relu', input_shape=(15, 1)))
    model.add(keras.layers.Dense(1))
    model.compile(optimizer='adam', loss='mse',
                  metrics=[keras.metrics.mae, keras.metrics.RootMeanSquaredError()])
    # the weights are loaded from the file which stored the best run of the model
    model.load_weights('lstm_vanilla_eur_gbp_univariate_15_past_observations_run_2.h5')
    return model
```

Figure 3.9: Code example of model implementation from the Prediction Server

```
@app.route('/prediction/EURGBP/<day>')
def predict_eur_gbp(day):
    """
    Function which exposes the route from which predictions for the EUR/GBP exchange rate can be retrieved
    :param day: String date, the date of the prediction
    :return: JSON Object, the value of the prediction as a JSON Object of the form {"prediction": "value"},
             where value is a String
    """
    # create date object
    date_obj = datetime.strptime(day, '%Y-%m-%d').date()
    # search for the prediction in the database
    prediction_object = Prediction.query.filter_by(day=date_obj, rate="EURGBP").first()
    if prediction_object is None:
        # compute the prediction since none was found
        # construct the model
        model = construct_model_eur_gbp()
        # get the data for the prediction
        data = get_data(date_obj - timedelta(days=1), 15, "EUR", "GBP")
        data = array(data)
        data = data.reshape((1, 15, 1))
        # compute the prediction
        new_prediction = model.predict(data)
        new_prediction = new_prediction[0][0]
        # store the prediction
        prediction_object = Prediction(day=date_obj, value=new_prediction, rate="EURGBP")
        db.session.add(prediction_object)
        db.session.commit()
        # expose the prediction as an JSON object
        prediction = {"prediction": str(new_prediction)}
        return jsonify(prediction)
    else:
        # since it was found in the database, expose the prediction as an JSON object
        prediction = {"prediction": str(prediction_object.value)}
        return jsonify(prediction)
```

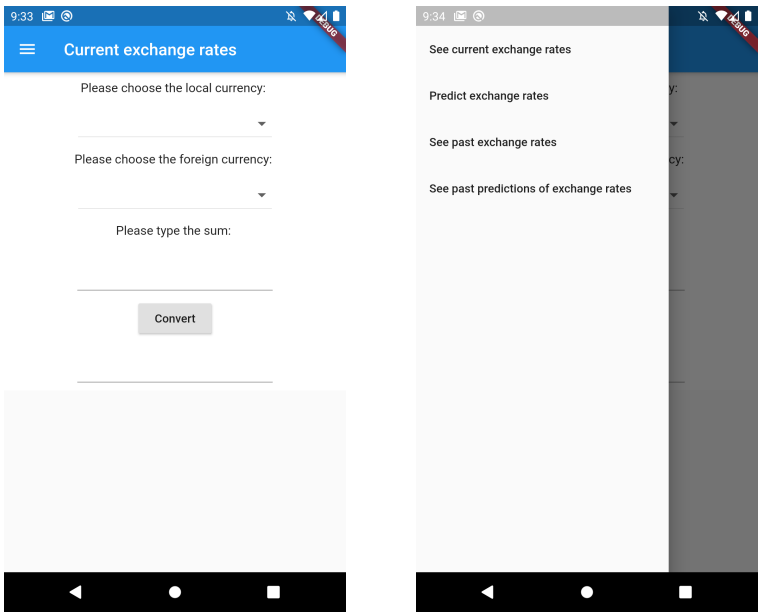
Figure 3.10: Code example of route implementation from the Prediction Server

```
class TestGetWeekdayNDaysAgo(unittest.TestCase):

    # test for 1 days back
    def test1(self):
        self.assertEqual(get_weekday_n_days_ago(datetime.strptime("2020-05-26", '%Y-%m-%d'), 1), datetime.
                         strptime("2020-05-25", '%Y-%m-%d'))

    # test for 29 days back
    def test2(self):
        self.assertEqual(get_weekday_n_days_ago(datetime.strptime("2020-04-27", '%Y-%m-%d'), 20), datetime.
                         strptime("2020-03-30", '%Y-%m-%d'))

    # test for 1st of May
    def test3(self):
        self.assertEqual(get_weekday_n_days_ago(datetime.strptime("2020-05-08", '%Y-%m-%d'), 5), datetime.
                         strptime("2020-04-30", '%Y-%m-%d'))

    # test for a Monday
    def test4(self):
        self.assertEqual(datetime.strptime("2020-05-04", '%Y-%m-%d'),
                         get_weekday_n_days_ago(datetime.strptime("2020-05-25", '%Y-%m-%d'), 15))

    # test for a Tuesday
    def test5(self):
        self.assertEqual(get_weekday_n_days_ago(datetime.strptime("2020-05-26", '%Y-%m-%d'), 15), datetime.
                         strptime("2020-05-05", '%Y-%m-%d'))

    # test for a Wednesday
    def test6(self):
        self.assertEqual(get_weekday_n_days_ago(datetime.strptime("2020-05-27", '%Y-%m-%d'), 15), datetime.
                         strptime("2020-05-06", '%Y-%m-%d'))

    # test for a Thursday
    def test7(self):
        self.assertEqual(get_weekday_n_days_ago(datetime.strptime("2020-05-28", '%Y-%m-%d'), 15), datetime.
                         strptime("2020-05-07", '%Y-%m-%d'))
```

Figure 3.11: Example of unit testing

3.1.5 User manual

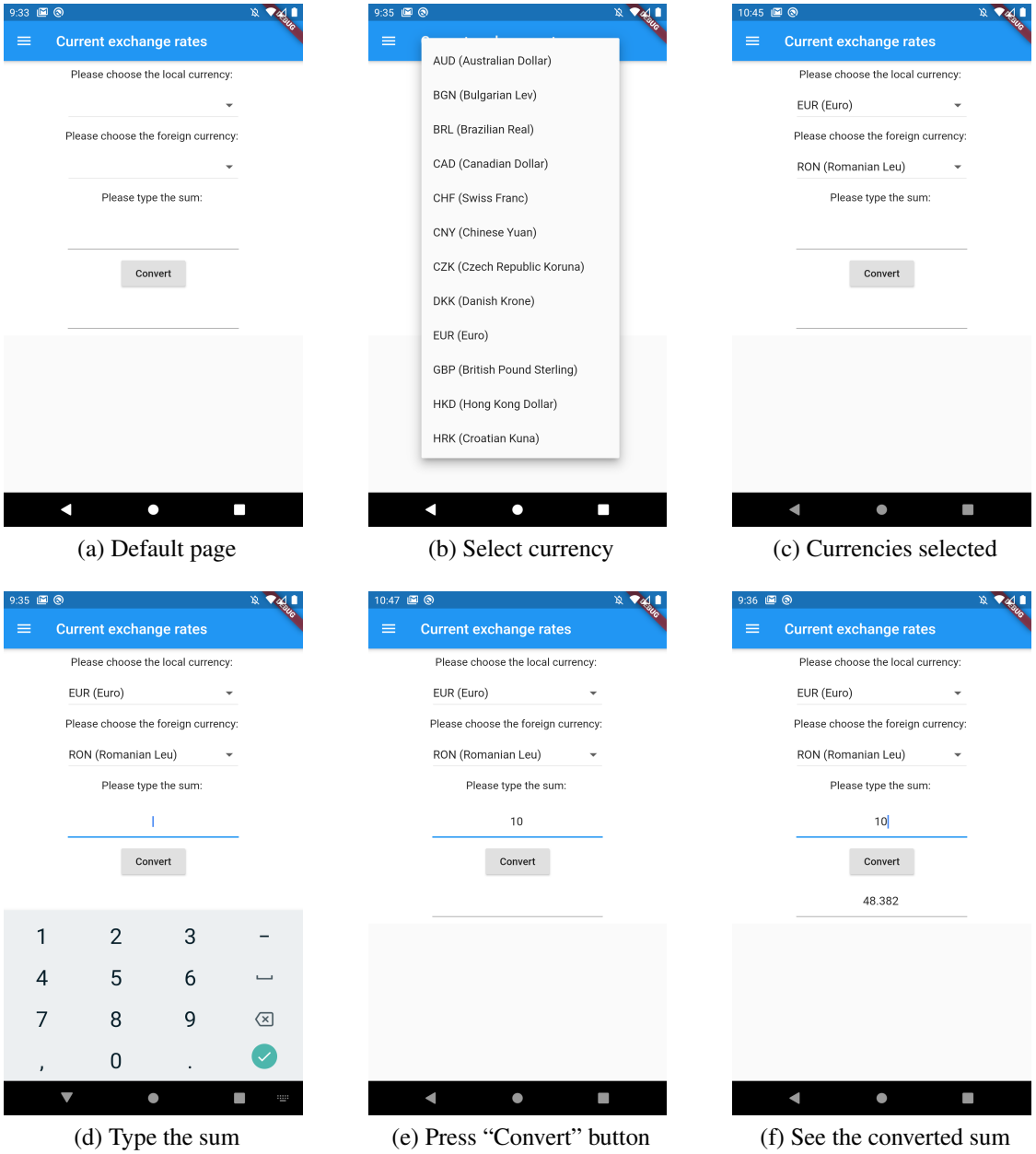
The outlook of the application and a basic step by step manual are presented in the pictures below. The Figure 3.12 depicts the default page of the application (which takes care of the conversion functionality) alongside the navigation drawer (which handles the navigation between the features and is triggered by pressing the drawer icon at the top right of each page).



(a) Home Page (b) Navigation Drawer

Figure 3.12: The first look at PRICE

The visual guide for the first feature, “Convert currencies”, is shown in the Figure 3.13. As it can be seen, first the the currencies which define the exchange rate must be chosen by



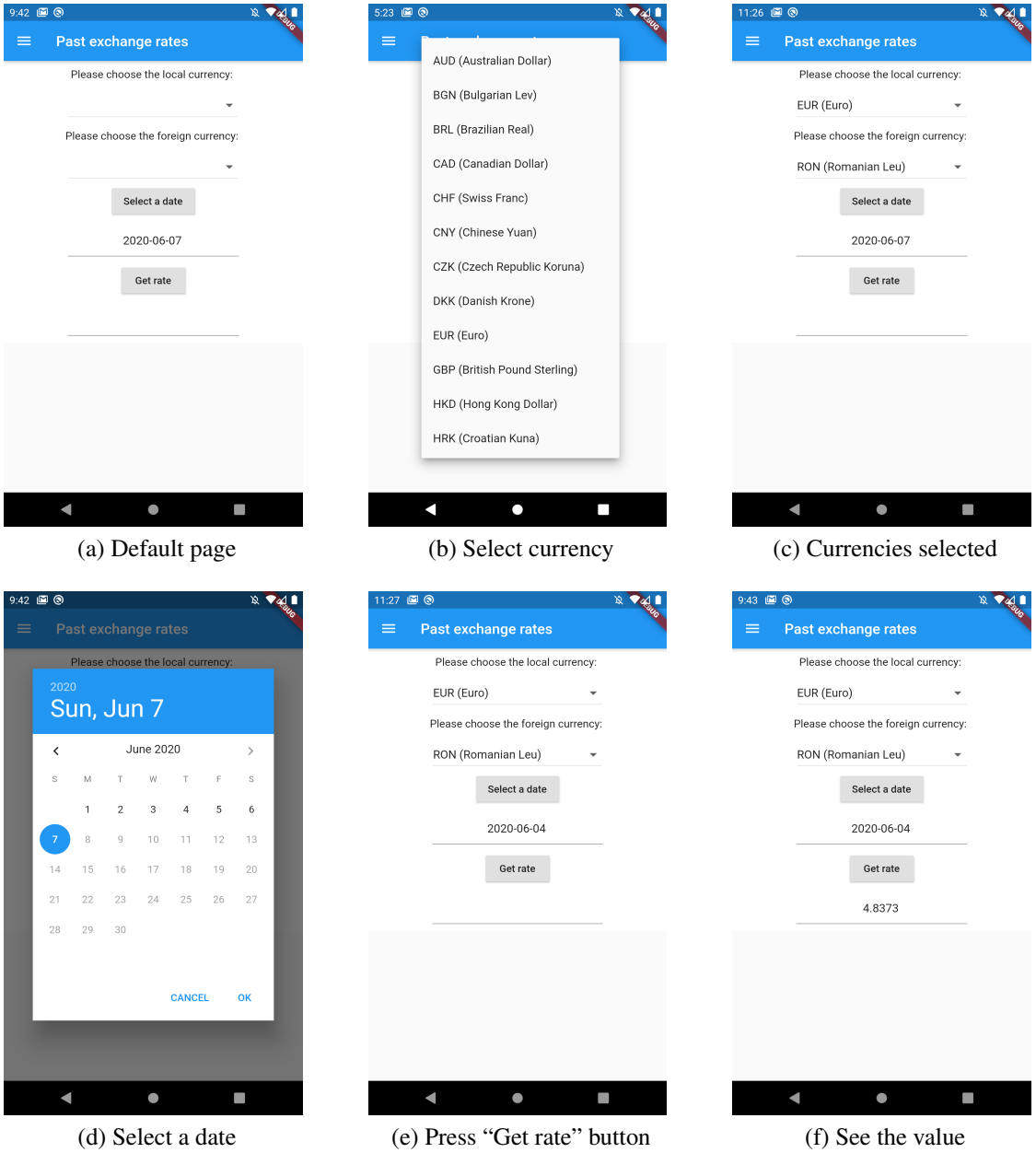
(a) Default page (b) Select currency (c) Currencies selected

(d) Type the sum (e) Press “Convert” button (f) See the converted sum

Figure 3.13: Convert Currencies Functionality

pressing on each of the drop down lists. Then the sum which the user want to convert has to be typed in the specified field. Finally, the “Convert button” needs to be pressed and the sum is converted.

Figure 3.14 illustrates the second functionality of PRICE, namely the visualisation of past values of specified exchange rates. As in the previous feature, the currencies which define have to be selected. Then, the date of interest for the exchange rate is chosen by tapping on the “Select a date” button and browsing through the calendar. As a last step, the “Get rate” button needs to tapped and the value is displayed.

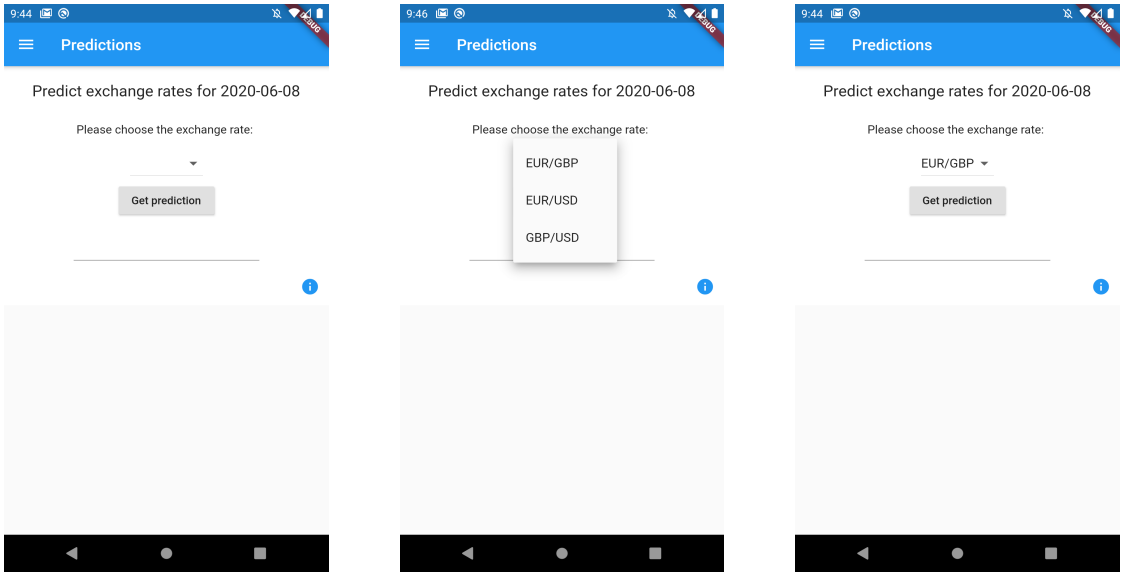


(a) Default page (b) Select currency (c) Currencies selected

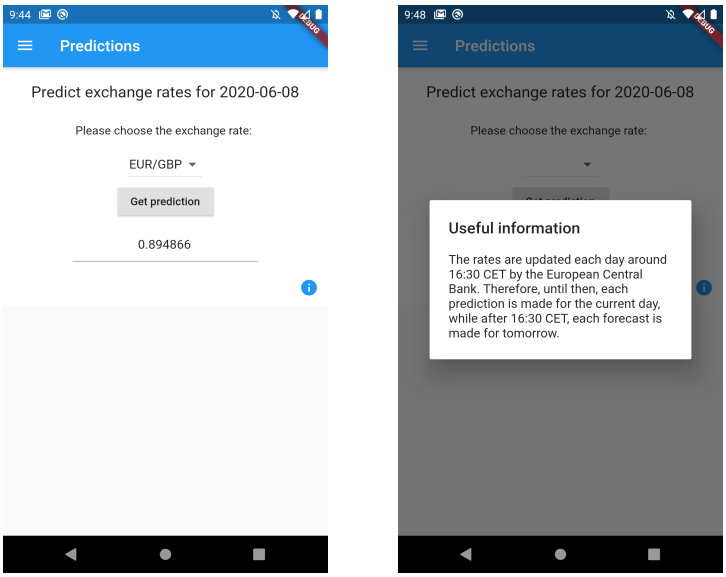
(d) Select a date (e) Press “Get rate” button (f) See the value

Figure 3.14: See Past Exchange Rates Functionality

The Predict Exchange Rates functionality is depicted in the Figure 3.15. In the case of this functionality, as mentioned before, the user can only select a predefined exchange rate from an existing set. After choosing it (from a drop down list as in the previous features), the user needs to tap the “Get prediction” button and then the value is displayed in the text field below the button. In the bottom right corner there is also an icon which, if pressed, offer information in regards with the time span of the prediction (more exactly, that the rates retrieved from the Exchange Rates API are updated each day around 16:30 CET by the European Central Bank, so, until then, each prediction is made for the current day, while after 16:30 CET, each forecast is made for the next one.)



(a) Default page (b) Choose exchange rate (c) Press “Get prediction” button

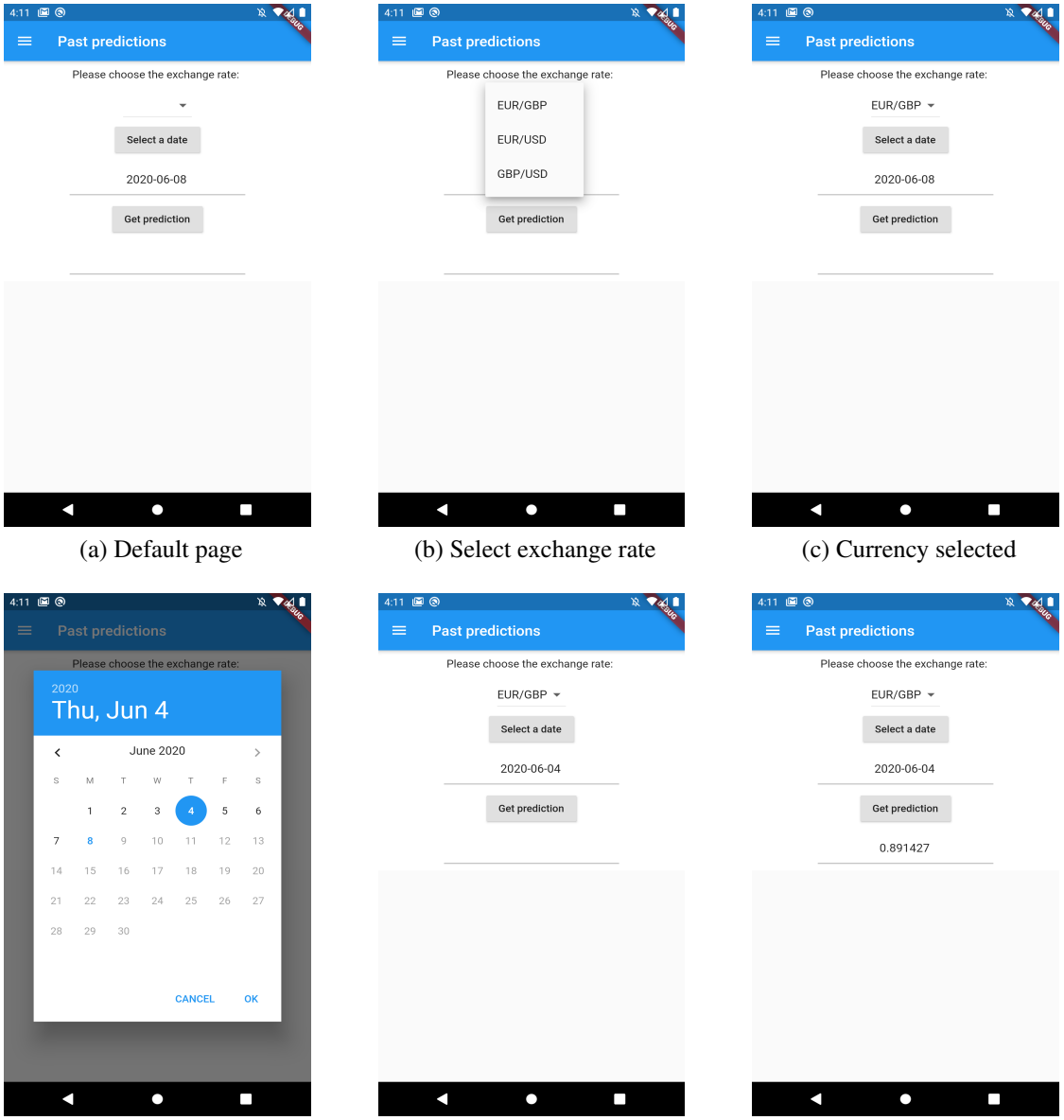


(d) See the prediction (e) Information icon pressed

Figure 3.15: Predict Exchange Rates Functionality

Figure 3.16 illustrates the last functionality of PRICE, namely See Past Predictions of Exchange Rates. As in the earlier one, the user chooses the exchange rate from the list of the available ones. Then, they choose the date for which they are interested to see the prediction (like in the case of See Past Exchange Rates). Finally, the “Get prediction” button is pressed, like in all the previous features, and the value of prediction is shown in the bottom text field.

It is worth mentioning that the screens which correspond to exceptional states (like missing or faulty input) are not presented here. This decision was made since the manual guide assumed the best case scenarios for each functionality. Nevertheless, these states are implemented and any user is invited to try them out.



(a) Default page (b) Select exchange rate (c) Currency selected

(d) Select a date (e) Press “Get prediction” button (f) See the prediction

Figure 3.16: See Past Predictions of Exchange Rates Functionality

3.2 Experimental results

The Prediction Server presented in the *Application Development* section was created in order to expose the models created for forecasting the EUR\USD, EUR\GBP and GBP\USD exchange rates. The models were chosen following comparisons between multiple architectures, combination of multiple time series as input, varying lengths of the input in the terms how many past observations were considered and different activation functions for the neurons. As mentioned earlier, the comparisons made and the final choice of the model were made in regards to predicting the EUR\GBP exchange rate, but the performance of the selected model for the other 2 exchange rate is also presented.

3.2.1 Data set and experimental configuration

The time-series considered in this study were daily observations of the EUR\GBP rate, which were retrieved from https://www.investing.com/currencies/eur-gbp-historical-data, daily observations of the EUR\USD rate, which were retrieved from https://www.investing.com/currencies/eur-usd-historical-data, daily observations of the GBP\USD rate, which were retrieved from https://www.investing.com/currencies/gbp-usd-historical-data, daily observations of gold price expressed in dollars per ounce, which were retrieved from https://www.usagold.com/reference/prices/goldhistory.php and daily observations of Brent Crude Oil Spot price expressed in dollar per barrel, which were retrieved from https://ycharts.com/indicators/brent_crude_oil_spot_price. The choice of including gold and oil prices in the study was made in order to see if a rather stable commodity (gold) or one which is varying (oil) or a combination of the 2 has any effect on predicting the exchange rate. These time-series were also refined to be in the range between [0,1] by dividing each observation by their respective maximum value. Each time-series included 2670 daily observations, spanning a period of approximately 10 years, from the 4th of January 2010 until the 27th of March 2020. These daily observations were divided for all the architectures in three categories (training, validation and test), in the ratio 60:20:20. The input length was varied to contain 10, 15 and 20 past observations (more exactly, for predicting tomorrow's exchange rate, we considered looking back at the last 10, 15 and 20 values of the exchange rate, gold price and oil price).

The architectures inspected were FFNN with a single dense hidden layer of 2000 neurons; LSTM neural network with a single hidden layer of 100 memory cells; LSTM neural network with 2 hidden layers of 100 memory cells; bidirectional LSTM neural network with a single hidden layer of 100 memory cells; CNN with a single one-dimensional convolutional layer of 100 filter maps and with a kernel size of 2 and a single one-dimensional pooling layer with the pool size of 2; and ConvLSTM with a single hidden layer of 100 filter maps and a kernel size of 2. The choice of the architectures was inspired from [Bro18].

The study and choice of the model was divided in 3 phases. In the first phase of the study, the problem was treated as univariate (predictions were made based only on the past observations of the EUR\GBP exchange rate) and all the architectures were considered. For all the architectures the activation function of the neurons was ReLU, the optimizer was Adam, the loss function was MSE (Mean Squared Error) and the input length was varied as explained earlier. Each combination of architecture and input length was trained, validated (over 500 epochs) and tested 5 times due to the stochastic nature of the ANN method and for gaining a better understanding of the stability of it. Based on the average performance of the runs, the best 3 architectures (with their respective best input length) were included in the second phase of the study.

The second phase of the study looked at the problem as being a multivariate one and, as mentioned earlier, considered the best 3 architectures (in combination with their best input

length) found in the previous phase. Therefore, for each model, the input was changed to include firstly only gold observations, secondly only oil observations and finally both. As in the previous phase, each model had ReLU as the activation function for the neurons, Adam as its optimizer, MSE as the loss function, was trained and validated over 500 epochs and was run 5 times.

In the third (and final) phase of the study, the best model out of all tried in the first two phases was selected and tested with different activation functions, namely sigmoid and tanh. The optimizer remained Adam, the loss function was still MSE and the model was again trained, validated (over 500 epochs) and tested 5 times.

3.2.2 Results and comparisons

The results of the study detailed in the previous subsection can be visualized in the figures and tables below. As it can be seen in the Table 3.1, in the phase one, the best runs of the best models achieved close results, with the model of bidirectional LSTM coupled with 15 past observations achieving the top performance.

Metric / Model[1]	MSE	MAE	NRMSE
FFNN, 10 past observations	2.5474e-05	0.00365918	4.486346%
FFNN, 15 past observations	3.3888e-05	0.00438173	5.174528%
FFNN, 20 past observations	2.6067e-05	0.00359527	4.538323%
LSTM, 1 hidden layer, 10 past observations	1.9504e-05	0.00309606	3.925599%
LSTM, 1 hidden layer, 15 past observations	**1.8768e-05**	**0.00303572**	**3.850829%**
LSTM, 1 hidden layer, 20 past observations	1.9229e-05	0.00306240	3.897839%
LSTM, 2 hidden layers, 10 past observations	1.9973e-05	0.00318270	4.287936%
LSTM, 2 hidden layers, 15 past observations	2.1215e-05	0.00332553	4.094180%
LSTM, 2 hidden layers, 20 past observations	1.9074e-05	0.00305046	3.882096%
BiLSTM, 10 past observations	1.9034e-05	0.00306866	3.878051%
BiLSTM, 15 past observations	**1.8694e-05**	**0.00303273**	**3.843212%**
BiLSTM, 20 past observations	**1.8887e-05**	**0.00303563**	**3.863069%**
CNN, 10 past observations	4.6090e-05	0.00500661	6.034617%
CNN, 15 past observations	2.6558e-05	0.00365642	4.580820%
CNN, 20 past observations	4.0828e-05	0.00464103	5.679697%
ConvLSTM, 10 past observations	1.9570e-05	0.00315895	3.932286%
ConvLSTM, 15 past observations	2.0376e-05	0.00325973	4.012408%
ConvLSTM, 20 past observations	2.0891e-05	0.00329801	4.062841%

[1] Each row in the table denotes a model which is described using a concatenation with commas of the architecture name, the number of hidden layers and the number of past observations used. If the number of hidden layers is omitted, like in the case of 'ConvLSTM, 10 past observations', it is assumed to be 1.

Table 3.1: Phase one best run

However, when the average performance over 5 runs was taken into account, the picture started to change, as depicted by the Table 3.2. No model with a bidirectional layer came

into the top 3, denoting the instability of this architecture in regards with this forecasting problem. The top spot was however occupied by the model with the second best run, namely the one with one LSTM hidden layer and 15 past observations. The second and third spot were the model with one ConvLSTM layer and 10 past observations and the model with 2 LSTM hidden layers combined with 20 past observations respectively.

Model[1] \ Metric	MSE	MAE	NRMSE
FFNN, 10 past observations	3.3048e-05	0.00441106	5.099069%
FFNN, 15 past observations	3.6677e-05	0.00464655	5.379130%
FFNN, 20 past observations	5.9119e-05	0.00552600	6.156998%
LSTM, 1 hidden layer, 10 past observations	2.9578e-05	0.00409457	4.792840%
LSTM, 1 hidden layer, 15 past observations	**2.0669e-05**	**0.00324845**	**4.032329%**
LSTM, 1 hidden layer, 20 past observations	3.7811e-05	0.00466179	5.192892%
LSTM, 2 hidden layers, 10 past observations	2.6591e-05	0.00392260	4.561696%
LSTM, 2 hidden layers, 15 past observations	3.6738e-05	0.00482068	5.269631%
LSTM, 2 hidden layers, 20 past observations	**2.1890e-05**	**0.00339765**	**4.153552%**
BiLSTM, 10 past observations	2.6781e-05	0.00390696	4.571052%
BiLSTM, 15 past observations	2.4407e-05	0.00365455	4.349091%
BiLSTM, 20 past observations	2.9648e-05	0.00404103	4.693120%
CNN, 10 past observations	6.1760e-05	0.00605346	6.947501%
CNN, 15 past observations	4.6561e-05	0.00539988	5.936248%
CNN, 20 past observations	6.3305e-05	0.00603738	6.964842%
ConvLSTM, 10 past observations	**2.1766e-05**	**0.00332457**	**4.144068%**
ConvLSTM, 15 past observations	2.8467e-05	0.00400320	4.708461%
ConvLSTM, 20 past observations	2.7368e-05	0.00386704	4.616218%

[1] Each row in the table denotes a model which is described using a concatenation with commas of the architecture name, the number of hidden layers and the number of past observations used. If the number of hidden layers is omitted, like in the case of 'ConvLSTM, 10 past observations', it is assumed to be 1.

Table 3.2: Phase one average

The main reasoning behind phase 2 was to see if the addition of external factors like the price of gold and/or the price of oil has a beneficial effect on the prediction performance. If we look at the Table 3.3 we can conclude that, with the exception of 2 models, the prediction performance diminished. Nevertheless, in the case of the model with 2 LSTM hidden layers and 20 past observations, the addition of gold observations seemed to have resulted in the best run obtained so far over the 2 phases.

If in the case of the best run there existed 2 outliers, when we analyze the average performance of the multivariate models there is a clear conclusion: the inclusion of gold prices and/or oil price in trying to predict the EUR\GBP rate is reducing the performance of our models, as shown by Table 3.4. Furthermore, if we look at the model mentioned earlier which produced the best run in the first 2 phases, we can see that its average performance is the worst of those tried in the second phase, denoting again a high instability.

Model[1] \ Metric	MSE	MAE	NRMSE
LSTM, 1 hidden layer, 15 past obs., gold	1.9962e-05	0.00314824	3.971494%
LSTM, 1 hidden layer, 15 past obs., oil	**1.8766e-05**	**0.00303181**	**3.850663%**
LSTM, 1 hidden layer, 15 past obs., both	3.0371e-05	0.00404156	4.898621%
LSTM, 2 hidden layers, 20 past obs., gold	**1.8550e-05**	**0.00300649**	**3.828414%**
LSTM, 2 hidden layers, 20 past obs., oil	2.0994e-05	0.00329573	4.072794%
LSTM, 2 hidden layers, 20 past obs., both	2.3568e-05	0.00349445	4.315293%
ConvLSTM, 10 past obs., gold	2.0060e-05	0.00315476	3.981231%
ConvLSTM, 10 past obs., oil	2.2328e-05	0.00328461	4.200244%
ConvLSTM, 10 past obs., both	2.2029e-05	0.00335438	4.171975%

[1] Each row in the table denotes a model which is described using a concatenation with commas of the architecture name, the number of hidden layers, the number of past observations used (obs. being an abbreviation for observations) and which time series were used additionally (with G denoting gold, O denoting oil and B denoting both). If the number of hidden layers is omitted, like in the case of 'ConvLSTM, 10 past observations, O', it is assumed to be 1.

Table 3.3: Phase two best run

Model[1] \ Metric	MSE	MAE	NRMSE
LSTM, 1 hidden layer, 15 past obs., gold	3.1894e-05	0.00427498	4.863918%
LSTM, 1 hidden layer, 15 past obs., oil	2.3309e-05	0.00354584	4.283770%
LSTM, 1 hidden layer, 15 past obs., both	4.0220e-05	0.00496762	5.589109%
LSTM, 2 hidden layers, 20 past obs., gold	**4.6178e-05**	**0.00543907**	**5.797110%**
LSTM, 2 hidden layers, 20 past obs., oil	2.8835e-05	0.00416965	4.746307%
LSTM, 2 hidden layers, 20 past obs., both	3.6053e-05	0.00461307	5.254791%
ConvLSTM, 10 past obs., gold	2.3217e-05	0.00349663	4.272158%
ConvLSTM, 10 past obs., oil	2.4983e-05	0.00362119	4.438720%
ConvLSTM, 10 past obs., both	3.0499e-05	0.00419381	4.868320%

[1] Each row in the table denotes a model which is described using a concatenation with commas of the architecture name, the number of hidden layers, the number of past observations used (obs. being an abbreviation for observations) and which time series were used additionally (with G denoting gold, O denoting oil and B denoting both). If the number of hidden layers is omitted, like in the case of 'ConvLSTM, 10 past observations, O', it is assumed to be 1.

Table 3.4: Phase two average

The phase 3 of the study was concerned with how much activation functions can influence the best model obtained so far. The choice of the activation functions was inspired from literature [NIGM18]. As revealed by the previous stages, the top performer model (from an average performance standpoint) was the one that combined one LSTM hidden layer with 15 past observations and had as an activation function ReLU. In Table 3.5 and Table 3.6 there can be seen the results when the activation function is replaced by sigmoid and tanh.

As long as the best run is brought into discussion, the switch to a tanh activation function seemed to have produced a better best run when compared with the original model, albeit

Model[1] \ Metric	MSE	MAE	NRMSE
LSTM, 1 hidden layer, 15 past obs., R	1.8768e-05	0.00303572	3.850829%
LSTM, 1 hidden layer, 15 past obs., S	2.2598e-05	0.00332108	4.225528%
LSTM, 1 hidden layer, 15 past obs., T	1.8641e-05	0.00301489	3.837802%

[1] Each row in the table denotes a model which is described using a concatenation with commas of the architecture name, the number of hidden layers, the number of past observations used and which activation function was used (with R denoting ReLU, S denoting sigmoid and T denoting tanh). For example, 'LSTM, 1 hidden layer, 15 past obs., R' denotes the model which uses the LSTM architecture with 2 hidden layers, 15 past observations and tanh as an activation function.

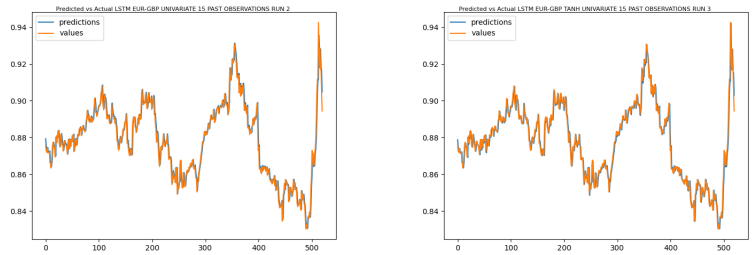
Table 3.5: Phase three best run

only slightly. Moreover, if we look at the average performance, while the switch to sigmoid appeared to have made no improvement whatsoever, the change to tanh looked to have produced the first real contender for the top model discovered in the first 2 phases. Thus, even though the original model which makes use of the ReLU activation function still reigns supreme in terms of average performance, we can conclude that there is not a significant difference if we choose to deploy this model with a tanh activation function or ReLU activation function (which is also reflected in Figures 3.17, 3.18).

Model[1] \ Metric	MSE	MAE	NRMSE
LSTM, 1 hidden layer, 15 past obs., R	2.0669e-05	0.00324845	4.032329%
LSTM, 1 hidden layer, 15 past obs., S	3.1488e-05	0.00422056	4.930744%
LSTM, 1 hidden layer, 15 past obs., T	2.0685e-05	0.00326033	4.032962%

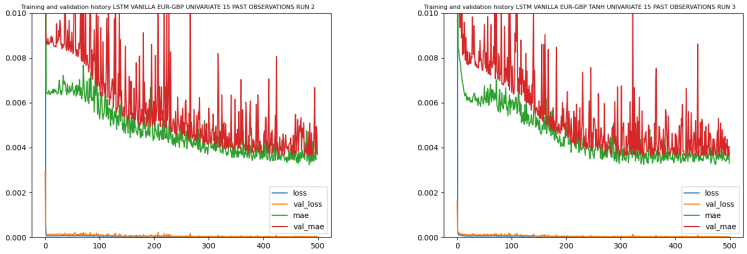
[1] Each row in the table denotes a model which is described using a concatenation with commas of the architecture name, the number of hidden layers, the number of past observations used and which activation function was used (with R denoting ReLU, S denoting sigmoid and T denoting tanh). For example, 'LSTM, 1 hidden layer, 15 past obs., R' denotes the model which uses the LSTM architecture with 2 hidden layers, 15 past observations and tanh as an activation function.

Table 3.6: Phase three average



(a) ReLU (b) Tanh

Figure 3.17: Predictions against actual values on test data for the best run of the chosen model

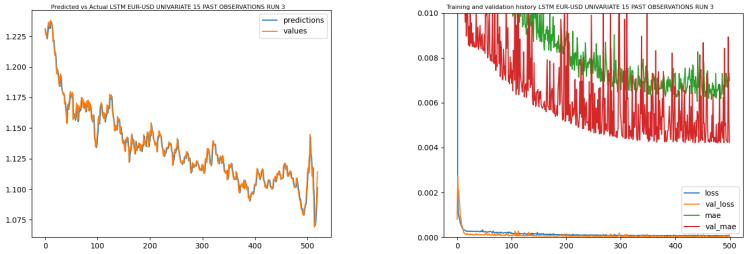


(a) ReLU (b) Tanh

Figure 3.18: Training and validation history for best run of the chosen model

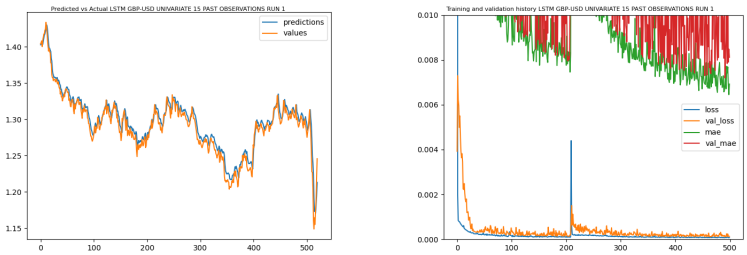
With the model being now chosen, its performance on the other 2 exchange rates can be assessed. The results can be visualized in Tables 3.7 and 3.8 as well as in Figures 3.19 and 3.20. The first conclusion is that this model is a good fit for trying to predict the EUR\USD exchange rate, achieving comparable results in performance like when it was used for predicting the EUR\GBP exchange rate.

However, not the same can be said about the GBP\USD exchange rate. The model seems to have a rather poor performance on this exchange rate in comparison with the other 2, raising again the question if maybe a different activation function may improve the result.



(a) Prediction VS Actual Values (b) Training and Validation History

Figure 3.19: Best run of the chosen model for the EUR\USD exchange rate



(a) Prediction VS Actual Values (b) Training and Validation History

Figure 3.20: Best run of the chosen model for the GBP\USD exchange rate

Metric / Exchange rate	MSE	MAE	NRMSE
EUR\USD	2.1166e-05	0.00339492	4.089426%
GDP\USD	0.00010767	0.00803783	9.223286%

Table 3.7: Model's best run on the other 2 exchange rates

Metric / Exchange rate	MSE	MAE	NRMSE
EUR\USD	2.7485e-05	0.00397120	4.595929%
GDP\USD	0.00013447	0.00944993	10.23021%

Table 3.8: Model's average performance on the other 2 exchange rates

Tables 3.9, 3.10 compares the model's results for predicting the GBP\USD rate with different activation functions. If for the EUR\GBP rate the ReLU activation function performed slightly better than its counterpart, the picture is very different in this case. Tanh clearly outperforms its rival, both for the best run and in average performance.

Metric / Function	MSE	MAE	NRMSE
ReLU	0.00010767	0.00803783	9.223286%
Tanh	5.7895e-05	0.00545111	6.763423%

Table 3.9: Model's best run with different activation functions for predicting the GBP\USD exchange rate

Metric / Function	MSE	MAE	NRMSE
ReLU	0.00013447	0.00944993	10.23021%
Tanh	7.0178e-05	0.00622916	7.374714%

Table 3.10: Model's average performance with different activation functions for predicting the GBP\USD exchange rate

With the models now chosen and since each one in this study was trained, validated and tested 5 times as mentioned before, it is also worth mentioning the 95% confidence intervals for the metrics average scores. These values are presented in the Table 3.11.

Metric / Exchange rate	MSE	MAE	NRMSE
EUR\GBP	2.067e-05 ± 1.433e-06	0.0032 ± 0.0003	4.032% ± 0.262%
EUR\USD	2.749e-05 ± 5.029e-06	0.0039 ± 0.0009	4.596% ± 0.755%
GDP\USD	7.018e-05 ± 1.062e-05	0.0062 ± 0.0012	7.375% ± 1.011%

Table 3.11: The metrics' confidence intervals of the exchange rates' models

Compared to the results obtained by Sermpinis et al. in their paper [SSD14], which were presented in Chapter 2, one can conclude that the model presented in this study achieves a slightly better performance. This is rather clear in the case of the **EUR\GBP** exchange rate, since the **95% confidence intervals for the MAE and RMSE metrics** were **0.0032 ± 0.0003** and **0.0045 ± 0.0003** for **the model in this paper**, while **the best model found by the aforementioned authors** achieved **MAE scores of 0.0036, 0.0049, 0.0035 and RMSE scores of 0.0071, 0.0081, 0.0068 for the F1, F2 and F3 periods** (which are also described in the Chapter 2). However, when the **EUR\USD** exchange rate is considered, the results seem more balanced, with **the 95% confidence intervals for the MAE and RMSE metrics** being **0.0039 ± 0.0009** and **0.0051 ± 0.0009** for **the model in this paper**, while **the best model found by the aforementioned authors** achieved **MAE scores of 0.0039, 0.0048, 0.0037 and RMSE scores of 0.0053, 0.0058, 0.0051 for the F1, F2 and F3 periods**. Nevertheless, when **the best run of the model** described in this study for the EUR\USD exchange rate is brought into discussion, the **MAE and RMSE metrics of 0.0033 and 0.0046** seem to tilt again the balance in favour of the prototype detailed above.

3.2.3 Summation

The final conclusions that can be drawn from this study are the following: the best model, from those trialed, for forecasting the EUR\GBP exchange rate is a rather simple one from a complexity point of view, namely the one which has one LSTM hidden layer of 100 memory cells, has ReLU as the activation function for its neurons, has MSE as its loss function, has Adam as its optimizer, is trained and validated for 500 epochs and uses 15 past observations of the exchange rate as input (nicknamed 'LSTM,1L,15PS,R'); this model also performs rather well for predicting the EUR\USD exchange rate, albeit poorer when compared with previous rate; the model achieves mediocre results when trying to predict the GBP\USD exchange rate and a switch to the tanh activation function improves its performance in this case (which creates the following altered nickname 'LSTM,1L,15PS,T'). Consequently, the models chosen for predicting the aforementioned rates inside the PRICE mobile application are **'LSTM,1L,15PS,R' for the EUR\GBP and EUR\USD exchange rate** and **'LSTM,1L,15PS,T' for the GBP\USD exchange rate**.

Conclusions and Future Work

The present thesis investigated the problem of forecasting exchange rates by making use of the method of artificial neural networks. The approach aimed to obtain reliable predictors for the EUR\GBP, EUR\USD and GBP\USD exchange rates by enganging in a study which considered multiple architectures (FFNN, CNN, LSTM, BiLSTM and ConvLSTM), multiple time series (gold and oil observations), multiple activation functions and varying lengths of past observations as input. A more detailed analysis of the configuration of the models was presented in the second section of the Chapter 3. The best runs of the most performant models achieved NRMSE values of 3.85%, 4.08% and 6.76% for the EUR\GBP, EUR\USD and GBP\USD exchange rates, while the 95% confidence intervals of the same metric were 4.032% ± 0.262%, 4.596% ± 0.755% and 7.375% ± 1.011% respectively. The results of other metrics and comparisons of these results with existing ones from literature were also described in the second section of the Chapter 3. The best models were further integrated in a mobile application, named PRICE, which also offer the functionality of converting money between different currencies at the current day's rates, the possibility to see past values of certain exchange rates and the option to analyze past predictions made by the exposed models, besides the already implied feature of forecasting the currency exchanges mentioned earlier one day into the future. The application development as well as an user manual was presented in the first section of the Chapter 3.

Further directions which the existing study could have explored contain the following: the inclusion of more neural networks architectures, such as Psi Sigma Neural Networks; analysis in regards to the addition of more external indicators, like other exchange rates which include the currencies in the original one; the variation of the input length, in terms of past observations considered, in a more granular way; the examination of more optimizers for the models considered; or the diversification of the data set by means of including more observations, less observations or different test data.

Areas in which the mobile application could be improved or expanded include the following: the usage of data mining to continually retrain and update the proposed model; the inclusion of more exchange rate for which the prediction functionality is offered; in the case of the Prediction Server, route security such that they can be accessed only through the mobile application; the caching of the data used from Exchange Rates API in order to eliminate the reliance on their service; or offering the possibility to predict exchange rates further into the future, for example one week or one month ahead.

Bibliography

[Agg19] Raghav Aggarwal. Bi-LSTM, 2019. https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0 [Online; accessed May-2020].

[Boy02] Nataliya Boyko. The monetary model of exchange rate determination: The case of ukraine. Master's thesis, Economic Education and Research Consortium The National University of "Kiev-Mohyla Academy", 2002.

[Bro18] J. Brownlee. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[CC18] Pierre Luc Carrier and Kyunghyun Cho. LSTM Networks for Sentiment Analysis, 2018. http://www.deeplearning.net/tutorial/lstm.html [Online; accessed March-2020].

[Cha19] Himadri Sankar Chatterjee. A Basic Introduction to Convolutional Neural Network, 2019. https://medium.com/@himadrisankarchatterjee/a-basic-introduction-to-convolutional-neural-network-8e39019b27c4 [Online; accessed March-2020].

[Eck07] Wayne W. Eckerson. PREDICTIVE ANALYTICS. Extending the Value of Your Data Warehousing Investment. Technical report, TDWI, 2007.

[Edw19] John Edwards. What is predictive analytics? Transforming data into future insights, 2019. https://www.cio.com/article/3273114/what-is-predictive-analytics-transforming-data-into-future-insights.html [Online; accessed March-2020].

[FD11] Codruța Maria Făt and Eva Dezsi. Exchange-rates forecasting: Exponential smoothing techniques and arima models. *Annals of Faculty of Economics*, 1, 2011.

[Gal16] Svitlana Galeshchuk. Neural networks performance in exchange rate prediction. *Neurocomput.*, 172(C):446–452, January 2016.

[Gér19] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[GR92] Javier Gardeazabal and Marta Regúlez. *The Monetary Model of Exchange Rates and Cointegration Estimation, Testing and Prediction*. Springer-Verlag, USA, 1992.

[Ima] Imanuel. What is predictive analytics? Transforming data into future insights. https://www.predictiveanalyticstoday.com/what-is-predictive-analytics/ [Online; accessed March-2020].

[KB10] Mehdi Khashei and Mehdi Bijari. An artificial neural network (p,d,q) model for timeseries forecasting. *Expert Systems with Applications*, 37(1):479 – 489, 2010.

[ker] https://keras.io/ [Online; accessed May-2020].

[Las20] William D. Lastrapes. Autoregressive models, 2020. https://www.encyclopedia.com/social-sciences/applied-and-social-sciences-magazines/autoregressive-models [Online; accessed April-2020].

[MR83] Richard A. Meese and Kenneth Rogoff. Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, 14(1):3 – 24, 1983.

[Nau] Robert Nau. Notes on nonseasonal ARIMA models. http://people.duke.edu/~rnau/Notes_on_nonseasonal_ARIMA_models--Robert_Nau.pdf [Online; accessed April-2020].

[NIGM18] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv e-prints*, page arXiv:1811.03378, November 2018.

[Pei19] Marco Peixeiro. The Complete Guide to Time Series Analysis and Forecasting, 2019. https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775 [Online; accessed March-2020].

[PMB12] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1211.5063, November 2012.

[Pra19] Selva Prabhakaran. ARIMA Model – Complete Guide to Time Series Forecasting in Python, 2019. https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/ [Online; accessed April-2020].

[Put18] Ocktavia Nurima Putri. Titanic prediction with artificial neural network in r., 2018. https://laptrinhx.com/titanic-prediction-with-artificial-neural-network-in-r-3087367370/ [Online; accessed April-2020].

[pyt] https://docs.python.org/3/faq/general.html [Online; accessed May-2020].

[RB15] SK Reddy and AS Babu. Exchange rate forecasting using arima, neural network and fuzzy neuron. *Journal of Stock & Forex Trading*, 04, 01 2015.

[RKM14] Mehreen Rehman, Gul Muhammad Khan, and S.A. Mahmud. Foreign currency exchange rates prediction using cgp and recurrent neural network. *IERI Procedia*, 10:239–244, 12 2014.

[Spe14] Ewan Spence. The Mobile Browser Is Dead, Long Live The App, 2014. https://www.forbes.com/sites/ewanspence/2014/04/02/the-mobile-browser-is-dead-long-live-the-app/#70b96d70614d [Online; accessed April-2020].

[SSD14] Georgios Sermpinis, Charalampos Stasinakis, and Christian Dunis. Stochastic and genetic neural network combinations in trading and hybrid time-varying leverage effects. *Journal of International Financial Markets, Institutions and Money*, 30, 05 2014.

[Sto16] Marijn Stollenga. *Advances in Humanoid Control and Perception*. PhD thesis, 05 2016.

[Sus] Rauli Susmel. FORECASTING EXCHANGE RATES. https://www.bauer.uh.edu/rsusmel/7386/ln5.pdf [Online; accessed April-2020].

[TG15] Philip E. Tetlock and Dan Gardner. *Superforecasting: The Art and Science of Prediction*. Crown Publishing Group, USA, 2015.

[Ven19] Mahendran Venkatachalam. Recurrent Neural Networks, 2019. https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce [Online; accessed June-2020].

[VL19] Fjodor Van Veen and Stefan Leijnen. The Neural Network Zoo, 2019. https://www.asimovinstitute.org/neural-network-zoo/ [Online; accessed March-2020].

[Wan19] Chi-Feng Wang. The Vanishing Gradient Problem, 2019. https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484 [Online; accessed June-2020].

[WP08] Florentin Woergoetter and Bernd Porr. Reinforcement learning, 2008. http://www.scholarpedia.org/article/Reinforcement_learning [Online; accessed March-2020].

[YDA+19] Muhammad Yasir, Mehr Yahya Durrani, Sitara Afzal, Muazzam Maqsood, Farhan Aadil, Irfan Mehmood, and Seungmin Rho. An intelligent event-sentiment-based daily foreign exchange rate forecasting system. *Applied Sciences*, 9(15), 2019.

[YF17] Cenk Ufuk Yıldıran and Abdurrahman Fettahoğlu. Forecasting usdtry rate by arima method. *Cogent Economics & Finance*, 5(1), 2017.

[Zha03] Peter G. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50, 2003.

[ZL07] Shidong Zhang and Thomas Lowinger. The monetary exchange rate model: Long-run, short-run, and forecasting performance. *Journal of Economic Integration*, 22:397–406, 06 2007.