LUISS

Department of

Business and Management

Bachelor's Degree in

Management and Computer Science

Course of Data Analysis for Business

Non-Fungible Tokens and Virtual Worlds: An Analysis on the Trading in Primary and Secondary Marketplaces

Prof. Francesco Iafrate

SUPERVISOR

Costanza Placanica - 248551

CANDIDATE

Academic Year 2021/2022

Abstract

In our fully digitalized world, assets such as NFTs are seeping more and more into our daily lives and understanding how to profit from them is now of utmost importance, both for business and for individuals. In this study, I will attempt to provide some answers regarding how they work, what can they be used for and, most importantly, which of them will be worth investing in, always relating to the category in which the NFTs I worked with could be found in, the metaverses. Indeed, with the aid of four different kinds of analysis, I will study the NFTs from different perspectives, in order to try and have the most comprehensive vision of this topic.
I will provide my considerations and hypothesis on the various patterns and results, and then discuss about the possible futures of NFTs and metaverses, which is still very complicated to predict.

List of Figures

Table of Contents

CHAPTER 1

Metaverses and Non-Fungible Tokens

In 1992, Neil Stephenson, an American writer, published his novel *Snow Crash* and it was the first time the world read about the *Metaverse*, a word that comes from the blending of meta, prefix with the meaning of transcending, and universe.

In Stephenson's novel, the *Metaverse* is regarded as a successor to the Internet and is a three-dimensional virtual place where, through user-created avatars, humans interact with each other, in a simulation of the real world[1].

Nowadays, virtual worlds like the one described by Stephenson are widely well known and many versions are available on the Internet, but in these recent years, aided by the Covid-19 pandemic, they have been making a huge comeback.

In fact, while the world was in lockdown, the sales of video gaming consoles, Virtual Reality devices, and the presence on social virtual worlds skyrocketed[23], which consequently brought more awareness on such topics to the general population.

At the same time, another, new kind of concept was getting a lot of attention: the NFTs.

The history of NFTs actually started in 2012, when *Colored Coins*, which are denominations of bitcoin and can be used to represent various series of assets, were first introduced. The creation of *Colored Coins* made many people realize the potential of distributing assets onto the blockchain, even though at the time bitcoin could not handle this kind of feature.

Then, in 2014, *Counterparty*[4], a peer-to-peer financial platform and an open source protocol, was founded and built on the bitcoin blockchain: it was an innovation in the sense that it allowed the creation of assets and, moreover, had a decentralized exchange platform.

In 2015, a game called *Spells of Genesis* issued its in-game assets onto the blockchain via *Counterparty*, and, in 2016, a popular card game, *Force of Will*, also launched its cards on the platform, signaling the start of mainstream companies (which up until then had no prior relationship with cryptocurrencies) issuing their assets on the blockchain.

In the meantime, another platform, called Ethereum, was gaining momentum: in Ethereum, developers were allowed to implement smart contracts, which are connected to digital assets and confirm the asset is unique, traceable, and verifiable.

In 2017, a particular type of meme started to be traded on Ethereum, the *Rare Pepe Meme Directory*[5] and as it was becoming more popular, John Watkinson and Matt Hall, two mobile

developers, decided to create a set of NFTs: unique characters, by the name of *Cryptopunks*[6], generated on the Ethereum blockchain, which were immediately claimed and sold again in a secondary marketplace.

In the same year, another popular collection, *Cryptokitties*[7], was released. *Cryptokitties,* which is a virtual game based on the blockchain in which users take care of virtual cats, opened people's eyes to the power and potential of non-fungible tokens.

In the following years, 2018 and 2019, the NFT ecosystem experienced a massive growth, and NFTs started to be included in virtual worlds, games, art, and even music.

However, what are NFTs exactly? For many years, the digital content available online was in general free of charge and easily duplicated, so that its authenticity was difficult to prove.

In particular, the owners of said content, if they wanted to profit from it, had the difficult task of verifying their ownership, while buyers had problems in finding authentic digital content to purchase online.

Nowadays, though, the blockchain and the NFTs changed the situation.

NFTs, or Non-Fungible Tokens, are tokens that are used to represent unique items, which are written in a *smart contract* (i.e. string of codes recorded in a decentralized ledger in the blockchain)[8]. Specifically, in the economy field, the words "non-fungible" mean items which are not interchangeable with anything, exactly because they have unique properties (for comparison, fungible items can be exchanged because they are defined by their value and not by their unique properties).

Further, NFTs can have only one official owner at a time and, being secured and traceable on the blockchain, no one can modify the record of ownership or copy an NFT[9].

For this reason, NFTs revolutionized the concept of digital content, for better or for worse. For instance, there are studies that consider NFTs to be the art's downfall, as in Frye (2021)[10].

Moreover, in studies such as *Crypto-marketing: how non-fungible tokens (NFTs) challenge traditional marketing*[11], the uniqueness of NFTs has been questioned, as many collections of NFTs (e.g. CryptoPunks) feature thousands of items that look almost identical to each other.

NFTs, however, are much more than "simple" art content. In recent years they expanded in the video-gaming world, but, more importantly, in the metaverses.

Nowadays metaverses are much more embedded in the blockchain than, of course, how Stephenson would have imagined: indeed, a recent study[12], mentions the "Crypto metaverses", where the blockchain supports both the economy, technology, and commerce. These kinds of metaverses are becoming increasingly more popular, and, as a consequence, can now include NFT applications or marketplaces to trade assets and virtual properties, such as land parcels.

The new crypto niche: NFTs, play-to-earn, and metaverse tokens[12] also highlights that in these metaverses, traders can invest in projects without even playing games or actually interacting with the surrounding world, but only to buy tokens that they believe might be profitable to resell. This means that, from a pure business approach, the identification of which NFTs are worth investing in, just like real-life stock, is crucial. Moreover, this would also lead to more companies investing in projects to transfer some of their assets to the metaverses, as some are already doing, especially in the fashion industry (e.g. Gucci)[13].

On this note, I must mention a particularly relevant aspect of the metaverses which is the ownership of land. I will take as example one of the most famous metaverses nowadays, Decentraland, which I will describe better in the next chapter: as Dowling reports in *Fertile LAND: Pricing non-fungible tokens*, in order to become a citizen in *Decentraland* all you have to do is to buy land, or more specifically, LAND. LAND is a coded piece of the virtual world, which you can buy and sell freely, and any change of ownership or money is recorded in an Ethereum smart contract[14]. When you buy it, you can use it as you prefer, maybe turn it into a shop or a venue, or perhaps build a villa and increase its value, then resell it.

There is one issue, however, that has been identified by Dowling. As the online markets for Non-Fungible Tokens are in their early stage, there are inefficiencies in the pricing of the tokens, which is often subjected to speculation or market manipulation resulting in difficulties in predicting it.

For this reason, I preferred to focus my analysis on another important issue: the primary and secondary sales in the trading of NFTs.

Primary sales are the sales between the creator and the collector, while secondary sales are between the buyer, or investor, to another buyer.

In marketplaces such as OpenSea, primary sales and secondary sales are both permitted, as users can create and sell their content on the platform.

Secondary sales are crucial both for the creator and for the investor/buyer. In fact, from the creator's point of view, in most marketplaces of NFTs the creator sets the royalties during the minting process (the process in which digital content is turned into a digital asset and stored on the blockchain): these royalties give a percentage of the sales to the creator each time their NFTs are sold on the marketplace, meaning that the creator will get a percentage of the profits for each time their NFTs will be sold in secondary sales. These royalties are all registered on the blockchain and are automatic as the terms of the smart contract of the NFT are created.

For this reason, the more the NFT is sold in secondary sales, the more the creator will earn.

Instead, from the investor's point of view, especially if they are informed investors, they might

buy a token and resell it at a higher price on secondary markets, and they may even earn a lot if that token is rare. Otherwise, especially in the case of land parcels, investors might buy land, build on it and sell it again at a higher price.

We will see in the next chapter that the NFTs sold on secondary marketplaces have generally higher prices than those in primary markets. For instance, in a study called *Influencing NFT Pricing on Secondary Markets: A case study of Vpunks*[15], it is reported that, when studying the secondary market data of the NFT project *Vpunks*, NFTs which were considered to be rarer were sold at a much higher price in the secondary sale. Moreover, the study also reported that a bigger community also results in an increase in secondary sales' prices. These patterns that the study found, despite being related to *Vpunks,* are also shown in my analysis in the next chapter.

Therefore, I would say that understanding which NFTs have the potential to be sold in the secondary marketplace is especially important as it is related to the profits of both the creators and the buyers.

Businesses are going to profit from this as well: I believe that one day it will become absolutely normal to buy the NFT of a particular brand, whether it be fashion or cars, and understanding from a business perspective which NFT to invest in more will be crucial in the marketing strategies.

Therefore, I gathered the data from a study, *Mapping the NFT revolution: market trends, trade networks, and visual features,* in which a dataset of 6 million NFT transactions had been collected through scraping the OpenSea NFT marketplace.

I decided to structure my work in the following way: I will first analyze the data, in the Exploratory Data Analysis section. Then, I will do a clustering and an object detection analysis and, finally, the predictive analysis, which I split into the theoretical framework and the results. In the end, I will draw the conclusions of my studies and make some considerations on the future of metaverses and NFTs.

CHAPTER 2

Exploratory Data Analysis

2.1 Dataset Description

The original dataset included 6.1 million observations, where NFTs that shared common features were classified in six different categories: Art, Collectible, Games, Metaverse and Utility. Of course, I was not going to use the entire dataset, as my intention was to only study the NFTs with a particular focus to their relationship with metaverses. Therefore, I wrote a Python code to create another *.csv* file, filtering only those NFTs with Metaverse as category. At this point, I had about 68 thousand observations, which was much more manageable than the original 6.1 million.

The first variable is *Smart_contract*. Smart contracts are digital contracts which exist on decentralized blockchain networks and whose terms of agreements between users are written in a code. A smart contract can hold assets, such as in the case of NFTs, and if the conditions included in the code are met, the assets can be distributed upon execution[16]. The NFTs are combined with the smart contracts, in the sense that an NFT can be embedded in a smart contract and vice-versa. In simpler words, a smart contract enforces a sale agreement between the owner and the buyer, and, in the case of NFTs, it assigns (or reassigns) the ownership when the NFT is sold or transferred.

Then, we have the *ID_token*, which is the ID of the NFT.
The *Transaction_hash*, which is a random string of both letters and numbers and represents the address of the transaction, recording the transaction, or in my case, the NFT purchase, on the blockchain.
We also have the addresses and usernames of both sellers and buyers for every transaction. Obviously, the addresses are only proxies and not real identities, as it is done in the Ethereum blockchain, where identities are decentralized.
We then have four different URLs (*Image_url_1, Image_url_2, Image_url_3, Image_url_4*), through which we can access the image of the NFT.

The next two variables are the price in crypto (*Price_Crypto*) and the cryptocurrency whose NFT has been paid with (*Crypto*). Of course, different cryptocurrencies have different correspondent rates in terms of dollars, which is why the dataset has another variable, *Price_USD,* where the price of the NFT has been converted into the price in US dollars. Indeed, for each transaction, the price in dollars has been computed considering the exchange rate at the time. This way, the price remains consistent in the entire dataset.
Then, we have the *Name* and *Description* variables. The former refers to the name of the NFT, while the latter is a short description of what kind of NFT that is.
In the *Collection* column is reported the raw name of the collection the NFT can be found in. NFTs collections are an assortment of NFTs that, in my case, come from the same Virtual World or Virtual Reality game.
The collections in this column have then been cleaned, meaning that the name has been stripped of any special character, numbers, or unusual patterns and the cleaned names have been reported in another column, *Collection_cleaned*.

Then, we have the market in which the NFT has been sold, which are Atomic, Decentraland and OpenSea.
We then see two different variables for the date of the transaction: *Datetime_updated* and *Datetime_updated_seconds:* the latter, apart from reporting the full date of the transaction, also has the hour, minutes, and seconds the transaction has been concluded in.
Next, the column *Permanent_link,* and, finally, we have *Unique_id_collection,* which puts together both the collection and the ID of the NFT, which is why I considered this as the key in the dataset.
Lastly, the *Category* column, whose value will only be "Metaverse", as we have filtered initially and which I will remove at the beginning.

For my analysis, I used both R and Python.
Indeed, I also used Python to perform object detection, which I will explain in the next chapters.
For the remaining analysis, meaning Exploratory data analysis, clustering, and predictions, I used R. I also used R for data cleaning and visualization.

2.2 Data Cleaning

The first thing I did was removing the columns I was sure I was not going to use in my analysis. I used the R library *dplyr* and removed all the URLs, *Description,* and *Category,* which I removed because its only value was "Metaverse", and therefore it would not have made any sense to keep it.

Afterwards, I removed all the rows that have NAs in the *Price_USD* column, in order to have a consistent analysis.

I rendered the variables *Transaction_hash, ID_token,* and *Datetime_updated* as factors and extracted the year from the *Datetime_updated,* which I inserted into the dataset as a new variable, *Year,* so that I could use it for the analysis. I called the updated dataset: *nfts_an.*

In this last dataset, I added the column which will be the focus of my analysis: *resold.*

It has only two values, 0 and 1, and therefore I factorized it to be able to perform classification.

2.3 Exploratory Data Analysis

As mentioned before, the dataset has 68180 observations and 18 variables. Most of them are factors, except for the price in cryptocurrencies (*Price_Crypto*), the price in USD (*Price_USD*), and the date (*Datetime_updated*).

Therefore, I started an analysis of the categories of NFTs present in the dataset and I plotted them in a bar plot in order to understand the proportions.

From the graph (**Figure 2.1**) we notice that the most common collection in the dataset is definitely *Decentraland*.

Since there was such a detachment of *Decentraland* among the other collections, I decided to look more into what it is and into what makes it so popular.

First of all, *Decentraland* is a Virtual Reality (VR) platform that runs on the Ethereum blockchain, and there is a particular reason as to why it is called *Decentraland*: it is the first

fully decentralized world, controlled by the DAO (*Decentralized Autonomous Organization*). Indeed, the users, through the DAO, are allowed to be in control of the policies created to determine how the world behaves and can decide what is and what is not allowed; for example, they can moderate the content, the LAND policies, and auctions. The community proposes and votes on updates of the policy, future auctions or on the whitelisting of NFT contracts allowed inside the World, and on other issues that the community deems relevant: the voting takes place on the Decentraland DAO's governance interface, which is powered by Aragon.

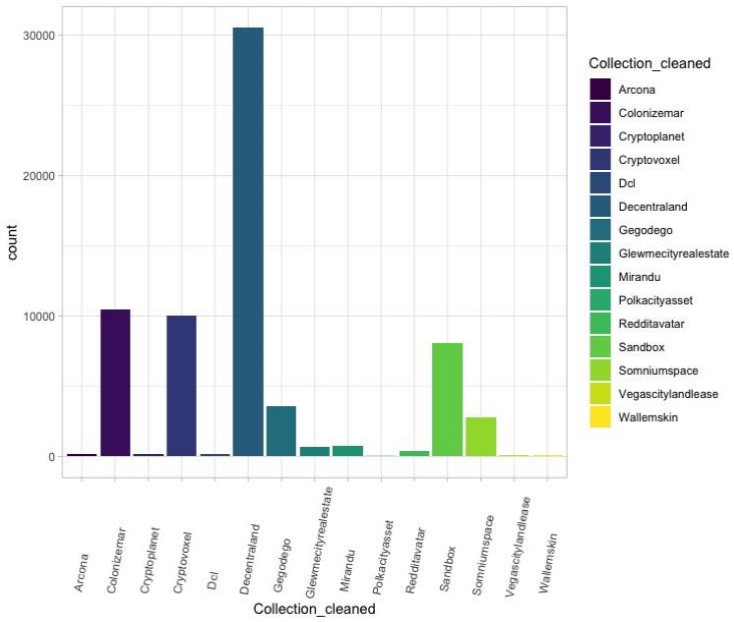
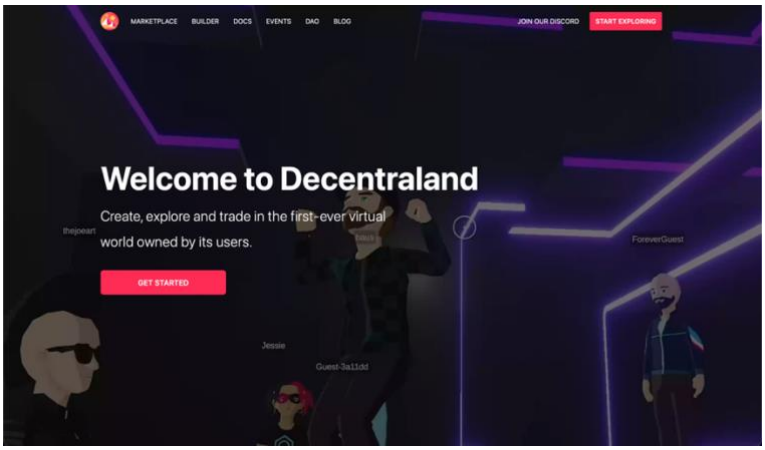


Figure 2.1 Bar plot with the number of observations in each collection of the dataset

Furthermore, as I mentioned before, Decentraland is built on the public Ethereum blockchain, and the Ether currency is the second most used on the platform, outnumbered by MANA. In fact, Decentraland owns a purse of MANA, which is one of the biggest metaverse cryptos, and it is the in-game currency. The important aspect of owning MANA is that Decentraland can be entirely anonymous, as MANA has already been fully decentralized, meaning that the private key that controlled its smart contract has been "thrown away"[17].



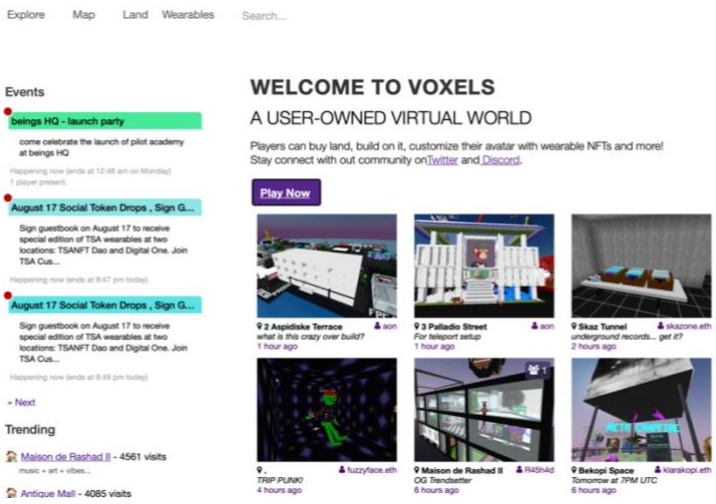
I also researched other common collections: in particular, the second most common was *Colonizemar*.

Colonize Mars is a "Blockchain Strategy Game", in which the users go on missions on Mars and build a life there. According to the White Paper on their website: "Creating a permanent, self-sustaining city on Mars will allow humanity to answer fundamental scientific questions, reignite younger generations' curiosity about space, lead to massive technological innovation, and enable new branches of human civilization.

That's why our company vision is to catalyze the will of humanity to become multi-planetary"[18]. In this game, the gameplay focuses on 4 main areas, which are designed to appeal to different kinds of users: economic strategy, Mars exploration, colony expansion, and leveling up. The user will be absorbed into the game through a very immersive storytelling, and they will be able to build their life and career, as well as explore the maps and find scientific evidence which can be turned into rare NFTs.

The main in-game token is MARTIA and can be earned through in-game activities. MARTIA is built on the WAX blockchain, therefore the users will have to open a WAX Wallet and add the MARTIA token to it.

Another I wanted to research more on was *Cryptovoxels*. In May 2022 they rebranded as only "Voxels".



It is a virtual world built on the Ethereum blockchain, where users buy, sell, and even rent land. They can also build stores and art galleries with NFTs, as well as personalize their avatars and interact with other players. The in-game currency is the native currency of the Ethereum blockchain, ETH, and the lands and other NFTs are traded on the OpenSea market. Furthermore, *Cryptovoxels* also has the Cryptovoxels Parcel, which is an NFT that represents the land in Origin City, and is auctioned weekly on Cryptovoxels and on OpenSea[19].

Other interesting collections, or virtual worlds, are *Wallemskins*, or Wallem Skins, where users buy "skins" to use as avatars, or, for instance, *Mirandu,* whose actual name is Mirandus, which is a fantasy metaverse[20].

Naturally, I also looked at what was the first sold NFT in my dataset. I used the variable *Datetime_updated_seconds* in order to have a more precise answer, and I found that the first sold NFT was on June 12th of 2018 and its name is "69 Block Fork". The NFT belonged to *Cryptovoxels*, which was launched in 2018 by Ben Nolan: indeed, I found this curious because the NFT seller's username is actually "bnolan", thus I assumed he is the *Cryptovoxels'* founder himself. The token was sold at around 0.197 ETH, which equals to 100 USD, and it was traded on the OpenSea market. As we can see from the picture, taken from the OpenSea website, the NFT is actually a land. At the moment, it is reported that the current price is 25 ETH, which amounts to 49,753 USD.

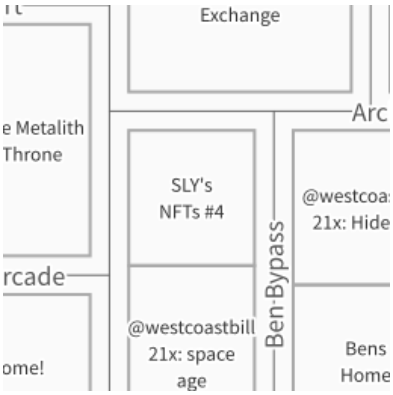


Image 2.1 First sold NFT in the dataset

Let us now look at the last sold NFT in my dataset. It was sold on the 27th of April in 2021 and belongs to the collection *Glewmecityrealestate*, whose real name is GlewMe City, and according to the OpenSea website, it is "the first photorealistic world ever on Ethereum"[21]. The NFT is a *.gif* and it has been sold for 0.7 ETH, the equivalent of around 1808 USD at the time.

I then studied the different Cryptocurrencies and their proportions in the dataset (**Figure 2.2**).

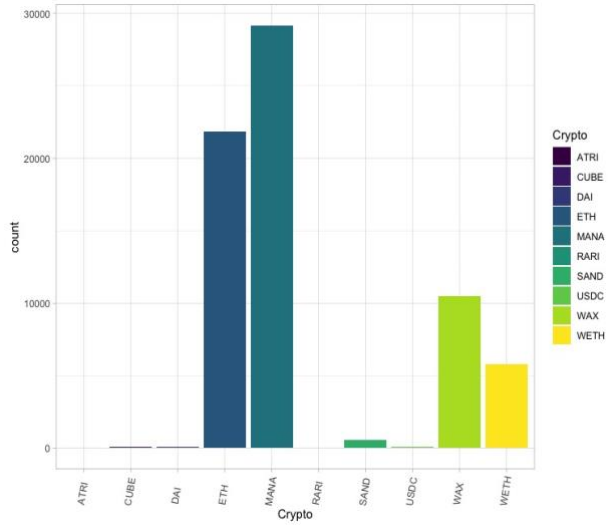


Figure 2.2 Bar plot of the proportions of the variable Crypto

First of all, the most used Cryptocurrency is definitely MANA, followed by ETH.

This is probably because MANA is the main currency of the platform *Decentraland*, which also was the most used one. At the same time, while MANA is only used in *Decentraland*, many other platforms use ETH as their currency. For example *Cryptovoxels*, but also others such as *Arcona* or collections whose NFTs have been sold both with ETH or WETH.

Moreover, I wanted to understand if ETH and WETH might have been related, and indeed, WETH is a "wrapped" version of ETH. WETH follows the ERC-20 standard, which means

that it can be used across "dApps", decentralized applications. Therefore, there is no price difference between ETH and WETH, as they are virtually the same[22].

Next, I analyzed the price. Firstly, the maximum price at which an NFT has been sold is 2.684.346,8 US Dollars. The NFT in question belongs to the *Somniumspace* collection, which is an open-source platform based on the Ethereum blockchain. The seller is "SomniumSpace", which means it was bought from the platform on April 1st, 2021, through the OpenSea market. Here in **Image 2.2**, I report the image of the NFT.

Image 2.2 Most expensive NFT sold

Next, I looked at the distribution of the price from 2018 to 2022. The number of transactions each year varied significantly: in 2018 and 2019, when NFTs were not mainstream yet, the transaction number was somehow low. In particular, in 2018 it was around 2.003 transactions, while in 2019 it was 9.772. From 2019 to 2020 we see the highest jump in the amount of NFTs sold, which goes up to 33.311, more than 240%. In 2021, instead, it decreased by 30%, reaching 23.074. I found this variation in the number of transactions intriguing: first of all, we see a huge spike in numbers from 2019 to 2020, and in fact, from a quick research on Google Trends, I saw that in 2020 the worldwide interest in the words "virtual world" was incredibly high, especially if compared to 2019 (**Figure 2.3**).

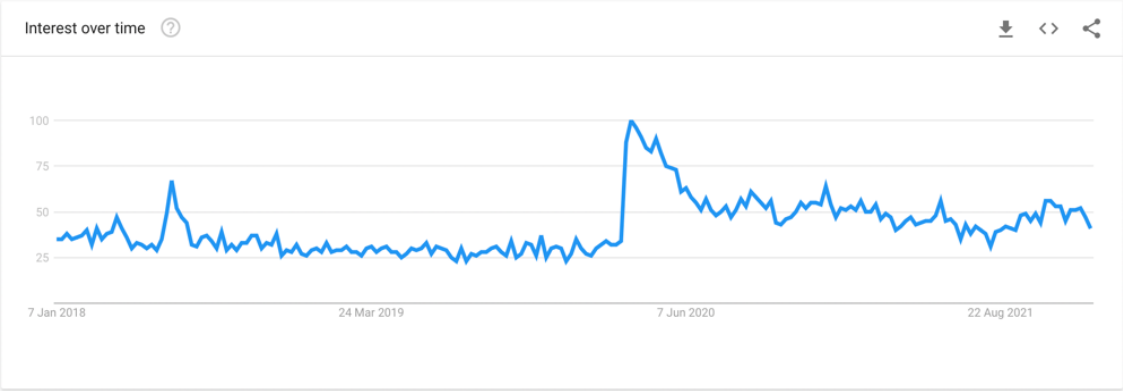


Figure 2.3 Google Trends: interest in the words "Virtual World" from 2018 to 2021

I could have searched for "metaverse", but it would have been futile, because it is a word that came to popularity with Meta, the famous rebranding of Facebook, which was announced on October 28th, 2021[23], and the final transaction in my dataset was in April, 2021.

Indeed, in **Figure 2.4** there is a noticeable spike in interest which is exactly corresponding to the 24th-30th of October 2021.

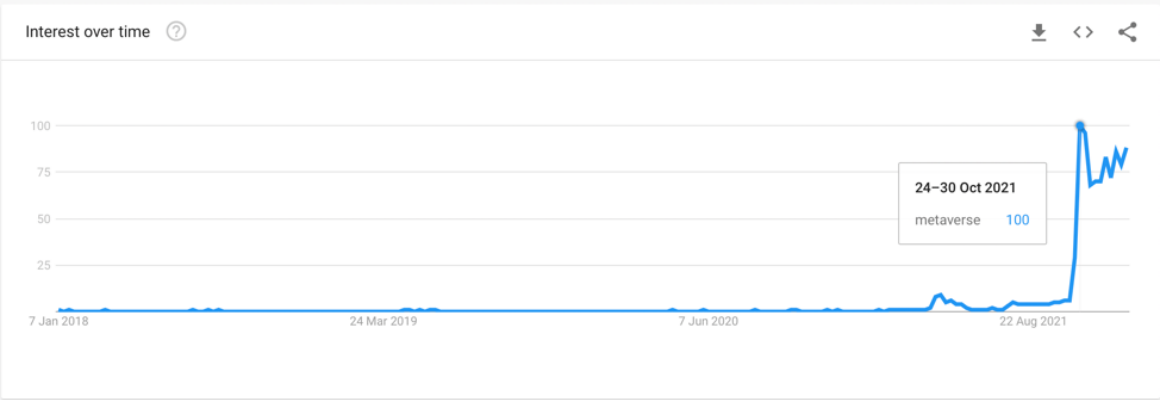


Figure 2.4 Google Trends: Interest in the word "Metaverse" from 2018 to 2021

Therefore, if I had searched for "Metaverse", I would not have gotten any insight into the trends, as people were more common with "virtual worlds". Of course, I do not mean that metaverses were not known, but only that the general population was less familiar with the word.

I can therefore conclude that the rising interest in virtual worlds in 2020 explains the rising number of transactions found in my dataset. It can also explain the diminishing of the numbers in 2021, as we see a decline in the interest from around June 2020 to August 2021.

I would explain such a high interest in virtual worlds in 2020 with the pandemic spreading and the various lockdowns.

Indeed, the highest spike in **Figure 2.3** was just before June, when the world was already mostly in lockdown, and people, who were forced to stay at home, were probably finding an escape from the real world into a virtual one. This also explains the decreasing interest in 2021, where lockdowns became less strict, and people could get out more.

Afterwards, I looked at the average price for NFTs during the years: in 2018 the average price was at its highest, at around 2.033 USD. It then decreased to a much lower price in 2019, when it was at around 720 USD, only to decrease again in 2020, at its lowest, 370 USD. It then rose again to 1.900 USD. This gives us an insight into how both the demand and offer, and the speculation on the NFTs in the market, shaped the price. Indeed, in economy, a very familiar concept is that the lower is the offer, and the higher the demand, the higher the price is. In our case, we see that the highest price was in 2018, when NFTs were not mainstream yet, and there were few of them except for some famous collections. It was also still a foreign concept to the majority of people, who, in fact, started looking into NFTs in 2021, as can be seen in **Figure 2.5**, where it is shown that the interest for the word "NFT" started at the beginning of 2021.

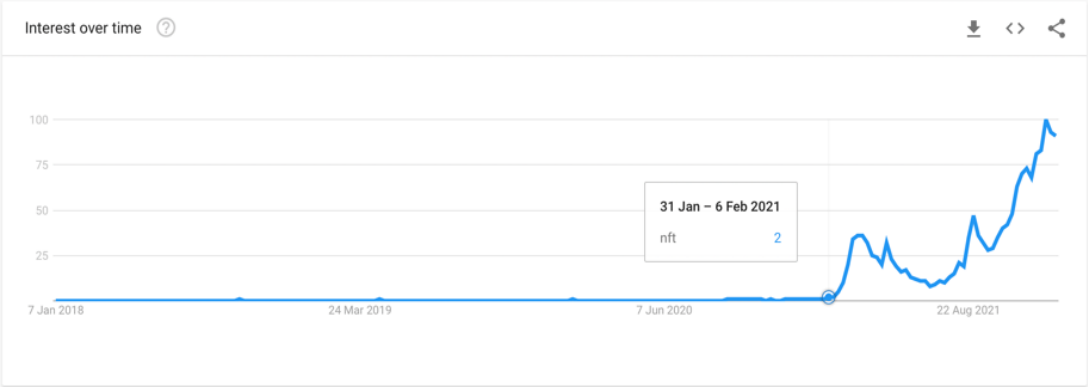


Figure 2.5 Google Trends: Interest in the word "NFT" from 2018 to 2021

However, here we are talking about NFTs related to the metaverses, which instead piqued in 2020, where, coincidentally, the average price is at the lowest.

I also found the maximum price each year, and in 2020 it is 95.607 USD, which is a high price in general, but remains low if we compare it with the maximum prices for the other years. In fact, in 2018 the maximum was 214.274 USD, much higher than in 2020. Here again, we see a pattern: in 2019, the maximum price was 110.907 USD, lower than in 2018, but definitely higher than in 2020. This reminds us of the pattern for the average price, where in 2019 it was higher than in 2020, but lower than in 2018. To understand what is going on, I also calculated the total volume each year, which the sum of how much money has been spent on the transactions, and I found that in 2018, when NFTs were not popular, the volume was at around 4 million USD. It keeps increasing in the following years, but first I will focus on the years 2018, 2019 and 2020, as 2021 needs a different kind of explanation. In 2019 and in 2020 the volume increased exponentially, going as far as more than 12 million USD spent in 2020.

I assumed, then, that as the NFTs became more popular the price went down, because of the highest number of available NFTs in general, which relates to a higher offer and perhaps not yet a particularly high demand. On this note, it is important to notice that in 2019 we have a drop in price, and a major increase in the volume spent, which might confirm the previous supposition.

This takes us to the year 2021, which I preferred to analyze separately from the other three years. Indeed, in 2021, we see a huge increase in the interest on NFTs and crypto in general, together with the "birth" of Meta. This translates into two different scenarios: on one hand, the total volume spent on NFTs is much higher than in 2020, as it reaches more than 44 million USD, meaning an increase of 258%. On the other hand, the price also goes up since 2020: the increase is as much as 416%, from 373 USD to 1930 USD in 2021.

Of course, this last finding goes against what I have stated previously, that as the volume goes up the price comes down, but I would say that this is because of the afore mentioned speculation that is on the NFTs. Indeed, the peak in popularity of the NFTs in the past year, has made it so that the sellers are increasing the price of their creations, because they know that their content will be sold. Furthermore, especially in the case of metaverses and virtual worlds, lands are being sold: we have already talked about virtual worlds like Decentraland, but also GlewMe City or Cryptovoxels, where people buy lands in the form of an NFT and they can build on it and resell it at a much higher cost. In a way, it is like investing in real estate, and since it is becoming very popular to own digital land, the price is increasing exponentially. What is more, people are now understanding more about the concept of NFTs, and I suppose that one aspect that really drives people to pay more is the idea of the uniqueness of the token. Its scarcity carries a unique market value, which increases the price.

From these reasonings, I believe that it is really no surprise if 2021 has the highest volume and the highest average price, just as it is of no surprise that the NFT with the highest price was exactly in 2021.

I also plotted the distribution of the price during the years. I used two kinds of plots, a box plot and a ridge plot. At first I had plotted the entire dataset, but I quickly realised that the plots were unreadable, as the outliers were too many and too high. Indeed, many of the prices were on a small scale, and they were complicated to observe if there was a much more expensive outlier. Thus, I removed some of the outliers and used the new dataset to plot the box plot and the ridge plot. I only used this dataset for the plots, as otherwise I would have lost a huge amount of data useful for the predictions of the next chapters.

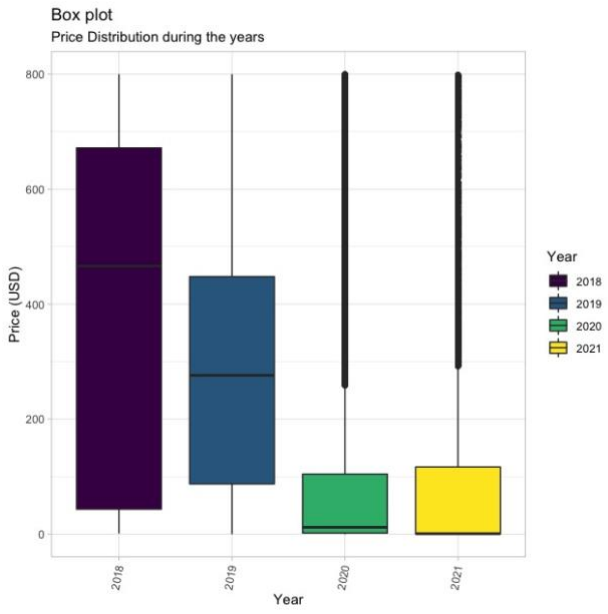


Figure 2.6 Boxplot for the distribution of the price each year

In **Figure 2.6**, I reported the box plot without the outliers. We can see that in 2018 the price had a much wider range than in the other three years, and there are no remaining outliers. In 2019, the range gets smaller and more towards a slightly higher price than in 2018. Interesting to notice that, instead, in 2020 and 2021 the range is much smaller and more towards lower prices, with many outliers remaining.

This gives us some new interesting insights: first of all, in 2018 the price was much less "standardized", as perhaps people were just discovering NFTs and the market was not saturated. In 2020 and 2021, the majority of NFTs had a much lower price, as people were buying them more, although the huge quantity of outliers are an indication that some NFTs actually spiked, probably due to an increase in popularity.

From the ridge plot in **Figure 2.7**, instead, I was able to see more clearly the distribution of the prices (again, I used the dataset without outliers). First, the distribution in 2018 and 2019 is definitely not Gaussian. It looks more normal in the years 2020 and 2021, where the prices started to be more "standardized". Indeed, in 2018, for instance, we notice a spike from 0 USD to what I assumed was around 50/100 USD, and then another curve towards much higher prices. In all the years we still see a peak in that price range, meaning that many NFTs probably tend to that price range. This interpretation stands particularly in the year 2020 and 2021, when the curve is almost flat after that first high peak.

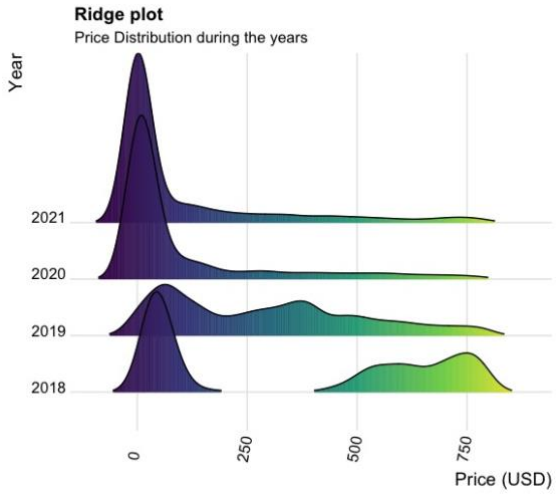


Figure 2.7 Ridge plot for the distribution of the price each year

Moreover, I used a scatter plot and looked closer at the prices. Just like before, I firstly plotted with the entire dataset, as shown in **Figure 2.8**. From this graph, we clearly see when the prices have gone exceptionally high, and we notice that in 2021 we have the most amount of NFTs with high prices, probably due to the hype.

However, I wanted to have a closer inspection at the variations during the years, which is why I then used the dataset without outliers.

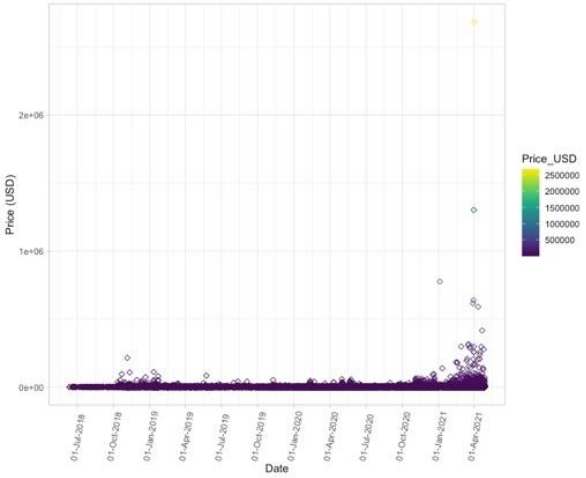


Figure 2.8 Scatter plot with the prices of the entire dataset

Firstly, as shown in **Figure 2.9**, in 2018 it was very uncommon for the NFT price to go higher than 200 USD, apart from some sporadic NFTs, whose price went much higher (as visible from the previous graph). Then, the price rose immediately in 2019, so much so that between July 2019 and October 2019, there is almost a curve, where the minimum price gets to a lot higher than before, except from a few pieces. From the ending of 2019 to the beginning of 2020, we see the pattern we have already found before. The minimum price gets lower, almost towards 0 USD, and by the deeper color I supposed that there was a high amount of NFTs at that price range. At the same time, there still were many much more expensive tokens, especially higher than 400 USD. Finally, in 2021, as the NFT numbers increased, we see more distributed prices, although there are still many who were pretty cheap.

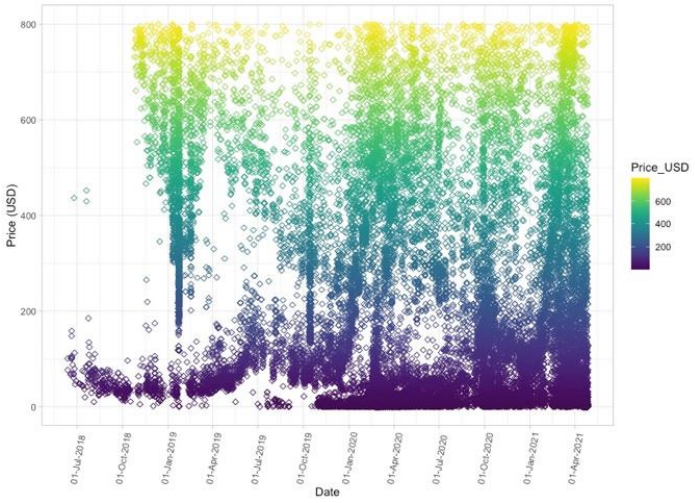


Figure 2.9 Scatter plot with prices and no outliers for the entire dataset

I then studied the distribution of the price across the different collections, shown with the boxplots in **Figure 2.10**.

As we can see from the graph, the distribution of prices looks very different depending on the collections.

For instance, many of them have prices that range towards very low ones: one example *Colonizemar* (Colonize Mars), whose tokens' prices are entirely towards 0, with relatively few outliers. On the other hand, also of note are *Cryptoplanet* and *Vegascitylandlease*, whose prices are much higher than the other collections.

The most popular virtual world, *Decentraland*, has relatively low prices

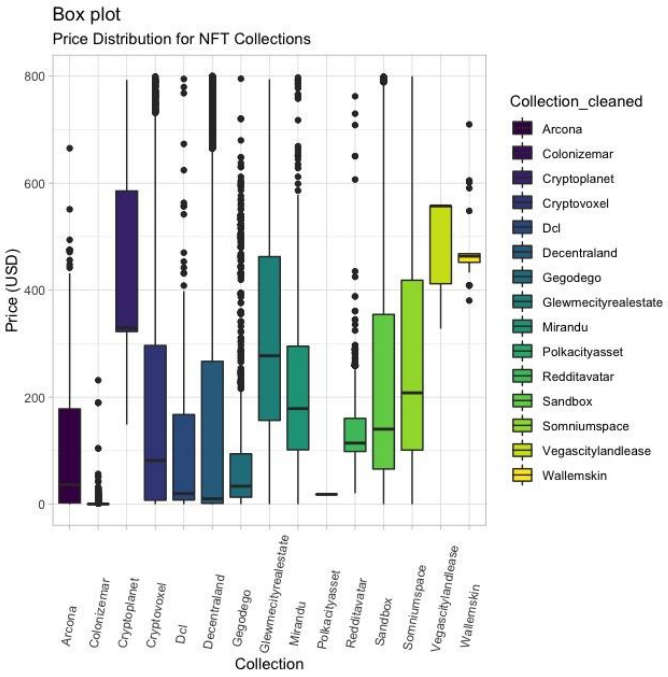


Figure 2.10 Box plot for the distribution of prices in NFT collections

compared to other collections, although it clearly has many outliers, which are also above 800 USD.

This might be an indication of the fact that *Decentraland*, being a more mainstream metaverse, and therefore having more transactions, has a lower price range than, for instance, *Cryptoplanet*, which is less known and probably has customers with more niche taste. I can say the same thing for collections such as *Wallemskin* or *Vegascitylandlease*.

When I created the "resold" column, I thought of getting some insights out of it as well.

Resold is the variable in which "1" stands for all those NFTs which were sold in secondary sales, whole "0" stands for those which were only sold in primary sales. I also considered as resold the original, first sold ones, so that I could have a clear distinction from those which have never been sold again.

First, I looked at how different the distribution of resold NFTs was among the collections. In **Figure 2.11** I reported a bar plot where the percentage of the resold NFTs is visible for each collection.

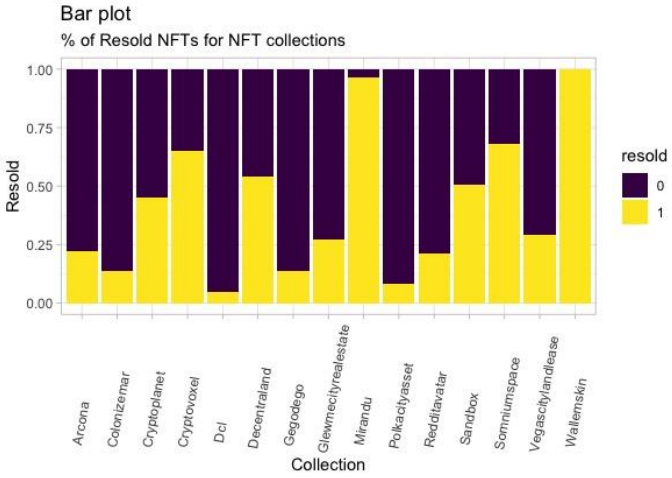


Figure 2.11 Bar plot on percentage of resold NFTs in the collections

I immediately noticed that the collection *Wallemskin* has 100% of resold NFTs. I looked at how many observations did *Wallemskin* have, and found they were 90, which is less than 1% of the total of the observations. Of all those NFTs, all of them have been sold in the secondary market. In the same way, the collection *Mirandu* also has more than 90% of NFTs resold, and it has 745 observations in the dataset. I would say that, because they have such few NFTs, those that are sold are more likely to be sold again in secondary sales due to their rarity.

Furthermore, I noticed that collections such as *Decentraland*, *Sandbox*, *Cryptoplanet* and *Colonizemar*, had a much lower percentage of resold NFTs. I supposed this happened because those collections belonged to virtual worlds where users bought mostly land, such as *Decentraland*, and therefore it would maybe be more difficult to sell. For instance, a user might purchase land, build on it, and then sell it again later at a higher price, but it would take investments and time. Indeed, *Wallemskin*, for instance, is an augmented reality game where users buy NFTs for "skins" or other items, which are much more easily resold than land.

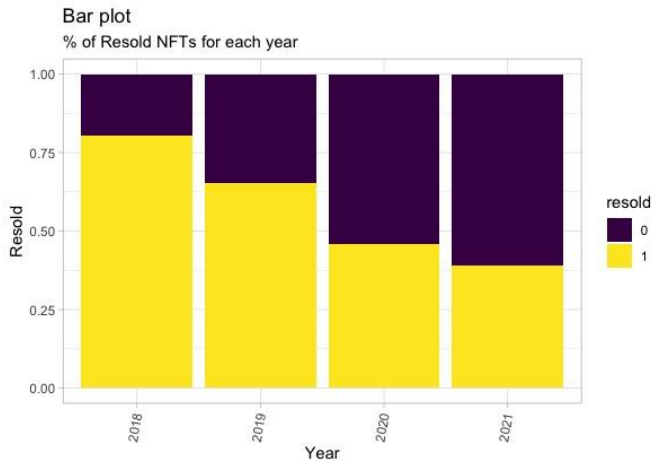


Figure 2.12 Bar plot on percentage of resold NFTs each year

Another interesting plot is the percentage of how many resold NFTs are there each year. As visible in **Figure 2.12**, the percentage decreases as the years advance. This might have happened because in the previous years there were very few collections of NFTs compared to these last two years, which meant users would buy NFTs more from the secondary market than the primary, especially as the demand increased. However, as more creators emerged, it would have been easier for buyers to buy the NFTs directly from the creator.

Moreover, buying directly from the creator, in this case, is even cheaper than buying it from the secondary market, and in fact, I computed the average price for resold and not resold NFTs and those which were only sold on the primary market were actually cheaper (average price: 617 USD) than the others (average price: 1425 USD).

As a confirmation of this pattern, I designed a box plot, which shows the distribution of the price with respect to the primary and secondary market (**Figure 2.13**).

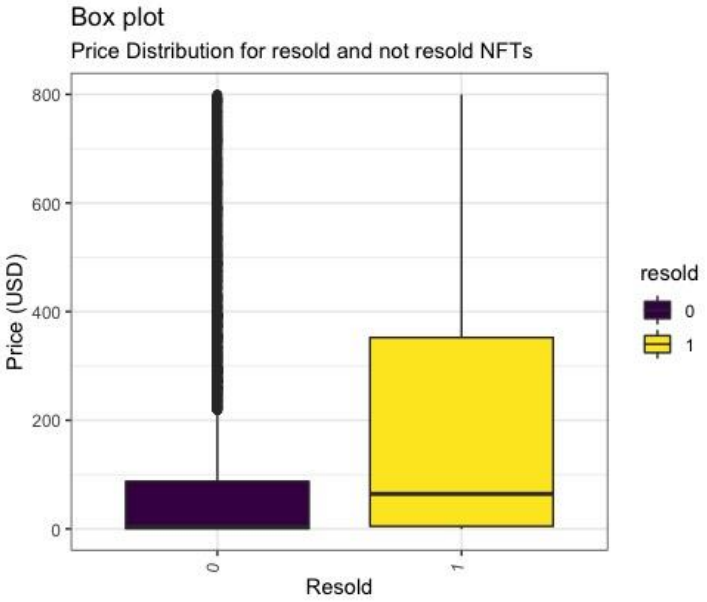


Figure 2.13 Box plot for price distribution of resold NFTs

Indeed, the distribution of the price has a much larger range in the case of resold NFTs. Those which were sold on the secondary market have a higher price compared to those which were not, an aspect I had already mentioned in **Chapter 1**. I supposed that there might be two different reasons for this. On one hand, in a way that also happens in the auctions of art pieces, sometimes the sellers are not aware of the actual worth of what they are selling, and clever clients, who instead know that they are buying a piece for a too low price, will sell it again at a higher price.

On the other hand, as many of the NFTs in my dataset are parts of land, shops or buildings, my supposition is that users would buy land for its initial price, and then would build on it, consequently increasing the value of the land. This is just like what happens with real estate, where people buy land and improve its value through investments in buildings and construction.

Afterwards, I looked at the distribution of the resold NFTs for each market. Out of the three markets we are considering, Decentraland and OpenSea have almost the same percentage of resold NFTs, which is more than 50%. For what regards OpenSea, it is considered to be a secondary marketplace, although creators can also sell their work directly from there as primary sales. Atomic, instead, has only a small percentage of secondary sales.

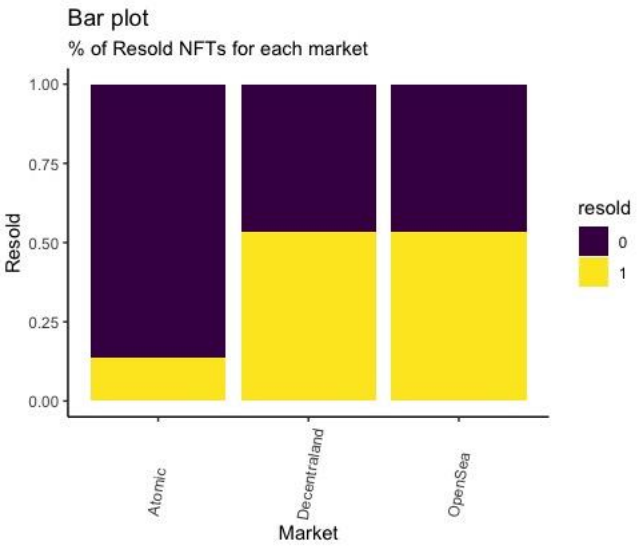


Figure 2.14 Bar plot on percentage of resold NFTs in each market

CHAPTER 3

Hierarchical Clustering Analysis

Clustering analysis is very useful to data analytics as it helps understanding patterns among data that might go unseen without it.
I decided to include clustering in my analysis because my intention was to build a profile of different NFTs' exchanges, which might potentially give me insights into the characteristics of the instances. Indeed, I will try to interpret the comparison between clusters and the different variables, which highlights the patterns in the dataset, in order to understand if there are any peculiar trends.

3.1 Theory Framework

Hierarchical Clustering is an unsupervised learning algorithm that groups data points into clusters. Differently from other techniques, such as K-Means clustering, the advantage of hierarchical clustering is not having to specify the number of clusters beforehand.
There are two kinds of approaches to hierarchical clustering: bottom-up and top-down.
Bottom-up, also called agglomerative, means that the algorithm combines together the closest pair of data points and continues until all the observations are grouped together in a single comprehensive cluster. The final form of this approach is a dendrogram.
Top-down, or divisive, recursively splits the clusters until only the individual data points remain.
In my analysis, I will use the agglomerative approach.
There are three different agglomerative methods:

1. Complete Linkage: in which the distance is defined as the farthest distance between two points of two different clusters.
2. Average Linkage: in which the distance is defined as the distance between the centroids of two different clusters.
3. Single Linkage: in which the distance is defined as the distance the closest distance between two points of two different clusters.

The distance in clustering is actually a function that measures the similarity between two data points.

Although the most used distance is the Euclidean, my dataset has many categorical variables, which means that I preferred to change the metric to the Gower distance metric.
The Gower[24] metric, or the Gower similarity coefficient, is a similarity measure that is considered to be simple, dynamic, and flexible as it has the ability to compare both numerical variable types and categorical variable types. It is based on the average value resulting from comparing the attributes of two objects[25].

3.2 Implementation of the Technique

Because hierarchical clustering is a particularly slow technique, I had to significantly reduce the size of my dataset. In order to achieve what I thought was the optimal number, 6 thousand observations, I used the function *slice_sample()* from the r package *dplyr* and randomly reduced the size.

Then, I computed the distance. As I mentioned above, I used the Gower similarity coefficient, because of the way my dataset was composed. Indeed, if I had wanted to use the Euclidean distance, I could have only used the variable *Price_USD,* but that would have significantly reduced the performance of the clustering. This is also the reason why I did not use another clustering algorithm, K-Means clustering. This last method, in fact, can only handle numerical variables, and trying to do cluster analysis with only *Price_USD* would not have made any sense, which is why I resorted to only hierarchical clustering.

Nevertheless, I compute the distance using the function *daisy()* from the r package *cluster,* and I set Gower as metric.
Then, I do hierarchical clustering with the three different linkages: complete, average, and single.
For all three of them, I use the function *hclust(),* from the r package *stats.* Afterwards, I plot the dendrogram and choose the clusters accordingly, looking at how many elements are there in each cluster.
Finally, I compared the clusters with the variables, to understand the patterns and the distribution.

3.3 Results

The first dendrogram I obtained was with the complete linkages.

I chose 5 clusters since I found it to be the optimal number considering the plot.

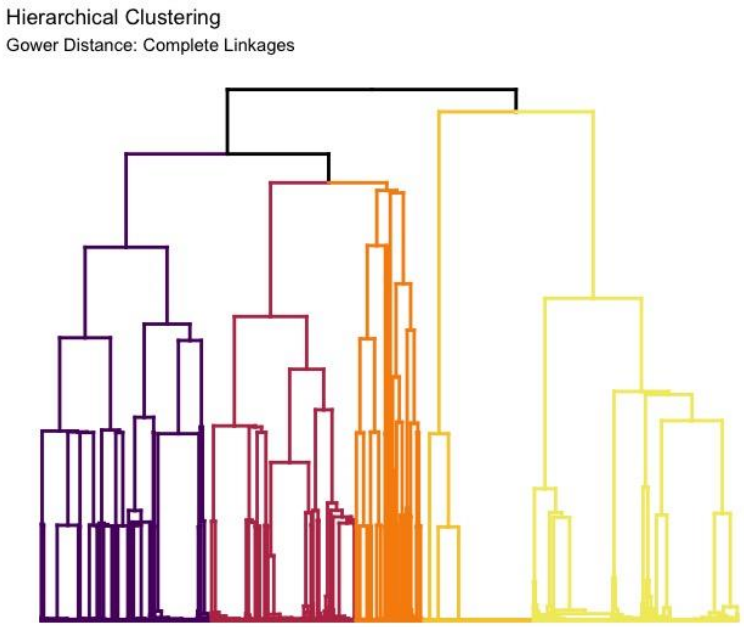


Figure 3.1 Dendrogram of Hierarchical Clustering Complete Linkages

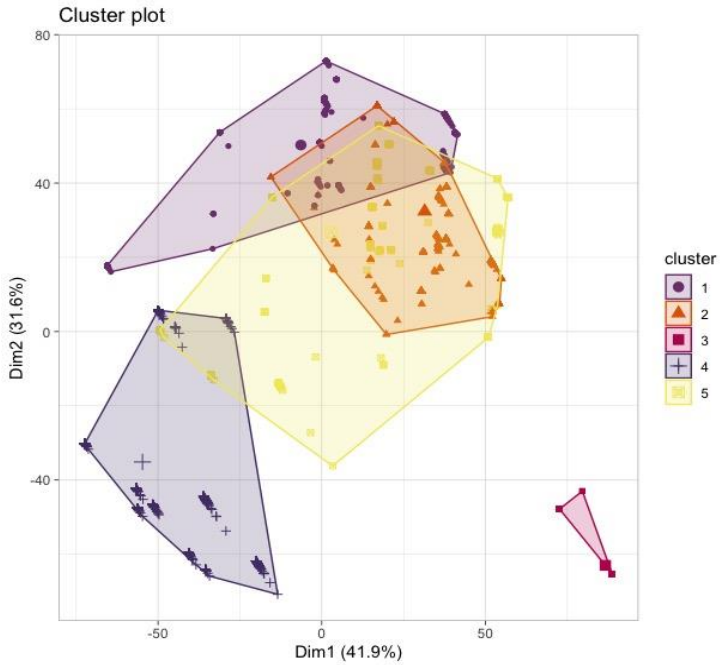


Figure 3.2 Plot of the clusters with respect to the first two principal components

Indeed, the clusters seem pretty balanced and distinct, and I did not want to have a higher number as it can limit the interpretability.

I confronted the clusters with the various variables in my dataset: *Crypto, Market, Collection_cleaned,* and *resold.*

For what regards *conf_hc_cr,* the distribution of the cryptocurrencies in the various clusters, it did not provide many insights on its own. Indeed, I first looked at the other comparisons. For instance, through the comparison with the variable *Market* (**Table 3.1**), *conf_hc_mk,* I noticed that most of the clusters (cluster 1, 2, 3, 5) had only one market. In particular, cluster 3 had the entirety of the NFTs sold on the Atomic market, while cluster 4 all the ones sold on Decentraland. Only one cluster, cluster 4, contains two markets, both Decentraland and OpenSea.

	hc.clusters_c				
	1	2	3	4	5
Atomic	0	0	1069	0	0
Decentraland	0	0	0	1654	0
OpenSea	1416	666	0	342	1669

Table 3.1 Market distribution in the clusters – complete linkage

Also of note, is that OpenSea, the biggest market, is present in all clusters except cluster 3.

I wanted to look for more interesting patterns and therefore confronted the clusters with the variable resold, which, as mentioned, will be the dependent variable, or the variable to predict in my analysis. From the confusion matrix *conf_hc_res* (**Table 3.2**), I already noticed two interesting aspects: in cluster 1 all the NFTs have been sold again on the secondary market, but in cluster 5 it is the contrary, as no NFTs have been resold. All the other clusters have more balanced results, for instance, cluster 2 has 437 resold NFTs and 229 which have only been sold once. Cluster 3, that had only NFTs from the Atomic market, has the high majority of NFTs not resold, and in fact, as seen in Chapter 2, most of the NFTs from the Atomic market have not been resold on the secondary market (of course, these suppositions are related to the very reduced data sample I am currently working on, although the pattern stands even in the entirety of the dataset). On this note, I would consider cluster 1 as the cluster with the majority of resold NFTs from OpenSea, while cluster 5 is the cluster with the majority of *not* resold NFTs from OpenSea.

	hc.clusters_c				
	1	2	3	4	5
0	0	229	933	776	1669
1	1416	437	136	1220	0

Table 3.2 Resold distribution in the clusters – complete linkage

Now that I had this information, I confronted the clusters with the variable *Collection_cleaned* (**Table 3.3**), so that I could understand the distribution of the collections in each cluster and gain more insights. The aspect I noticed in the confusion matrix *conf_hc_coll,* was that cluster 4 had only one collection inside, *Decentraland*. Indeed, we had noticed before that cluster 4

had the majority of NFTs bought in the Decentraland market, and, moreover, the majority of them has been resold. If I also compare it with the variable *Year,* it looks like most of the observations in cluster 4 are also from the year 2020, while some are also from the year 2019.

In cluster 3, instead, where the majority of NFTs have not been resold, almost all the observations are in the *Colonizemar* collection, and they are all from the year 2021.

	hc.clusters_c				
	1	2	3	4	5
Arcona	8	8	0	0	9
Colonizemar	0	0	1067	0	0
Cryptoplanet	6	4	0	0	8
Cryptovoxel	506	263	2	0	270
Dcl	1	0	0	0	20
Decentraland	428	8	0	1996	602
Gegodego	40	0	0	0	318
Glewmecityrealestate	16	14	0	0	40
Mirandu	4	74	0	0	2
Polkacityasset	0	0	0	0	1
Redditavatar	9	1	0	0	27
Sandbox	279	150	0	0	341
Somniumspace	111	143	0	0	27
Vegascitylandlease	1	1	0	0	4
Wallemskin	7	0	0	0	0

Table 3.3 Collection distribution in the clusters – complete linkages

Now, I can come back to *conf_hc_c* (**Table 3.4**), confusion matrix between cryptocurrencies and clusters, and try to gain some more insights: for instance, cluster 3 has all its data points with the WAX cryptocurrency, while in cluster 4, the same thing happens with MANA.

Putting together all this information, cluster 3 presents a very interesting pattern: the almost entirety of NFTs in the cluster are from the collection *Colonizemar* and they were all traded on the Atomic market. Also, the observations in the cluster are all in the year 2021 and the cryptocurrency is WAX.

It makes sense that cluster 3 grouped together all *Colonize Mars* users' transactions: *Colonize Mars* has been launched in 2021[26], and therefore it makes sense that the cluster in which all *Colonize Mars* transactions are grouped, also has only the year 2021. What is more, the cryptocurrency WAX is also related to *Colonize Mars*, as it is based on the WAX Wallet. Another interesting aspect of cluster 3 is the amount of resold NFTs, which is lower than the not resold ones: I suppose that this is because, as I said earlier, *Colonize Mars* is almost like a video game, which means that generally, the users prefer to keep their tokens to themselves, as they represent their effort in the game.

	hc.clusters_c				
	1	2	3	4	5
ARCONA	0	0	0	0	0
ATRI	0	0	0	0	0
CUBE	0	7	0	0	0
DAI	2	5	0	0	6
ETH	968	214	0	0	993
GALA	0	0	0	0	0
MANA	334	0	0	1996	560
PGU	0	0	0	0	0
RARI	0	0	0	0	0
SAND	3	45	0	0	7
SWAG	0	0	0	0	0
USDC	1	4	0	0	5
WAX	0	0	1069	0	0
WETH	108	391	0	0	98
WIPC	0	0	0	0	0

Table 3.4 Crypto distribution in the clusters – complete linkages

This last pattern can also be found both in the model with the average linkage and with the single linkage. Indeed, in both models, there is always a cluster with the same exact characteristics as in *hc_model_c.* This is very peculiar, as generally the clusters differ with the different linkage methods, since the way to compute the distance between data points changes.

However, this stands as a confirmation of the strength of this pattern in the data, which is much more ingrained than other possible trends I could find.

Nevertheless, I tried to understand if there were other tendencies and I came to some more conclusions. For example, in cluster 4, which has only *Decentraland* tokens, we notice that many of them are in the year 2020 and that the majority were resold. This is probably because of two factors: on one hand, the pandemic, like I mentioned in earlier chapters, had a huge impact in the rise of virtual worlds, as people could not go out and escaped reality through a virtual one. This also brought more awareness to people, who started understanding that buying land online could be very profitable for the future.

hc.clusters_c	1	2	3	4	5
2018	3	41	0	160	0
2019	0	285	0	551	218
2020	1033	14	0	1060	1162
2021	380	326	1069	225	289

Table 3.5 Year distribution in the clusters - complete linkage

This last pattern is also confirmed in cluster 1, even more strongly, as all the NFTs in the cluster have been resold and they mostly belong to *Cryptovoxel*, *Decentraland* and *Sandbox*, and most of the transactions have been carried out in 2020.

Cluster 5, instead, is the opposite. All the NFTs in the cluster have never been resold, therefore I suppose that it has grouped those NFTs which have no further market in the virtual worlds, or perhaps NFTs of users who did not necessarily think about profit when buying them, but instead simply preferred to enjoy the virtual reality they were in.

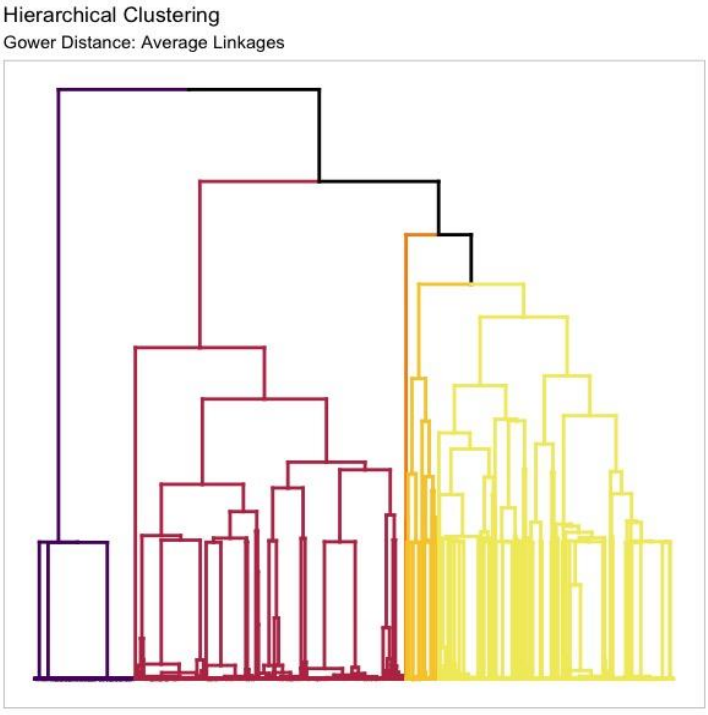


Figure 3.3 Dendrogram of Hierarchical Clustering Average Linkages

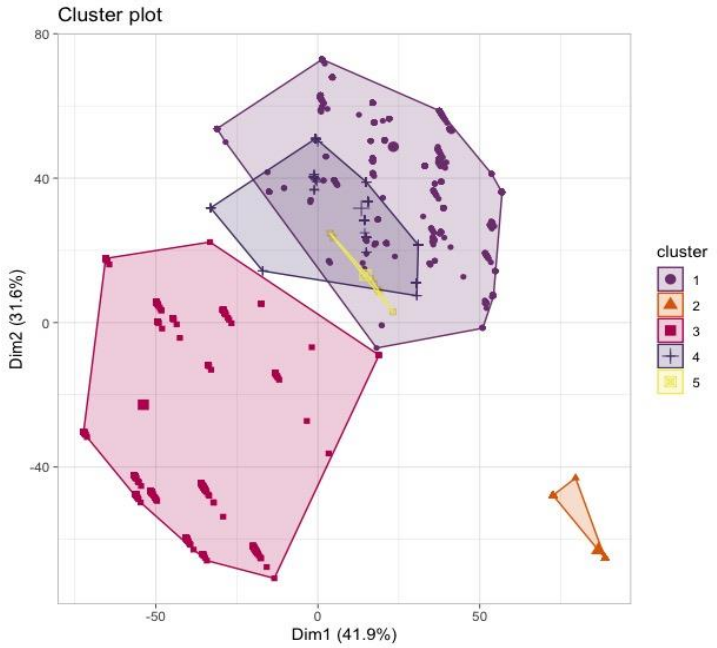


Figure 3.4 Plot of the clusters with respect to the first two principal components

Regarding the model obtained with the average linkage (**Figure 3.3**), I decided on splitting in 5 clusters, and then I created the confusion matrices as before.

I have already mentioned that in *hc_model_a,* cluster 2 is the exact same as the cluster 3 in the previous model; however, it also has a cluster identical to one which we will see later in *hc_model_s,* the model with the single linkage method.

This cluster is cluster 5, which contains only 7 data points, and they all belong to the *Somniumspace* collection, have been traded entirely in 2020 and most of them have not been resold. Furthermore, all of them have been bought and with the CUBE cryptocurrency in the market OpenSea. However, I believe that this does not give many insights, except that it is a cluster which has its correspondent in the single linkage. Other patterns that I was able to notice are, for instance, that there are less clusters that contain only data points from the OpenSea market, as only cluster 1, 2 and 5 do, compared to the previous model.

hc.clusters_a					
	1	2	3	4	5
ARCONA	0	0	0	0	0
ATRI	0	0	0	0	0
CUBE	0	0	0	0	7
DAI	13	0	0	0	0
ETH	2175	0	0	0	0
GALA	0	0	0	0	0
MANA	0	0	2890	0	0
PGU	0	0	0	0	0
RARI	0	0	0	0	0
SAND	55	0	0	0	0
SWAG	0	0	0	0	0
USDC	10	0	0	0	0
WAX	0	1069	0	0	0
WETH	252	0	0	345	0
WIPC	0	0	0	0	0

Table 3.6 Crypto distribution in clusters – average linkage

From *conf_hcs_res* (**Table 3.8**), confusion matrix with the clusters and the variable resold, I also noticed that this time there were no clusters with NFTs resold or not resold entirely. Indeed, this time, the distribution was more balanced, apart from cluster 2, where the proportion of resold and not resold NFTs was the same as previously in cluster 3: 933 not sold on the secondary market, 136 resold.

	hc.clusters_a				
	1	2	3	4	5
Arcona	17	0	0	8	0
Colonizemar	0	1067	0	0	0
Cryptoplanet	18	0	0	0	0
Cryptovoxel	858	2	0	181	0
Dcl	10	0	11	0	0
Decentraland	149	0	2877	8	0
Gegodego	341	0	0	17	0
Glewmecityrealestate	70	0	0	0	0
Mirandu	80	0	0	0	0
Polkacityasset	1	0	0	0	0
Redditavatar	37	0	0	0	0
Sandbox	770	0	0	0	0
Somniumspace	145	0	0	129	7
Vegascitylandlease	4	0	2	0	0
Wallemskin	5	0	0	2	0

Table 3.7 Collection distribution in the clusters – average linkage

Lastly, I use the single linkage method (**Figure 3.5**), and this time I chose 3 clusters, as it was the only visible way to split them, although, as can be seen from the dendrogram, the clusters were not particularly well defined.

	hc.clusters_a				
	1	2	3	4	5
0	1173	933	1336	160	5
1	1332	136	1554	185	2

Table 3.8 Resold distribution in the clusters - average linkage

Also, there are only three clusters, but two have already been analyzed with the previous models, and the third one does not give us any more particular insights.

Figure 3.5 Dendrogram of Hierarchical Clustering Single Linkages

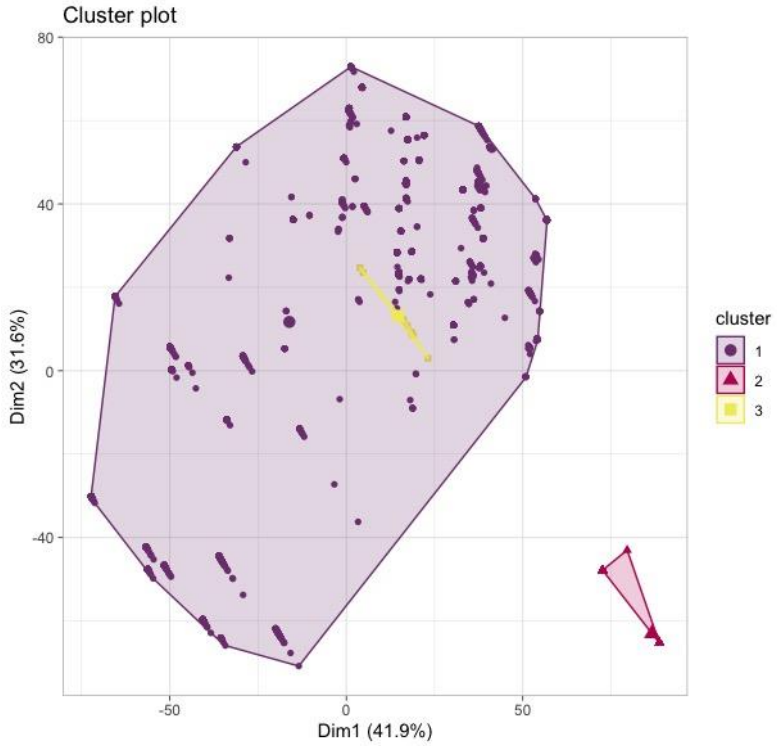


Figure 3.6 Plot of the clusters with respect to the first two principal components

CHAPTER 4

Predictive Analysis: Theoretical Framework

The first models I computed are the linear ones: Logistic Regression, Linear Discriminant Analysis, and penalization techniques, such as Ridge, Lasso, Elastic Net and Grouped Lasso.
Then I also used non-linear models, like Quadratic Discriminant Analysis, K-Nearest-Neighbors and the tree-based algorithms (Decision tree, pruning, Random forest and Boosting).
For each method, I will explain the theory behind it and how I implemented it, in this chapter then I will discuss the results in **Chapter 5**.
For the entire analysis, I will use the dataset comprising 6 variables: 5 predictors and the response. The predictor variables are *Crypto*, *Price_USD*, *Market, Collection_cleaned* and *Year*, while the response is, of course, *resold*.
After the predictions, I will measure the accuracy both with the AUC (Area Under the Curve), which indicates the overall performance of the classifier, and with the percentage of correct predictions, which is computed through the function *mean()*.
In addition, I also used the function *prop.table()* to create a confusion matrix that compares the predicted classes with the real values, out of which I will always check the *sensitivity*. The sensitivity is the percentage of true positive, meaning the times that the classifier correctly assigned 1, which means the NFT is among the resold ones.

4.1 Logistic Regression

Logistic regression is a classification model that allocates the response, Y, to a particular category in a set of categories.
Indeed, logistic regression actually estimates the probability of Y belonging to the category, and, consequently, its function must give values between 0 and 1.
For this reason, a simple linear regression model cannot be used, as it would give as output $p(X) < 0$ or $p(X) > 1$, which are outside the range for the probabilities' values.
The regression function specific for logistic regression, called *logistic function*, avoids this problem, giving outputs between 0 and 1 for every value of X.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

This function's curve will be *S-shaped*, and we will be able to obtain sensible predictions.

The coefficients β_0 and β_1 are initially unknown and must be estimated based on the training data, usually through the method of *maximum likelihood* and its equation, the *likelihood function* (down below).

$$\ell(\beta_0, \beta_1) = \prod_{i:\, y_i = 1} p(x_i) \prod_{i':\, y_i' = 0} (1 - p(x_{i'}))$$

Furthermore, the values of $\hat{\beta}_0, \hat{\beta}_1$ are chosen in order to maximize the likelihood function.[27]

For the implementation of logistic regression in my code, I used *glm*(), from the R package *stats*.

I split my dataset into a training and a validation set and I fit the model *lin_model* with glm() with both. Afterwards, I compute the predictions on both sets and then measure the accuracy, together with the AUC.

4.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is another statistical method for the classification of data. Its objective is to find an optimal linear transformation so that the original data is transformed to a lower dimensional space and the class separability is maximized[28].

Indeed, in LDA, the data space is divided into K distinct regions, which represent the classes, and classification is performed by allocating a variable to one specific class if it belongs in the region; the predictions are computed through the use of linear combinations of predictors[29], because LDA assumes a linear distribution of data.

Generally, it is used especially in the cases of multi-class classification, as it is more stable than logistic regression. The two differ in their underlying assumptions of the explanatory data: while Logistic Regression has no assumptions, Linear Discriminant Analysis assumes normally distributed predictor variables, which is also the reason why LDA can only handle numerical predictors, as they can be standardized.

As a matter of fact, the assumptions of LDA are generally two: the Gaussian (or normal) distribution of the predictor variables and the homoscedasticity, meaning the homogeneity of the variance[27].

For the model *lda.fit*, I used the function *lda*() from the *MASS* R package[30].

First, I removed the categorical variables from my training and validation sets. I remained with only one variable, *Price_USD*, which I then scaled because, as I said in the theory paragraph, LDA assumes Gaussian distribution.
I did the same for the validation set and then computed both the training and validation predictions. I used two different thresholds when computing the predictions on the validation set, since I prefer to have a higher sensitivity with respect to the specificity, and therefore I had to find the threshold that would give me the best results.

4.3 Regularization Techniques

When dealing with high dimensionality data, which presents a large number of features, one of the major problems is overfitting. Overfitting is very common and represents a serious challenge for the traditional learning methods, whose performance worsens as the noise of the dataset increases. Indeed, if the data collected has a high level of noise it becomes complicated for the model to find patterns and useful knowledge, which is why dimensionality reduction is one of the most popular techniques for dealing with noisy and high dimensionality datasets.
There are two kinds of dimensionality reduction techniques: feature extraction and feature selection. We have already seen Linear Discriminant Analysis, which is a feature extraction technique, together with, for instance, Principal Component Analysis, which however I did not use in my work.[31]
Feature selection is still a complex problem to solve, as it involves understanding and defining which is the set of optimal features without much loss of information.[32]
The techniques I will use in my analysis are Ridge Regression, Lasso Regression, Elastic Net Regression, and Grouped Lasso, which all work through shrinking (thus why they are also called Shrinkage Methods) the coefficients of the least contributive variables towards zero.

4.3.1 Ridge Regression

Ridge regression is realized once the likelihood function is maximized with a penalized parameter which is applied to all the coefficients, apart from the intercept.
The estimator is dependent on the choice of λ, which is the tuning parameter and will be determined separately[33], generally with a k-fold Cross-Validation approach.

The coefficient estimates are the values that minimize the Residual Sum of Squares (RSS) in the following function:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

The $\lambda\sum_{j=1}^{p}\beta_j^2$ term is called the shrinkage penalty (it is also known as the L2-norm): when λ, the tuning parameter, is 0, the penalty term has no effect, while as λ increases, the penalty also increases and the coefficients approach 0[27].

In order to fit the Regression model, I first used the function *model.matrix()* from the package *stats*, which, as reported in the official documentation, "creates a design (or model) matrix, e.g., by expanding factors to a set of dummy variables (depending on the contrasts) and expanding interactions similarly"[34], and I applied it to both the training and the validation set.

Then, I performed a Cross-Validation in order to find the best model given a value of λ and plotted both the AUC for each value of λ, as seen in **Figure 4.1**, as well as the visualization of the shrinking as λ increased (**Figure 4.2**).

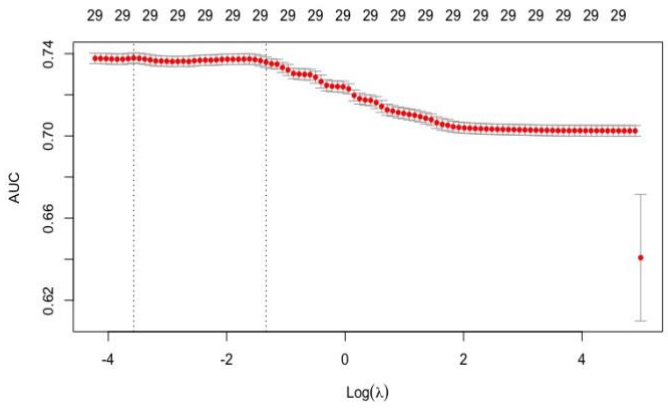


Figure 4.1 Ridge: variations of the AUC as Log(λ) increases

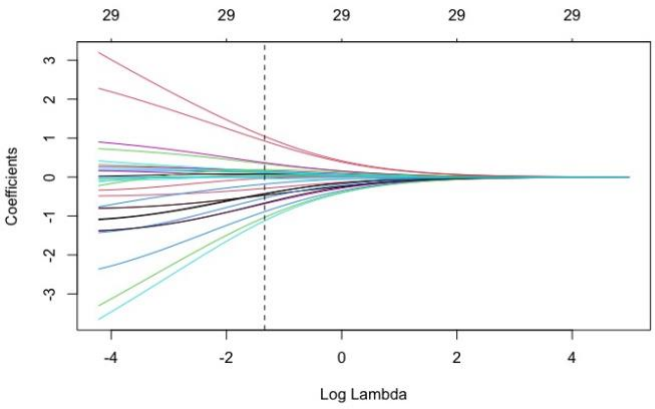


Figure 4.2 Ridge: shrinking with the increase of Log(λ)

I decided to train the model using the value of *lambda.1se,* as it is generally preferable.

I fit the model *ridge.1se* through the function *glmnet()* of the r package *glmnet* and compute the predictions, with a threshold of 0.5. I look at the accuracy and then do the same for the validation set, this time using both a threshold of 0.5 and 0.2.

4.3.2 Lasso Regression

Lasso Regression is an alternative to Ridge Regression. In fact, Ridge has one particular problem, which is that the coefficients can be shrunk towards zero but will still remain in the model, as they will never be *exactly* zero. In other words, in Ridge Regression, no variables will be excluded in the model, and therefore there is no reduction in the number of predictors. However, Lasso overcomes this problem. Indeed, in this case the formula will be:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

We see that the only change is in the term $\lambda\sum_{j=1}^{p}|\beta_j|$, where the term β_j^2 which we found in Ridge Regression, is now $|\beta_j|$. This term, $\lambda\sum_{j=1}^{p}|\beta_j|$ is called the L1-norm and is the penalty applied in Lasso for the shrinkage[35].

Moreover, also in the case of Lasso, λ will be estimated through Cross-Validation.

Therefore, I perform Cross-Validation on the model matrix I had constructed earlier on and plot both the AUC (**Figure 4.3**) and the shrinkage of the coefficients to 0 (**Figure 4.4**).

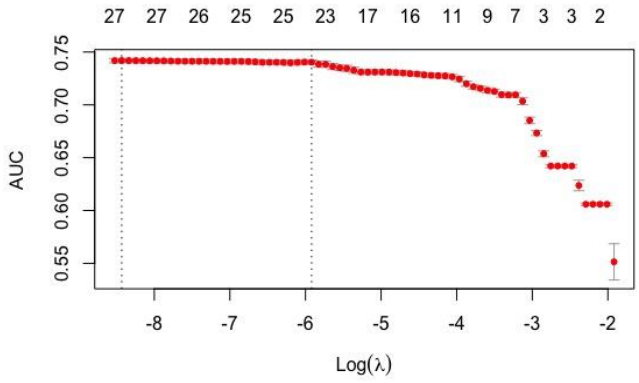


Figure 4.3 Lasso: variations of the AUC as Log(λ) increases

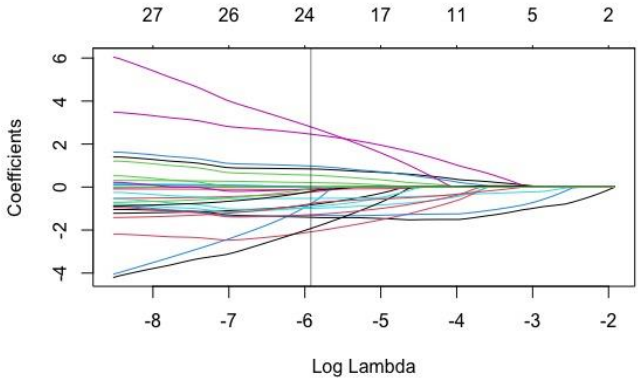


Figure 4.4 Lasso: shrinking with the increase of Log(λ)

From **Figure 4.4**, it is obvious the difference between Lasso and Ridge, as previously, the predictors were approaching zero, while in Lasso Regression's case, they are directly brought to zero.

As before, I use the *lambda.1se* and fit the model with the function *glmnet(),* for the training set. Thus, I compute the predictions and measure the accuracy and do the same for the validation set.

4.3.3 Elastic Net Regression

Elastic Net Regression, like Lasso, does together variable selection and continuous shrinkage, although it also has a "grouped selection" ability.
The main characteristic of the Elastic Net is that the penalty is a convex combination of the Lasso and Ridge penalties[36], which enables the algorithm to shrink some coefficients and also set some to exactly 0. Indeed, it is penalized both by the L1-norm and the L2-norm making it effectively a combination of Ridge and Lasso.
Elastic Net will print two parameters, α and λ. The former, α, is the mixing parameter and can have a range of values from 0 to 1, in particular:

- $\alpha = 0$, for Ridge Regression
- $\alpha = 1$, for Lasso Regression
- $0 < \alpha < 1$, for Elastic Net Regression

In fact, α balances the proportions of Ridge and Lasso penalties in the model, thus, for α close to 0, the L2-norm will be prevalent, while for α close to 1, the L1-norm will.
The parameter λ, instead, is the numeric value determining the amount of shrinking and can range between 0 and infinity.
For the implementation of the Elastic Net, just as I did with Ridge and Lasso, I first set up a 10-fold Cross-Validation on the training set.
Then, I use the *train()* function, from the r library *caret,* to fit the model and find the best parameters λ and α. In my case, the best value of α was 0.1, which means the L1-norm is definitely predominant as α is close to 0, while λ is 0.029.
I then fit the model with the function *glmnet(),* using, as before, the training model matrix.
As before, I computed the predictions on the training set and used the default threshold of 0.5, and on the validation set, with also the 0.2 threshold.

4.3.4 Grouped Lasso

Grouped Lasso is a variant of Lasso which introduces an extension to the Lasso penalty.
In fact, Lasso Regression has two specific problems: on one hand, when there are categorical variables in the dataset, it uses the dummy variables, which leads to a selection of individual instances instead of the variable itself. On the other hand, it is also dependent on how the

dummy variables are encoded, as when changing the contrasts for a categorical predictor, the solutions change consequently.
The Grouped Lasso, instead, performs variable selection in a group manner and remains invariant under orthogonal transformations like in the case of Ridge.
As I mentioned in the theory part, Grouped Lasso selects the explanatory factors at the group level, thus, I wrote for loop in order to get the groups, by grouping the various instances of the variables, and then used the function *cv.grpreg()* from the r package *grpreg* to fit the model.
As with the other regularization models, I used the training set transformed into a matrix, and in this case I also used the function *as.matrix()* on the response variable, both for the training and validation set.

4.4 Quadratic Discriminant Analysis

Up until now, I have considered models which are represented by a linear equation of the form of $Y = b_1X_1 + b_2X_2 + \cdots + b_nX_n$, meaning, simply, that its terms are added, instead of being multiplied or divided. However, they sometimes might be limited in their predictions, as the assumption of linearity is often an approximation.
For this reason, I will now be using models that go beyond linearity, such as Quadratic Discriminant Analysis, K-Nearest-Neighbors, and classification trees.

Quadratic Discriminant Analysis differs from Linear Discriminant Analysis because it assumes that every class has its own covariance matrix. Indeed, LDA, with its assumption of a common covariance matrix for all classes, is much less flexible than QDA, and in the case of a wrong assumption of common variance, the accuracy of the predictions can be affected greatly.
Furthermore, the LDA assumption of linearity of data is very strict, which can be another reason for low accuracy, as it is rare that the experimental data is linearly distributed.
Instead, QDA, which is generally preferred if the training set is particularly large or if the assumption of common variance is impossible to maintain[27], does not assume linearity in data, but rather a quadratic distribution, which again makes QDA a more flexible and thus a more suitable option, in general.
As we have seen for Linear Discriminant Analysis, also in this case the training and validation set must be all numerical and scaled, therefore I reutilize the same datasets as LDA and I fit the model with the function *qda()* from the r package *MASS.*

I computed the predictions on the training and validation set and measured the accuracy, both with the mean and with the AUC.

4.5 K-Nearest-Neighbors

KNN is a very popular method for data mining and statistics and is a simple but effective approach to classification.
The concept behind KNN is to take as input a *k*, which is the number of nearest neighbors that are considered when computing the predictions. With "nearest neighbors", we mean the data points which are up to a certain distance, that is generally the Euclidean distance.
Consequently, the labels are assigned by the majority rule, meaning that they are predicted based on the majority class of its k most similar data points in the feature space.
Since I need to compute the distance between data points, KNN can only handle numerical data, even better if normalized.
For implementing KNN, I first wrote a for loop, whose objective was to find the best k for my model. In fact, the choice of k is very important as it impacts the accuracy of the model, and the loop finds the k with the best accuracy up until k = 130.
After I found the best k, I used it to fit the model, for which I used the dataset with numerical scaled variables, in my case only *Price_USD*.
I also plotted the accuracy of the various k that have been considered, in order to understand whether the model was overfitting or not.
Then, I computed the predictions on both the training and the validation set and used the mean and the AUC to measure the overall accuracy.

4.6 Tree-based Algorithms

In this section, I will discuss the theory behind the algorithms based on decision trees, as well as how I implemented them.
These kinds of models are strongly non-linear, because, as I will mention later in the chapter, they mimic the thinking processes of the human mind.

Decision trees are another set of methods for supervised learning, meaning they are made from pre-classified data. They are sequential models that mimic the human decision-making process,

as they logically combine a sequence of simple tests[37]. Like normal trees, they have a root node, which is the parent node, branches nodes, representing the links and containing the rule, and leave nodes, which show the outcomes.

The concept behind it is to segment the predictor space into regions through a recursive binary splitting, which is a greedy approach, starting from the top of the tree, when all data points belong to the same region. Each split is determined by a predictor variable and a cut-point that minimizes the error in the region, meaning the percentage of observations in the region that do not belong to the prevalent class of that region. The error, in this case, is the misclassification error. The split is determined by the feature that aptly divides the data so that the observations are also split according to the values of those features, as each data point is assigned to the class predominant in the region it belongs to.
Furthermore, as the misclassification error is not always considered to be the best to measure the quality of a decision tree model, I used the Gini Index, or Gini impurity, which is a function used to measure the quality of the split and has a range from 0 to 0.5. However, I also built another model using the "deviance" split method, which generally gives more clear and comprehensible trees.

Down below, I discuss the different algorithms I will be using: decision tree, which I then tried to improve through pruning, Random Forest and Boosting, which are both based on decision trees and are generally more accurate.

4.6.1 Decision Trees

I decided to start with a "simple" decision tree, which is the basic form of tree-based algorithm, as it only uses one tree, because I preferred to begin with a model which would have been more easily comprehensible.
Indeed, decision trees mimic the thinking processes of the human brain, which makes them more easily interpretable to humans, as the logic behind their decisions is much more similar to ours.
However, I recognized that with this method the risk of overfitting was pretty high, thus, I attempted an additional technique, called Pruning, in order to remove any part of the tree which did not provide any help in the classification of instances.

Pruning starts from the recursive splitting in the creation of the larger tree, which is then reduced to a smaller sub-tree. Each sub-tree is associated to α, a positive tuning parameter. When α increases, the more there are terminal nodes, the higher will the price be, as α controls the trade-off between the fit to the data and the model's complexity.

For the simple Decision tree, I first fit one model with the Gini Index and then one with the deviance. For both trees I used the training set with the six variables I had selected for the analysis and did the same for the predictions.
I computed the predictions on the training set and validation set and measured the accuracy, together with printing the confusion matrix. Finally, for both models, I also measured the overall accuracy with the AUC, on the training as well as on the validation set.
Once I had the simplest models, I attempted to improve the results with pruning.
First of all, I use cross-validation, with the function *cv.tree()* from the r package *tree,* to find the optimal tree complexity in terms of the number of terminal nodes in each tree and use that number to fit the pruning model. After cross-validation, I found that the optimal number was 84 for the first model, *fit.tree*, and 5 for the second model, *fit.tree2.* Thus, I used them to fit the model.
I used the function *prune.misclass()* also from the r package *tree* and I applied it for both previous decision tree models.

4.6.2 Random Forest

Sometimes pruning might not be as useful as we would like for improving the models.
Indeed, a decision tree might be too simplistic a model for my data, which is why I decided to also use the Random Forest technique.
Random Forest is an efficient method for different kinds of datasets and can also handle very well categorical variables. It has many advantages over the decision tree: for instance, it mostly overcomes the problem of overfitting (this does not mean that it cannot overfit the data, only that is more robust against it) and it does not need pruning; moreover, it is not very sensible to outliers, of which I have a significant number in my data.
In particular, Random Forest is a group of classification (or regression, but this is not my case) trees, all unpruned, that are built from a random selection of samples of the training data, and the Random features are selected in the induction process.

The extent of each tree is grown to the largest possible.
For these reasons, in general, Random Forest has higher performance compared to decision trees.
Actually, Random Forest is an improved extension of another technique, Bagging, which is an ensemble algorithm.
The idea behind bagging is to fit a model to each sample of the training data and then combine the predictions from all the models, which reduces the variance and improves the accuracy of the predictions. The way that it differs from Random Forest is that in RF each tree is grown with a randomized set of predictors, which is also the reason for its name "Random Forest"[38].

I first implemented bagging, then Random Forest.
In order to perform bagging, I have to use the *randomForest()* function from the r package *randomForest* and I set the mtry equal to the number of predictor in the dataset, which is 5. Then, I also set *importance = T,* in order to assess the relevance of each predictor.

Afterwards, I fit Random Forest with the same function and I set the mtry at 3, as typically, the size of the sample of predictors should be equal to the square root of the number of predictors. Actually, as my predictors were 5, the square root of 5 was 2.24, but I decided to use 3 because I obtained a higher accuracy.
To find the best number of trees, I use a for loop that fits multiple models and measures the accuracy for every number of trees between 1 and 200 and I then use the number for which the accuracy was maximized. As the number was 191 trees, I fit the model and then compute the predictions on both training and validation set. I also set *importance = T* to check the importance of the predictors and compare the result with the one I obtained with bagging, which I will comment in the next chapter.

4.6.3 Boosting

By "Boosting", in general, it is meant a method used to boost the accuracy of any learning algorithm. The boosting algorithm I used is AdaBoost, or Adaptive Boosting.
The idea behind AdaBoost is to combine multiple weak learners to create a strong classifier.

Indeed, in AdaBoost, the weak learners are the simplest decision trees possible as they are trees with only one split and are called decision stumps[39]. They are combined together so to give better results than random guessing, as they eliminate bias and therefore boost performance.
The process behind it starts with the first weak classifier which is fit on training data, where every unit is initially given equal weight, and then the units are re-weighted at each step. If an observation has been misclassified, this approach will assign more weight to it, in order to try to get the right prediction. This last part is the reason why it is called AdaBoost, "adaptive boosting".
In order to implement boosting, I fit the model with the function *adaboost()* from the r package *fastAdaboost* and compute the predictions on the training set. I do the same for the validation set and I estimate the accuracy, both as *1- error*, and with the AUC.

CHAPTER 5

Predictive Analysis: Results

5.1 Logistic Regression

The results I had with logistic regression were not the best of my entire predictive analysis, although in general they were pretty good.
Overall, the accuracy on the training set was about 67%, while the sensitivity 61%.
On the validation set, the results were essentially the same, as the accuracy was 67% and the sensitivity 62%.
One aspect that I found interesting, was the significance of some levels of the variables. For instance, from the Crypto variable, ETH, MANA, SAND and WETH were considered as very significant (*** next to the variable). We have seen before that those were the most common cryptocurrencies, and it makes sense that their significance would be high.
However, for what concerns the collections, I noticed that many smaller ones also had high significance, like *Mirandu*, or *Gegodego*. This seemed strange to me at first, as they are two less known metaverses, but I suppose that their significance can be attributed to them representing a certain kind of transactions, more specifically niche transactions, that probably have different patterns than other transactions in the main metaverses, and are therefore more useful to the predictions.
Finally, for what regards the AUC for this model, despite this being a "simple" model, it was still 0.73, and therefore was a much higher AUC than some other most complex models I will discuss later in the chapter.

5.2 Linear Discriminant Analysis

The first thing I must mention, is that because I computed LDA with only one predictor variable, *Price_USD*, the performance of the model was clearly affected.
Indeed, the accuracy of the predictions on the training set was 52%, which was much lower than the one I had with the **Logistic Regression model**, and was not a particularly encouraging result in general.

For the validation set, I tried the predictions with the 0.5 threshold, which also gave an accuracy of 52%, however, the sensitivity was incredibly low, as it was around 0.009. Therefore, I lowered the threshold to 0.2 in order to find out if the results would change. However, 0.2 gave even worse results, as the predictions were only 1, and never 0, apart from the accuracy which was even lower, at 47%.

For this reason, I decided to look at the minimum of the probabilities in the vector, and found that the minimum was actually 0.4745318, which explained why, if I lowered the threshold to 0.2, the predictions were entirely 1. Indeed, the accuracy was 47%, which meant that probably the model was assigning randomly the classes to the data, leading to an inaccurate prediction.
Furthermore, few predictions had a probability higher than 0.5, which made it complicated to find a good threshold. However, after some trials, I managed to find an optimal threshold that was satisfying enough for my analysis. The threshold I used was 0.4746, because in this way, my accuracy was much higher than before (63%) and the sensitivity was also a pretty good 66%.

Now, there are two aspects that I need to mention here: first of all, I could have chosen another threshold which could have given me a particularly high sensitivity, at 99%, but this would have meant that also the False Positive error was going to increase. If this was the case, and the model had been used for really predicting whether a NFTs was going to be sold on the primary market or not, it would have meant a huge loss for the seller. Indeed, the model would incorrectly predict as resold too many NFTs, leading the seller to not understanding which NFT would be worth selling or not. In case of an incorrect prediction, the seller would bring on the market an NFT that does not actually have the potential to be sold on the secondary market, resulting in a loss of time and potential to the seller who could have been creating new NFTs or concentrating on others instead of this one.
For this reason, I preferred a more balanced result, even when considering the False Negative percentage, 34%, which I considered it to be pretty good, especially considering the accuracy of the model.

I also need to mention my assumption as to why the model is performing so badly: the reason stands in one of the assumptions of Linear Discriminant Analysis, the linearity of the data, which is particularly stringent, especially when considering that the only variable used as predictor, *Price_USD*, is clearly impossible to consider as linear. Because of this assumption,

the model is way too simple for dealing with this sort of data, and it leads to a huge loss in accuracy.
This can be shown also directly by the AUC, which is 0.68, even less than Logistic Regression.

5.3 Ridge Regression

The results on the training set for Ridge Regression were similar to those of **Linear Discriminant Analysis**. Indeed, the accuracy was around 54% although the AUC was not particularly low, as it was even on the higher side, considering it was 0.73.
However, with the training set and a threshold of 0.5, the sensitivity was only 8%, while the False Negative percentage was 91%.
Thus, I looked at the predictions on the validation set. For what regards the predictions with the threshold of 0.5, they were basically the same of the ones on the training set, with 54% accuracy, 8% sensitivity ad False Negative percentage of 91%.
Setting the threshold at 0.2 gave slightly better results as the accuracy increased to 63%, while the sensitivity increased to 40%.
Finally, the AUC on the validation set predictions was 0.728, which still indicated an overall good performance.

5.4 Lasso Regression

Overall, the results were better than with **Ridge Regression model**.
On the training set, with the default threshold of 0.5, the accuracy of the predictions was 62% and the sensitivity 39%, though the False Negative percentage was 60%, which was still quite high.
On the validation set, the results with the different thresholds differ significantly. In fact, if we set 0.5, the accuracy and the sensitivity are the same as with the training set; however, if we set the threshold at 0.2, the accuracy will increase to 66% as well as the sensitivity which will increase up to 58%. Furthermore, the False Negative error will decrease to 42%, which is a very good improvement.
The Lasso Regression model, finally, has an AUC of 0.73, which is definitely not bad, as it is in line with the best ones we had so far.

5.5 Elastic Net Regression

The accuracy of the predictions on the training set was around 68%, which was also the accuracy on the validation set with the same threshold.
Indeed, for the 0.5 threshold, the results are the same for both the predictions training and the validation set, which is also the reason why I decided to use the 0.2 threshold.
The results with 0.2 were more encouraging: although the accuracy was slightly lower, at 61%, the sensitivity was actually a very high 93%. However, the False Positive percentage was also high, at 67%.
The results were slightly higher than with **Lasso Regression** and especially higher than **Ridge**. This might have happened because Elastic Net removed some features that were possibly generating noise, leading to a more accurate model. Furthermore, the fact that the results were particularly close to the ones obtained with Lasso, is also due to the parameter α, which, as mentioned before, is very close to 0, meaning a prevalence of the L1-norm, Lasso's penalty.
Moreover, the AUC was 0.73, still close to the one for Ridge Regression, which was 0.728, and the same as Lasso.

5.6 Grouped Lasso

The overall accuracy, both on the predictions on the training set and on the validation set, was in line with the other regularization techniques: in fact, the accuracy on the training set with a threshold of 0.5, was 67%, as it was on the validation set.
With the threshold of 0.2, the accuracy of the predictions on the validation set was 61%, which was slightly lower, but the sensitivity was much higher, because it went from 62% to 93%. Another good aspect is that the False Negative percentage decreased to 7%, although the False Positive error increased significantly, from 28% to 66%
However, The AUC, which was 0.7323, was higher than in other models.
The slightly higher AUC, despite the other results, is probably an indication of what I had explained before in Lasso: despite the method of Grouped Lasso having its problems, it still solves the inaccuracy that Lasso has, and therefore it makes sense that the AUC, the overall accuracy of the model, is higher, even if only of 0.023.

5.7 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis, despite not assuming linearity of the data like in LDA, and therefore not being particularly strict, still assumes a quadratic distribution.
Indeed, the problems that we have encountered with LDA are still not entirely solved with QDA: because the model only handles numerical variables, the only predictor we have is *Price_USD,* and assuming a quadratic distribution for this variable is not very feasible. Indeed, from **Figure 2.8** and **Figure 2.9**, it was clear that the assumption of quadratic distribution was too simplistic and possibly even further away than the one of linearity.
I was not surprised, therefore, when I looked at the results and noticed that the accuracy was similar to the one of **Linear Discriminant Analysis**: in the case of the AUC, actually, it was even worse, as for QDA it reached only 0.61, the worst result among all the methods.

I also used the same method that I had used for LDA regarding the optimal threshold: I had first used the 0.5 threshold for both the training and validation set, but the highest accuracy was reached on the validation set and was 53%, still very low compared to the other models I had used up to now.
Therefore, I looked at the minimum in the predictions, which was 0.1567274, and from there I used trial and error to get to the optimal one. Indeed, I had noticed that with a threshold of 0.5, the sensitivity was one of the lowest, 4.5%, and therefore I tried to find a value between 0.5 and the minimum.
The optimal one, in my opinion, was 0.1618, as it gave the best possible results for what regards sensitivity and False Positive error, at least for what I was looking for, which was a good equilibrium in the two values. It also lowered the False Negative error, although it still remained pretty high, at 87%.

5.8 K-Nearest-Neighbors

Before getting to the actual results, I first wanted to look at the accuracy measured for each k. Indeed, it is shown in **Figure 5.1** that for smaller values of k the accuracy is not entirely low, about 60%, which is probably because it is overfitting the data, and this is also proven

especially by the peak at k = 60. However, for higher values of k, the accuracy tends to stabilize, as the overfitting is reduced.

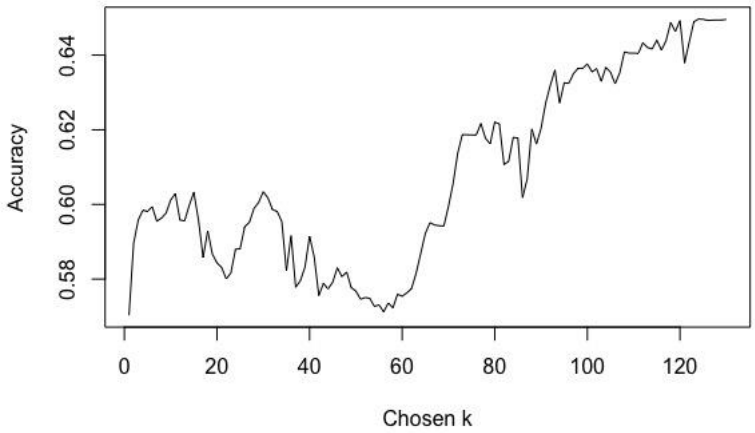


Figure 5.1 KNN: Accuracy variations as K increases

For what regards the accuracy of the final model, despite not being the lowest, it is a not particularly high 60%, which also confirmed by the AUC, whose value is 0.65.
I would say that, as with other models which could only handle numerical variables, the accuracy was negatively impacted by the fact that the model could only be fitted with one predictor, *Price_USD*, which was also a complicated variable to deal with.

5.8 Decision Trees

The tree I obtained specifying the Gini index split (**Figure 5.2**) is much more complex than the one with deviance (**Figure 5.3**). Indeed, from comparing the two trees, we see that the first one is much less interpretable and noisy.

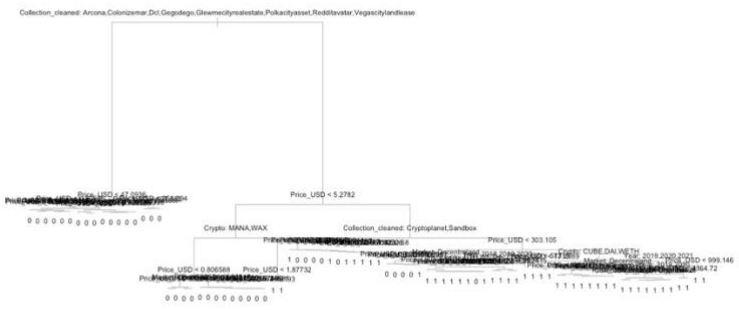


Figure 5.2 Decision Tree with Gini Split

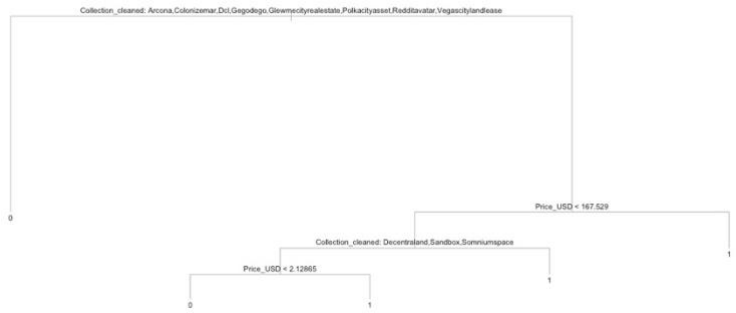


Figure 5.3 Decision Tree with split = deviance

However, the second is a bit too simplistic, which is also confirmed by the accuracy of the predictions on the validation set, that in the case of *fit.tree* (69%) is slightly higher than in *fit.tree2* (68%). However, the sensitivity gave better results with *fit.tree2,* as it was 41%, although the False Positive False Negative rate were also higher.

Finally, the AUC for *fit.tree* was higher than in *fit.tree2,* as it was 0.7723 compared to 0.7277, which again confirms that the second model was slightly too simple.

Now, I look at the results after I try to improve the accuracy with pruning.

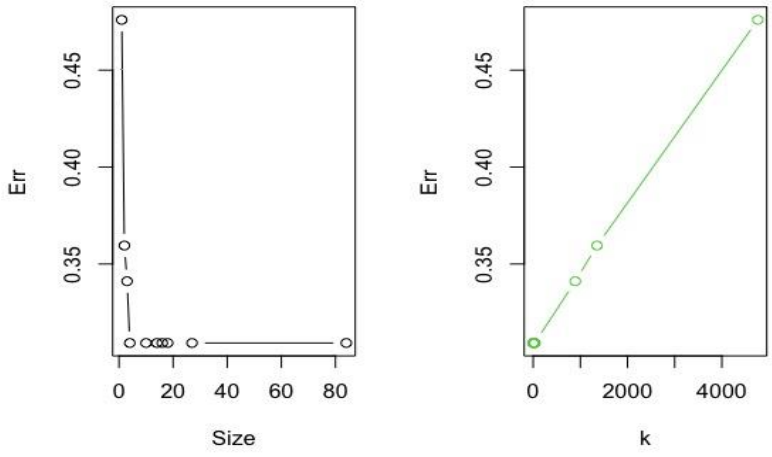


Figure 5.4 Pruning: fit.tree error has size and K increase

First of all, fot *fit.tree,* I look at which is the optimal number of trees for the model: from **Figure 5.4**, the size after slightly more than 0 shows a basically constant error; however, this is only

what is visible in the plot, because there are the other three errors (more than 0.34) that mess up the scale. We can also see that as k grows, the error grows as well.
I did the same thing for *fit.tree2*. Here below, in **Figure 5.5**, the same pattern is shown, where the minimum error is reached at a size 5.

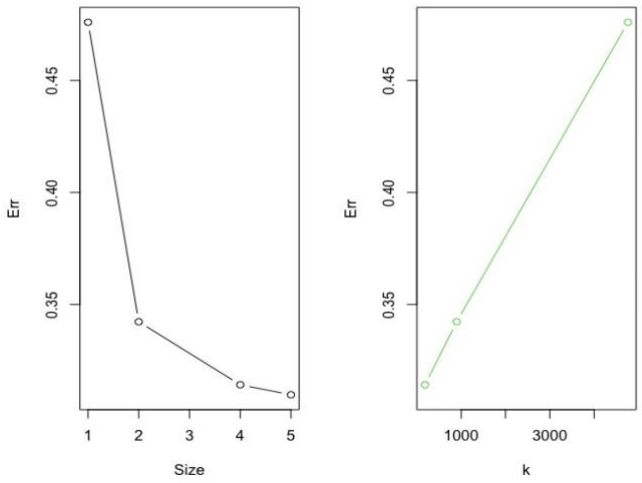


Figure 5.5 Pruning: fit.tree2 error as size and K increase

However, the accuracy, especially measured by the AUC, did not increase significantly. Furthrmore the trees were actually the same as without pruning: indeed, the size that minimizes the error according to Cross-Validation, is the same which I had obtained with the decision tree. This might be because the original trees did not have great overfitting issues.

5.9 Random Forests

The results I had with bagging were slightly better than with the simple decision tree, even after pruning, but also worse than Random Forest.
Indeed, the AUC of the predictions on the training set was 0.955, which I supposed is a sign of overfitting, although the AUC on the validation data predictions is still a very good performance, especially compared to the one we have obtained in previous models.
Moreover, I wanted to look at the importance that the model gave each variable, especially comparing it to the results I obtained with Random Forest, to find if we could have some additional insights into the data.

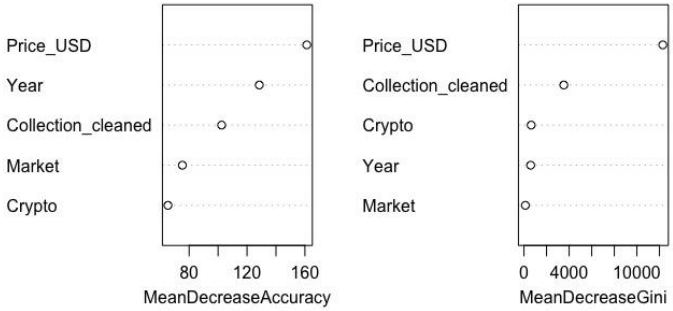


Figure 5.6 Bagging variable importance

Model importance is measured as how much the predictor variable contributes to the accuracy. Indeed, we notice that in **Figure 5.6**, *Price_USD* is the most important predictor, as the accuracy with this it is at the highest. Moreover, *Collection_cleaned* and *Price_USD* are both very impactful on the homogeneity of the nodes, (*Price_USD* in particular), while the three variables *Crypto*, *Year*, and *Market* are all very close and have a very limited effect on the homogeneity.

Instead, Random Forest (**Figure 5.7**) considers *Collection_cleaned* to be the most important variable, directly followed by *Price_USD.* Overall, the mean decrease in Gini coefficient is even more pronounced: again, *Crypto*, *Year,* and *Market* are very close, with a huge detachment from both *Collection_cleaned* and *Price_USD,* which instead have a big impact on the homogeneity of the nodes.

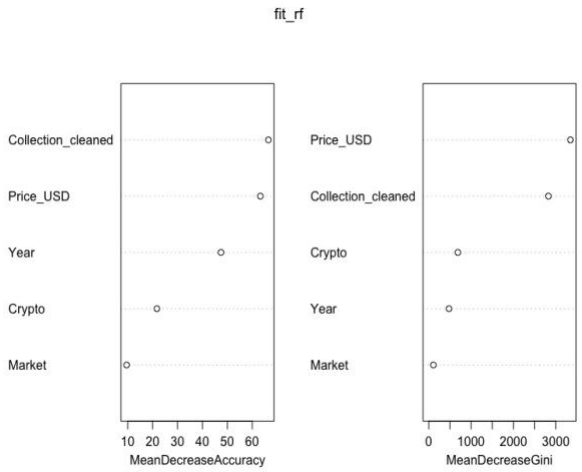


Figure 5.7 Random Forest variable importance

The fact that *Price_USD* is always somehow significant, is probably because in general people firstly look at the price when considering whether to buy something or not, and, moreover, the fact that *Collection_cleaned* is important, also stresses the flow with which a transaction is made: for instance, I am a user of a specific virtual world and I buy, based on the price, whatever NFT I prefer from that virtual world, or that collection in our case.

Nevertheless, Random Forest gave the best results out of all the models. The accuracy was 72%, while the AUC was a pretty high 79%, which is not bad considering that these are real life data and that the other models were all around 70% or less.

5.10 Boosting

Although the results were not entirely bad, I was expecting a bit higher than what I have obtained. Indeed, while on the training set it seemed that the accuracy was pretty high, 85%, on the validation set it was significantly reduced, 70%. While 70% is not bad per se, I have obtained way higher accuracies with other models such as Random Forest.
The same can be said for the AUC. Indeed, on the training data, the AUC is very high, 96%, while on the validation data, the AUC is 78%, slightly less than with Random Forest.

5.11 Final Considerations on the Results

Table 5.9 Results of predictions

Technique	Accuracy	AUC	Sensitivity	False Negative	False Positive
LR	67%	73%	62%	37%	27%
LDA (t = 0.4746)	63%	68%	66%	39%	34%
Ridge (t = 0.2)	63%	72.82%	40%	59%	16%
Lasso (t = 0.2)	66%	72.93%	58%	25%	42%
El. Net (t = 0.5)	68%	73%	64%	35%	28%
Gr. Lasso (t = 0.5)	67%	73.23%	62%	37%	28%
QDA (t = 0.1618)	55%	61%	13%	6%	87%
KNN	61%	65%	35%	16%	64%
Dec. Tree	70%	77%	35%	18%	12%
Pruning	70%	77%	35%	18%	12%
R. Forest	72%	79%	33%	14%	13%
Boosting	70%	78%	37%	10%	19%

I will now report some final conclusions on the results I have obtained in the predictions on the validation set, which I have grouped in **Table 5.1**. All the results were the best of out each technique: in particular, for LDA, QDA, and the regularization techniques I chose the thresholds which gave me the best accuracy overall.

The best results were clearly obtained with the tree-based algorithms, where the AUC reached a minimum of 77%, a much higher result than in the other methods. However, Decision trees and pruning were slightly less accurate than Random Forest and Boosting, probably because they were only producing one tree. For what regards Random Forest, overall, it had the best AUC and the best accuracy, and therefore I would probably choose this method if I wanted to compute the predictions with other data.

Nevertheless, I also wanted to discuss the worst results and the reason behind them.

First of all, regarding the results obtained with Linear Discriminant Analysis and Quadratic Discriminant Analysis: the one problem (in my case) that these techniques have in common is that they only handle numerical predictors because they assume linear and quadratic distribution in the data respectively. Therefore, because I only had one numerical variable among, the results I obtained were obviously inaccurate. Furthermore, for both LDA and QDA the assumptions of linear or quadratic distribution on the predictor *Price_USD* were a bit too strict, though from the results it also seems like the assumption of quadratic was even worse than the one of linearity. For what regards KNN, despite having more flexible assumptions, it can still only handle numerical variables and therefore results in being poorly compatible with this specific dataset.

Another important aspect that I noticed, was that Logistic Regression, the easiest and more straightforward method, did not actually give bad results: indeed, although the concept behind is basic, LR managed to get an AUC of 73%, on par with the regularization techniques, and in most cases the accuracy is even higher.

Regarding the regularization techniques, they have basically the same AUC, although it still varies. Elastic net, for instance, is better than both Ridge and Lasso, because is the best possible combination between the two penalties, L1-Norm and L2-Norm. Nevertheless, it is still slightly worse than Grouped Lasso, which is the best out of the four models, as Elastic Net's problem is the fact that, through α, it balances Ridge and Lasso and thus it cannot incorporate Grouped Lasso. However, Grouped Lasso is fundamental in this case as most of the predictors are factors.

In conclusion, if I had to really predict which of my NFTs have the potential to be then resold in secondary sales, I would definitely use Random Forest. Moreover, a sensitivity of 33% is a small price to pay for a much higher accuracy than, for instance, having a sensitivity of 68% in LDA, while the AUC is much lower.

CHAPTER 6

Object Recognition

As I anticipated in Chapter 1, I wanted to analyze the NFTs from a different perspective: indeed, as I had the URLs of the images, I thought of performing object recognition to look for potential new patterns which could have been useful in the analysis.
Object detection is a kind of image classification in which neural networks predict and highlight objects in the form of bounding boxes.

After a bit of research, I found the YOLO[40] v3 algorithm. YOLO stands for You Only Look Once and is a real-time object detection system. It is incredibly fast as it makes predictions of bounding boxes and the class probabilities at the same time, thanks to the use of an end-to-end neural network[41].
However, even despite YOLO's quickness in predictions, I had to reduce the dataset, as 60 thousand observations were too many to download and analyze; I removed all the duplicated URLs and the NAs, and remained with about 11 thousand observations.
I implemented a code in file *images_download.py* to download the images, using the library *requests,* which served to get the content of the URL and the correct extension, and then saved the images in a folder that I called *inputs.*
I then created another file, *object_detection.py,* in which I ran the actual object detection.

First of all, though, because I had some gifs in the dataset, I split the detection between gifs and other kinds of extensions: for what regards the gifs, I added another code snippet in which I only take the first frame of the image and use that frame to perform the analysis.

For the actual object detection, I use the library *cvlib,* and, in particular *cvlib.object_detection* and I performed the analysis; then, I saved the results on a dictionary, called *labels.*
Finally, I saved the dictionary in a *.csv* file, so that I could access it at any time without having to run the code again.

Regarding the results, the YOLO detection system managed to outline some objects out of the images of the various NFTs.

The labels detected are actually a pretty good amount, 34. This is because of the heterogeneous nature of the dataset which is used as input: it contains different kinds of NFTs, as they come from different collections, and therefore the classes are also going to be very different.

However, some objects were much more common than others. In **Table 6.1**, I have reported the labels which have been detected more frequentely than others (I filtered using Occurrences > 10).

Label	Occurrences
person	367
clock	117
bench	85
bird	33
cell phone	22
car	20
traffic light	16
fire hydrant	13
suitcase	11
boat	11

Table 6.1 Most common labels detected with object detection

The most common object detected was "person" with 367 occurrences: although it is peculiar and might seem strange, it is also true that, in my case, the NFTs are all related to the various metaverses, meaning that they might sometimes be NFTs of accessories for the users' avatars, which are probably shown on the avatar itself, or perhaps the *person* is the NFT of a character, or a "skin", like we have seen previously in Wallem Skins (collection *Wallemskin*).

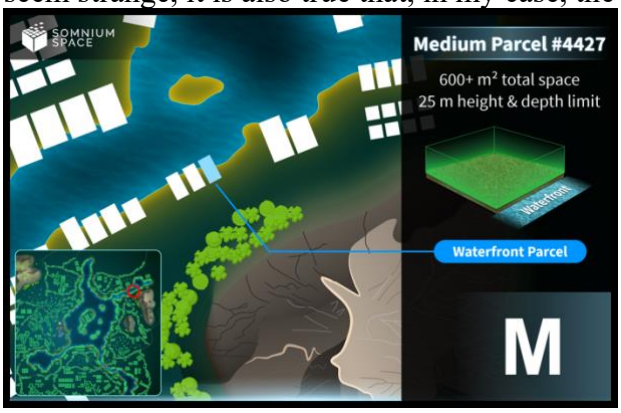


Image 6.2 Land Parcel in Somnium Space

It is, however, important to notice a peculiarity of the majority of NFTs in the dataset: many of them are land, and therefore no objects will be recognized, especially because many of those NFTs are like in **Image 6.1**, **Image 6.2** and **Image 6.3**.

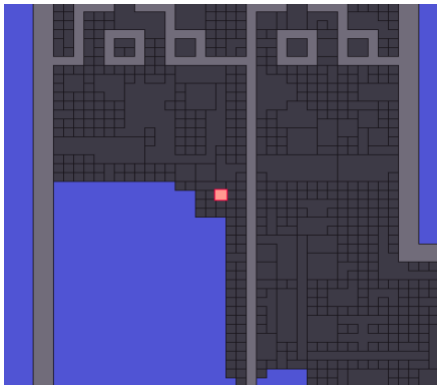


Image 6.1 Land Parcel in Decentraland

Moreover, the YOLO system has also been shown to have trouble in the identification of objects in groups, such as small objects in bigger group.

Of course, on these premises, the detection is extremely difficult, as there is not much to detect in the first place. For this reason, the results I have obtained are not particularly satisfying, especially considering the amount of objects classified compared to the amount of images I have given as input.

Image 6.3 Land parcel in Cryptovoxel

I can, therefore, conclude that object detection did not give many useful insights. However, I must note that if my dataset had been slightly different (perhaps images with more identifiable items), I could have used the labels obtained in my predictive analysis. Indeed, at first I had thought about adding columns with the labels, and use them as additional predictors, but, to computational problems, I refrained from doing it even though I believe that it might be a very interesting approach for future studies.

CHAPTER 7

Conclusions

When I first thought about the topic on which I wanted to develop my final thesis, Facebook had just announced Meta and the world was still amid the Covid-19 pandemic.

Although I was very familiar with virtual worlds, the release of Meta sparked a newfound interest on the topic, and I decided that I was going to turn my curiosity into my final project. However, data on Meta, or metaverses in general, was almost impossible to find and the only dataset I had found was a scraping of 6 million NFT transaction, which was, I must admit, slightly intimidating for me. Indeed, to me the concept of NFTs was, in some way, incomprehensible, and I feared that my lack of knowledge on this topic was going to negatively impact the quality of my analysis. However, after some months spent on working on this thesis, I have built an incredible baggage of knowledge, on metaverses, virtual world, NFTs and on the blockchain in general.

In this paper, I first gave a general overview of the concepts of *metaverse,* or virtual world if you will, and of NFTs. Then, although it is still very limited due to how recent these topics are, I found the most interesting literature and reported it to give a broader picture of the studies and the general opinions on the subject.

I decided that I wanted to study the NFTs and the metaverses and how they related to each other, especially in the context of primary and secondary markets, as I found it to be of particular importance for future considerations of both creators of NFTs, investors, and even companies which want to export some of their assets into the virtual worlds: with this objective in mind, I created a new variable, called *resold,* in which I had "1" when the NFT was resold in secondary sales and "0" if the NFT was only sold once, in primary sales.

I analyzed the data from four different perspectives: I first attempted an exploratory data analysis, through which I could interpret, with graphs and various statistics, the data I was dealing with, the distribution, and the proportions of various levels in the variables. Then, I used clustering analysis to discover patterns in the data that had gone unnoticed before, such as different kinds of transactions with respect to different collections, or how the year might have influenced the secondary sales.

Further, I put myself in the mind of a creator, or a business, and tried different prediction techniques to understand which NFT was going to be sold in the secondary market or not. From

this analysis I obtained very different results, as I used linear and non-linear models, which all had different assumptions. Although not all these methods gave good accuracy, from the results I also had additional insights into the nature of the data or of the relationship the predictor variables had with each other, as well as with the dependent variable.
After the predictive analysis I could conclude that, with this kind of data, the best prediction technique is Random Forest.
Finally, because NFTs are actually about images, I tried the technique of object recognition in order to understand if there were some patterns in the images which might have influenced the transactions. Unfortunately, due to the nature of many of my NFTs, this analysis did not give satisfactory results.

Now that I have reached the end of my thesis, however, I ask myself: what happens now?
The data I used was collected up until the beginning of 2021 and we are now approaching the end of 2022, which means that more than a year has passed, and with these kinds of innovations, in one year anything can change.
Actually, it seems to me that the hype that the metaverse had at the end of 2021 and at the beginning of 2022, has quiet down. At the time it seemed like the revolution of how we looked at reality and it felt as if everything was going to be adapted to these new virtual worlds. However, as the article *The Metaverse Will Reshape Our Lives. Let's Make Sure It's for the Better* from *TIME* points out, nowadays people believe the metaverse to be already gone.
As Facebook lost almost $460 billion dollars of market capitalization, people are out of lockdown and the Crypto world is in decline[42], many are speculating that the metaverse was simply a trend that has passed.
Despite this general opinion, many are still investing in the metaverses. Indeed, the famous consulting firm McKinsey & Company, has estimated that in the first five months of 2022, corporations, venture capitalists, and private equity companies have made more than $120 billion in investments which were metaverse-related. For example, Microsoft, in January 2022, paid $75 billion dollars for the acquisition of the gaming company Activision Blizzard, which, according to them, was going to provide building blocks for the metaverse. The fashion industry is also still investing: for instance, the famous brand Hermès, is venturing into the metaverse with fashion shows and related NFTs. This shows that the metaverse hype has only faded away to the average person, but huge companies understand the potential that it will have in the future.

At the same time, NFTs are also undergoing some adjustments. For instance, due to environmental reasons, Ethereum is transitioning from *proof-of-work* to *proof-of-stake,* in what is called a "Merge". This relates to the way the transactions are validated, which are now being done through the *proof-of-work* system that is very energy-intensive, and will instead be carried out through the *proof-of-stake* that uses far less power[43][44].

Another interesting aspect that is now being discussed is the use of NFTs in law, as it happened in the United Kingdom, where the court allowed the lawsuit to be delivered via an NFT[45].

However, as reported in *Brands, celebs double down on NFTs, but the market keeps tanking*[46]*,* the market of NFTs is not doing so well either. Indeed, especially because the Crypto market has been in decline, the highly speculative NFT market has been feeling the side effects, as the popularity of NFTs is dropping.

From these premises, I cannot very well predict what exactly will happen, although I believe that, as it happens with the Crypto market, there will be periods of recession and periods of prosperity, and we are currently experiencing a recession.

I also feel that the metaverse itself (not necessarily Meta, but the concept of it) will remain for a long time and will keep shaping our reality. We are currently only in the primordial phase of such a powerful technology, and we are building it as we go, removing features that are not liked and adding even more.

All in all, I believe these innovations are now ingrained in our society and they are here to stay for a long time.

Bibliography

1. Frey, T. The History of the Metaverse. *Futurist Speaker* https://futuristspeaker.com/future-trends/the-history-of-the-metaverse/ (2021).
2. Video gaming soars in popularity during lockdown. https://www.scotsman.com/business/consumer/video-gaming-soars-popularity-during-lockdown-3037377 (2020).
3. Vardomatski, S. Council Post: Augmented And Virtual Reality After Covid-19. *Forbes* https://www.forbes.com/sites/forbestechcouncil/2021/09/14/augmented-and-virtual-reality-after-covid-19/.
4. Counterparty. https://counterparty.io/ (2014).
5. Rare Pepe Directory – Rare Pepes on the Bitcoin Blockchain. http://rarepepedirectory.com/.
6. OpenSea. CryptoPunks - Collection. *OpenSea* https://opensea.io/collection/cryptopunks.
7. CryptoKitties | Collect and breed digital cats! https://www.cryptokitties.co/.
8. Chandra, Y. Non-fungible token-enabled entrepreneurship: A conceptual framework. *J. Bus. Ventur. Insights* **18**, e00323 (2022).
9. Non-fungible tokens (NFT). *ethereum.org* https://ethereum.org.
10. Frye, B. L. *NFTs & the Death of Art*. https://papers.ssrn.com/abstract=3829399 (2021) doi:10.2139/ssrn.3829399.
11. Hofstetter, R. *et al.* Crypto-marketing: how non-fungible tokens (NFTs) challenge traditional marketing. *Mark. Lett.* (2022) doi:10.1007/s11002-022-09639-2.
12. Vidal-Tomás, D. The new crypto niche: NFTs, play-to-earn, and metaverse tokens. *Finance Res. Lett.* **47**, 102742 (2022).
13. Nast, C. Gucci goes deeper into the metaverse for next NFT project. *Vogue Business* https://www.voguebusiness.com/technology/gucci-goes-deeper-into-the-metaverse-for-next-nft-project (2022).
14. Dowling, M. Fertile LAND: Pricing non-fungible tokens. *Finance Res. Lett.* **44**, 102096 (2022).
15. Davis, T. J. Influencing NFT Pricing on Secondary Markets: A case study of Vpunks. 8.
16. Ray, S. NFTs And Smart Contracts. *Lansaar* https://medium.com/lansaar/nfts-and-smart-contracts-6c4c5516d5a0 (2022).
17. Welcome to Decentraland. https://decentraland.org/.
18. Why Colonize Mars? https://mars.cards/whitepaper/why-colonize-mars.
19. Perkmann, M. Cryptovoxels: THIS Is the Future of the Metaverse. *Medium* https://medium.datadriveninvestor.com/cryptovoxels-this-is-the-future-of-the-metaverse-4467326d4102 (2022).
20. The blockchain-based MMORPG from Gala Games | Mirandus. https://mirandus.game/.
21. OpenSea. GLEWME CITY - Collection. *OpenSea* https://opensea.io/collection/glewme-city-real-estate.
22. ETH vs WETH: What's the Difference? | CoinMarketCap. *CoinMarketCap Alexandria* https://coinmarketcap.com/alexandria/article/eth-vs-weth-what-s-the-difference.
23. The Facebook Company Is Now Meta. *Meta* https://about.fb.com/news/2021/10/facebook-company-is-now-meta/ (2021).
24. Gower, J. C. A General Coefficient of Similarity and Some of Its Properties. *Biometrics* **27**, 857–871 (1971).
25. dos Santos, T. R. L. & Zárate, L. E. Categorical data clustering: What similarity measure to recommend? *Expert Syst. Appl.* **42**, 1247–1260 (2015).

26. Mars, C. Gameplay Launches October 29th. *Medium* https://medium.com/@colonizemarsnft/gameplay-launches-october-29th-6b4132752d94 (2021).
27. Gareth, J., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning*. (Springer).
28. Park, C. H. & Park, H. A comparison of generalized linear discriminant analysis algorithms. *Pattern Recognit.* **41**, 1083–1097 (2008).
29. Xiaozhou, Y. Linear Discriminant Analysis, Explained. *Medium* https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b (2022).
30. Ripley, B. *et al.* MASS: Support Functions and Datasets for Venables and Ripley's MASS. (2022).
31. Tang, J., Alelyani, S. & Liu, H. Feature Selection for Classification: A Review. 33.
32. Sechidis, K. *et al.* Efficient feature selection using shrinkage estimators. *Mach. Learn.* **108**, 1261–1286 (2019).
33. The Logistic Lasso and Ridge Regression in Predicting Corporate Failure | Elsevier Enhanced Reader. https://reader.elsevier.com/reader/sd/pii/S2212567116303100?token=CCC5356A819A42079592677679A5D7CA7CE67951865B0DA7538CA7C4A348D4F8176391F3FB0BBA4F2334F0BFB30E3454&originRegion=eu-west-1&originCreation=20220824175638 doi:10.1016/S2212-5671(16)30310-0.
34. model.matrix function - RDocumentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/model.matrix.
35. Tibshirani, R. Regression shrinkage and selection via the lasso: a retrospective. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **73**, 273–282 (2011).
36. Zou, H. & Hastie, T. Regression shrinkage and selection via the elastic net, with applications to microarrays. (2004).
37. Kotsiantis, S. B. Decision trees: a recent overview. *Artif. Intell. Rev.* **39**, 261–283 (2013).
38. Prasad, A. M., Iverson, L. R. & Liaw, A. Newer Classification and Regression Tree Techniques: Bagging and Random Forests for Ecological Prediction. *Ecosystems* **9**, 181–199 (2006).
39. Desarda, A. Understanding AdaBoost. *Medium* https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe (2019).
40. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 779–788 (2016). doi:10.1109/CVPR.2016.91.
41. YOLO: Real-Time Object Detection Explained. https://www.v7labs.com/blog/yolo-object-detection, https://www.v7labs.com/blog/yolo-object-detection.
42. Il mercato delle criptovalute fatica ad invertire il trend ribassista: Weekly Crypto Recap. *Cointelegraph* https://it.cointelegraph.com/news/bitcoin-fatica-a-invertire-la-tendenza.
43. Hayward, D. / A. What the Ethereum Merge Means for NFTs. *Decrypt* https://decrypt.co/108862/what-ethereum-merge-means-for-nfts (2022).
44. Stevens, D. / R. What Is 'The Merge'? Ethereum's Move to Proof of Stake. *Decrypt* https://decrypt.co/resources/what-merge-ethereum-move-proof-stake (2022).
45. UK court allows lawsuit to be delivered via NFT. *Cointelegraph* https://cointelegraph.com/news/uk-court-allows-lawsuit-to-be-delivered-via-nft.
46. Binder, M. Brands, celebs double down on NFTs, but the market keeps tanking. *Mashable* https://mashable.com/article/nft-market-down-opensea-brands-celebrities (2022).