Predicting price trends of digital products using various forecasting techniques: On the example of the Steam Community Market

Matthias Roth

Dissertation written under the supervision of professor Nicolò Bertani

Dissertation submitted in partial fulfilment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, 04.01.2023.

Abstract

Title: Predicting price trends of digital products using various forecasting techniques: On the example of the Steam Community Market

Author: Matthias Roth

The gaming industry has experienced steady growth over the years, contributing to its increasing commercialisation. One factor adding to this trend is the growing popularity of online marketplaces for in-game items, some of which are traded using real-world currencies and considered by a growing amount of young people as some new asset class. This study addresses the question of price predictability for these items, as the efficient market hypothesis posits that it is impossible to consistently predict future prices based on past prices. While this topic has been extensively discussed in the literature for classical financial time series forecasting, it has not yet been explored in the context of in-game item marketplaces.

This study used data from the Steam Community Market to investigate the predictability of in-game item prices in the context of online marketplaces. Multiple linear and non-linear forecasting models are applied to the data. This study shows that the price is predictable to some degree for many items, although the improvement is small compared to the naïve benchmark. Specially, linear models showed auspicious results for stationary data and short-term predictions, while non-linear models rarely delivered a strong performance. These findings suggest that forecasting digital items may be as challenging as forecasting traditional assets.

Keywords: Steam Community Market, digital products, machine learning, efficient market hypothesis

Abstract Portuguese

Title: Predicting price trends of digital products using various forecasting techniques: On the example of the Steam Community Market

Author: Matthias Roth

A indústria do jogo tem verificado um crescimento constante ao longo dos anos, contribuindo para a sua crescente comercialização. Um fator que contribui para esta tendência é a crescente popularidade dos mercados online para artigos dentro do jogo, alguns dos quais são comercializados utilizando moedas do mundo real e considerados por uma quantidade crescente de jovens como uma espécie de nova classe de ativos. Este estudo aborda a questão da previsibilidade de preços para estes itens, uma vez que a hipótese de mercado eficiente postula que é impossível prever de forma consistente os preços futuros com base nos preços do passado. Embora este tópico tenha sido amplamente discutido na literatura para a previsão clássica de séries cronológicas financeiras, ainda não foi explorado no contexto dos mercados de itens dentro do jogo. Este estudo utilizou dados Steam Community Market para investigar a previsibilidade dos preços dos itens no contexto dos mercados online. Modelos múltiplos de previsão linear e não-linear são aplicados aos dados. Este estudo mostra que o preço é previsível até certo ponto para muitos itens, embora a melhoria seja pequena em comparação com a referência naïve. Especialmente os modelos lineares mostraram resultados auspiciosos para dados estacionários e previsões a curto prazo, enquanto os modelos não lineares raramente proporcionaram um forte desempenho. Estes resultados sugerem que a previsão de itens digitais pode ser tão desafiante como a previsão de bens tradicionais.

Keywords: Steam Community Market, digital products, machine learning, efficient market hypothesis

Contents

List of figures

List of tables

List of abbreviation

Abbreviation	Definition
ADF	Augmented Dickey-Fuller test
AIC	Akaike Information Criterion
ANN	Artificial Neural Networks
API	Application Programming Interface
AR	AutoRegressive
ARCH	AutoRegressive Conditional Heteroscedastic
ARIMA	AutoRegressive Integrated Moving Average
CNN	Convolutional Neural Network
CS:GO	Counter-Strike: Global Offensive
DL	Deep Learning
DMLP	Deep Multilayer Perceptron
EMH	Efficient Market Hypothesis
GMRAE	Geometric Mean Relative Absolute Error
GRU	Gated Recurrent Unit
HTML	Hypertext Markup Language
LSTM	Long Short-Term Memory
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MIMO	Multi-Input Multi-Output
MRAE	Mean Relative Absolute Error
NFT	Non-Fungible Token
NN	Neural Network
ReLU	Rectified Linear Units
RNN	Recurrent Neural Network
SES	Simple Exponential Smoothing
sMAPE	Symmetric Mean Absolute Percentage Error

Acknowledgement

I would like to express my sincere gratitude to my family for their unwavering support and encouragement throughout my academic journey. I am especially thankful to my parents, for their love and sacrifice.

I would also like to thank my friends for being a constant source of motivation and for being there for me through thick and thin.

I am deeply indebted to my advisor, Nicolò Bertani, for his guidance, patience, and invaluable insights. His expertise and mentorship have been crucial in shaping my research and helping me grow as a scholar.

Finally, I would like to thank all of the individuals who have contributed to this work in any way, whether through helpful discussions or simply offering a listening ear. Your support has meant the world to me.

1. Introduction

The efficient market hypothesis (EMH) (Malkiel & Fama, 1970) proposes that prices of financial assets reflect all available information, making it impossible to consistently earn abnormal returns by applying linear and non-linear forecasting techniques. While the EMH has been extensively studied in the context of traditional financial markets, its relevance to digital marketplaces remains unclear. Marketplaces that use a bid/ask mechanism, similar to e.g., a stock exchange, exist for physical products like sneakers (StockX, 2022) but are especially popular for digital products like Non-Fungible Token (NFT) (OpenSea, 2022) or items that are used in video games, so-called in-game items. In particular, it has yet to be studied whether future prices in these marketplaces are predictable, thus violating the EMH.

In this study, I focus on whether the prices of these in-game items listed on marketplaces are predictable and whether the EMH can be applied in this context. Additionally, I consider several state-of-the-art forecasting models and explore whether any of them is particularly well suited for predictions in this context.

Specifically, first I collect data from the Steam Community Market (Steam, 2022c) and focus on their most popular game "Counter-Strike: Global Offensive" (Steam, 2022a). The dataset contains the daily average sales price and volume per item since they got listed on the marketplace. In total, I obtained the historical price history of 17319 items. As the gaming market continuously grows (Newzoo, 2022), contributing to its increasing commercialisation (Brame, 2021), the popularity of online marketplaces for in-game items is expanding likewise. Some of these are traded using real-world currencies and are considered by a growing amount of young people as some new asset class (Cleghorn & Griffiths, 2015; Hamari et al., 2017), for which gaining insights could potentially benefit players, traders, or investors financially.

Then I analyse these data based on univariate time series forecasting using the price variable. The forecasting methods are, on the one hand, linear models like the simple exponential smoothing (SES; for the details on this model refer to section 2.3.2) and AutoRegressive Integrated Moving Average (ARIMA; for the details on this model refer to section 2.3.3) model and, on the other hand, non-linear models from the recurrent neural network class. To measure the performance of each model, a naïve model is used as a benchmark, as it would, in theory, deliver the best results if the time series followed the EMH. Additionally, the two error metrics Symmetric Mean Absolute Percentage Error (sMAPE) and Geometric Mean Relative Absolute Error (GMRAE), deliver further insights into the accuracy of the forecasts.

The study's results show that the future price of in-game items is predictable to some degree for many items, although the improvement in prediction accuracy is small compared to the naïve benchmark. Yet, the naïve model still produced better forecasting results in approximately 40% of the time series for linear models and approximately 60% for non-linear models when compared directly to each of them, as presented in Figure 1 (orange bars). Additionally, when all models were compared against each other at once, the naïve model delivered the best results most frequently (approximately 26%; blue bars). Nevertheless, this also shows that approximately 75% of the studied time series are predictable if the right model is used, which is proving to be a challenge.

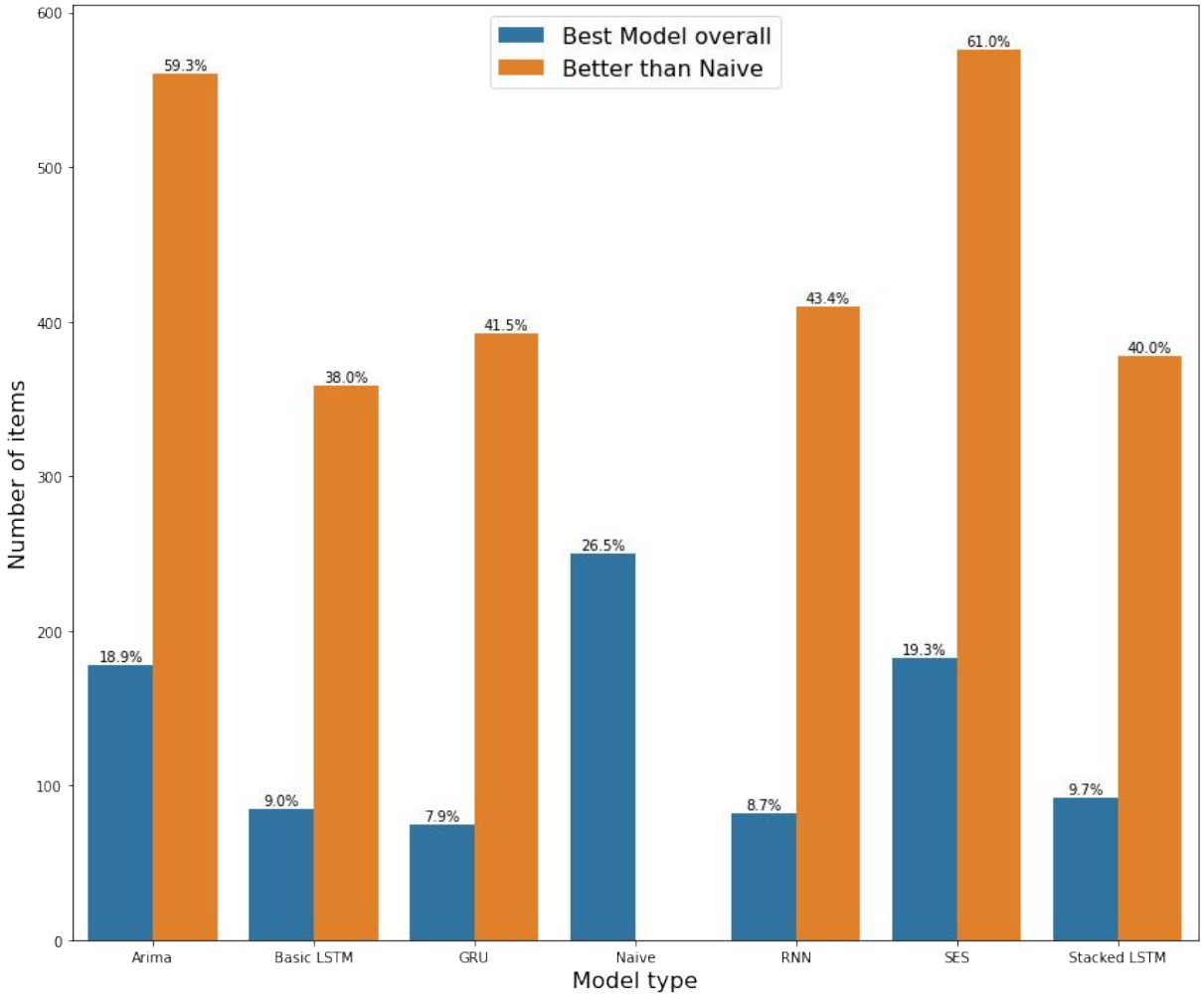


Figure 1: Number of times a model performs as the best. The plot shows the absolute number on the y-axis and the relative value on top of the bars.

However, simple methods were particularly effective in stationary data and predicting short-term horizons. Non-linear models performed worse than linear models in this study and only outperformed the naïve benchmark in a few evaluation points, with the basic RNN as the consistently best model. The most positive insights regarding the performance of non-linear

models are that their median sMAPE for stationary data is mostly better than the benchmark and their relative performances improve drastically for longer prediction horizons, as shown exemplarily for the median GMRAE in Figure 2.

Finally, it can be concluded that forecasting the time series of in-game items is a very challenging and complex task, similar to other financial time series forecasting fields. However, this study delivers some support for the critics of the efficient market hypothesis.

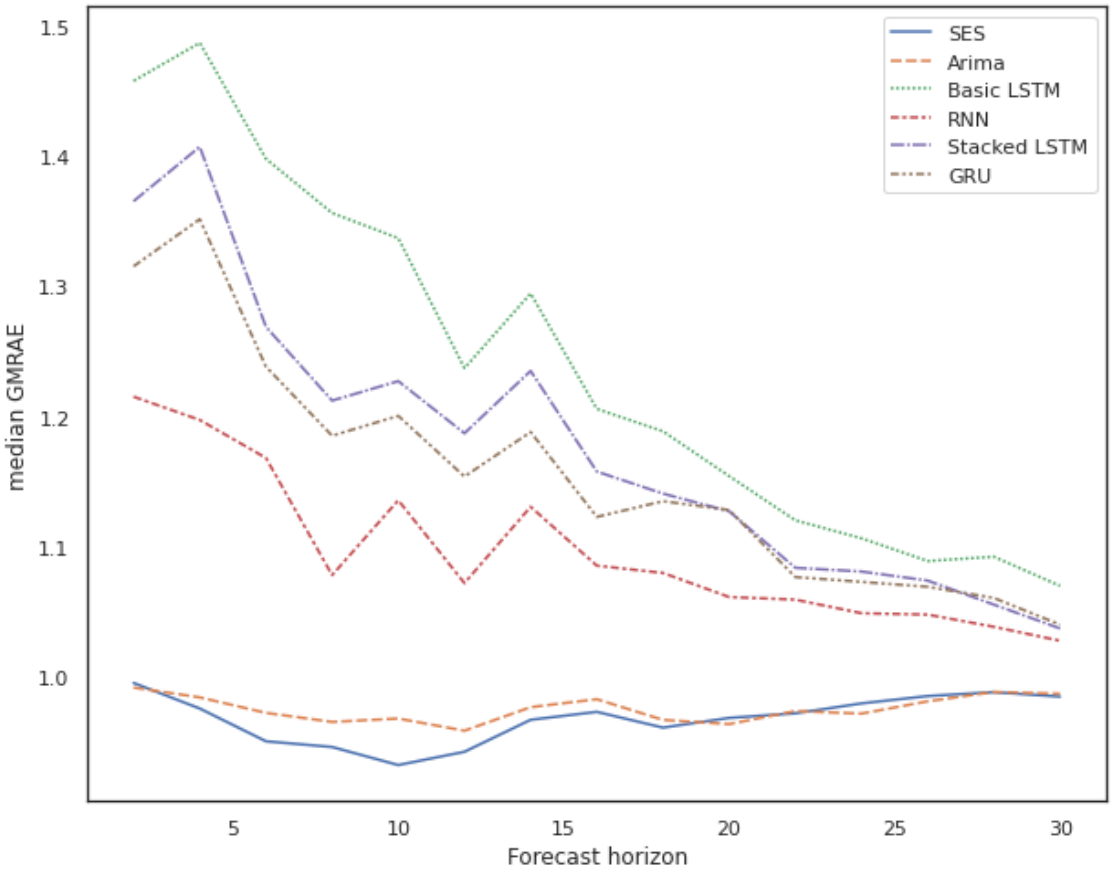


Figure 2: Median GMRAE metric over the time horizon per model

The following chapter provides background information and context for the present study. Specifically, it will detail the gaming market and relevant marketplaces, such as the Steam Community Market and introduce Counter-Strike: Global Offensive. Additionally, a review of related research on general time series forecasting and financial time series forecasting will be presented, followed by a review of the prediction models, accuracy evaluation measurements, and other methods used in the analysis. The follow-up chapter will outline the dataset, model settings, and evaluation procedures utilised in this study. In the next chapter, the study results will be presented in terms of a general evaluation, an evaluation based on stationarity, and an

analysis based on different time horizons. Finally, the last chapters will provide a conclusion followed by a discussion of the study's limitations and potential directions for future research.

2. Background

This chapter starts with a background section that provides context for the study. The first part covers the gaming industry, digital marketplaces, and the Steam marketplace, with a brief overview of Counter-Strike: Global Offensive and the rationale for focusing on this specific game. It also includes statistics on player count and daily trading volume. The second part briefly introduces time series forecasting and highlights major developments in this field. The focus then shifts to financial time series forecasting, introducing the efficient market hypothesis and discussing relevant research and developments in the field. The following part explains the basic theory behind the linear and non-linear models and the stationarity testing method used in this study. The final section is focused on accuracy evaluation, where different forecasting strategies are assessed, and two error metrics are presented.

2.1 Context

The gaming industry has experienced significant growth and popularity in recent years, with 3.2 billion video game players worldwide, and the global market is expected to reach $196.8 billion in revenue in 2022 (Newzoo, 2022).

Marketplaces like Steam have gained considerable popularity and become major gaming industry revenue sources. Steam is an online platform developed by Valve Corporation that allows users to purchase and download games and participate in a social network for gamers. According to Steam's data, as of December 2022, the platform had a daily peak concurrent player count of around 30 million players (Steam, 2022b). In addition to selling games, Steam also allows users to sell in-game items through its community marketplace.

The Steam Community Market is an online platform that allows users of the Steam gaming platform to buy and sell virtual items, such as in-game items, trading cards, and emoticons. Players have obtained these items by playing games on Steam or purchasing them from other users. The market operates on a listing and bidding system, like other online marketplaces or traditional financial systems like stock markets, in which users can list items for sale, and buyers can place bids on items they wish to purchase (Steam Support, 2022).

Items listed on the market are typically priced in a real-world currency, such as Euros, and transactions are facilitated through the Steam Wallet. However, money on the Steam Wallet can only be spent on the Steam platform and cannot exit that system (Steam Support, 2022).

All in all, the Steam Community Market is a popular destination for gamers looking to buy or sell virtual items, as it provides a convenient and secure way to transact. It also serves as a source of revenue for some gamers, who may sell items they have obtained through playing games to fund their gaming habits or make a profit.

Out of the over 50.000 games listed on Steam (PCGamesN, 2022), Counter-Strike: Global Offensive (CS:GO henceforth) is currently the most popular game on the platform, with peak concurrent players of over 1 million in December 2022 (Steam Charts, 2022). This demonstrates the game's enduring popularity, which has been a mainstay of the competitive gaming scene since its release in 2012.

CS:GO, owned by Valve Corporation, is a game where players are divided into two teams of five and engage in objective-based gameplay. The objective for one team is to eliminate the opposing team or plant a bomb, while the objective for the other team is to disarm a bomb set by the first team or eliminate them before the bomb is planted. In addition to its core gameplay, CS:GO also features a wide range of in-game items, including skins for weapons and characters, which players can purchase with real money from the Steam Community Market or third-party websites. Other ways to get skins are by unboxing them through the game's "Case" system, which requires a bought key to unlock them, or by trading with other players (Steam, 2022a). Skins are used to customise the appearance of a player's weapons or other equipment displayed in the game but do not impact gameplay and are purely cosmetic.

These skins have become an integral part of the game and are indispensable. One factor contributing to the popularity of skins in CS:GO is their integration with fundamental aspects of the game. The ability to view the painted guns and knives of your in-game model and other players during gameplay adds a visually appealing aspect to the game. The game also features a command for inspection animation that allows players to show off their skins as a form of pride or taunt. Additionally, obtaining and using an opponent's decorated rifle as a visual trophy adds an extra layer of satisfaction to gameplay (PCGamer, 2015).

Furthermore, the presentation of the gun owner's name in the user interface further enhances the sense of ownership and personalisation of skins (PCGamer, 2015). These elements have helped to make skins an integral part of many players' CS:GO experience, and some skins are

highly sought after. Therefore, some of them are worth a significant amount of money, depending on their rarity and demand, with prices that can be thousands of Euros. This leads to some players enjoying collecting and trading them for a hobby or as an asset.

In summary, the Steam Community Market is a widely utilised player-to-player (C2C) market for buying and selling in-game items for various games. As the top game on the market, CS:GO also serves as a prime example of activities and dynamics present within this digital market. The market's popularity, combined with the abundance of items available for CS:GO, makes it an ideal platform for studying the price development of digital products in my study.

As depicted in Figure 3, there appears to be a correlation between the observed 30-day average trading volume in millions of Euros in this study's dataset (represented on the left y-axis in red) and the average daily player count per month in millions (represented on the right y-axis in blue) over the illustrated period. The introduction of skins in August 2013 (Counter-Strike Blog, 2013), exactly one year after the game's release, coincides with a significant increase in the game's popularity, as indicated by the sharp rise in both variables. The graph also shows a second surge in popularity beginning in 2020, coinciding with the onset of the COVID-19 pandemic. These findings suggest that the introduction of skins and the COVID-19 pandemic may have contributed to the significant growth in the game's popularity.

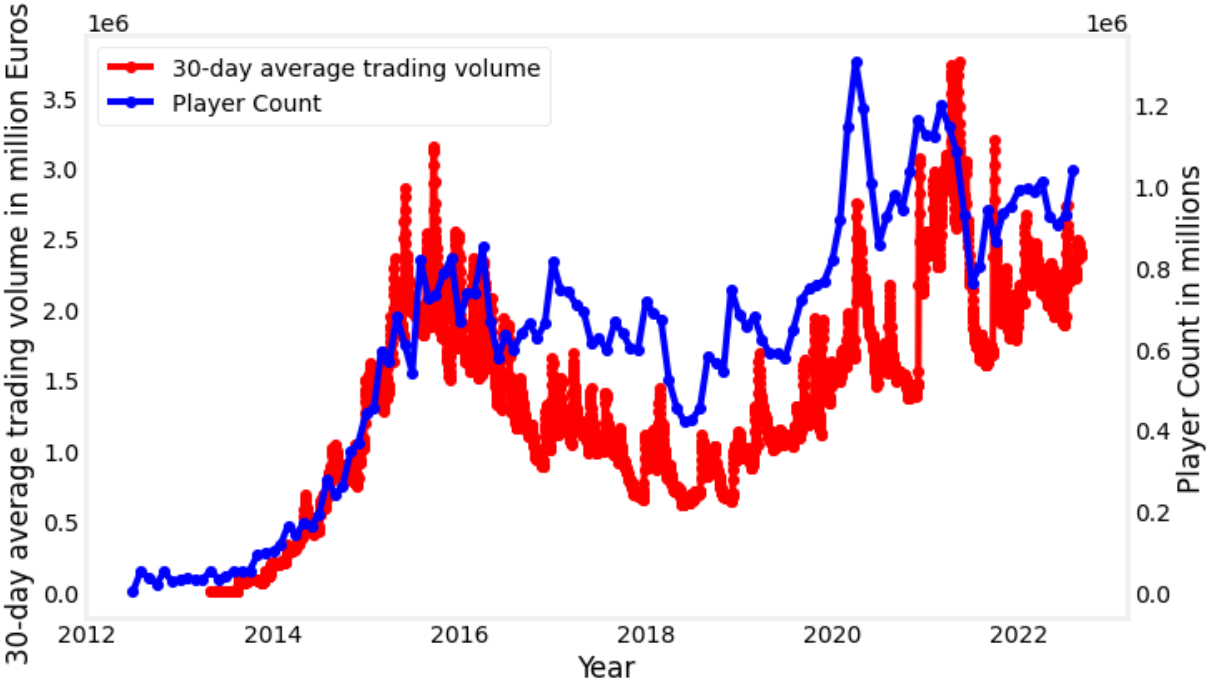


Figure 3: 30-day average trading volume in millions of Euros versus player count in millions, data: (Steam, 2022c; Steam Charts, 2022)

2.2 Literature Review

2.2.1 Time series forecasting introduction

Forecasting future values of an observed time series is an area of expanding interest crucial to almost all scientific and technical disciplines, including economics, finance, meteorology, and communications (Palit & Popovic, 2005). This significance, in combination with an ever-increasing amount and complexity of available data, calls for forecasting tools that are more and more reliable, scalable, and, most importantly, accurate. On the one hand, the accuracy of such methods is also dependent on the forecast horizon as multi-step ahead forecasting generally leads to an accumulation of errors, growing uncertainty, and therefore lower accuracy (Crone et al., 2011; Sorjamaa et al., 2007; Tiao & Tsay, 1994). On the other hand, this metric is also influenced by the dataset used, as randomness varies.

Time series forecasting methods can be divided into linear and non-linear statistical models. For a long time, ad hoc approaches for different kinds of univariate time series were SES (Brown & Meyer, 1961; Muth, 1960) and other more complex linear statistical techniques like ARIMA (Box et al., 1994) models. They significantly impacted the time series forecasting research during this period (de Gooijer & Hyndman, 2006). Simultaneously, non-linear methods like the "AutoRegressive Conditional Heteroscedastic" (ARCH) (Engle, 1982) family were introduced. Over time, machine learning methods, like "Artificial Neural Networks" (ANN), became increasingly popular as they severely threatened the dominant traditional linear univariate methods in forecasting competitions (Ahmed et al., 2010; Crone et al., 2011). For example, in the NN3 competition (SAS & IIF, 2006), traditional models dominated the short-term forecasting horizon (h=1-3). However, more sophisticated models caught up or even outperformed them in medium to long-term time horizons (h=4-18) (Crone et al., 2011). Especially as machine learning models like ANNs are non-linear, they are more precise in predicting big, noisy, and non-linear time series data with complex relations and patterns (Li & Ma, 2010). Other machine learning methods like decision-tree-based methods like Gradient boosting and XGBoost got also introduced gradually.

Deep learning (DL), a type of machine learning, has seen great success in recent years as GPUs' availability, price, and performance improved tremendously (Jiang, 2021). DL models consist of multiple ANN layers, which provide a high-level abstraction of the modelling data (LeCun et al., 2015).

Nowadays, the most common DL method family for time series forecasting is “Recurrent Neural Network” (RNN). These models have a feedback loop, contrary to ANNs, which have a forward connection. However, RNNs are still not generally better than traditional models, which also always have the advantage of using less computational power and are more beginner-friendly (Hewamalage et al., 2021).

2.2.2 Financial time series forecasting

Financial time series forecasting is a highly attractive but challenging research area, as demonstrated by the enduring debate surrounding the efficient market hypothesis (EMH) (Malkiel & Fama, 1970).

The EMH (Malkiel & Fama, 1970) is a theoretical framework that describes how financial markets operate. It is based on the idea that the prices of assets in financial markets reflect all available information. Therefore, it is impossible to systematically outperform the market by using past or current prices as a basis for investment decisions.

In other words, the efficient market hypothesis suggests that financial markets are highly efficient in incorporating new information into asset prices. It is difficult or impossible to gain an advantage by analysing past or current prices (Malkiel & Fama, 1970). One implication of the EMH is that asset prices would follow a random walk, which shows the unpredictable nature of asset price movements (Fama, 1995). While the efficient market hypothesis has been widely accepted and studied in the field of finance, it has also been the subject of significant debate and criticism. The number of studies that offer evidence in opposition to what the efficient market hypothesis and random walk hypotheses indicate is growing, as they provide evidence that, for example, the stock market is predictable to some degree (Atsalakis & Valavanis, 2009; Malkiel, 2003; Nofsinger, 2005).

Nevertheless, the prediction of financial time series is very challenging due to its chaotic, non-stationary, noisy nature (Abu-Mostafa & Atiya, 1996; Swanson & White, 1997). The term “noise” refers to information that is not relevant or significant for a particular model. More specifically, it often refers to the lack of complete data from past financial market activity necessary to accurately reflect the relationship between future and previous prices. The “non-stationary” means that the financial time series' distribution is altering over time. “Chaotic” refers to financial time series that are long-term deterministic but short-term unpredictable (Tay & Cao, 2001).

There is a wealth of research on financial time-series prediction on many different assets using different methods, from linear to non-linear techniques, to challenge this problem. Other reasons for this large number of research are, among other things: financial motives and the large amount of data to experiment, train, and test with. Datasets used in these studies vary a lot as underlying assets, time frames, predicting horizon, and the models, even model settings, compared against each other change from study to study.

Generally, previous research on financial issues points out that a significant number of study outcomes revealed that the accuracy of non-linear approaches is superior to that of traditional linear and statistical methods (Bahrammirzaee, 2010; Stock & Watson, 1998; Zhang, 2003). Methods based on ANN layers are the most used non-linear technique (Gandhmal & Kumar, 2019) as it has a data-driven approach. The model specifies itself based on the given data and suits the characteristics mentioned above of financial time series well. This automatic specification makes ANNs less prone to the issue of model formation. Additionally, the ability to learn complex patterns makes it more noise resilient and retrainable as it can adapt to new patterns (Tay & Cao, 2001). On the contrary, linear models like ARIMA assume a linear relationship in the data (Zhang, 2003), which makes financial time series harder to predict.

With the recent general advances and successes in DL, current research in financial time series forecasting is mainly based on these methods. Around 50% of the DL studies use RNN-based models, ∼20% Deep Multilayer Perceptron (DMLP)-based models, and ∼15% Convolutional Neural Network (CNN)-based models as their primary or benchmark method. Out of RNN-based methods, Long Short-Term Memory (LSTM; for the details on this model, refer to section 2.3.5) dominates as they make up 60% due to their relatively simple model-building steps. Vanilla RNN follows with a share of ∼30% and Gated Recurrent Unit (GRU; for the details on this model, refer to section 2.3.6) with ∼10 %. The most frequently studied subgroup of financial time series price forecasting is, by far, stock price forecasting, followed by index forecasting (Sezer et al., 2020). Due to the previously mentioned difference in the setups of published studies, it is difficult to make consolidated conclusions. LSTM seems to be the most competitive DL method and better than ANN (Nabipour et al., 2020; Sethia & Raut, 2019). XGBoost and Adaboost are the most accurate decision tree methods, which are also better than ANN (Jabeur et al., 2021; Nabipour et al., 2020).

Overall, the literature suggests that there is no one-size-fits-all approach to financial time series prediction and that the choice of method will depend on the specific context and data available.

There are no published studies of price forecasting of digital products like in-game items in the examined literature. Therefore, this study aims to answer whether their price movements are predictable. Furthermore, studying some models' performance makes it intriguing to determine the most precise method. A naïve model will be used as a baseline to determine predictability and as a performance comparison for other models. More precisely, it is compared against the linear models SES and ARIMA and the non-linear models LSTM, RNN, and GRU.

2.3 Brief review of prediction models and other methods

2.3.1 Naïve model

The naïve model is often used as the baseline model as it would, in expectation, outperform other models if the time series were a random walk process. It is a simple method that forecasts the value of the n data point in a time series as the current value. The model can be summarised as follows:

$$\widehat{y_{t+1}} = y_t$$

With y_t as the value y in the current period t and $\widehat{y_{t+1}}$ the predicted value for the next period $t+1$.

2.3.2 Simple exponential smoothing

The simple exponential smoothing (SES) approach uses little computational power and bases his forecast on prior time steps that are exponentially weighted and, in the end, combined:

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots + \alpha(1-\alpha)^k y_{t-k}$$

In the above equation y_t is the value y in the current period t. Therefore, y_{t+1} is the predicted value for the next period $t+1$ and is a weighted moving average of the considered observations. Furthermore, y_{t-k} is the know past value of period $t-k$ (the current period t minus the backwards looking periods k). Finally, α is the smoothing coefficient that gives the rate of weight decay (Brown & Meyer, 1961). The weights $\alpha, \alpha(1-\alpha)^k$ of each data point decline exponentially as we move backwards in the time series, resulting in an ever-smaller influence on the outcome value (Ostertagova & Ostertag, 2011).

2.3.3 Auto-Regressive Integrated Moving Average

An ARIMA model is a statistical model commonly used in time series analysis to forecast future data points of a time series based on its past values. Generally, it is a type of linear regression model that incorporates the concepts of autocorrelation and stationarity. An ARIMA model uses these concepts to account for the dependencies and trends in a time series and to forecast

future values based on the data. To create an ARIMA model, a time series is first differenced to remove any non-stationarity, afterwards, the autocorrelations and partial autocorrelations of the differenced time series are analysed. The results of this analysis determine the appropriate values for the parameters of the ARIMA model. The default notation of ARIMA is (p,d,q) together with its seasonal parts (P,D,Q) (Box et al., 1994) and consists of the following parts:

$$y_t = \underbrace{\emptyset_1 y_{t-1} + \cdots + \emptyset_p y_{t-p}}_{\text{Non-seasonal AR part}} + \underbrace{\psi_1 y_{t-m} + \cdots . + \psi_P y_{t-Pm}}_{\text{Seasonal AR part}} +$$

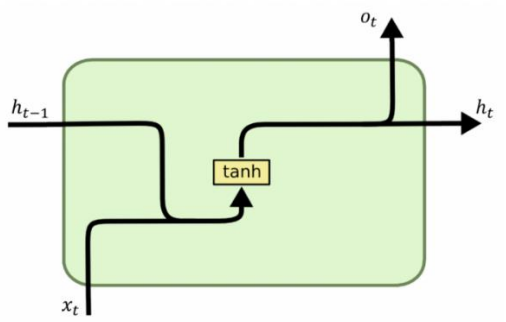
$$\underbrace{\theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}}_{\text{Non-seasonal MA part}} + \underbrace{\xi_1 \epsilon_{t-m} + \cdots + \xi_Q \epsilon_{t-Qm}}_{\text{Seasonal MA part}} + \epsilon_t$$

With y_t as the forecasting value, the two seasonal and non-seasonal AutoRegressive parts, the two seasonal and non-seasonal Moving Average parts and ϵ_t as the noise.

2.3.4 Recurrent Neural Network

A recurrent neural network (RNN) is a type of ANN specifically designed to process sequential data. Unlike a traditional feedforward neural network, which maps fixed-sized inputs to fixed-sized outputs, an RNN can process data of varying lengths and produce outputs of varying lengths. RNNs are composed of units called cells, which have internal states that allow them to retain information from previous inputs in the sequence. This allows an RNN to process data sequentially as it captures long-term dependencies, making it well-suited for time series forecasting tasks (Connor et al., 1994).

An RNN layer comprises one or more "cells" that are organised into a sequence. As shown in Figure 4, each cell has a set of weights and biases that are used to make predictions based on the input data and the previous cell's output. The output of each cell is passed as input to the next cell in the sequence, allowing the network to process the input data one step at a time (Elman, 1990; Williams & Zipser, 1989).



x_t: input vector
h_t: hidden layer vector
o_t: output vector
$tanh$: activation function

Figure 4: RNN cell structure, *graphic from:* (dProgrammer lopez, 2020)

2.3.5 Long Short-Term Memory

LSTM network (Hochreiter & Schmidhuber, 1997) is a type of RNN developed to tackle the vanishing gradient problem that occurs in RNN for long-term predictions. Instead of using hidden layers like RNN, LSTM contains cells to store long and short-term data of a sequential data set. Especially significant is the cell state (C) that stores the necessary information over a long duration and is, therefore, the main advantage to RNN. Figure 5 shows the architecture of a basic LSTM unit. In the input gate, new information (X_t) is fed and combined with the output of the previous block (H_{t-1}). Generally, sigmoid layers (σ) decide how much information is passed by using values between zero and one. The cell state (C) is the memory of the whole network and uses gates to remove, with the help of a forget gate layer, or add, with new vectors generated by a tanh layer ($tanh$), information. Finally, the output gate (H_t) gives out the generated information.

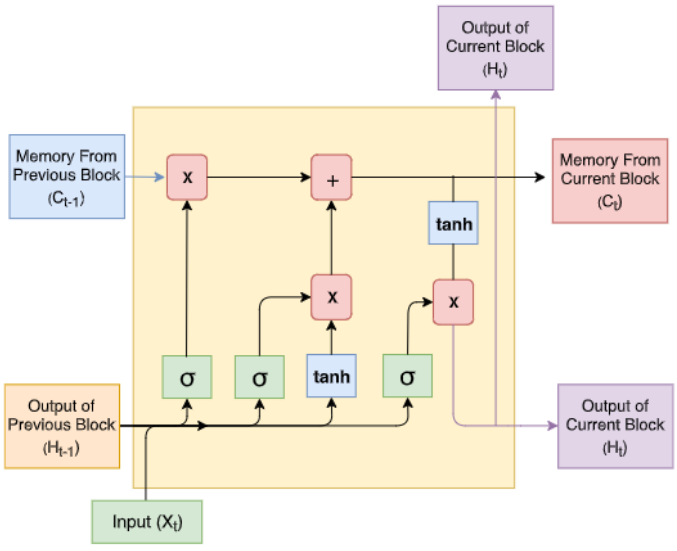


Figure 5: LSTM cell structure (Hochreiter & Schmidhuber, 1997) , graphic from: (Murat Ozbayoglu et al., 2020)

2.3.6 Gated Recurrent Unit

A Gated Recurrent Unit (GRU) (Chung et al., 2014) is a type of recurrent neural network designed to reduce the amount of computation required during training while still allowing the network to capture long-term dependencies in the data. This makes it an effective alternative to other types of RNNs, such as LSTMs. Compared to LSTM, the GRU does not have an output gate. Figure 6 shows that it has an update gate and a reset gate. These gates are vectors that determine which information from the input should be passed to the network's output. In this way, the update and reset gates help the GRU to efficiently capture long-term dependencies in the data (Yamak et al., 2019).

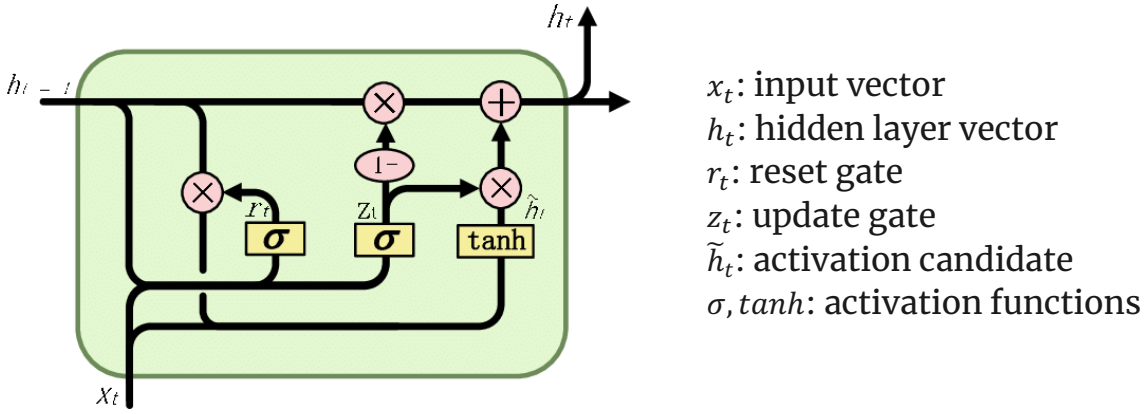


Figure 6: GRU cell structure, graphic from: (dProgrammer lopez, 2020)

2.3.7 Stationarity testing

In this study, the forecasting abilities of several prediction methods are also compared for different time series structures, more precisely, their stationarity. To distinguish the underlying time series structures, the Augmented Dickey-Fuller test is applied.

The Augmented Dickey-Fuller (ADF) test is a statistical test to check the stationarity of the time series. It is a type of unit root test, which means it is used to determine whether a time series exhibits a unit root, a feature that indicates the presence of a trend in the data. The test is based on a linear regression model with lagged time series values as independent variables. According to the null hypothesis of the ADF test, the time series has a unit root, which means it is non-stationary and has a trend. The alternative hypothesis is that the time series is stationary, which means it does not have a trend (Dickey & Fuller, 2012).

2.4 Accuracy evaluation

2.4.1 Forecasting strategies

Forecasting tasks are, in most cases, challenges that predict not only one but multiple data points into the future, so-called multi-step-ahead forecasting. There are multiple strategies to tackle

this problem, the most notable four are: the recursive strategy, direct strategy, DiRec strategy, and multi-input multi-output strategy (MIMO).

The recursive strategy involves making a series of one-step-ahead forecasts by using the prediction from the previous time step as input for the next prediction. Another technique is the direct strategy, which involves using separate models to make predictions for each time step of the forecasting horizon. A third technique, the DiRec strategy, merges elements of both previously mentioned strategies. The output of each model for each step is the basis of the next consecutive model. These techniques are all designed to produce a scalar value for a single forecasting point in the prediction timeframe (Xiong et al., 2013).

The last method, the Multiple-Input Multiple-Output (MIMO) strategy, however, forecasts a vector of future values rather than a single scalar quantity. This approach, which involves using a single multiple-output model, has the advantage of preserving the temporal dependencies within the predicted timeframe, unlike the direct strategy, which relies on multiple models to estimate each individual future value (Bontempi et al., 2013). Multiple studies compared these methods against each other and concluded that the MIMO strategy delivers the highest quality results among them (Bao et al., 2014; Taieb et al., 2012; Xiong et al., 2013). The major disadvantage of the recursive and DiRec strategies is their reliance on the prediction for the current time period being based on the previous time period. This can lead to a compounding of errors, resulting in decreased accuracy and reliability compared to the MIMO strategy. In particular, this issue becomes more significant for more extended time horizons or when the time series exhibits complex patterns or trends (Bao et al., 2014; Taieb et al., 2012; Xiong et al., 2013).

2.4.2 Performance measurements

The performance of the studied models is evaluated based on two different methods. The first one is the symmetric mean absolute percentage error (sMAPE) used by many other studies and forecast competitions. It is calculated as the average of the absolute percentage errors of the forecast and the actual value. sMAPE is denoted as follows (Shcherbakov & Tyukov, 2013):

$$sMAPE = \frac{100\%}{H} \sum_{k=1}^{H} \frac{|F_k - Y_k|}{(|Y_k| + |F_k|)/2}$$

With F_K being the forecast, Y_k the actual value, H the sample size, and the output as percentage value between 0% and 200%.

The main advantage of sMAPE compared to mean absolute percentage error (MAPE) is that due to its symmetric nature, it treats the forecast and the actual value equally, regardless of which is larger. This avoids bias that may appear while handling values of different magnitudes (Goodwin & Lawton, 1999).

Furthermore, as sMAPE is calculated as the average of the absolute percentage errors rather than the absolute percentage error itself, it is less affected by outliers or extreme values in the data than MAPE. (Armstrong & Collopy, 1992).

Nevertheless, sMAPE also has shortcomings due to its symmetric measure. Mainly, it does not consider the direction of the error (i.e., whether the forecast is over or under the actual value) and exhibits instability when the true values of the data are close to zero. This instability can cause the sMAPE to produce unreliable results and should be considered when using this measure (Hyndman & Koehler, 2006). This study is not heavily affected by the second point as all time series have a minimum average price of around 5.5 Euros, as presented in the upcoming Chapter 3.2. Furthermore, another limitation is the interpretability, as it is not straightforward to compare the symmetric error (Armstrong & Collopy, 1992).

To tackle these issues, a second error metric is needed. Another way to ensure that the scale of the data does not influence accuracy measures is to use relative errors based on the errors generated by a benchmark method, such as the naïve method. Two commonly employed measures of this type are the mean relative absolute error (MRAE) and the geometric mean relative absolute error (GMRAE). It is generally accepted that the geometric mean is a more suitable measure for averaging relative quantities than the arithmetic mean. As a result, GMRAE is often preferred for this purpose (Armstrong & Collopy, 1992; Fildes, 1992) and is also applied in this study. It is defined as follows (Chen et al., 2017):

$$GMRAE = \sqrt[n]{\prod_{t=1}^{n} \left(\frac{|y_t - \hat{y}_t|}{|y_t - y_{t-1}|}\right)}$$

Where n are the observations, y_t the observation at time t, $\hat{y}_t$ the forecast at time t , y_{t-1} the benchmark forecast at time t. A GMRAE value above 1 indicates that the reference method performs better. In contrast, a value below 1 indicates that the forecast produced by the model in question is superior. However, it has been noted that the GMRAE is still sensitive to outliers (Chen et al., 2017). It can be significantly impacted by either a single large outlier or a very

small error close to zero due to the lack of an upper or lower bound on the log-scaled error ratios calculated by the GMRAE. This can result in large values being returned if the error of the benchmark method is zero.

3. Data and models

In this chapter, the first section describes and visualises the dataset and its source. The dataset was then clustered in this study, with the clustering process and results explained in the subsequent section. The following section describes the data split method, afterwards, the model settings are presented. Finally, the evaluation procedure is explained in a detailed fashion.

3.1 Dataset description

This study is based on a dataset from the Steam Community Market with an example overview shown in Figure 7. The data records the average price and daily traded volume for CS:GO in-game items and was obtained through API connections listed in this website's HTML source code.

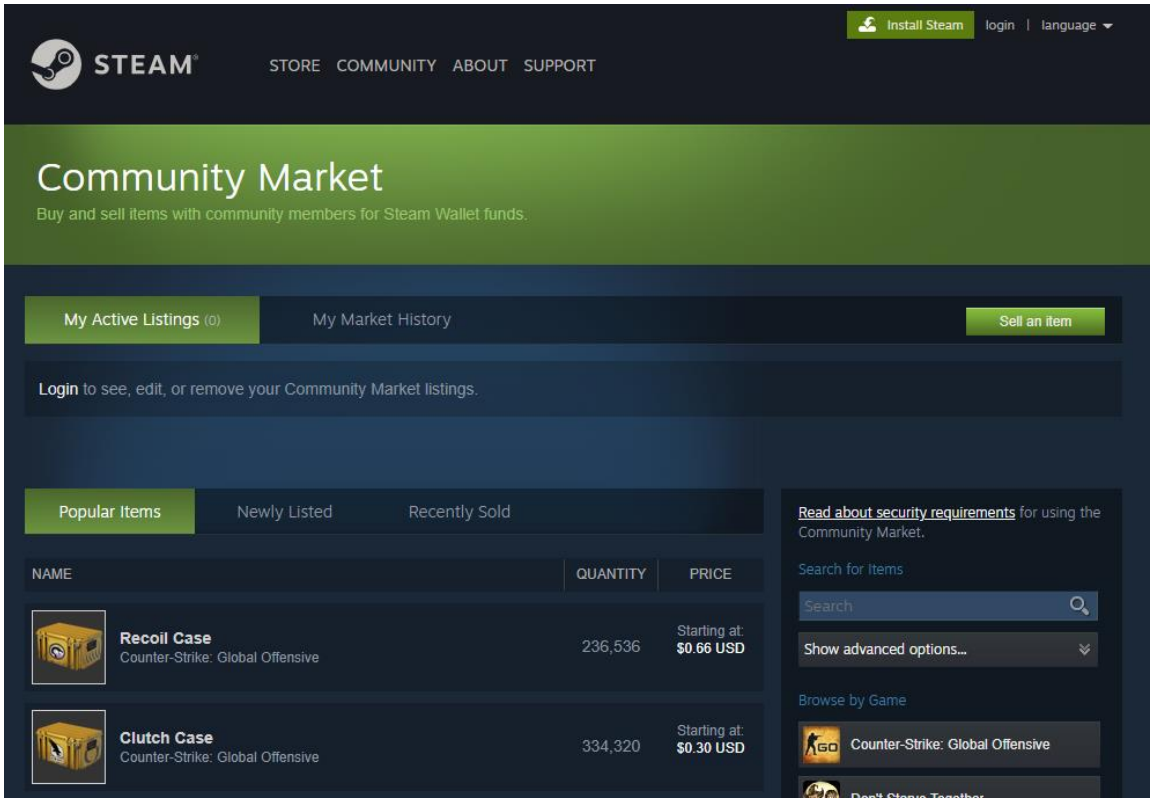


Figure 7: Steam Community Market landing page, from (Steam, 2022c)

The data acquisition process consists of two parts. First, a "search" request is utilised to obtain the names and basic information, such as the current price, of all items on the Community market for a particular game. Secondly, a "price history" search was performed on a single item at a time to obtain more detailed history price information. The data is gathered using Python's "requests" package to obtain the API data. Steam limits the number of items returned in a single request to 100, so multiple requests are necessary to retrieve all item names. In total, 17319 item names are collected in this process. For each item, a large amount of price history data is gathered. The data includes three variables, the average sale price and sales volume for each day the item has been available for sale and is being sold on the market and the corresponding date. Days without a sale for a specific item are not being registered. The price data is given in Euros, and to avoid confusion in the further course of this study, the variable "average daily sale price" is called "price".

The dataset ranges from 26.04.2013 to 31.08.2022 and consists of 23.414.201 observations. The minimum price is 0.03 Euros and the highest price is 2545 Euros. Regarding volume, the minimum is 1 with the maximum being 964250 per day.

Figure 8 shows the number of items in the dataset on the y-axis and the date in the form of the year on the x-axis with the intention of comparing the total number of available items over time to the number of actual traded items per day. Over the observation time span, more and more items are added to the game in a steady growth. However, the slope of traded individual items per day is not quite as steep. This means more and more products on the marketplace are not traded daily and therefore are more illiquid.

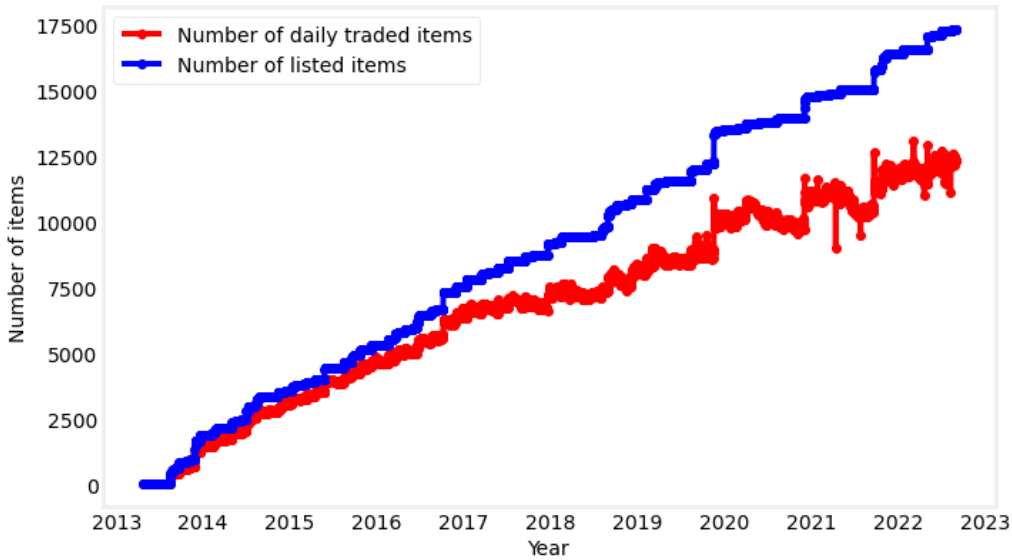


Figure 8: Total amount of items listed versus daily traded items, data (Steam, 2022c)

The following Figure 9 also supports this conclusion. It shows on the x-axis the volume per observation, and on the y-axis the days a product is listed on the market. When an item is brought into the game, it often starts with a higher trading volume and keeps this level before it levels off the longer it is in the game. One of the reasons for this is that in-game items are getting dropped less and less the longer in the game (CSGOSKINS.GG, 2022).

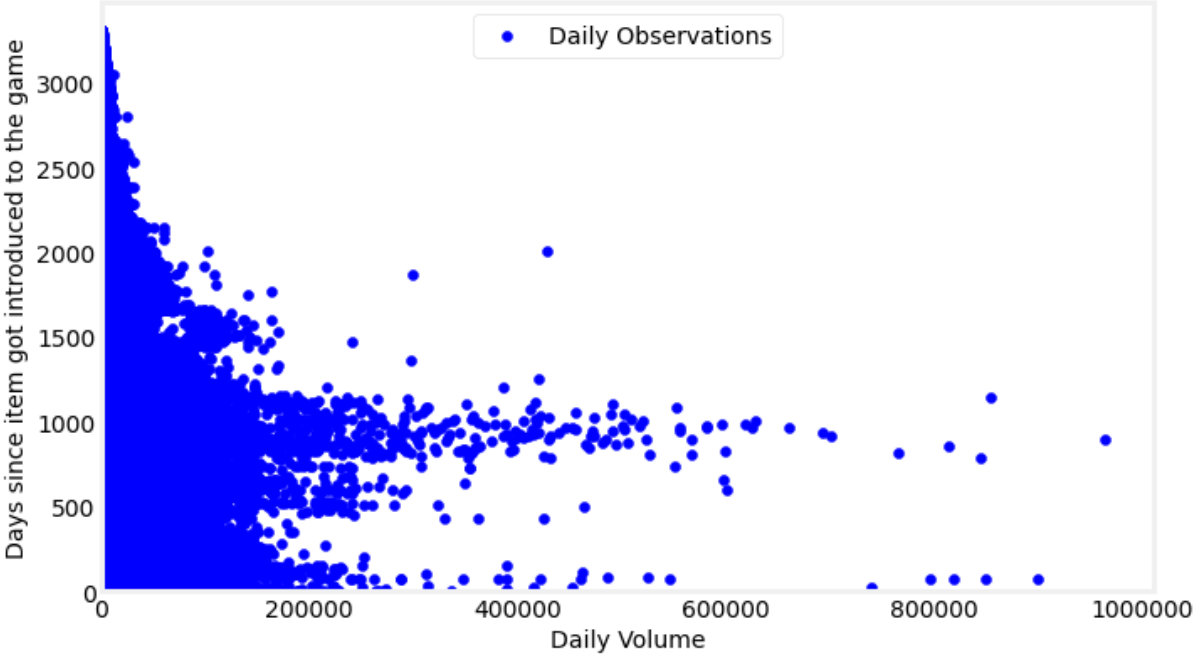


Figure 9: Daily volume in relation to days since the product was introduced in the game, data (Steam, 2022c)

Another takeaway from Figure 9 above is that the dataset contains time series with all kinds of different lengths. In fact, there are only a few sale data points for some products. However, some of the prediction models and applied techniques used in this study, presented in chapter 3.4, need a certain amount of data points to work. For this reason, the dataset is reduced to time series with more than 100 observations, which reduces the number of items to 16623.

3.2 Data clustering

Chapter 3.1 showcases the study's large data set with 16623 time series after cleaning and filtering. Due to this immense amount of data, clustering is an excellent strategy to further understand the dataset and group items into clusters based on similarity. Another objective is to select a cluster to focus on, as my computational power is limited, and the machine learning methods applied in this study require a lot of resources.

One of the most widely-used algorithms for this task is K-means clustering due to its simplicity (Jain & Dubes, 1988). The K-Means algorithm is a clustering method that aims to separate a given dataset into a specified number of clusters by minimising the sum of within-cluster variance, also known as inertia. This algorithm handles large datasets efficiently. The algorithm works by selecting a predetermined number of clusters, k, first and then iteratively assigning each element to the cluster with the closest mean or centroid. The centroid of a cluster is the average of all its data points and is updated at each iteration to reflect the current assignments of data points to the cluster. The algorithm continues to iterate and update the cluster assignments and centroids until the assignments stabilise and no further improvements can be made (Pedregosa et al., 2011). It is important to note that the number of clusters must be specified prior to running the K-Means algorithm, and this limitation can be challenging to determine in practice.

The simplest and oldest technique to tackle this problem is the elbow method. It is a visual method for determining a dataset's optimal number of clusters. It involves iteratively increasing the number of clusters, starting at 2, and calculating the cost of training the model with each step. The optimal number of clusters is determined by identifying the point at which the cost decreases rapidly, followed by a plateau. This is known as the "elbow" of the curve. The rationale behind this method is that adding additional clusters beyond the elbow point does not significantly decrease the cost and may lead to overfitting. Therefore, the optimal number of clusters is often identified at the elbow point (Makwana et al., 2013).

Before applying K-Means clustering and the elbow method, each time series of the dataset presented in chapter 3.1 needs to be summarised. This extension also provides a dataset with additional variables for the algorithm to calculate with. Table 1 lists and explains each of the variables.

Variable name	**Explanation**
Price Maximum	The maximum price of a particular time series
Price Minimum	The minimum price of a particular time series
Price Average	The average price of a particular time series
Volume Maximum	The maximum volume of a particular time series
Volume Minimum	The minimum volume of a particular time series
Volume Average	The average volume of a particular time series
Length	The number of days the product got traded

Table 1: Overview of the variables used for the K-Means algorithm

The new dataset is visualised in Figure 10. The data of most of the first six variables are heavily skewed towards one axis and do not follow a Gaussian Distribution. Furthermore, these variables have extreme outliers. The data is being normalised to correct for these two aspects as it does not affect the K-Means algorithms where no assumptions about the distribution are needed (Visalakshi & Kuttiyannan, 2009).

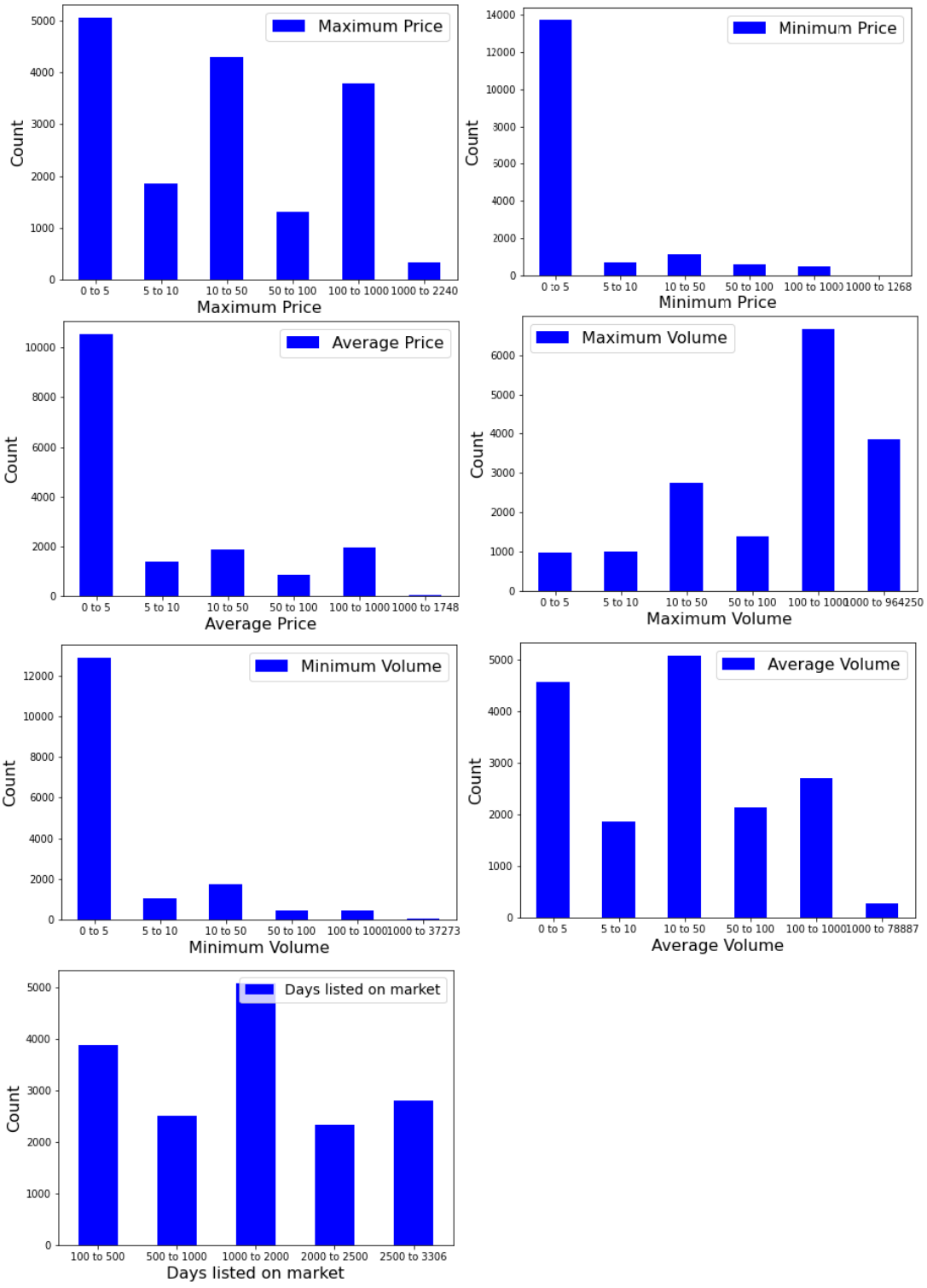


Figure 10: Distribution of the values of the variables used for the K-Means algorithm

The normalised data is fed into the algorithm. The aforementioned elbow method, with cluster sizes between two and twenty is tried to get more insights about the cluster size.

The outcome of applying the elbow method to this dataset is shown in Figure 11. The added circle is the area where the optimal number of clusters lies. Due to an immense dataset and the mentioned limitation of computing power, the cluster size of five, at the upper end of the number of needed clusters, is defined.

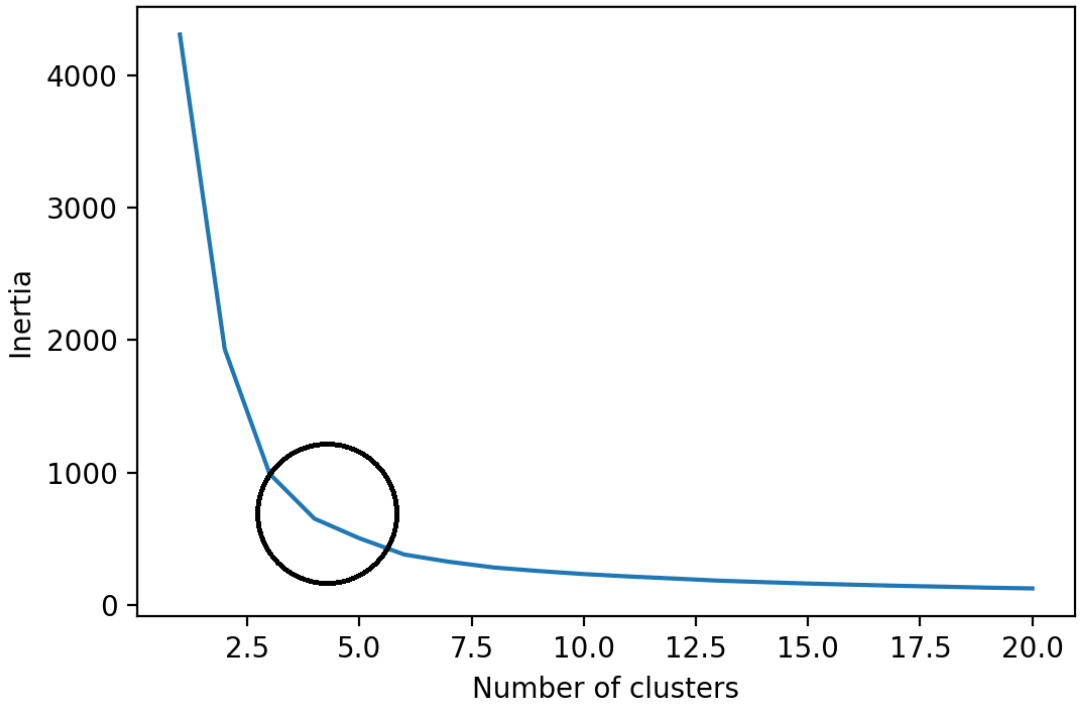


Figure 11: Outcome of the elbow method

The results of the K-Means clustering are listed in Table 2. The algorithm divided the dataset into five groups with different sizes. Cluster zero has the most items with almost half of the dataset. The rest is split into the other four groups, with group two being the smallest and group three the biggest. As the goal is to find a cluster worth studying concerning the earlier mentioned trading and investing focus, prices and volume are essential. Groups zero, one, and three have low-priced items, with the lowest maximum price of all assigned time series of under ten cents. The lowest maximum price for clusters two and four is way higher at 115 euros and 30 euros each.

Regarding the highest maximum price, the groups are split into two subgroups with one, two, and four on one side and zero and three on the other side, with each similar price. Group two is clearly standing out in terms of its average of the maximum, minimum, and mean prices of approximately 760 euros, 94 euros, and 280 Euros, respectively. Group four is the closest, with around half of the prices. The most significant difference between clusters two and four in terms of price is the minimum average price, where cluster two stands out with a nearly ten times higher price. There is also a clear picture regarding the trading volume as the clusters with higher prices are traded less frequently. In contrast to the price variables, these two have similar characteristics when it comes to volume. Group two stands out regarding the number of traded days for each product, as it has the lowest mean by far.

In conclusion, group two consists of items with the highest prices and the lowest volume and are most comparable with group four, and the difference in volume is negligible. A cluster with a lower number of items is preferred due to the mentioned componential limitations. It should also have a relatively high price as the error metrics used in this study are potentially sensitive to small numbers. Therefore, cluster two is the one being closely examined in the further course of this study.

	Number of	Maximum Price			Minimum Price		
Cluster	items	min	max	average	min	max	average
0	8009	0.09	829.97	63.58	0.03	134.6	4.26
1	2916	0.03	2022.2	20.64	0.03	188.7	0.85
2	944	115.19	2239.8	758.95	0.03	1267.91	94.16
3	3178	0.1	804.73	18.87	0.03	114.98	0.99
4	1576	30.08	2182.37	378.76	0.03	302.47	32.2

	Average Price			Maximum Volume			Minimum Volume		
Cluster	min	max	average	min	max	average	min	max	average
0	0.03	343.43	15.18	3	1198	200.86	1	101	3.36
1	0.03	247.98	3.25	161	964250	11467.62	1	37273	101.83
2	5.79	1748.48	279.06	1	781	17.39	1	15	1.02
3	0.03	159.78	3.21	47	4267	1017.76	1	304	13.61
4	0.67	755.59	129.24	2	499	19.91	1	41	1.09

	Average Volume			Days listed on market		
Cluster	min	max	average	min	max	average
0	1.06	400.02	27.24	103	3305	1757.66
1	7.5	78886.69	807.36	106	3305	1079.02
2	1	33.66	1.8	101	1768	320.14
3	2.81	1084.72	108.27	106	3306	1463.74
4	1.03	78.65	2.76	106	3102	761.69

Table 2: Results of the K-Means clustering

3.3 Dataset split

In developing a machine learning model, it is standard practice to divide the available time series data for each item into distinct portions for training and testing. This allows for a thorough assessment of the model's performance and ability to generalise to new, unseen data. Specifically, the training portion of the dataset is used to fit the model, and the testing set is used to evaluate the fully-trained model's ability to make accurate predictions on data that it has not seen during the training process. This multi-faceted data division approach helps ensure that the resulting model is effective and reliable.

The training process for linear and non-linear models differs significantly, which impacts how the time series data is divided for the development of each model. Linear models rely on a linear combination of input features to make predictions, while non-linear models, such as neural networks, apply non-linear transformations to the input data. As a result, the training process for these models involves different optimization algorithms and requires different amounts of data.

Considering these differences, the time series data must be divided differently to develop linear and non-linear models. While both types of models typically require a training set and a testing set to evaluate the model's performance on unseen data, non-linear models also benefit from an additional validation set to fine-tune the model's hyperparameters.

In all cases, the test set for each model and item consists of the final 30 observations, equivalent to the forecast horizon. The training set for linear models includes the remaining portion of the time series. As a result, the size of the training set may vary based on the specific total length of each dataset.

For non-linear models, the “sliding window technique” (Vafaeipour et al., 2014) is applied, which divides the whole training set into multiple training and evaluation sets. This technique involves specifying a window size for the training horizon and a step size. The step size determines how far the window moves along the time series for consecutive windows. In this study, the window size is 40, and the step size is one, meaning each window contains 40 observations. This length was determined by following insights from another study, which concluded that training horizons for RNN-based models should be slightly longer than the anticipated prediction horizon to achieve the best results (Hewamalage et al., 2021). Furthermore, before applying it to all models and items, different lengths between 40 and 60 were tested on randomly chosen products, with no significant differences in accuracy identified.

Figure 12 illustrates the sliding window technique applied to a single time series over three steps. The blue sections represent the training data, and the orange sections represent the validation data. This method allows for the systematic evaluation of the model's performance on different sections of the time series, providing a more robust assessment of the model's ability to generalise to unseen data.

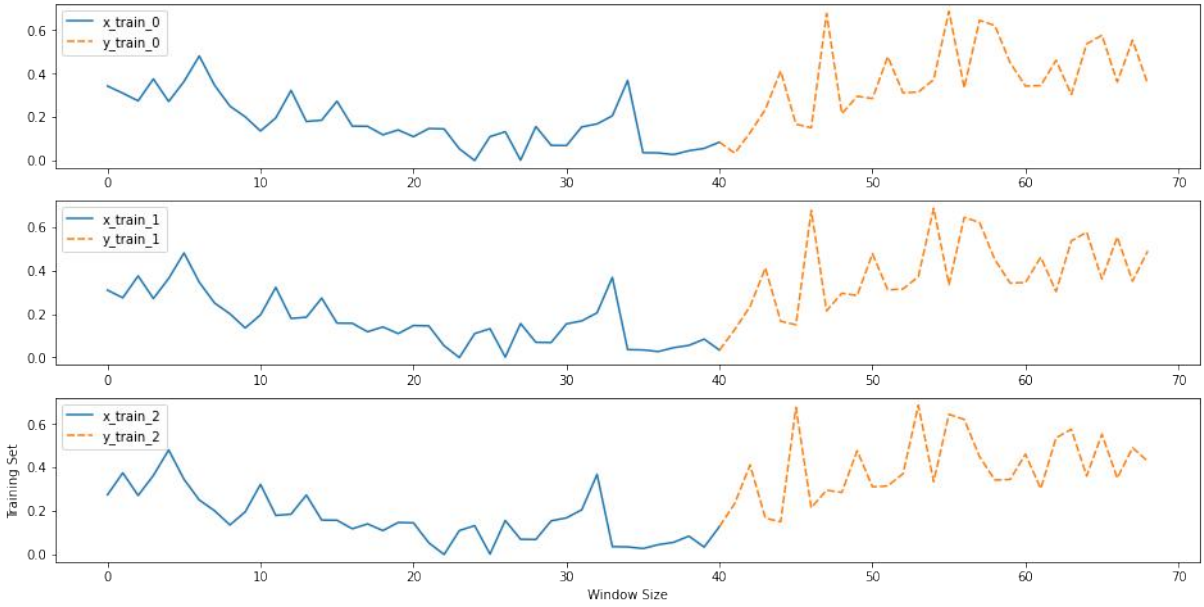


Figure 12: Sliding window technique for the non-linear model

3.4 Modelling

In selecting the models for this study, previous research in financial time series forecasting was considered, considering that this study uses univariate data. This approach was chosen to build upon the existing knowledge base and identify effective models in similar contexts. This study aimed to evaluate the performance of different modelling approaches and gather insights on the price predictability of items on in-game marketplaces.

This study consists of one benchmark model, two linear models, and four non-linear models. For each item, a new model is created and specially trained on its data. All in all, taking into account the 944 items and seven models, there are 6608 different models developed.

In this study, the benchmark model is a naïve model based on the EMH assumption that prices of items on the steam marketplace are not predictable and follow a random walk. The value for the naïve model during the whole forecast horizon is the last price point of the training set.

The following method, the SES model, was implemented using the "statsmodel" package in Python (statsmodel, 2022). One key feature of this package is its ability to automatically determine the optimal value of the smoothing parameter (α) within the training data set for each time series. The optimisation of α is critical for the accurate representation of the underlying trend and seasonality of the data, as it controls the weight given to past observations in the forecast. By utilising the automatic optimisation feature, the SES model is able to effectively capture the dynamic nature of the data and produce more reliable forecasts.

This study utilised the auto-ARIMA process (Smith, 2017) to automatically discover the optimal order for an ARIMA model on the training set. This process involves a systematic search for the most suitable combination of the three key parameters of an ARIMA model: the autoregressive (AR) order, the differencing order, and the moving average (MA) order.

To determine the optimal differencing order, d, the auto-ARIMA process conducts various differencing tests, such as the Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller, or Phillips–Perron tests. After identifying the optimal value of d, the process then fits a range of ARIMA models with different values of the AR and MA orders. It selects the model that minimises the Akaike Information Criterion (AIC) information criterion. In the case of seasonal data, the auto-ARIMA process also determines the optimal order of seasonal differencing, D, through the Canova-Hansen test and includes this parameter in the optimisation process.

For all machine learning models, the input sequence length was set to 40 data points based, ReLU is used as the activation function, and the Adam optimisation algorithm was utilised due to its demonstrated effectiveness in another study (Hewamalage et al., 2021). A batch size of 64 and a maximum of 200 epochs were also selected. An early stopping mechanism is implemented to prevent overfitting, which occurs when a model becomes too closely adapted to the training data and performs poorly on unseen data (Chollet, 2015a). It is set to terminate the training process if the loss function does not improve for five consecutive steps. The best model was subsequently chosen based on loss and applied to predict the final forecast.

The structure of the machine learning models was carefully designed to reflect the characteristics of the data being analysed. The number of neurons in the first layer is set to match the input data size, which is 40, due to the length of the training set and its univariate nature. The number of neurons in the output layer is determined by the desired period length of the predictions, which is set to 30 in this study.

The output layer was consistently implemented as a dense layer, as all the models utilised in this study were recurrent neural networks (RNNs). This configuration was chosen based on previous research indicating that it was optimal for RNN models (Hewamalage et al., 2021).

More precisely, the RNN model types are basic LSTM, stacked LSTM, basic RNN, and basic GRU and are built using Keras open-source package in Python (Chollet, 2015b). The models were selected based on popularity and accuracy in the time series study field. Table 3 offers an overview of the layer structure for each model. Due to the vast number of datasets and computational constraints, it is impossible to optimise the model's structure for each item, which is, therefore relatively basic. The Stacked LSTM has one more layer compared to the other three models. Adding another LSTM layer to the network allows the stacked LSTM model to learn more abstract representations of the input data, allowing it to capture better the underlying structure and patterns (Graves et al., 2013).

Model	**Layer type, neurons**
Basic LSTM	LSTM, 40 Dense layer, 30
Stacked LSTM	LSTM layer, 40 LSTM layer, 32 Dense layer, 30
Basic RNN	SimpleRNN layer, 40 Dense layer, 30
Basic GRU	GRU layer, 40 Dense layer, 30

Table 3: Structure of the applied machine learning models

Finally, I applied min-max scaling to the price feature to transform its values to a range between 0 and 1. Min-max scaling is a common data pre-processing technique that involves subtracting the minimum value of a feature from each data point and dividing the result by the range of the feature (i.e., the difference between the minimum and maximum values).

3.5 Evaluation procedure

This study uses multiple linear and non-linear models that are fundamentally described in Chapter 2.3 and setting-wise described in Chapter 3.4. These models differ in their data-handling capacities. Nevertheless, to give each model the same data foundation for predicting future timeframes, all models are fed only with univariate data. Therefore, they are predicting a single dependent variable based on a single independent variable, both being in this case, the price variable. The analysis is based on a 30 data points forecast using the MIMO strategy, taking into account the findings outlined in Chapter 2.4.1 and my computational constraints.

The prediction performances of the different models are measured by the two error metrics presented in Chapter 2.4.2. GMRAE is used to compare the model to a naïve benchmark, and sMAPE compares the accuracy of all models, including the naïve model. To assess the statistical significance of the differences in performance among all the models, a non-parametric Friedman rank-sum test is conducted. The Friedman test compares the differences between the multiple techniques in a repeated-measures design. The test calculates a test statistic based on the ranks of the observed responses rather than the actual response values. Then it uses this statistic to determine whether the differences between the methods are statistically significant (Sheldon et al., 1996). The formula is as follows (Friedman, 1937):

$$X_r^2 = \frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3N(k+1)$$

Where k represents the number of ranked methods, N represents the number of time series, and R represents the sum of the ranked scores in each column. This technique is applied to each model's calculated sMAPE and GMRAE errors by using the Python package "scipy" with its function "scipy.stats.friedmanchisquare" (Virtanen et al., 2020).

After evaluating if the outcomes of the overall study are statistically significant, it is also essential to evaluate this in a more detailed way. To achieve this, the Wilcoxon rank-sum test (Wilcoxon, 1992) determines whether the difference between the two sample means is significant. This test is rank-based, like the Friedman test, and likewise available in the same Python package with the function name "scipy.stats.ranksums" (Virtanen et al., 2020). The most crucial objective here is to determine if the models significantly differ from the naïve benchmark and can be interpreted confidently. Both statistical test hypothesis based and will be evaluated with the significance level of $\alpha = 0.05$ to either reject H_0 or not. Figure 13 shows the entire process of testing for significance in this study.

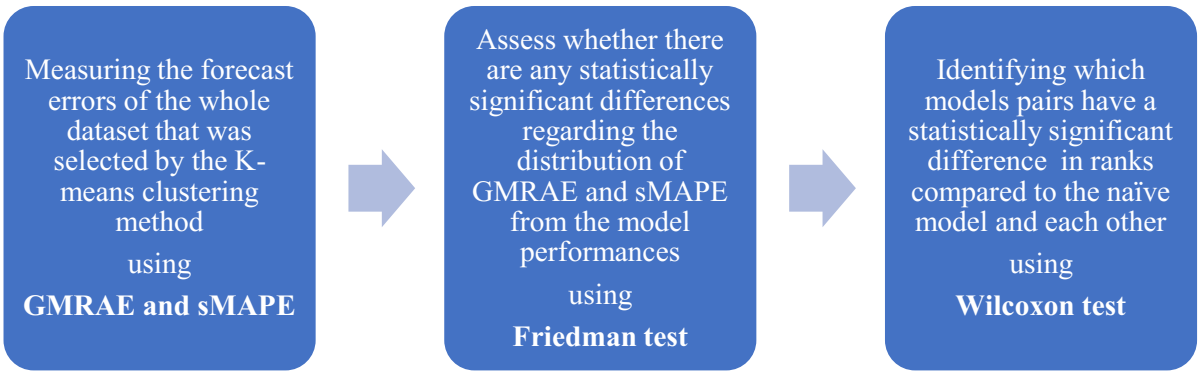


Figure 13: Steps of the performance evaluation

4. Analysis of results

In this section, I present a thorough analysis of the results of my study. As mentioned, I will consider the out-of-sample sMAPE and GMRAE for each model across all time series. Since I have multiple testing sets for each time series, the accuracy of a model on a given time series is the average over all the possible testing sets. The mean sMAPE and mean GMRAE metric results over the whole observed timeframe for each time series of all the models I used can be found on my GitHub in CSV or feather format. Additionally, I also uploaded my code there in three parts: data collection and processing, data visualisation, and, data clustering combined with the prediction and evaluation.

This chapter starts by providing a general evaluation of all models based on the sMAPE and GMRAE error metrics. Mainly the mean and median sMAPE values are initially examined, followed by an analysis of the distribution of sMAPE for each model and an overview of the frequency with which each model was the most accurate. The GMRAE results are presented afterwards. In the next section, the time series in the dataset are divided based on their stationarity to gain a deeper understanding of the performance of each model under different conditions. Finally, the entire observation horizon is divided into multiple 2-step windows, allowing for an evaluation of the accuracy of the models for different prediction lengths.

4.1 General performance evaluation

In order to assess the performance of the various models, I compare their results on all datasets using the sMAPE and GMRAE error metrics. The sMAPE results are summarised in Table 4, which shows the minimum, maximum, mean, median, and count values for each time series and model. Upon examining the mean values, it becomes apparent that the SES model was the most effective in predicting all 30 data points into the future, followed by the ARIMA model. The naïve model also performs relatively well, with results close to those of the SES and ARIMA models. The machine learning models perform significantly worse and are not competitive. Among the non-linear models, the basic RNN model demonstrates the best performance, followed by the stacked LSTM model, which performs similarly to the GRU model. The basic LSTM model was the least effective of the non-linear models.

An interesting finding emerges when examining the median values: the discrepancy between the linear and non-linear models is much smaller than the mean values suggest. Even the relative rankings of the models in terms of performance changed, with the RNN and GRU methods having a slightly better median than the naïve method. However, the RNN model still performs

the best of all non-linear methods. This change indicates the presence of more and heavier outliers in the prediction errors made by the machine learning models. When looking at the minimum and maximum sMAPE values, the ARIMA model has the lowest and highest values, respectively. This suggests that the ARIMA model has a broader range of performance than the other models, but only by a slim margin.

The Friedman test confirms a statistical significance in the difference of sMAPE errors between all models at a significance level of 0.05, with a p-value indistinguishable from zero. Furthermore, the paired Wilcoxon test confirms that the sMAPE difference between nearly all combinations of models is also statistically significant at a 0.05 level. Only the combinations of SES and ARIMA, and Basic and Stacked LSTM have p-values of approximately 0.108 and 0.195, respectively. This means that despite the large difference in mean and median sMAPE, the rank-based Wilcoxon test attests to a more similar performance than it looks at first glance.

index	Naive	SES	Arima	Basic LSTM	Stacked LSTM	RNN	GRU
min	2.341402	2.337488	2.219444	2.552327	2.609358	3.004278	2.749733
max	182.272162	151.034180	200.000000	199.923763	186.871751	181.119041	199.704752
mean	28.382976	26.254722	26.953373	40.623568	36.077755	33.970407	36.813305
median	15.765176	13.970466	14.427042	17.655732	16.381227	15.593709	15.726478
count	944.000000	944.000000	943.000000	944.000000	944.000000	944.000000	944.000000

Table 4: sMAPE metrics overview for each model

To investigate the hypothesis that the non-linear models produced more and heavier outliers in the errors, a violin plot combined with a strip plot was used to visualise the sMAPE metrics, as shown in Figure 14. This visualisation confirms that there are more and more substantial outliers for the machine learning models. Additionally, the distribution of errors indicates that the number of errors decreases as the sMAPE values increase for the linear models. In contrast, for the non-linear models, especially basic LSTM and RNN, the forecasts are either relatively good or very bad, with fewer errors occurring in the intermediate range of sMAPE values.

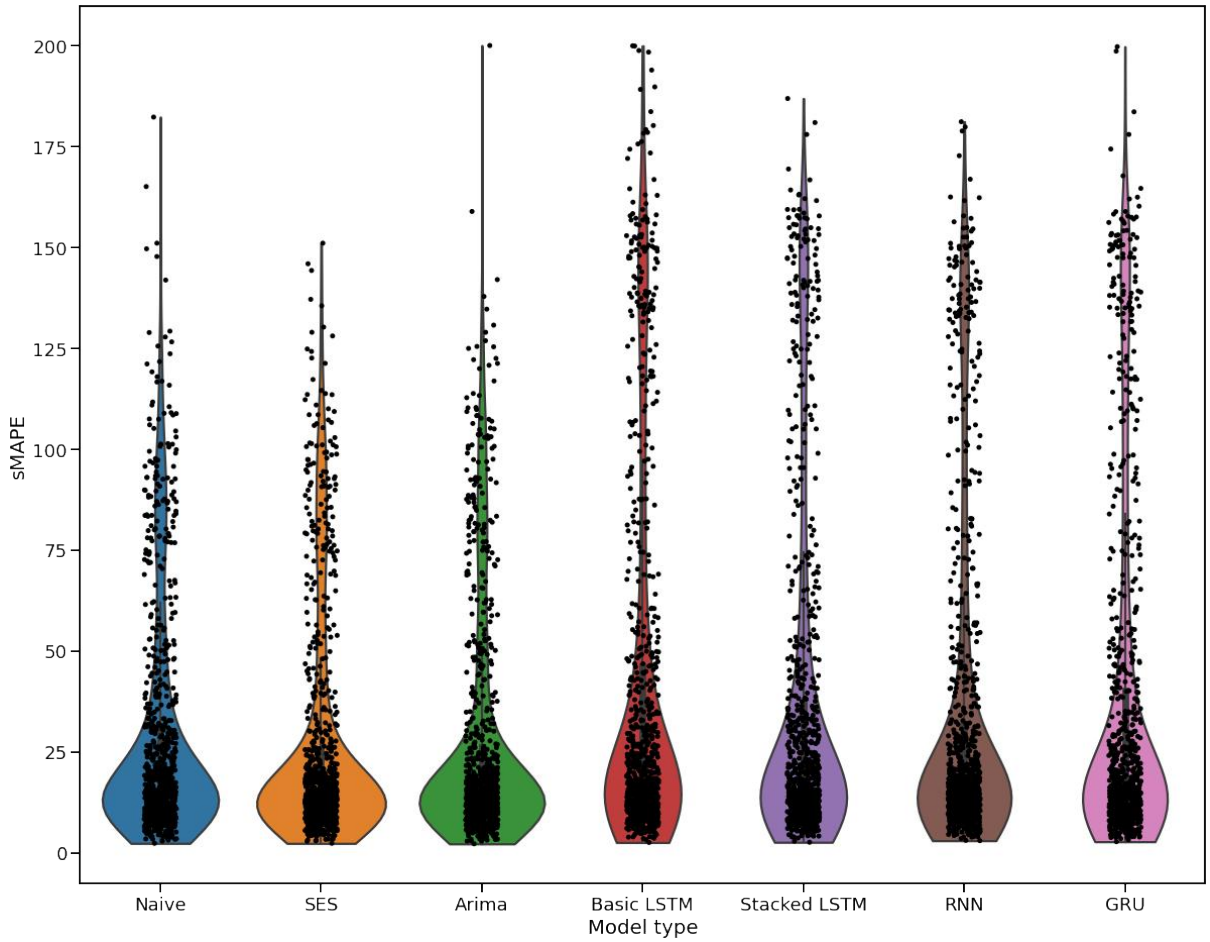


Figure 14: sMAPE distribution for each model

It is further of interest to analyse the frequency with which each model delivers the best forecasting results. Figure 15 illustrates that the naïve model, despite having a worse mean, median, and minimum sMAPE compared to the SES and ARIMA models (as seen in Table 4), delivers the best results the most frequently, with 250 out of 944 time series. This represents more than 25% of all observed items, suggesting that a significant number is not predictable. The SES model comes in second with 182 best predictions, followed by the ARIMA model with 178. These three models combined account for approximately 65% of the best predictions, indicating that the machine learning models lag significantly in performance.

Focusing on each model individually and comparing it to the naïve model, the results are consistent with the findings of the initial sMAPE Table 4. The SES and ARIMA models outperform the naïve model more than half the time, with 576 and 560 instances, respectively, approximately 60% of the time. Among the non-linear models, the RNN model performs the best, followed by the GRU and Stacked LSTM models, which have similar performance. In

general, each RNN-based methods only outperform the naïve method about 40% of the time. These findings further highlight the difficulties that both linear and non-linear models face in forecasting the price developments of these in-game items.

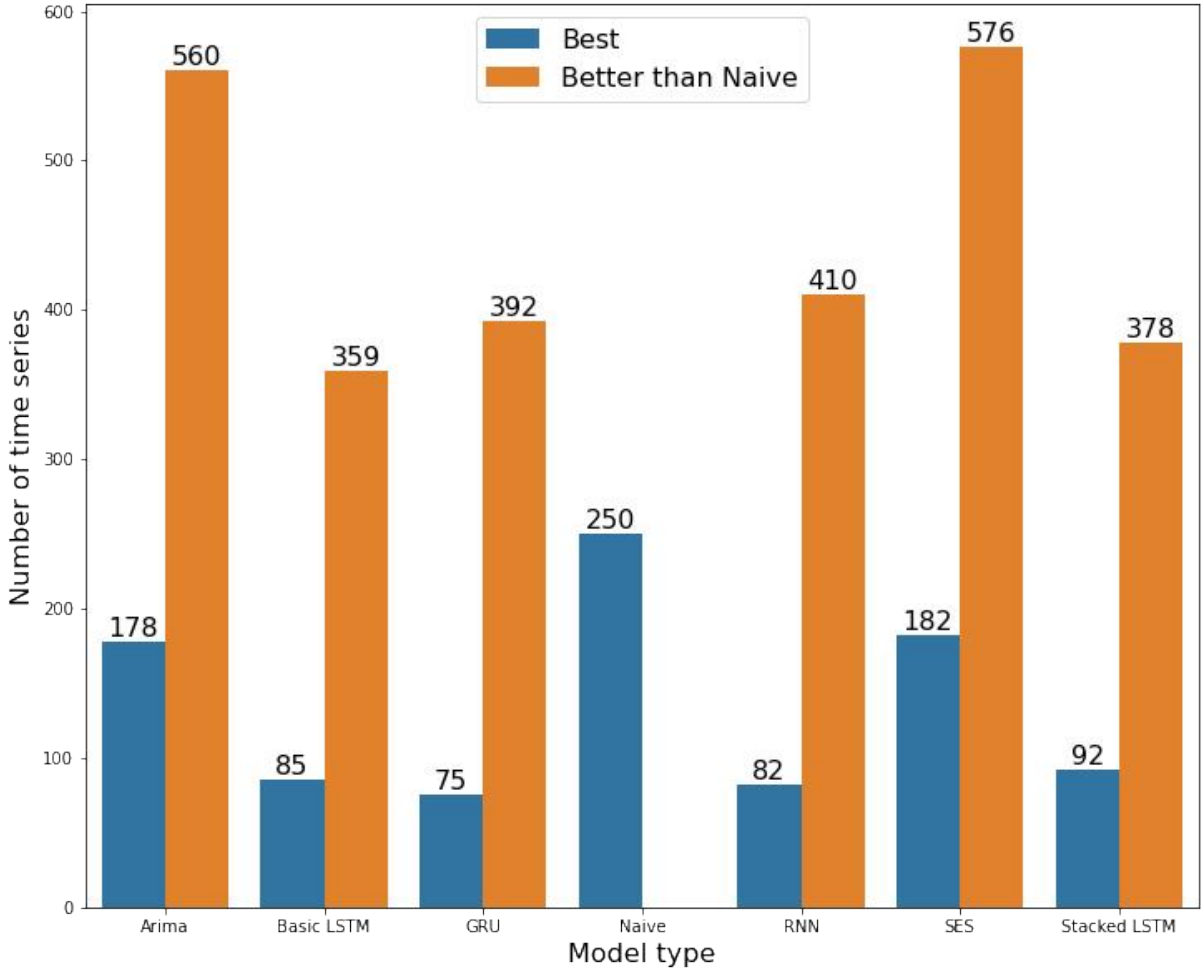


Figure 15: Count of best performances per model

Table 5 examines how each model performs against the naïve method as a benchmark, using the GMRAE error metric. A GMRAE value greater than one indicates that the model is worse than the benchmark. The mean GMRAE across all items shows that none of the models outperforms the naïve model. The rankings are similar to those observed for sMAPE, except for the GRU model, which performs worse than the stacked LSTM model. Therefore, SES is still the best model, followed by ARIMA and RNN. The basic LSTM model has an exceptionally high mean GMRAE, as does the GRU model. Examining the maximum value for each model reveals that these models have significant outliers compared to the other methods, which greatly influence their average performance. These high GMRAE errors stem from the

problem that GMRAE is vulnerable to outliers and very small numbers (Chapter 2.4.2) that can occur over the time horizon.

A different pattern emerges when considering the median error, which is less affected by outliers. Both linear models perform better than the naïve benchmark, with values of 0.976 and 0.977, respectively. This means that they slightly outperform the naïve model. However, the non-linear models are still ranked behind the benchmark, with the best performer being RNN, with a value of 1.096. The remaining methods maintain their order compared to the mean GMRAE.

According to the Friedman test, the differences in the models are statistically significant, with a p-value of approximately zero. The Wilcoxon shows that the mean ranks between the model pairs are all statistically significant at a 0.05 level, except the pair of SES and ARIMA with a p-value of approximately 0.261. This does not allow any clear conclusion on who performs better.

	SES	Arima	Basic LSTM	Stacked LSTM	RNN	GRU
min	0.104869	0.098316	0.067733	0.056073	0.086678	0.076939
max	164.878981	38.289466	55986.033238	274.318606	268.346621	2663.410594
mean	1.184606	1.121983	94.215845	1.833022	1.713472	5.668944
median	0.976498	0.977199	1.248062	1.159256	1.096152	1.171548
count	944.000000	943.000000	944.000000	944.000000	944.000000	944.000000

Table 5: GMRAE metrics overview

Overall, predicting the time series better than the naïve benchmark is very difficult, and only linear models achieve this more than half the time. The results will be further studied in the next part by dividing the dataset according to the stationarity of the underlying time series. The goal is to provide a more detailed view of when the naïve and the other models perform best.

4.2 Evaluation based on stationarity

Dividing the dataset by stationarity is important as stationary time series are typically easier to model and produce more reliable predictions. As previously elaborated, a time series is stationary if its statistical properties, such as the mean and variance, remain constant over time. Non-stationary time series, on the other hand, may exhibit trends, seasonality, or other changing

patterns that can complicate the modelling and forecasting process. The stationarity of a time series in this study is statistically assessed through the ADF test presented in Chapter 2.3.7.

First, I will examine the stationary time series presented in Table 6. The historical prices of 569 out of 944 items over the testing period are stationary in nature. For these stationary time series, the mean sMAPE values provide the same rankings as when the dataset is not divided, but all values are significantly lower, as expected. The drop in median performance is less significant. However, the accuracy of the models becomes more similar, as not only the RNN and GRU models outperform the naïve model but also the Stacked LSTM model.

Second, the forecasting ability of all models is considerably worse for non-stationary data, as the theory presented in Chapter 2.2 suggests and Table 7 shows. The mean performance rankings remained the same as for the entire dataset. Interestingly, for non-stationary data, where non-linear models are expected to outperform linear models, the median accuracy of these two groups is farther apart, and only the RNN model performs slightly better than the naïve model, unlike for stationary data, where both the GRU and Stacked LSTM models also outperformed the naïve model.

The Friedman test shows that the sMAPE differences in the stationary and non-stationary datasets are statistically significant, with p-values of approximately zero and 0.003. For the stationary time series, the pairs of the naïve and RNN models, SES and ARIMA models, and Basic and Stacked LSTM models are not statistically significant at a 0.05 level according to the Wilcoxon signed-rank test, with p-values of 0.119, 0.216, and 0.194, respectively. As the higher p-value of the Friedman test for the non-stationary compared to the stationary time series suggests, the pair difference is not that clean-cut here. None of the model pairs is statistically significant at the 0.05 level. However, most of the pairs have a p-value of around 0.06 except the combinations of naïve and SES (~0.144), naïve and ARIMA (~0.715), SES and RNN (~0.144), ARIMA and RNN (~0.144), Basic LSTM and Stacked LSTM (~0.144), and Stacked LSTM and GRU (~0.465).

index	Naive	SES	Arima	Basic LSTM	Stacked LSTM	RNN	GRU
min	2.341402	2.337488	2.219444	2.552327	2.609358	3.072906	2.749733
max	182.272162	145.930140	200.000000	199.923763	166.711103	162.463127	167.709176
mean	21.964606	19.144522	19.767684	30.030735	25.465479	23.535953	26.170003
median	14.438901	13.069624	13.236588	15.691823	14.313660	13.983905	14.044282
count	569.000000	569.000000	568.000000	569.000000	569.000000	569.000000	569.000000

Table 6: sMAPE metrics for stationary time series

index	Naive	SES	Arima	Basic LSTM	Stacked LSTM	RNN	GRU
min	3.361379	3.315104	3.338186	3.871993	4.137513	3.004278	3.782814
max	151.034180	151.034180	158.887529	199.872996	186.871751	181.119041	199.704752
mean	38.121781	37.043265	37.837296	56.696427	52.180115	49.802953	52.962741
median	21.371991	17.949437	18.314135	26.923866	23.860477	21.197303	23.087453
count	375.000000	375.000000	375.000000	375.000000	375.000000	375.000000	375.000000

Table 7: sMAPE metrics for non-stationary time series

Since the models in these two groups behave similarly to each other, as previously noted, they have been combined for this part of the analysis. Figure 16 below, a combination of a violin and strip plot, represents the sMAPE distribution for the naïve, linear, and non-linear models for the test dataset's stationary and non-stationary time series. Of the 6608 data points, 2625 belong to the non-stationary dataset, and 3983 belong to the stationary dataset. Generally, for all model types, the sMAPE errors are more widely spread for the non-stationary time series than for the stationary time series, visible in the more elongated body shape of the non-stationary violins. This is expected, given the increased difficulty in predicting the former data type. The decrease in the occurrence of sMAPE with its increase in strength, previously noted in Figure 14, applies especially to stationary data for the naïve, linear models and is also somewhat accurate for non-linear models for this data type. In contrast, the previously identified grouping of errors for non-linear models in non-stationary data at the higher ends of the sMAPE range in the general model (Figure 14) is particularly pronounced in non-linear time series. This pattern is also visible for the benchmark method and linear models, suggesting that these time series are either very easy or hard to predict.

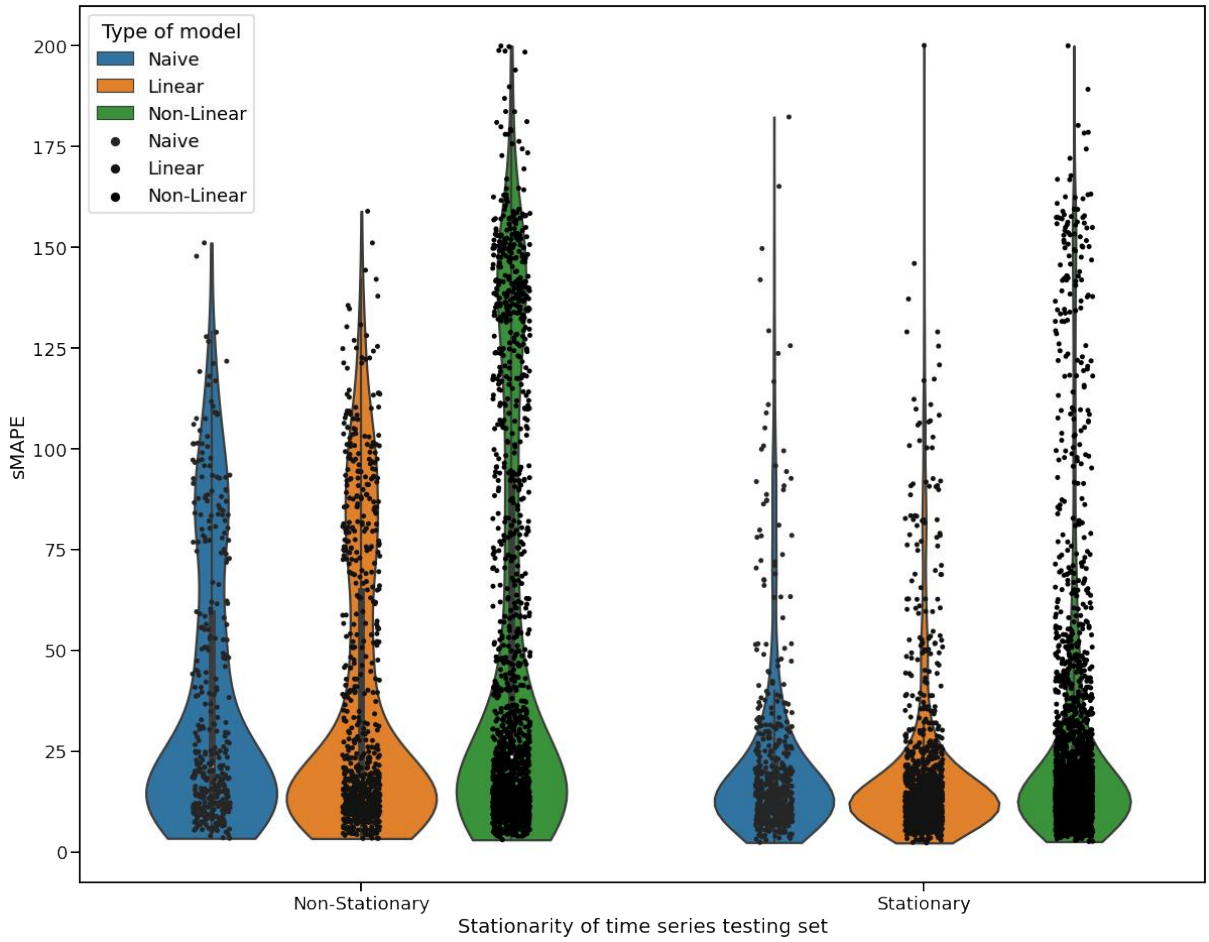


Figure 16: sMAPE distribution for different stationarity

Tables 8 and 9 present the GMRAE error metrics for stationary and non-stationary time series, respectively. The median GMRAE errors of all models are significantly lower overall, and some are even better than the naïve model for stationary time series compared to non-stationary. The SES and ARIMA models outperform the naïve model for the stationary data, but this is not the case for the RNN, GRU, and Stacked LSTM models, which performed better than the benchmark when sMAPE was considered. For non-stationary time series, the SES and ARIMA methods perform similarly to the naïve model, while the machine learning methods show significantly worse performance.

	SES	Arima	Basic LSTM	Stacked LSTM	RNN	GRU
min	0.104869	0.098316	0.067733	0.056073	0.086678	0.076939
max	164.878981	38.289466	55986.033238	274.318606	268.346621	272.059055
mean	1.288983	1.135830	102.289312	1.973774	1.857527	2.268110
median	0.948782	0.958094	1.205422	1.109419	1.049147	1.087897
count	569.000000	568.000000	569.000000	569.000000	569.000000	569.000000

Table 8: GMRAE error metrics for stationary time series

	SES	Arima	Basic LSTM	Stacked LSTM	RNN	GRU
min	0.156547	0.144411	0.085570	0.118333	0.165774	0.091190
max	5.015531	10.331493	28600.461866	13.913975	26.600352	2663.410594
mean	1.026232	1.101011	81.965705	1.619454	1.494892	10.829143
median	0.990962	0.993766	1.366780	1.276099	1.180641	1.247351
count	375.000000	375.000000	375.000000	375.000000	375.000000	375.000000

Table 9: GMRAE error metrics for non-stationary time series

4.3 Evaluation of the time horizon

Thus far, the sMAPE has always been calculated over the entire predicted time horizon. This chapter divides the time horizon into non-overlapping 2-datapoint long observation windows that move across all 30 forecasting data points. The goal of this analysis is to evaluate the differences in the performance of each model as the prediction horizon extends further into the future, also considering the performance of other models. The theory presented in Chapter 2.2 states that the accuracy of every model decreases with the increasing prediction horizon. Additionally, non-linear models should perform better than linear models for data points farther from the training data.

Figure 17 illustrates the mean sMAPE over the previously mentioned period. The mean accuracy of all presented models decreases steadily without any notable outliers. Additionally, the linear and machine learning models maintain a constant performance difference among each other over the entire time frame, with the linear models performing better throughout the observation period. However, the slope of all the simpler models is steeper than the non-linear ones, leading to them almost losing their sMAPE gap after 30 data points. This graph suggests that if the prediction horizon were longer, the non-linear models would soon outperform the linear models after these 30 data points.

Among the linear models, the best-performing model is the SES model, followed by the ARIMA model and the benchmark model. This ranking order remains unchanged throughout the entire time frame. For the non-linear models, the rankings changed somewhat. The GRU model had a better sMAPE in the first two data points than the Stacked LSTM model but performed worse for the rest of the time horizon. However, these two models were only the

second and third best models, respectively, as the RNN model was the most accurate over the 30 days. Nonetheless, the RNN model's gap towards the Stacked LSTM model narrowed towards the end, and the Basic LSTM model is far behind in last place.

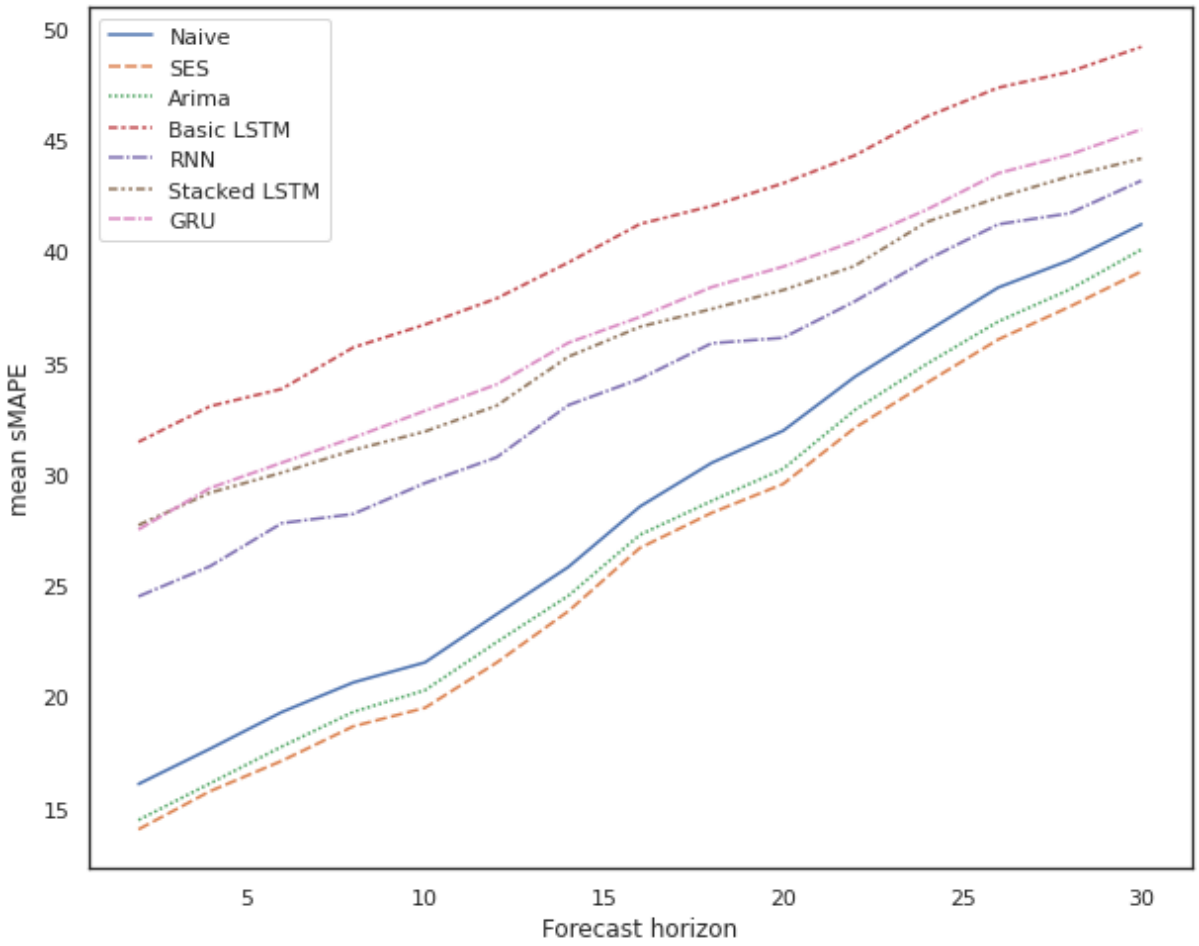


Figure 17: mean sMAPE per model over the forecast horizon

A different picture emerges when examining the median performance of all models, as shown in Figure 18. The naïve method, previously consistently the third-best method throughout the period, is now only at this rank for the first 12 days while experiencing the highest growth in median sMAPE errors during this observation window, which causes it to be surpassed by the RNN model. This development is only reversed for two data points towards the end. In addition, the accuracy of the GRU model also exceeds that of the naïve model after 16 days, and the Stacked LSTM model briefly surpasses it for a two-data point period. The GRU model also challenges the RNN model for some parts of the time horizon but can never entirely surpass its performance. The sMAPE ranking of the top two models and the worst model from the previously evaluated mean sMAPE remains the same, with only the ARIMA model sharing a

similar performance to the RNN model for the last four observations. Additionally, the accuracy gap between the linear models and the naïve model reduces significantly towards the end of the time horizon. Generally, the slopes for all models are not as clean as those of the mean sMAPE but are growing over time.

The Friedman test confirms that the mean and median sMAPE errors over the time horizon are statistically significant at a 0.05 level, with p-values of approximately zero for both. Additionally, the results of the Wilcoxon test show that the mean sMAPE for all pairs is different at a significance level of 0.05. A similar pattern is observed for the median sMAPE, except for the pairs naïve and RNN, and naïve and GRU, which have p-values of approximately 0.847 and 0.277, respectively. These results suggest that the median sMAPE errors for these pairs are not statistically different, even at a 0.10 level.

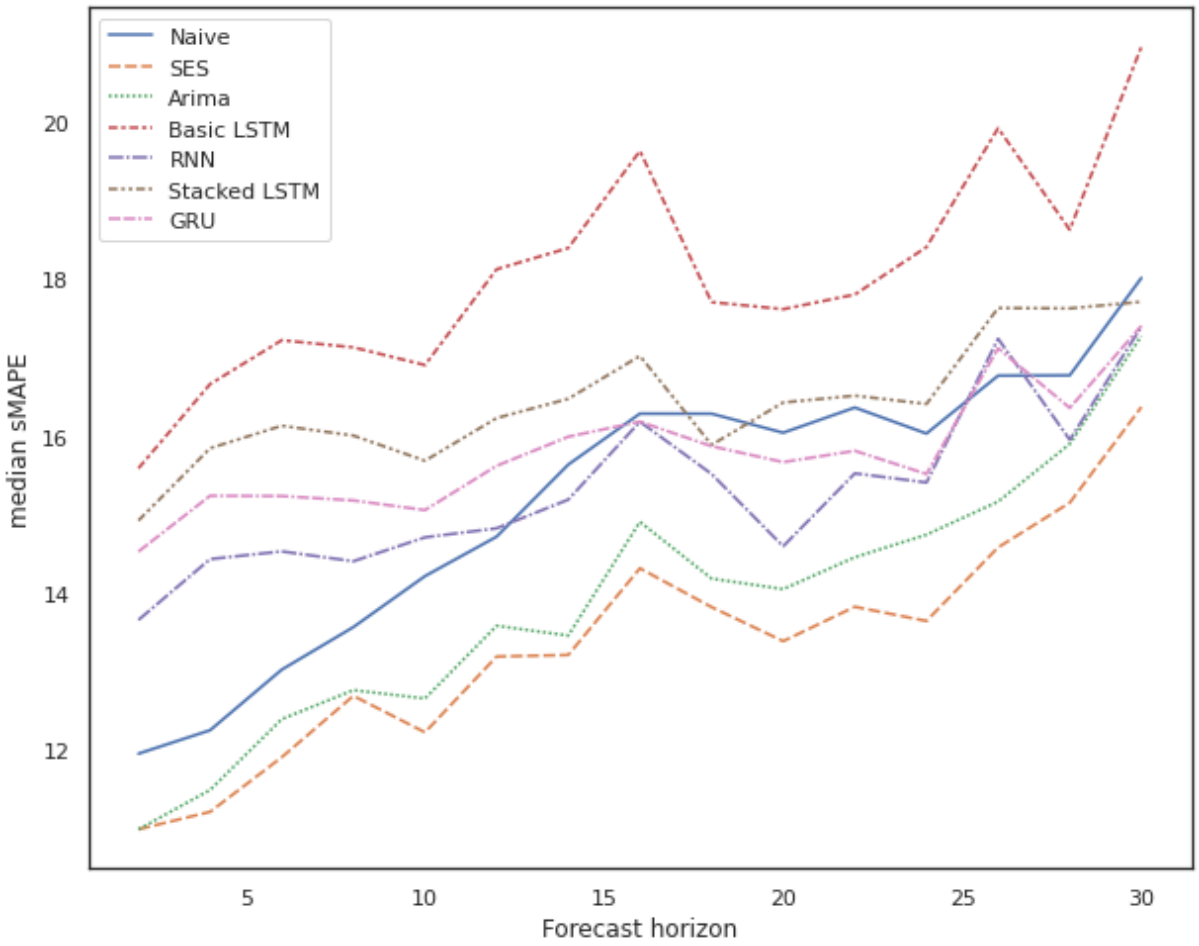


Figure 18: median sMAPE per model over the forecast horizon

The analysis of GMRAE in a previous section of this study revealed that the error metrics produced by this analysis contained outliers. As a result, only the median GMRAE was considered and depicted in Figure 19 for the time horizon analysis. The linear models presented

in blue (SES) and orange (ARIMA) are initially very close to one, which represents the benchmark of the time series. However, as the time horizon progresses, both models move and remain below one, indicating that they were superior to the naïve model throughout the entire period. Specifically, the SES model constantly improves over the first ten days, with the lowest median GMRAE of approximately 0.933 at day 10. The ARIMA model displays its lowest median GMRAE on day 12, with a value of approximately 0.959. After this point, both models move closer to one again, suggesting that they are particularly effective at outperforming the benchmark in the short term.

The median GMRAE for non-linear models falls within the range of 1.2 to 1.5 during the initial two days and remains relatively high for the subsequent 15 to 20 days. This suggests that these models exhibit inferior forecasting performance compared to the naïve and linear models in the short term. However, the performance of non-linear models gradually improves over the time series and approaches the benchmark of a GMRAE value of 1, although this benchmark is not reached within the observed time frame. These findings are consistent with the results obtained from the mean sMAPE. The RNN model displayed the best performance among the machine learning models examined, followed by the GRU and Stacked LSTM models. The Friedman and the Wilcoxon for every pair are statistically significant at a 0.05 level.

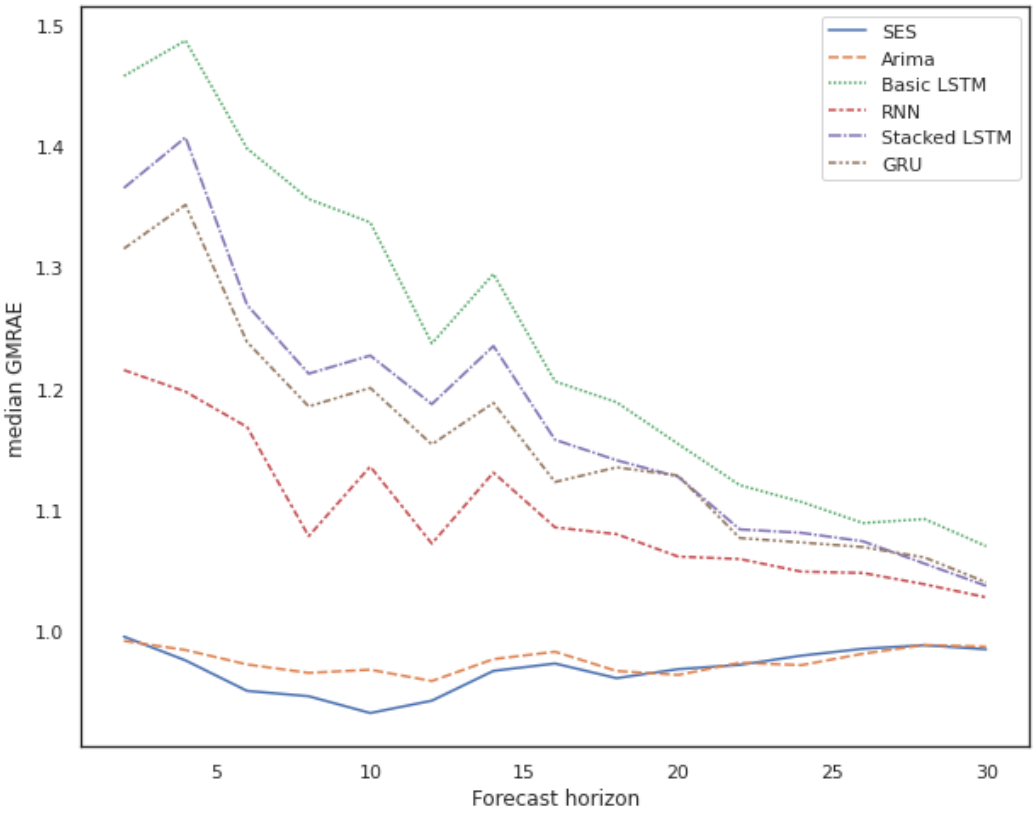


Figure 19: mean sMAPE per model over the forecast horizon

5. Conclusion

The motivation for this study was to gain insights into the price predictability of in-game items on digital marketplaces and evaluate different forecasting models based on their performance. To achieve this, the study is based on a multitude of perspectives, two error metrics, and a dataset with 944 individual time series, which varied in terms of the length of individual time series, price, and stationarity. All this allows me to make general conclusions based on this diverse range of time series characteristics and applied methods.

This study shows that the price is predictable to some degree for many items, although the improvement is small compared to the naïve benchmark. While the sMAPE and GMRAE results of the overall dataset indicate that linear models perform better than the naive method, the naive model still ranks as the top method most of the time when the models are compared in a group. However, it had a slightly lower probability of outperforming the linear models when compared individually. This suggests that multiple methods may be required to predict the time series better than a random walk consistently. This leads to the problem that there is no clear evidence to know which methods are favourable in certain events.

Nevertheless, the stationary time series analysis results suggest an overall higher level of predictability for these items as large forecasting errors, particularly for non-linear models, often occur with non-stationary data. The median sMAPE results for stationary time series show that all models except the Basic LSTM model outperform the naive benchmark. However, this was not supported by the median GMRAE results, which indicate that only the linear models outperform the naive model in predicting the time horizon for stationary data.

A mean sMAPE and median GMRAE analysis of smaller time steps indicate that linear models significantly outperform the benchmark in predicting shorter time periods and also deliver better accuracy over the remaining time frame. None of the non-linear models can achieve this, although the RNN and GRU models show better results than the benchmark in terms of median sMAPE after 12 data points and reduce the mean sMAPE and median GMRAE gap towards the other models significantly towards the end of the prediction horizon.

Furthermore, it can be generally concluded that the linear models outperform the non-linear models in every examined aspect. SES has the upper hand for linear models, though not always statistically significant, and for non-linear methods, the RNN model was the best at predicting prices. The study's results suggest that LSTM models, which possess a specialised structure for effectively remembering long-term dependencies in data, perform worse than the simpler RNN

model. Moreover, linear methods outperform both LSTM and RNN models. These two findings point in the direction that past price points do not contain a wealth of significant additional information and patterns. Finally, it can be concluded that forecasting digital markets is at least as challenging as any other financial market, however, this study delivers some support for the critics of the efficient market hypothesis.

6. Limitations and future directions

The results of this study are limited to a univariate context, which benefits linear methods as machine learning models cannot show their full strength. One of their main advantages is their ability to make decisions based on multiple variables. Furthermore, these non-linear models are not necessarily adapted to the specific "needs" of each time series and have the same basic settings for each one of them. This basic setting is another limitation, as more layers could have better reflected the data's complexity and possibly picked up price patterns that are not recognised in this study. It is worth noting that more sophisticated models with more optimised structures may potentially yield even better results in cases where more computational resources are available, and the focus lies on only one model that can be optimised. These are, in my opinion, the main future research targets regarding modelling for time series forecasting of digital marketplaces. Additionally, alternative approaches, such as constructing a global model for predicting all time series, utilising hybrid methods that combine multiple forecasting techniques, or using classification models that predict the direction of price movement, could also be evaluated.

Future research could expand upon the dataset by considering marketplaces of in-game items of different games, considering different items from Counter-Strike: Global Offensive, utilising a different time horizon, or even multiple time horizons, as the findings of this study are specific to only one particular period.

7. References

Abu-Mostafa, Y. S., & Atiya, A. E. (1996). Introduction to Financial Forecasting. *Applied Intelligence*, *6*, 205–213. https://doi.org/10.1007/BF00126626

Ahmed, N. K., Atiya, A. F., el Gayar, N., & El-Shishiny, H. (2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews*, *29*(6), 594–621. https://doi.org/10.1080/07474938.2010.481556

Armstrong, J. S., & Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, *8*(1), 69–80. https://doi.org/10.1016/0169-2070(92)90008-W

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques - Part II: Soft computing methods. *Expert Systems with Applications*, *36*(3 PART 2), 5932–5941. https://doi.org/10.1016/j.eswa.2008.07.006

Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, *19*(8), 1165–1195. https://doi.org/10.1007/s00521-010-0362-z

Bao, Y., Xiong, T., & Hu, Z. (2014). Multi-step-ahead time series prediction using multiple-output support vector regression. *Neurocomputing*, *129*, 482–493. https://doi.org/10.1016/j.neucom.2013.09.010

Bontempi, G., ben Taieb, S., & le Borgne, Y.-A. (2013). Machine Learning Strategies for Time Series Forecasting. In E. Aufaure Marie-Aude and Zimányi (Ed.), *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures* (pp. 62–77). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36318-4_3

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control: Vol. 3rd Edition*.

Brame. (2021). *Commercialization of the gaming industry | Brame*. https://brame.io/blog/commercialization-of-the-gaming-industry/

Brown, R. G., & Meyer, R. F. (1961). The Fundamental Theorem of Exponential Smoothing. *Https://Doi.Org/10.1287/Opre.9.5.673*, *9*(5), 673–685. https://doi.org/10.1287/OPRE.9.5.673

Chen, C., Twycross, J., & Garibaldi, J. M. (2017). A new accuracy measure based on bounded relative error for time series forecasting. *PloS One*, *12*(3), e0174202. https://doi.org/10.1371/JOURNAL.PONE.0174202

Chollet, F. and others. (2015a). *Keras EarlyStopping*. GitHub. https://keras.io/api/callbacks/early_stopping/

Chollet, F. and others. (2015b). *Keras Recurrent layers*. GitHub. https://keras.io/api/layers/recurrent_layers/

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. https://doi.org/10.48550/arxiv.1412.3555

Cleghorn, J., & Griffiths, M. (2015). Why do gamers buy "virtual assets"? An insight in to the psychology behind purchase behaviour. *Digital Education Review*, *27*, 85–104.

Connor, J. T., Martin, R. D., & Atlas, L. E. (1994). Recurrent Neural Networks and Robust Time Series Prediction. *IEEE Transactions on Neural Networks*, *5*(2), 240–254. https://doi.org/10.1109/72.279188

Counter-Strike Blog. (2013). *Counter-Strike: Global Offensive » 8/13 – The Arms Deal Update*. https://blog.counter-strike.net/index.php/2013/08/7425/

Crone, S. F., Hibon, M., & Nikolopoulos, K. (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, *27*(3), 635–660. https://doi.org/10.1016/j.ijforecast.2011.04.001

CSGOSKINS.GG. (2022). *How The Weekly CS:GO Drop System Works - CSGOSKINS.GG*. https://csgoskins.gg/blog/how-the-weekly-csgo-drop-system-works

de Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, *22*(3), 443–473. https://doi.org/10.1016/j.ijforecast.2006.01.001

Dickey, D. A., & Fuller, W. A. (2012). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Https://Doi.Org/10.1080/01621459.1979.10482531*, *74*(366a), 427–431. https://doi.org/10.1080/01621459.1979.10482531

dProgrammer lopez. (2020). *RNN, LSTM & GRU*. http://dprogrammer.org/rnn-lstm-gru

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/S15516709COG1402_1

Engle, R. F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, *50*(4), 987. https://doi.org/10.2307/1912773

Fama, E. F. (1995). Random Walks in Stock Market Prices. *Financial Analysts Journal*, *51*(1). https://doi.org/10.2469/faj.v51.n1.1861

Fildes, R. (1992). The evaluation of extrapolative forecasting methods. *International Journal of Forecasting*, *8*(1), 81–98. https://doi.org/10.1016/0169-2070(92)90009-X

Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, *32*(200), 675–701. https://doi.org/10.1080/01621459.1937.10503522

Gandhmal, D. P., & Kumar, K. (2019). Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, *34*, 100190. https://doi.org/10.1016/j.cosrev.2019.08.001

Goodwin, P., & Lawton, R. (1999). On the asymmetry of the symmetric MAPE. *International Journal of Forecasting*, *15*(4), 405–408. https://doi.org/10.1016/S0169-2070(99)00007-2

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947

Hamari, J., Alha, K., Järvelä, S., Kivikangas, J. M., Koivisto, J., & Paavilainen, J. (2017). Why do players buy in-game content? An empirical study on concrete purchase motivations. *Computers in Human Behavior*, *68*, 538–546. https://doi.org/10.1016/J.CHB.2016.11.045

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting*, *37*, 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688. https://doi.org/10.1016/J.IJFORECAST.2006.03.001

Jabeur, S. ben, Mefteh-Wali, S., & Viviani, J. L. (2021). Forecasting gold price with the XGBoost algorithm and SHAP interaction values. *Annals of Operations Research*. https://doi.org/10.1007/s10479-021-04187-w

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Inc.

Jiang, W. (2021). Applications of deep learning in stock market prediction: Recent progress . *Expert Systems With Applications*, *184*, 957–4174. https://doi.org/10.1016/j.eswa.2021.115537

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Li, Y., & Ma, W. (2010). Applications of artificial neural networks in financial economics: A survey. *Proceedings - 2010 International Symposium on Computational Intelligence and Design, ISCID 2010*, *1*, 211–214. https://doi.org/10.1109/ISCID.2010.70

Makwana, P., Kodinariya, T. M., & Makwana, P. R. (2013). Review on Determining of Cluster in K-means Clustering Review on determining number of Cluster in K-Means Clustering. *International Journal of Advance Research in Computer Science and Management Studies*, *1*(6). https://www.researchgate.net/publication/313554124

Malkiel, B. G. (2003). The Efficient Market Hypothesis and Its Critics. *Journal of Economic Perspectives*, *17*(1), 59–82. https://doi.org/10.1257/089533003321164958

Malkiel, B. G., & Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383–417. https://doi.org/10.1111/j.1540-6261.1970.tb00518.x

Murat Ozbayoglu, A., Gudelek, U., & Sezer, O. B. (2020). Deep learning for financial applications : A survey. *Applied Soft Computing Journal*, *93*, 106384. https://doi.org/10.1016/j.asoc.2020.106384

Muth, J. F. (1960). Optimal Properties of Exponentially Weighted Forecasts. *Journal of the American Statistical Association*, *55*(290), 299–306. https://doi.org/10.1080/01621459.1960.10482064

Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & Shahab, S. (2020). Deep learning for stock market prediction. *Entropy*, *22*(8). https://doi.org/10.3390/E22080840

Newzoo. (2022). *Newzoo Global Games Market Report 2022*. https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2022-free-version

Nofsinger, J. R. (2005). Social Mood and Financial Economics. *Journal of Behavioral Finance*, *6*(3), 144–160. https://doi.org/10.1207/s15427579jpfm0603_4

OpenSea. (2022). *OpenSea, the largest NFT marketplace*. https://opensea.io/

Ostertagova, E., & Ostertag, O. (2011). The simple exponential smoothing model. *He 4th International Conference on Modelling of Mechanical and Mechatronic Systems, Technical University of Košice, Slovak Republic, Proceedings of Conference*, 380–384. https://www.researchgate.net/publication/256088917

Palit, A. K., & Popovic, D. (2005). *Computational intelligence in time series forecasting: Theory and engineering applications (Advances in industrial control)*. Springer-Verlag. https://doi.org/10.1007/1-84628-184-9

PCGamer. (2015). *How $400 virtual knives saved Counter-Strike | PC Gamer*. https://www.pcgamer.com/how-400-virtual-knives-saved-counter-strike/

PCGamesN. (2022). *Steam just reached 50,000 total games listed | PCGamesN*. https://www.pcgamesn.com/steam/total-games

SAS & IIF. (2006). *NN3 - Neural Forecasting Competition* . http://www.neural-forecasting-competition.com/NN3/

Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://scikit-learn.org/stable/modules/clustering.html#k-means

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., & Cournapeau, D. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Sethia, A., & Raut, P. (2019). Application of LSTM, GRU and ICA for stock price prediction. *Smart Innovation, Systems and Technologies*, *107*, 479–487. https://doi.org/10.1007/978-981-13-1747-7_46

Sezer, O. B., Gudelek, U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005-2019. *Applied Soft Computing Journal*, *90*, 106181. https://doi.org/10.1016/j.asoc.2020.106181

Shcherbakov, M., & Tyukov, A. (2013). A Survey of Forecast Error Measures. *World Applied Sciences Journal 24 (Information Technologies in Modern Industry, Education & Society)*, 171–176. https://doi.org/10.5829/idosi.wasj.2013.24.itmies.80032

Sheldon, M. R., Fillyaw, M. J., & Thompson, W. D. (1996). The use and interpretation of the Friedman test in the analysis of ordinal-scale data in repeated measures designs. *Physiotherapy Research International : The Journal for Researchers and Clinicians in Physical Therapy*, *1*(4), 221–228. https://doi.org/10.1002/PRI.66

Smith, T. (2017). *pmdarima.arima.auto_arima — pmdarima 2.0.2 documentation*. https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html

Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., & Lendasse, A. (2007). Methodology for long-term prediction of time series. *Neurocomputing*, *70*, 2861–2869. https://doi.org/10.1016/j.neucom.2006.06.015

statsmodel. (2022). *SES — statsmodels*. https://www.statsmodels.org/dev/examples/notebooks/generated/exponential_smoothing.html

Steam. (2022a). *Counter-Strike: Global Offensive on Steam*. https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/

Steam. (2022b). *Steam Charts*. https://store.steampowered.com/charts/

Steam. (2022c). *Steam Community : Steam Community Market*. https://steamcommunity.com/market/

Steam Charts. (2022). *Steam Charts - Tracking What's Played*. https://steamcharts.com/top

Steam Support. (2022). *Steam Support: Community Market FAQ*. https://help.steampowered.com/en/faqs/view/61F0-72B7-9A18-C70B

Stock, J. H., & Watson, M. W. (1998). A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series. *National Bureau of Economic Research*.

StockX. (2022). *StockX: Sneaker, Streetwear, Sammelkarten, Handtaschen, Uhren*. https://stockx.com/de-de

Swanson, N. R., & White, H. (1997). Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models. *International Journal of Forecasting*, *13*, 439–461.

Taieb, B. S., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, *39*, 7067–7083. https://doi.org/10.1016/j.eswa.2012.01.039

Tay, F. E. H., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, *29*, 309–317. www.elsevier.com/locate/dsw

Tiao, G. C., & Tsay, R. S. (1994). Some Advances in Non-linear and Adaptive Modelling in Time-series ARMA model Adaptive forecasting Fractional difference Gibbs sampler Long-memory Outlier Random variance-shift model Threshold autoregressive model. In *Journal of Forecasting* (Vol. 13).

Vafaeipour, M., Rahbari, O., Rosen, M. A., Fazelpour, F., & Ansarirad, P. (2014). Application of sliding window technique for prediction of wind velocity time series. *International Journal of Energy and Environmental Engineering*, *5*(2–3), 1–7. https://doi.org/10.1007/S40095-014-0105-5/FIGURES/7

Visalakshi, N. K., & Kuttiyannan, Dr. T. (2009). Impact of normalization in distributed K-means clustering. *International Journal of Soft Computing*, *4*, 168–172.

Wilcoxon, F. (1992). *Individual Comparisons by Ranking Methods*. 196–202. https://doi.org/10.1007/978-1-4612-4380-9_16

Williams, R. J., & Zipser, D. (1989). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, *1*(2), 270–280. https://doi.org/10.1162/NECO.1989.1.2.270

Xiong, T., Bao, Y., & Hu, Z. (2013). Beyond one-step-ahead forecasting: Evaluation of alternative multi-step-ahead forecasting models for crude oil prices. *Energy Economics*, *40*, 405–415. https://doi.org/10.1016/j.eneco.2013.07.028

Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019). A comparison between ARIMA, LSTM, and GRU for time series forecasting. *ACM International Conference Proceeding Series*, *7*, 49–55. https://doi.org/10.1145/3377713.3377722

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. In *Neurocomputing* (Vol. 50). www.elsevier.com/locate/neucom