NFT Price Analysis Data Challenge

Ocean Protocol - Desights

AVRIL 30, 2023

philippepro36@gmail.com

Table of contents

Introduction

In the fast-growing world of non-fungible tokens (NFTs), accurately determining their fair market value is crucial for investors looking to make informed decisions about buying and selling. However, the value of an NFT is determined by a complex set of factors that are difficult for humans to evaluate. But thanks to machine learning, we can now process this vast amount of data and extract relevant insights.

That’s why Ocean Protocol and Desights have organized a data challenge challenging participants to analyze NFT transaction data and develop a machine learning model to determine the right price of NFTs based on their characteristics.

The objective of this data challenge is thus to determine the fair market price of NFTs, which can help investors identify undervalued NFTs to buy or determine the optimal selling price for their NFTs. We will analyze the historical sales data of five famous NFT collections – Bored Ape Yacht Club, Mutant Ape Yacht Club, Azuki, Moonbirds, and Otherdeed – using advanced data analysis and machine learning techniques. The data we will use is reliable and transparent, as it comes directly from the Ethereum blockchain.

The insights gained from this analysis will be of great value to investors, enabling them to make informed investment decisions and better manage their portfolios. With a more accurate understanding of NFT value, investors can minimize pricing risk associated with NFT financial products and identify profitable opportunities.

Methodology

In this section we will explain the methodology used for the overall data analysis of this report.

The questions we will address here are as follows:

I. Analyze how the number of daily transactions for the collections has changed over time.

II. Determine the correlation between the number of transactions in a collection and the price of ETH.

III. Determine the correlation between the number of transactions in a collection and its floor price.

IV. What are the most liquid traits (those with the most sales) for each collection?

V. Provide a visual overview of the NFT collections of your choice and its characteristics (e.g. size, type of NFTs, date range)

Questions I, II, III and IV will be answered for each of the five NFT collections, while question V will focus on the Bored Ape Yacht Club collection only. Answering these questions will allow us to determine the transaction behaviour within a collection and to understand which factors influence the price. Question V. will be focused on the BAYC collection only and will allow us to introduce the machine learning model which is also on the same collection.

The methodology used for the explanation and display of the answer will be as follows:

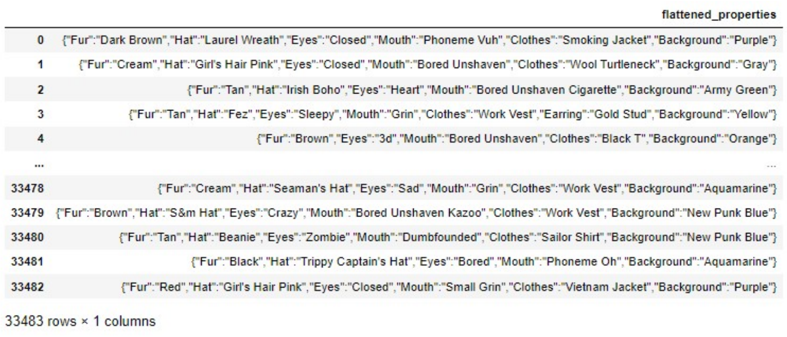
For the questions concerning the 5 collections, we will briefly explain the Python code of a collection allowing us to answer it and then we will display the result for the 5 collections. The graphs will have a colour for each collection to help understanding: Azuki: red, MAYC: green, BAYC: blue, Moonbird: purple, Otherdeed: brown. Finally, we will comment on the results to explain the important information to be drawn from them as well as potential inter-collection similarities.

For the question concerning the BAYC collection we will use the same methodology except that the results and comments will only concern the BAYC collection.

Data pre-processing and data enrichment

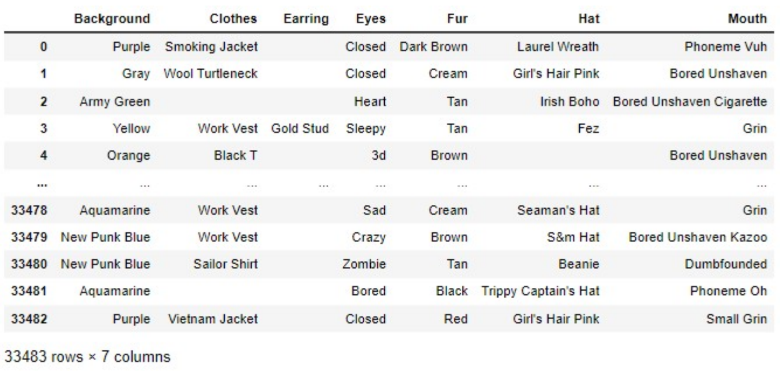
In this analysis we will use data provided by Ocean Protocol and Transpose with all sales transactions of the Azuki, BAYC, MAYC, Moonbirds and Otherdeed collections. For further analysis, we will also use a dataset with the daily price of ETH, a dataset with the transaction volumes of Opensea and 5 other datasets with the daily floor price of the 5 processed collections. All these data are on-chain data fetched from Dune Dashboard.

All these datasets were processed with Power Query to remove duplicates and empty data. The transaction dataset was divided into 5 datasets (one per collection). Special processing was then performed on these 5 datasets using Pandas and Power Query to obtain for each transaction the values of each of the traits of the transferred NFT:



	flattened_properties
0	{"Fur":"Dark Brown","Hat":"Laurel Wreath","Eyes":"Closed","Mouth":"Phoneme Vuh","Clothes":"Smoking Jacket","Background":"Purple"}
1	{"Fur":"Cream","Hat":"Girl's Hair Pink","Eyes":"Closed","Mouth":"Bored Unshaven","Clothes":"Wool Turtleneck","Background":"Gray"}
2	{"Fur":"Tan","Hat":"Irish Boho","Eyes":"Heart","Mouth":"Bored Unshaven Cigarette","Background":"Army Green"}
3	{"Fur":"Tan","Hat":"Fez","Eyes":"Sleepy","Mouth":"Grin","Clothes":"Work Vest","Earring":"Gold Stud","Background":"Yellow"}
4	{"Fur":"Brown","Eyes":"3d","Mouth":"Bored Unshaven","Clothes":"Black T","Background":"Orange"}
...	...
33478	{"Fur":"Cream","Hat":"Seaman's Hat","Eyes":"Sad","Mouth":"Grin","Clothes":"Work Vest","Background":"Aquamarine"}
33479	{"Fur":"Brown","Hat":"S&m Hat","Eyes":"Crazy","Mouth":"Bored Unshaven Kazoo","Clothes":"Work Vest","Background":"New Punk Blue"}
33480	{"Fur":"Tan","Hat":"Beanie","Eyes":"Zombie","Mouth":"Dumbfounded","Clothes":"Sailor Shirt","Background":"New Punk Blue"}
33481	{"Fur":"Black","Hat":"Trippy Captain's Hat","Eyes":"Bored","Mouth":"Phoneme Oh","Background":"Aquamarine"}
33482	{"Fur":"Red","Hat":"Girl's Hair Pink","Eyes":"Closed","Mouth":"Small Grin","Clothes":"Vietnam Jacket","Background":"Purple"}

33483 rows × 1 columns



	Background	Clothes	Earring	Eyes	Fur	Hat	Mouth
0	Purple	Smoking Jacket		Closed	Dark Brown	Laurel Wreath	Phoneme Vuh
1	Gray	Wool Turtleneck		Closed	Cream	Girl's Hair Pink	Bored Unshaven
2	Army Green			Heart	Tan	Irish Boho	Bored Unshaven Cigarette
3	Yellow	Work Vest	Gold Stud	Sleepy	Tan	Fez	Grin
4	Orange	Black T		3d	Brown		Bored Unshaven
...	...	...	...	...	...	...	...
33478	Aquamarine	Work Vest		Sad	Cream	Seaman's Hat	Grin
33479	New Punk Blue	Work Vest		Crazy	Brown	S&m Hat	Bored Unshaven Kazoo
33480	New Punk Blue	Sailor Shirt		Zombie	Tan	Beanie	Dumbfounded
33481	Aquamarine			Bored	Black	Trippy Captain's Hat	Phoneme Oh
33482	Purple	Vietnam Jacket		Closed	Red	Girl's Hair Pink	Small Grin

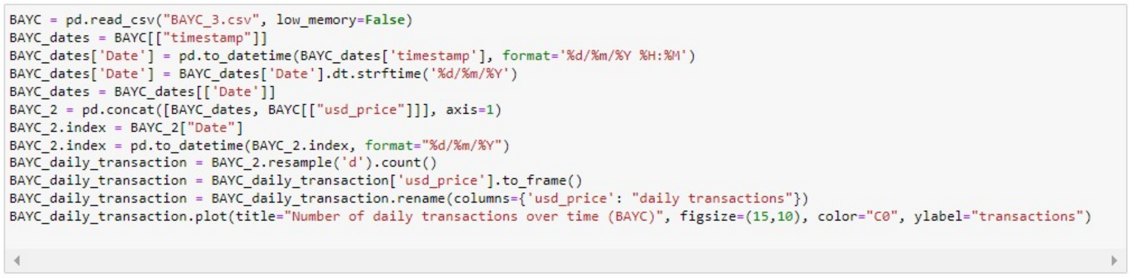
33483 rows × 7 columns

This transformation will allow us to properly analyse the features of the NFTs in each of the 5 collections.

Data Analysis

I. Analyze how the number of daily transactions for the collections has changed over time.

a) Code explanation

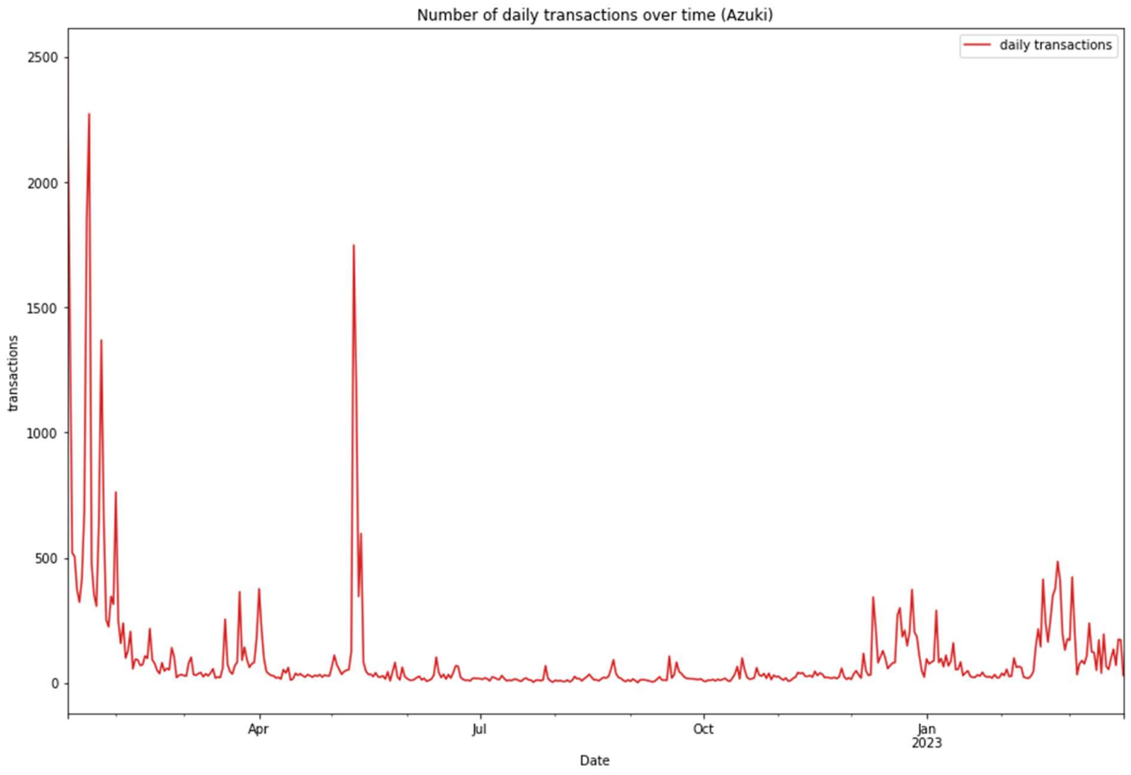


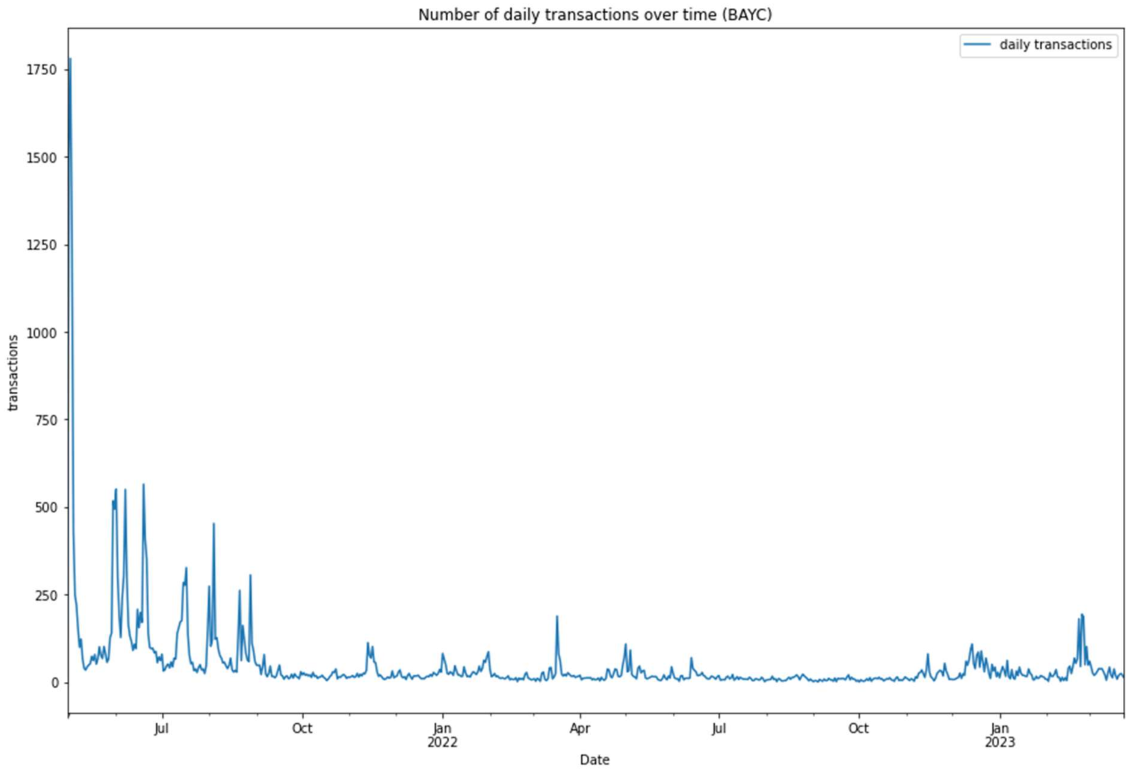
```python
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)
BAYC_dates = BAYC[["timestamp"]]
BAYC_dates['Date'] = pd.to_datetime(BAYC_dates['timestamp'], format='%d/%m/%Y %H:%M')
BAYC_dates['Date'] = BAYC_dates['Date'].dt.strftime('%d/%m/%Y')
BAYC_dates = BAYC_dates[['Date']]
BAYC_2 = pd.concat([BAYC_dates, BAYC[["usd_price"]]], axis=1)
BAYC_2.index = BAYC_2["Date"]
BAYC_2.index = pd.to_datetime(BAYC_2.index, format="%d/%m/%Y")
BAYC_daily_transaction = BAYC_2.resample('d').count()
BAYC_daily_transaction = BAYC_daily_transaction['usd_price'].to_frame()
BAYC_daily_transaction = BAYC_daily_transaction.rename(columns={'usd_price': "daily transactions"})
BAYC_daily_transaction.plot(title="Number of daily transactions over time (BAYC)", figsize=(15,10), color="C0", ylabel="transactions")
```

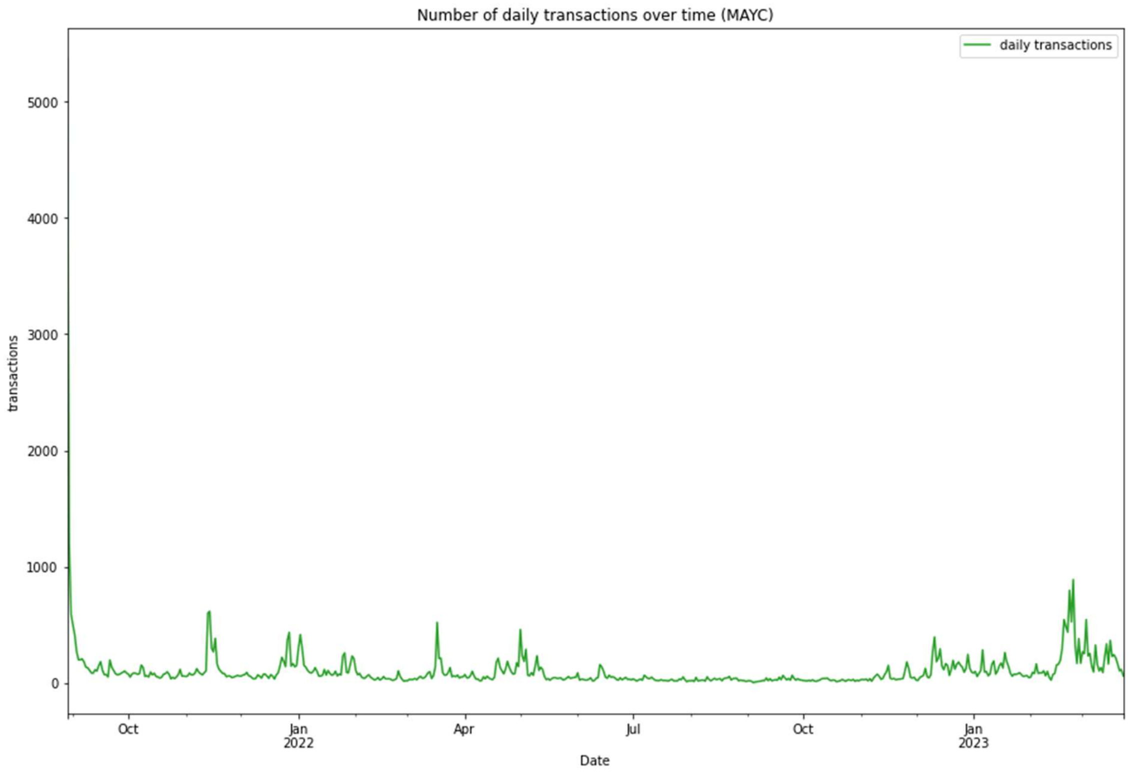
In this code, we first manipulate the transaction DataFrame called BAYC to extract and format the date information from the 'timestamp' column. We then concatenate the resulting dates with the 'usd_price' column of the original BAYC DataFrame to create a new DataFrame called BAYC_2. We set the index of BAYC_2 to be the 'Date' column and resample it at a daily frequency using the resample('d') method. We then count the number of rows for each day using the count() method and store the resulting DataFrame in a variable called BAYC_daily_transaction. We select the 'usd_price' column from this DataFrame, convert it into a new DataFrame and rename the column to 'daily transactions'. Finally, we plot the resulting DataFrame using the plot() method with a specified title and figure size.

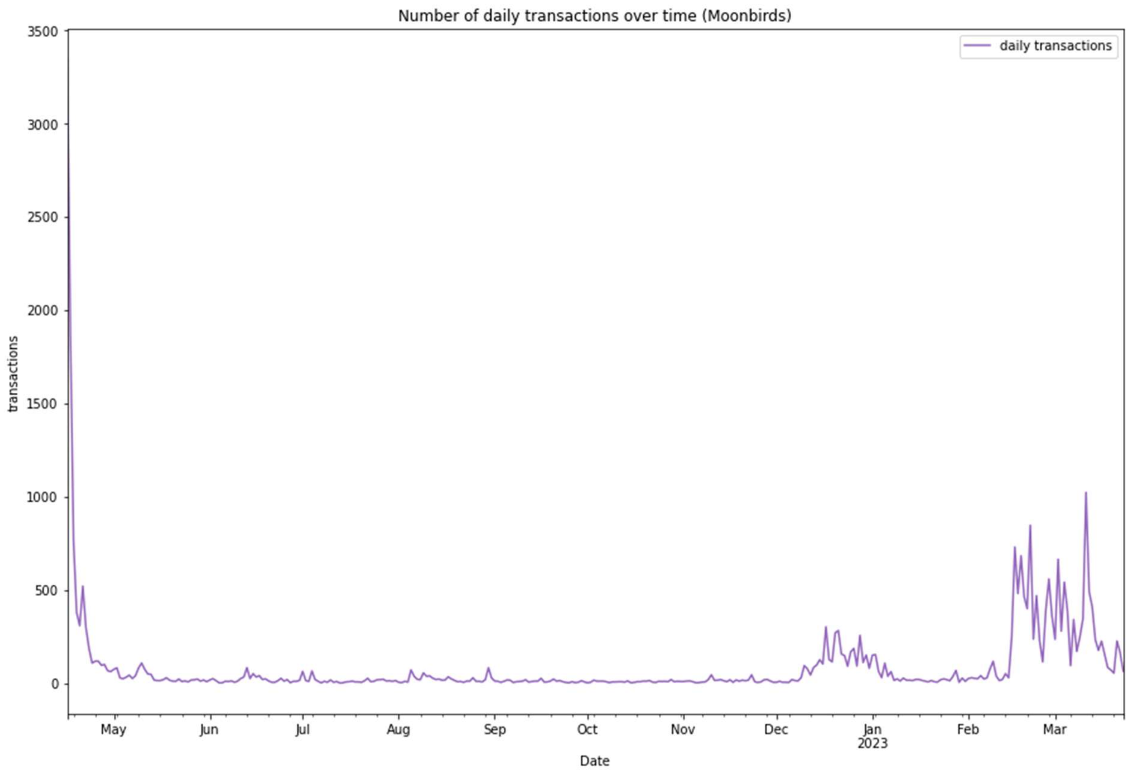
The main purpose of this code is to count the number of daily transactions over time and display them on a graph.

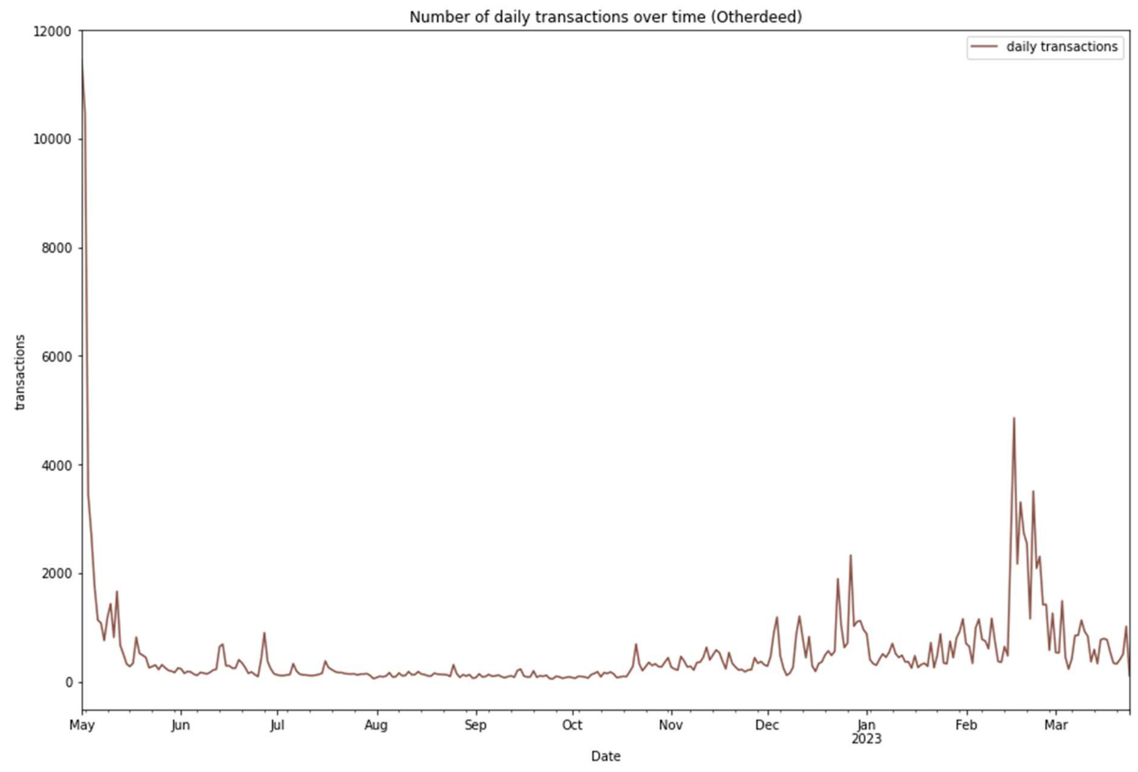
b) Results


Number of daily transactions over time (Azuki)
daily transactions
transactions
Date
0
500
1000
1500
2000
2500
Apr
Jul
Oct
Jan
2023


Number of daily transactions over time (BAYC)
daily transactions
transactions
Date
0
250
500
750
1000
1250
1500
1750
Jul
Oct
Jan
2022
Apr
Jul
Oct
Jan
2023


Number of daily transactions over time (MAYC)
daily transactions
transactions
0
1000
2000
3000
4000
5000
Oct
Jan
2022
Apr
Jul
Oct
Jan
2023
Date


Number of daily transactions over time (Moonbirds)
daily transactions
transactions
0
500
1000
1500
2000
2500
3000
3500
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2023
Feb
Mar
Date

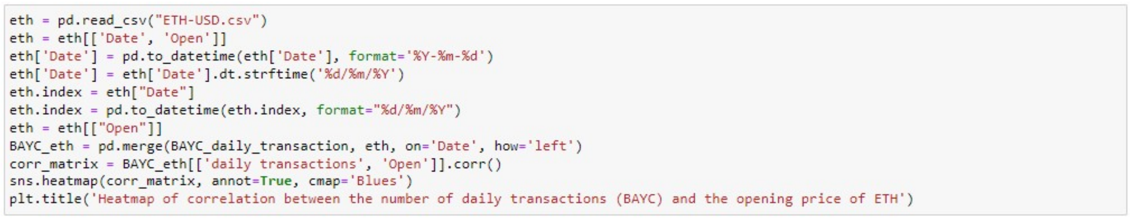


c) Observations

We observe here that the highest number of daily transactions occurs for each collection in the first days of its launch, this is explained by the fact that the mint generates many transactions and that many investors resell their NFTs before the reveal. There is a certain similarity between the 5 collections, indeed 2022 seems to be calm with few transactions which shows a lack of enthusiasm from investors. However, there has been a rebound in activity from December 2022 to the present. We can therefore conclude that the NFT market has seen a decline in interest in 2022 but that investor interest is returning, which is potentially a good investment opportunity.

II. Determine the correlation between the number of transactions in a collection and the price of ETH.

a) Code explanation



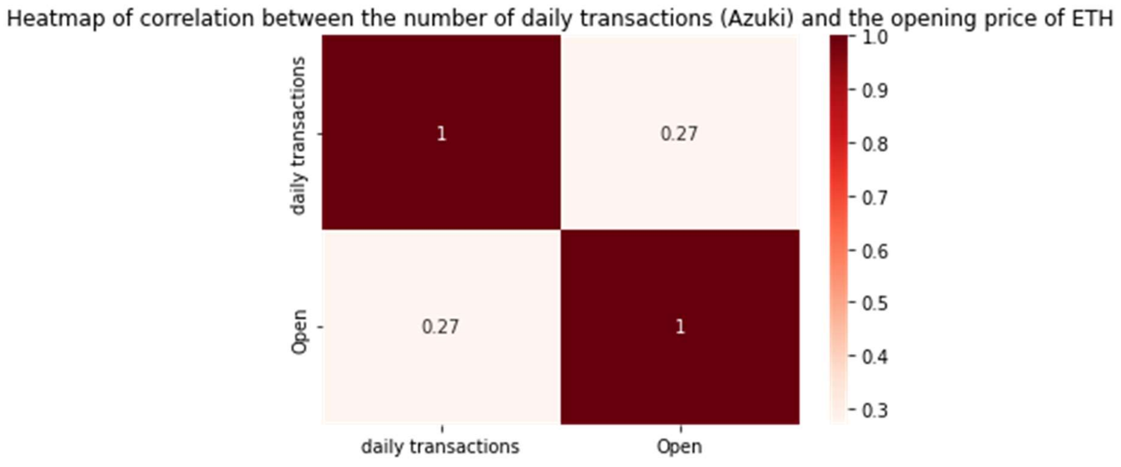
```
eth = pd.read_csv("ETH-USD.csv")
eth = eth[['Date', 'Open']]
eth['Date'] = pd.to_datetime(eth['Date'], format='%Y-%m-%d')
eth['Date'] = eth['Date'].dt.strftime('%d/%m/%Y')
eth.index = eth["Date"]
eth.index = pd.to_datetime(eth.index, format="%d/%m/%Y")
eth = eth[["Open"]]
BAYC_eth = pd.merge(BAYC_daily_transaction, eth, on='Date', how='left')
corr_matrix = BAYC_eth[['daily transactions', 'Open']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='Blues')
plt.title('Heatmap of correlation between the number of daily transactions (BAYC) and the opening price of ETH')
```

In this code, we get the ETH price data in a DataFrame called `eth`. We then select the 'Date' and 'Open' (which is the opening price) columns of this DataFrame and manipulate the 'Date' column to convert it to a datetime object and format it as a string using the `pd.to_datetime()` and `strftime()` methods. We set the index of `eth` to be the 'Date' column and convert it to a datetime object using the `pd.to_datetime()` method with a specified format.

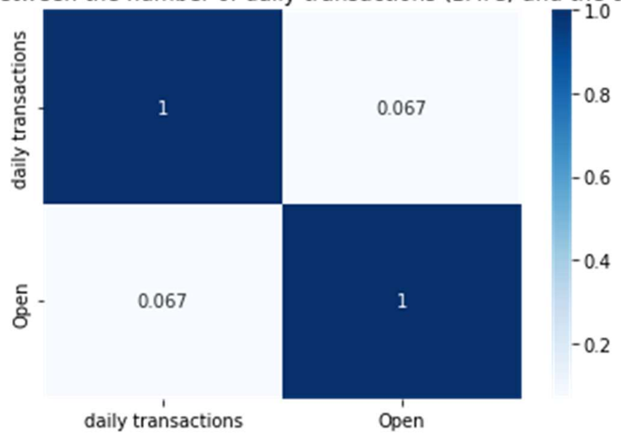
Next, we merge the resulting `eth` DataFrame with our previously designed daily transaction DataFrame called `BAYC_daily_transaction` on the 'Date' column using the `pd.merge()` method with a specified merge type of 'left'. The resulting merged DataFrame is stored in a variable called `BAYC_eth`. We then select the 'daily transactions' and 'Open' columns of this DataFrame and calculate their correlation matrix using the `corr()` method. The resulting correlation matrix is stored in a variable called `corr_matrix`. Finally, we plot this correlation matrix as a heatmap using the `heatmap()` method of the Seaborn library.

The main objective of this code is to calculate and visualise the correlation between the number of daily trades and the opening price of ETH.

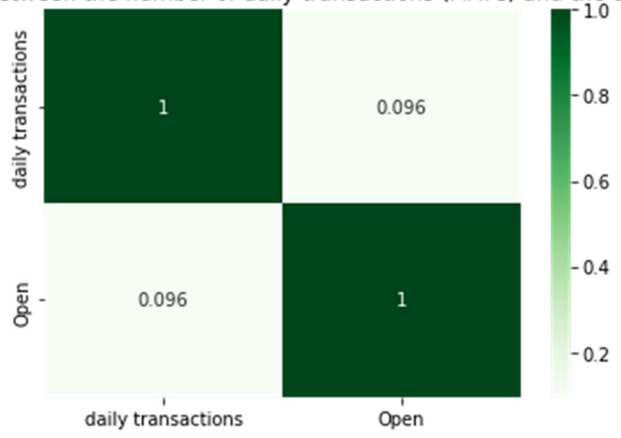
b) Results



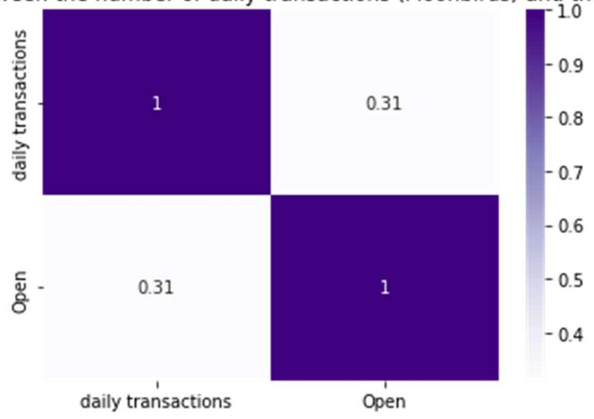
Heatmap of correlation between the number of daily transactions (BAYC) and the opening price of ETH

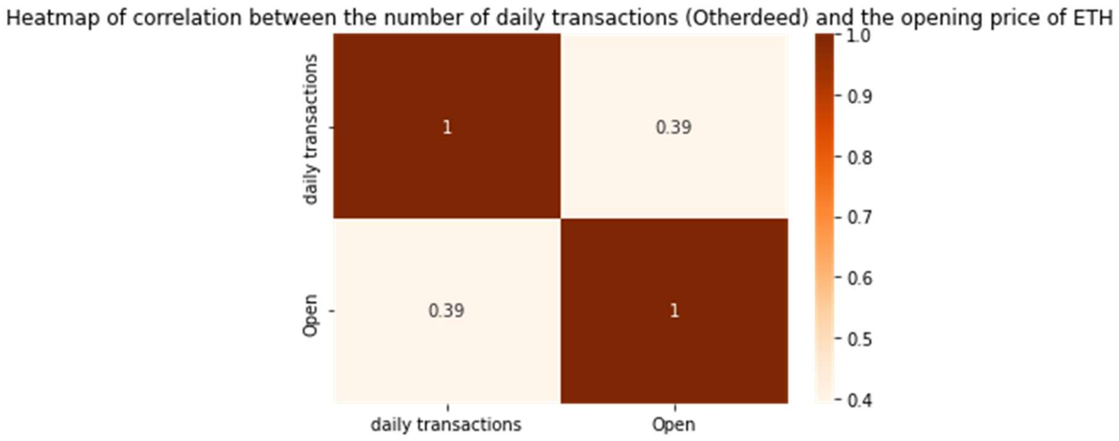


Heatmap of correlation between the number of daily transactions (MAYC) and the opening price of ETH



Heatmap of correlation between the number of daily transactions (Moonbirds) and the opening price of ETH



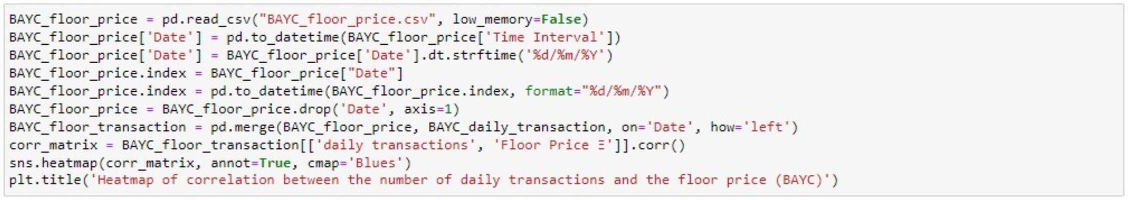


c) Observations

We observe here that there is no significant correlation between the number of daily transactions and the price of ETH for the BAYC and MAYC collections. Nevertheless, there is a slight positive correlation between the price of ETH and the number of transactions for the Azuki, Moonbirds and Otherdeed collections. We can therefore conclude that the price of ETH does not influence the number of transactions of the BAYC and MAYC collections, but it does slightly influence the number of transactions of the Azuki, Moonbirds and Otherdeed collections. We will see in part V. that it is actually the average selling price of the NFTs in a collection that is influenced by the price of ETH and not its number of daily transactions.

III. Determine the correlation between the number of transactions in a collection and its floor price.

a) Code explanation



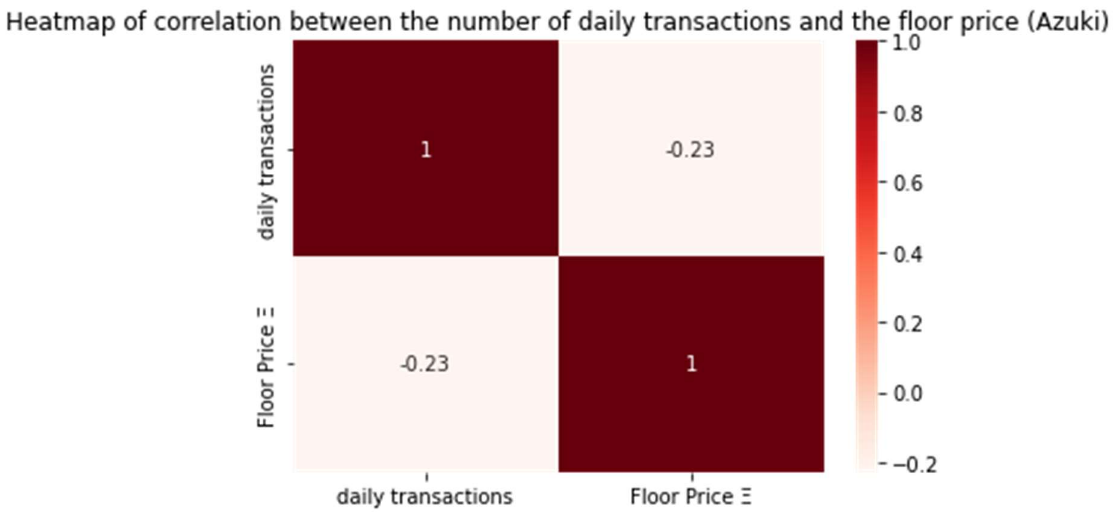
```
BAYC_floor_price = pd.read_csv("BAYC_floor_price.csv", low_memory=False)
BAYC_floor_price['Date'] = pd.to_datetime(BAYC_floor_price['Time Interval'])
BAYC_floor_price['Date'] = BAYC_floor_price['Date'].dt.strftime('%d/%m/%Y')
BAYC_floor_price.index = BAYC_floor_price["Date"]
BAYC_floor_price.index = pd.to_datetime(BAYC_floor_price.index, format="%d/%m/%Y")
BAYC_floor_price = BAYC_floor_price.drop('Date', axis=1)
BAYC_floor_transaction = pd.merge(BAYC_floor_price, BAYC_daily_transaction, on='Date', how='left')
corr_matrix = BAYC_floor_transaction[['daily transactions', 'Floor Price Ξ']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='Blues')
plt.title('Heatmap of correlation between the number of daily transactions and the floor price (BAYC)')
```

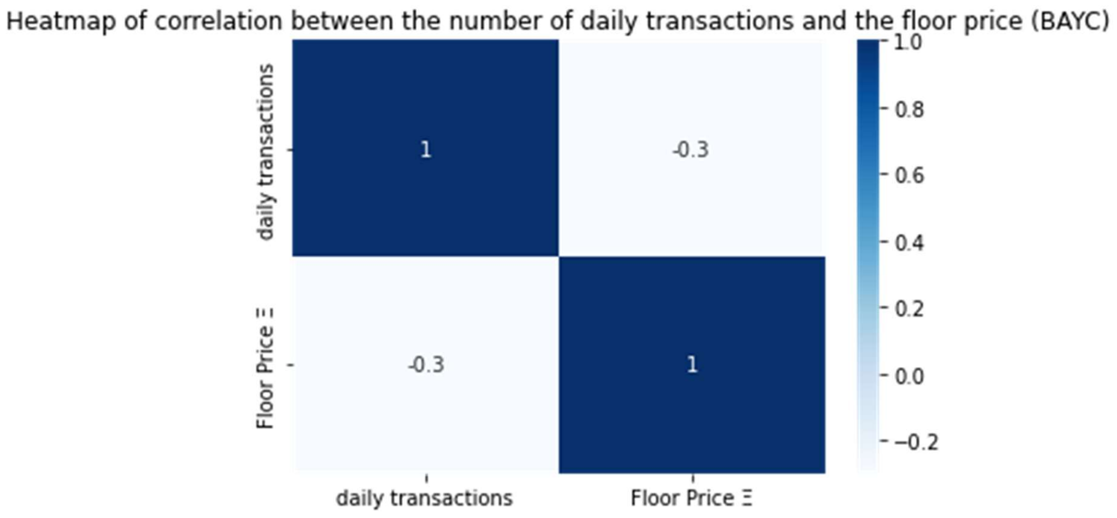
In this code let's get the floor price data from the collecton in a DataFrame called `BAYC_floor_price`. We then manipulate the 'Time Interval' column to convert it into a datetime object and format it as a string using the `pd.to_datetime()` and `strftime()` methods. We set the index of `BAYC_floor_price` to be the 'Date' column and convert it to a datetime object using the `pd.to_datetime()` method with a specified format.

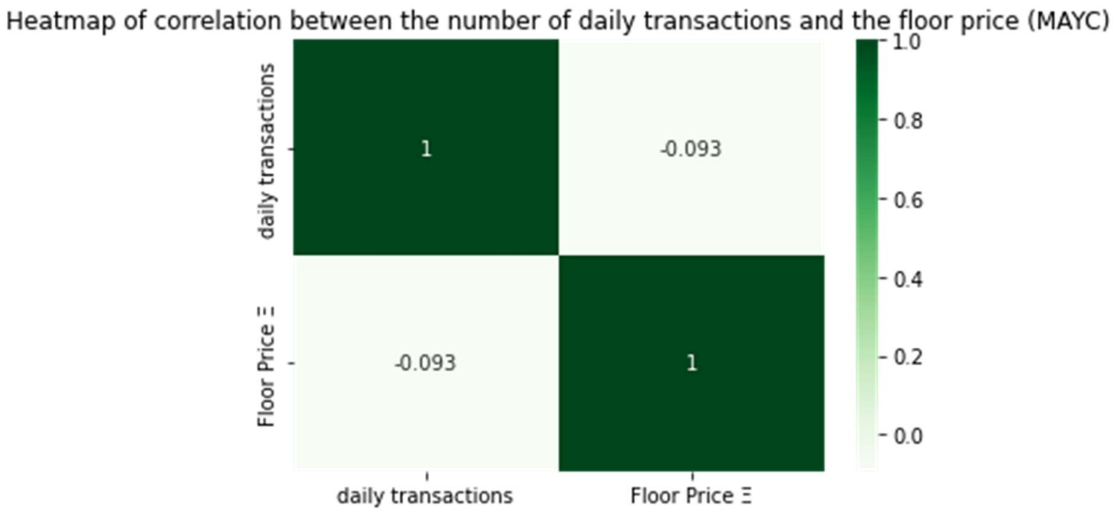
Next, we merge the resulting floor price DataFrame `BAYC_floor_price` with our daily transaction DataFrame `BAYC_daily_transaction` on the 'Date' column using the `pd.merge()` method with a specified merge type of 'left'. The resulting merged DataFrame is stored in a variable called `BAYC_floor_transaction`. We then select the columns `daily trades` and `floor price Ξ` from this DataFrame and calculate their correlation matrix using the `corr()` method. The resulting correlation matrix is stored in a variable called `corr_matrix`. Finally, we plot this correlation matrix as a heatmap using the `heatmap()` method of the Seaborn library.

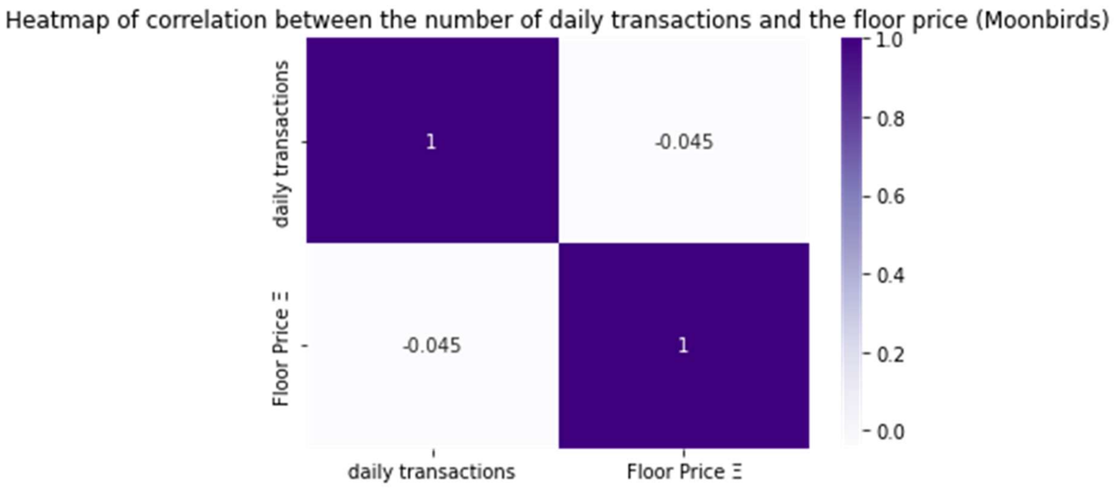
The main objective of this code is to calculate and visualise the correlation between the number of daily transactions in the collection and its floor price.

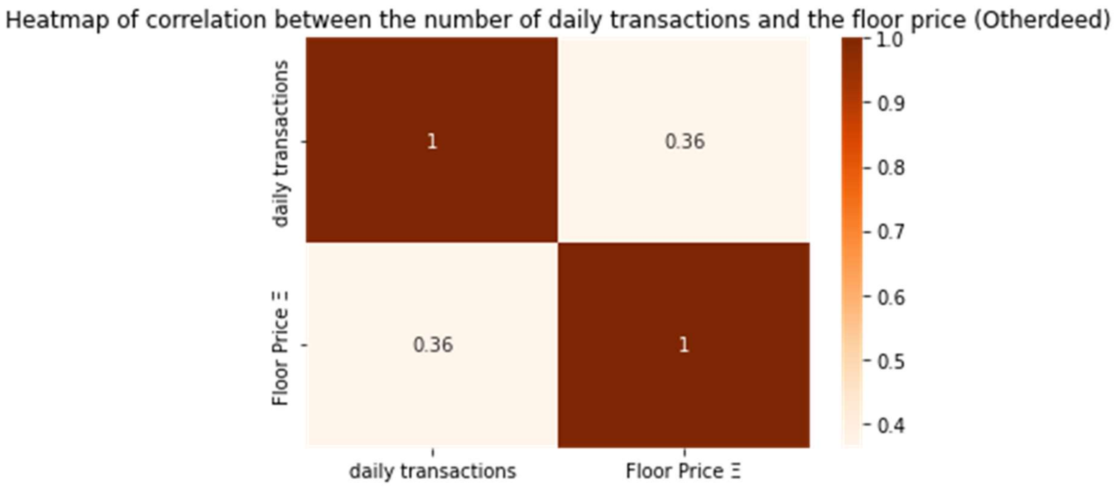
b) Results


Heatmap of correlation between the number of daily transactions and the floor price (Azuki)
daily transactions
Floor Price Ξ
1
-0.23
-0.23
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Heatmap of correlation between the number of daily transactions and the floor price (BAYC)
daily transactions
Floor Price Ξ
1
-0.3
-0.3
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Heatmap of correlation between the number of daily transactions and the floor price (MAYC)
daily transactions
Floor Price Ξ
1
-0.093
-0.093
1
1.0
0.8
0.6
0.4
0.2
0.0



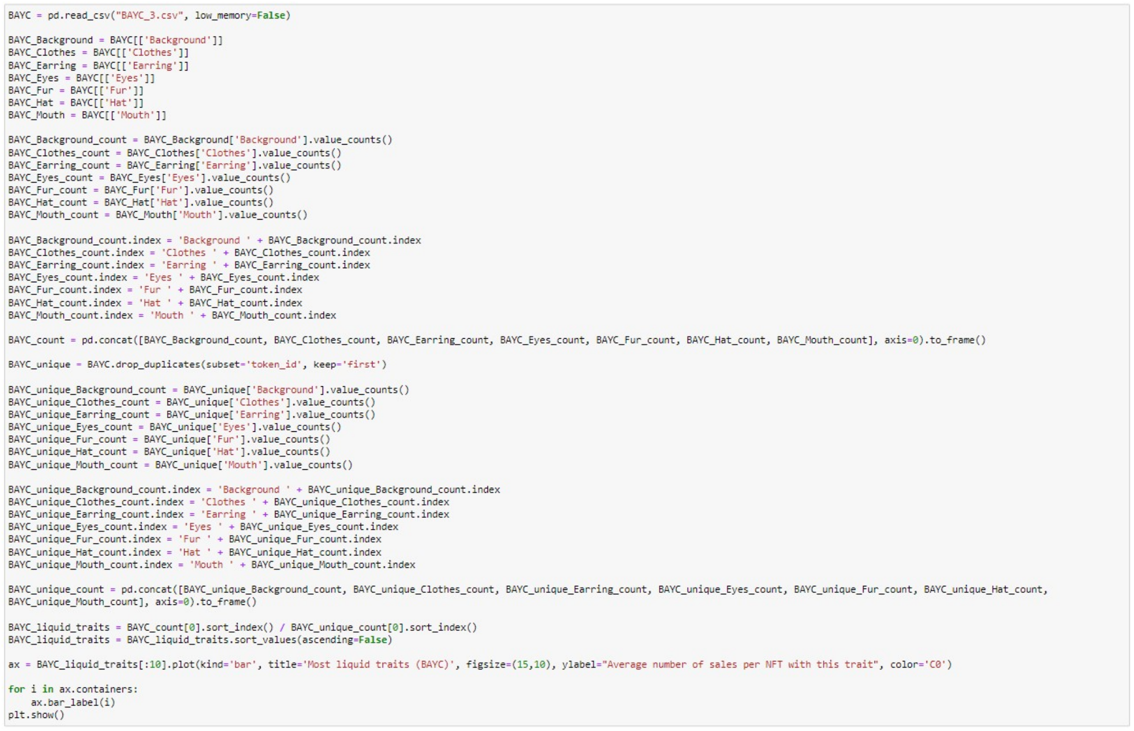


c) Observations

We observe here that there is no significant correlation between the number of daily transactions and the floor price of the MAYC and Moonbirds collections. Nevertheless, we observe a weak negative correlation for the Azuki and BAYC collections and a weak positive correlation for the Otherdeed collection. For the Azuki and BAYC collections, which are very expensive collections, this negative correlation can be explained by the fact that a greater number of investors are able to buy these collections at times when their floor price is low, which results in a higher number of transactions at times when the floor price is low. In contrast, the Otherdeed collection, which is an inexpensive collection, does not observe the same phenomenon because the price is still affordable by investors. For this collection, a drop in the floor price results in fewer transactions because there is less excitement about the collection.

IV. What are the most liquid traits (those with the most sales) for each collection?

a) Code explanation



```python
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)

BAYC_Background = BAYC[['Background']]
BAYC_Clothes = BAYC[['Clothes']]
BAYC_Earring = BAYC[['Earring']]
BAYC_Eyes = BAYC[['Eyes']]
BAYC_Fur = BAYC[['Fur']]
BAYC_Hat = BAYC[['Hat']]
BAYC_Mouth = BAYC[['Mouth']]

BAYC_Background_count = BAYC_Background['Background'].value_counts()
BAYC_Clothes_count = BAYC_Clothes['Clothes'].value_counts()
BAYC_Earring_count = BAYC_Earring['Earring'].value_counts()
BAYC_Eyes_count = BAYC_Eyes['Eyes'].value_counts()
BAYC_Fur_count = BAYC_Fur['Fur'].value_counts()
BAYC_Hat_count = BAYC_Hat['Hat'].value_counts()
BAYC_Mouth_count = BAYC_Mouth['Mouth'].value_counts()

BAYC_Background_count.index = 'Background ' + BAYC_Background_count.index
BAYC_Clothes_count.index = 'Clothes ' + BAYC_Clothes_count.index
BAYC_Earring_count.index = 'Earring ' + BAYC_Earring_count.index
BAYC_Eyes_count.index = 'Eyes ' + BAYC_Eyes_count.index
BAYC_Fur_count.index = 'Fur ' + BAYC_Fur_count.index
BAYC_Hat_count.index = 'Hat ' + BAYC_Hat_count.index
BAYC_Mouth_count.index = 'Mouth ' + BAYC_Mouth_count.index

BAYC_count = pd.concat([BAYC_Background_count, BAYC_Clothes_count, BAYC_Earring_count, BAYC_Eyes_count, BAYC_Fur_count, BAYC_Hat_count, BAYC_Mouth_count], axis=0).to_frame()

BAYC_unique = BAYC.drop_duplicates(subset='token_id', keep='first')

BAYC_unique_Background_count = BAYC_unique['Background'].value_counts()
BAYC_unique_Clothes_count = BAYC_unique['Clothes'].value_counts()
BAYC_unique_Earring_count = BAYC_unique['Earring'].value_counts()
BAYC_unique_Eyes_count = BAYC_unique['Eyes'].value_counts()
BAYC_unique_Fur_count = BAYC_unique['Fur'].value_counts()
BAYC_unique_Hat_count = BAYC_unique['Hat'].value_counts()
BAYC_unique_Mouth_count = BAYC_unique['Mouth'].value_counts()

BAYC_unique_Background_count.index = 'Background ' + BAYC_unique_Background_count.index
BAYC_unique_Clothes_count.index = 'Clothes ' + BAYC_unique_Clothes_count.index
BAYC_unique_Earring_count.index = 'Earring ' + BAYC_unique_Earring_count.index
BAYC_unique_Eyes_count.index = 'Eyes ' + BAYC_unique_Eyes_count.index
BAYC_unique_Fur_count.index = 'Fur ' + BAYC_unique_Fur_count.index
BAYC_unique_Hat_count.index = 'Hat ' + BAYC_unique_Hat_count.index
BAYC_unique_Mouth_count.index = 'Mouth ' + BAYC_unique_Mouth_count.index

BAYC_unique_count = pd.concat([BAYC_unique_Background_count, BAYC_unique_Clothes_count, BAYC_unique_Earring_count, BAYC_unique_Eyes_count, BAYC_unique_Fur_count, BAYC_unique_Hat_count,
BAYC_unique_Mouth_count], axis=0).to_frame()

BAYC_liquid_traits = BAYC_count[0].sort_index() / BAYC_unique_count[0].sort_index()
BAYC_liquid_traits = BAYC_liquid_traits.sort_values(ascending=False)

ax = BAYC_liquid_traits[:10].plot(kind='bar', title='Most liquid traits (BAYC)', figsize=(15,10), ylabel="Average number of sales per NFT with this trait", color='C0')

for i in ax.containers:
    ax.bar_label(i)
plt.show()
```

In this code we fetch the transaction data from the collection in a DataFrame called `BAYC`. We then select the columns with features of the same type such as 'Background', 'Clothes', 'Earring', 'Eyes', 'Fur', 'Hat' and 'Mouth', and store them in separate DataFrames. We then calculate the number of occurrences of each trait for each of these columns using the `value_counts()` method and store the resulting series in separate variables. The indexes of these series are therefore the traits.

Next, we manipulate the index of each of these series by concatenating the trait type with the original index values (the traits) using string concatenation. This step allows us to distinguish between two traits with the same name but not the same type e.g. the trait type "Background" may take the value "Blue" and the trait type "Fur" may also take the value "Blue", so we need to distinguish between them.

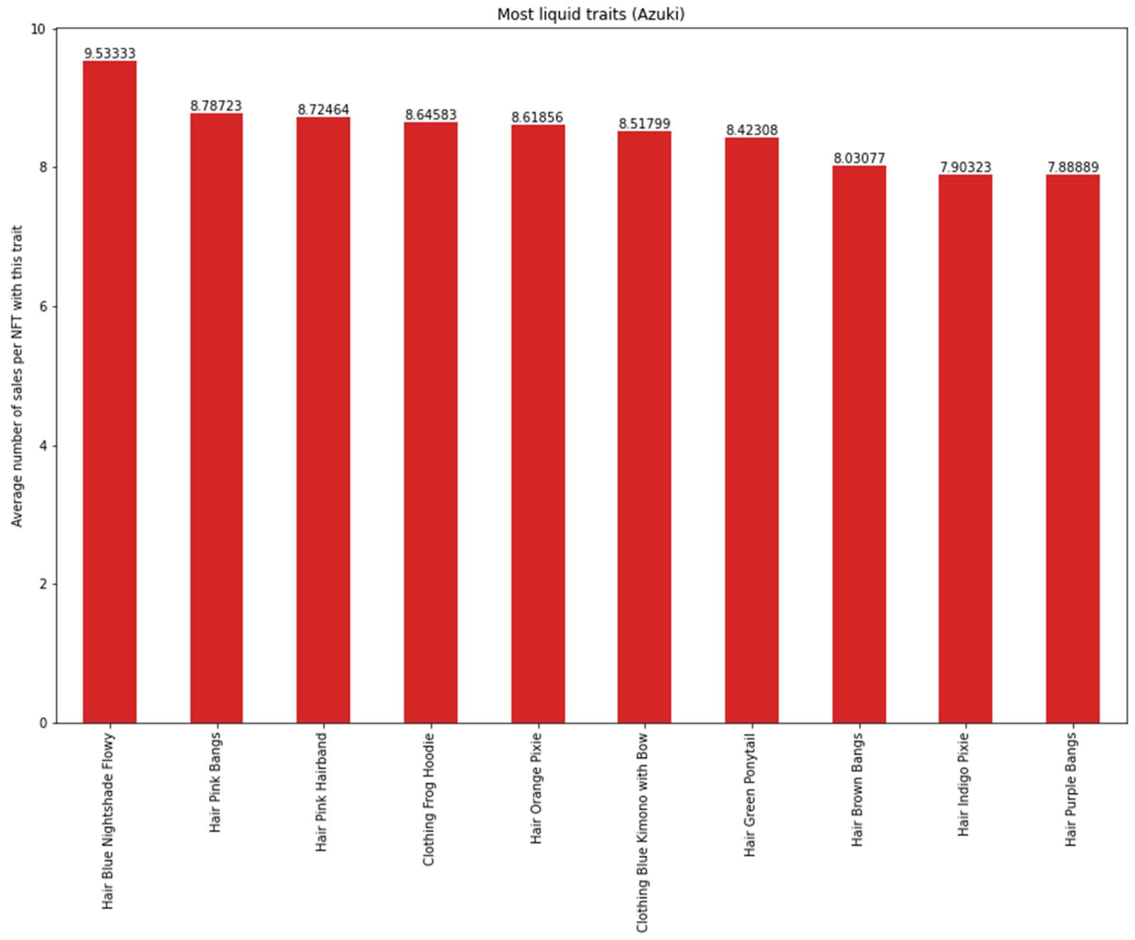
Finally, we concatenate all these series along the line axis using the `pd.concat()` method and convert the resulting series into a DataFrame using the `to_frame()` method. The resulting DataFrame is stored in a variable called `BAYC_count`. This variable is therefore the number of transactions per trait.

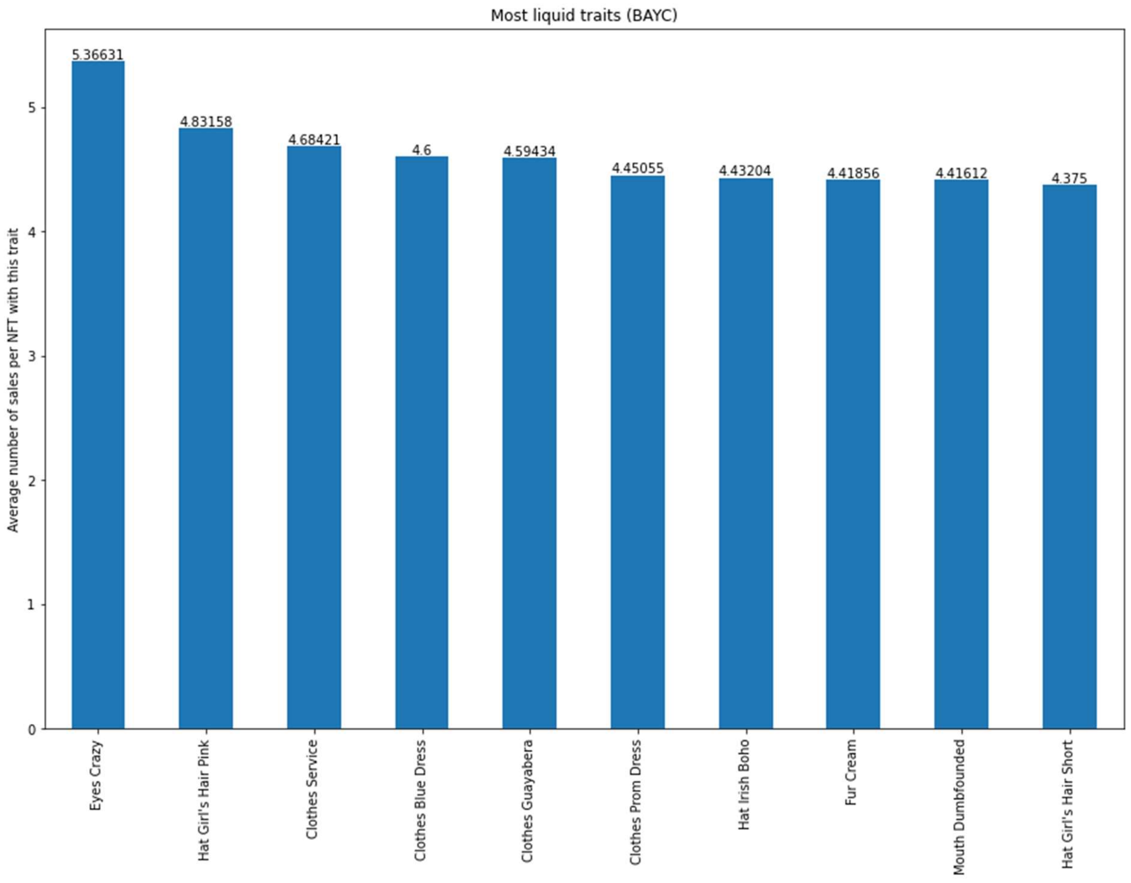
We then do the same for the `BAYC_unique` variable which does not represent transactions like 'BAYC' but only the NFTs in the collection. The resulting DataFrame is stored in a variable called `BAYC_count`. This variable is the number of NFTs per trait, it tells us how many NFTs carry a specific trait for each of the traits in the collection. The indexes of these series are therefore the traits.

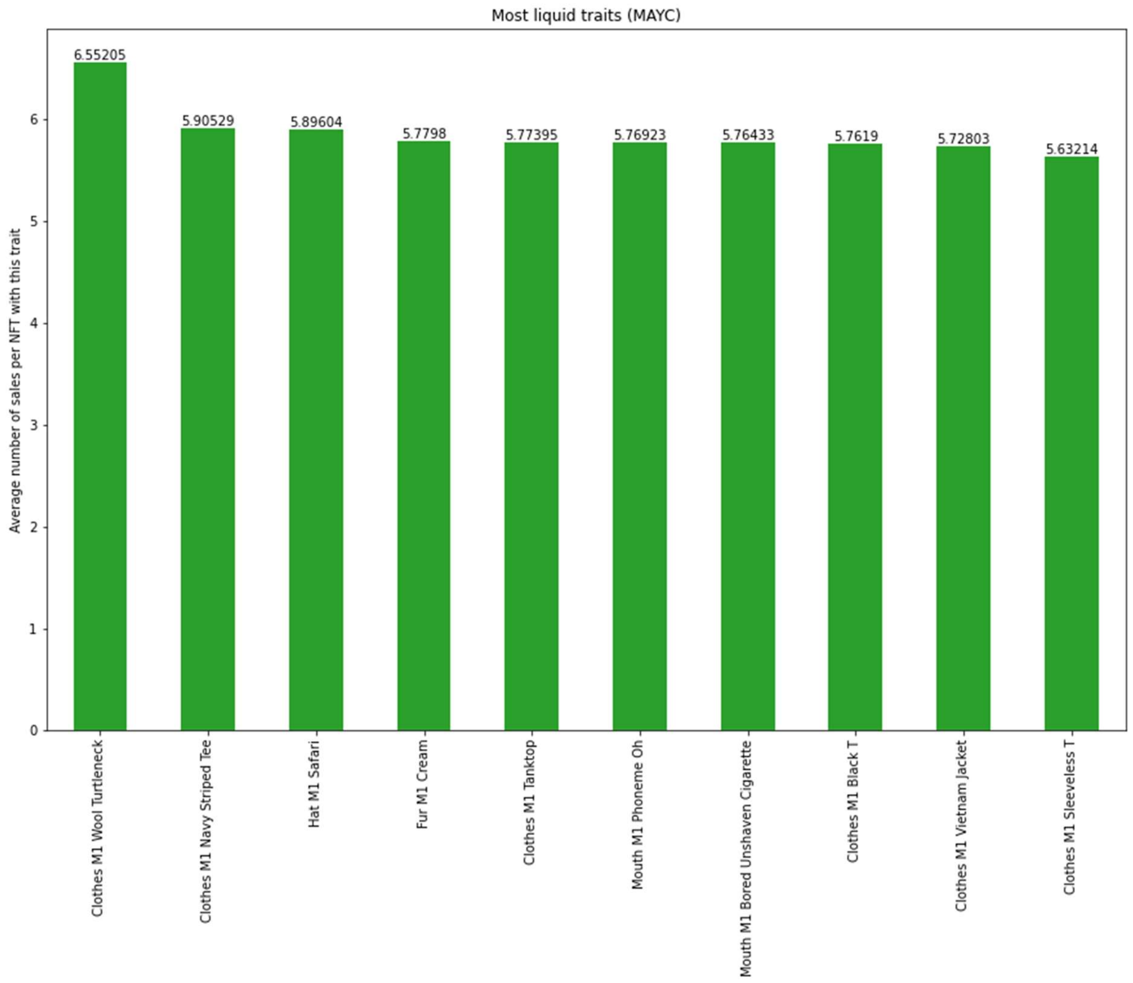
We then calculate the ratio of two series by dividing their values element by element after sorting them in ascending order of index and store the resulting series in a variable called `BAYC_liquid_traits`. This variable therefore represents the average number of sales of an NFTs carrying a specific trait for each of the traits in the collection. For example, the NFTs in the BAYC collection with the "Crazy" trait of type "Eyes" were sold on average 5.36631 times making this trait the most liquid trait in the BAYC collection.

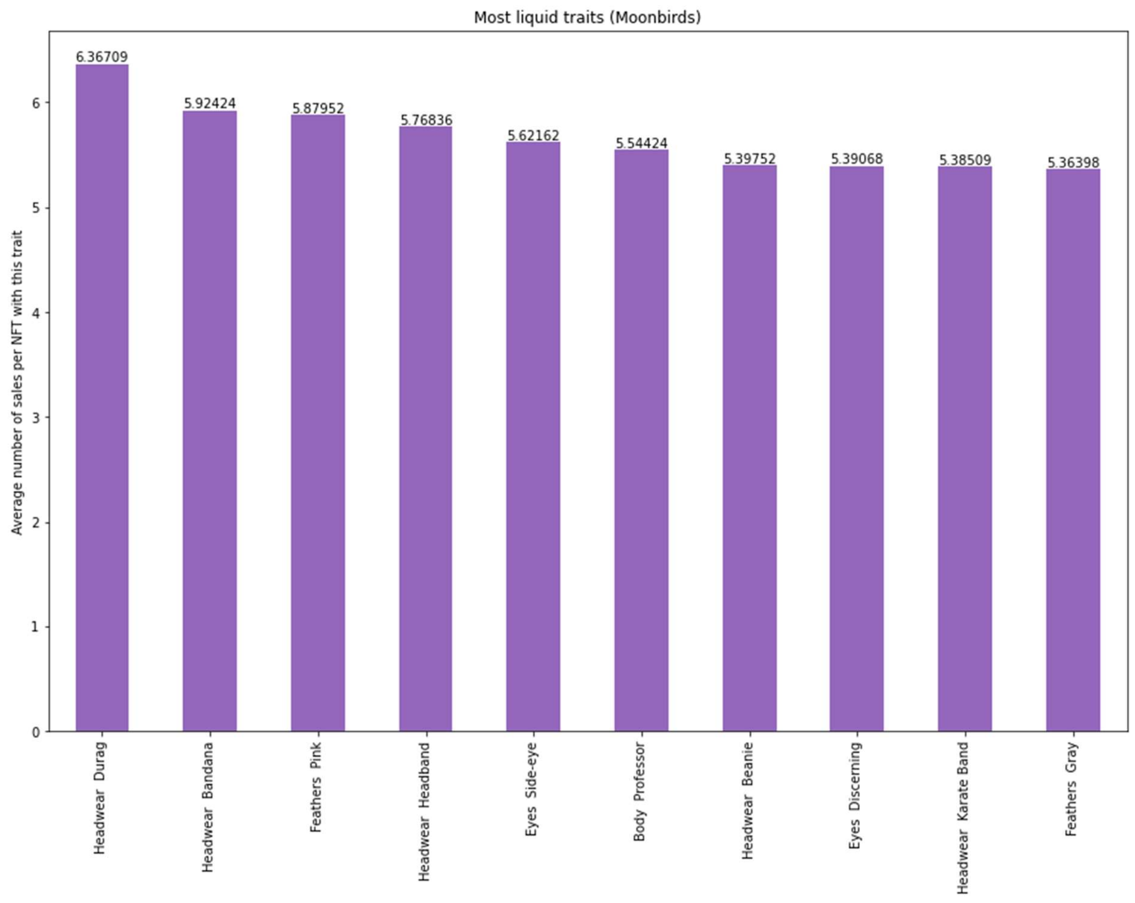
Finally, we plot the first 10 values of this series as a bar chart using the `plot()` method of the Matplotlib library. This gives us the 10 most liquid traits in the collection.

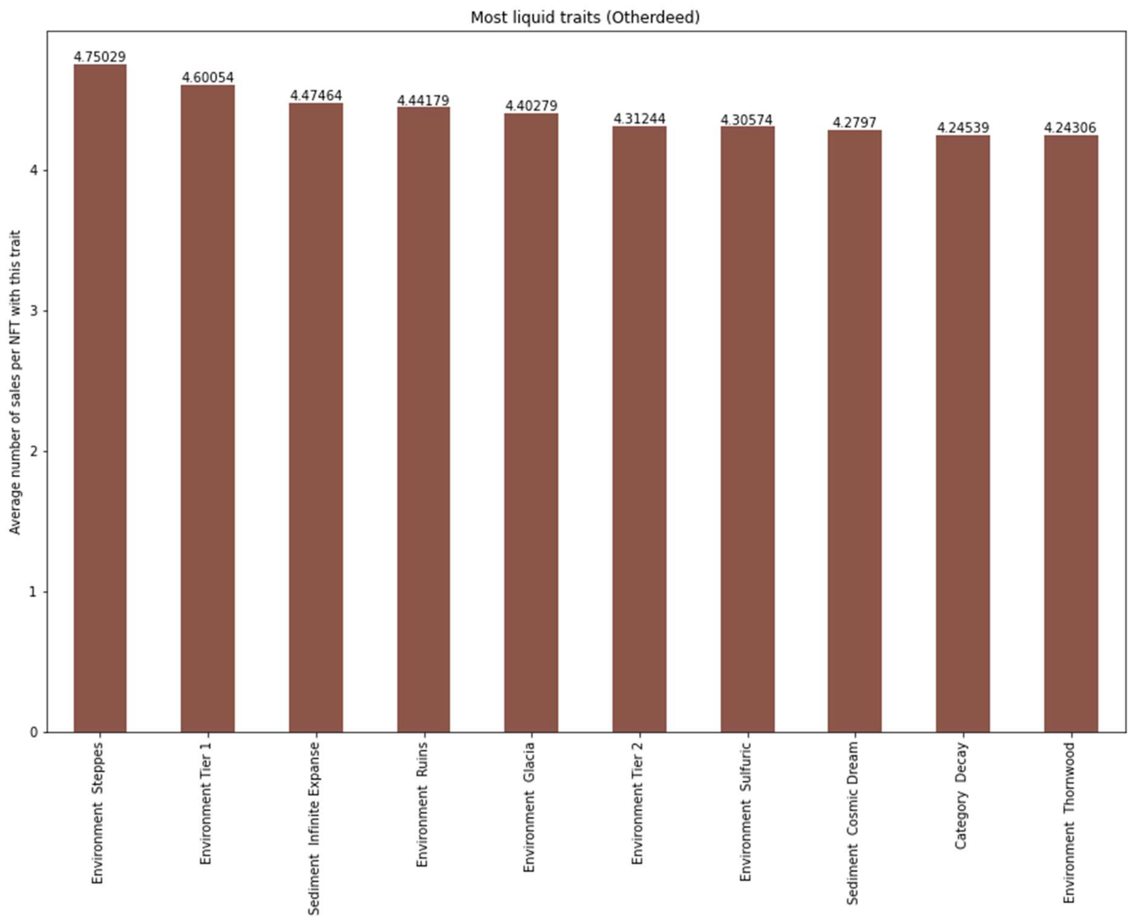
b) Results




Most liquid traits (BAYC)
Average number of sales per NFT with this trait
0
1
2
3
4
5
5.36631
4.83158
4.68421
4.6
4.59434
4.45055
4.43204
4.41856
4.41612
4.375
Eyes Crazy
Hat Girl's Hair Pink
Clothes Service
Clothes Blue Dress
Clothes Guayabera
Clothes Prom Dress
Hat Irish Boho
Fur Cream
Mouth Dumbfounded
Hat Girl's Hair Short

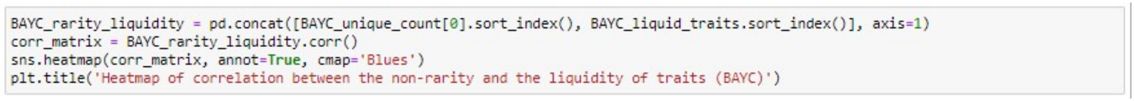

Most liquid traits (MAYC)
Average number of sales per NFT with this trait
0
1
2
3
4
5
6
6.55205
5.90529
5.89604
5.7798
5.77395
5.76923
5.76433
5.7619
5.72803
5.63214
Clothes M1 Wool Turtleneck
Clothes M1 Navy Striped Tee
Hat M1 Safari
Fur M1 Cream
Clothes M1 Tanktop
Mouth M1 Phoneme Oh
Mouth M1 Bored Unshaven Cigarette
Clothes M1 Black T
Clothes M1 Vietnam Jacket
Clothes M1 Sleeveless T

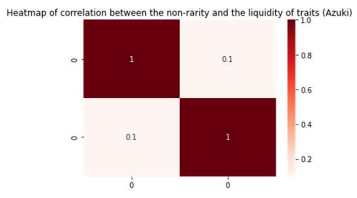

Most liquid traits (Moonbirds)
Average number of sales per NFT with this trait
6.36709
5.92424
5.87952
5.76836
5.62162
5.54424
5.39752
5.39068
5.38509
5.36398
Headwear Durag
Headwear Bandana
Feathers Pink
Headwear Headband
Eyes Side-eye
Body Professor
Headwear Beanie
Eyes Discerning
Headwear Karate Band
Feathers Gray
0
1
2
3
4
5
6

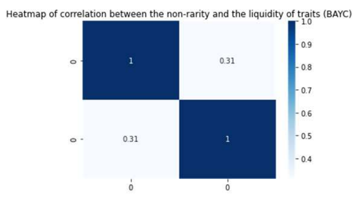


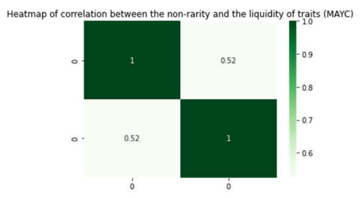
c) Observations

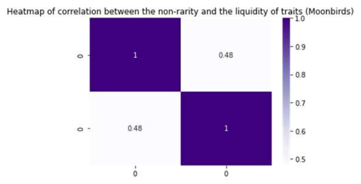
We observe here that the NFTs with the most liquid traits are on average sold 4 to 6 times for the BAYC, MAYC, Moonbirds and Otherdeed collections and 8 to 9 times for the Azuki collection. The Azuki collection has the highest liquidity. For each collection there is a 20% difference in sales between the most liquid trait and the 10th most liquid trait. An interesting information to remember is that the most liquid traits are very common. There is a positive correlation between the non-rarity of a trait and its liquidity as shown in these heatmaps:

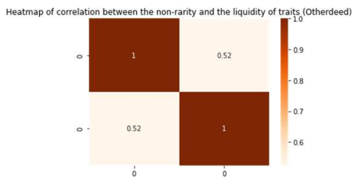

```
BAYC_rarity_liquidity = pd.concat([BAYC_unique_count[0].sort_index(), BAYC_liquid_traits.sort_index()], axis=1)
corr_matrix = BAYC_rarity_liquidity.corr()
sns.heatmap(corr_matrix, annot=True, cmap='Blues')
plt.title('Heatmap of correlation between the non-rarity and the liquidity of traits (BAYC)')
```











V. Provide a visual overview of the NFT collections of your choice and its characteristics (e.g. size, type of NFTs, date range)

In this section, we will deal with the Bored Ape Yacht Club collection as it is on this same collection that we would train our machine learning model. The aim here is to determine the characteristics that determine the selling price of the NFTs in the collection.

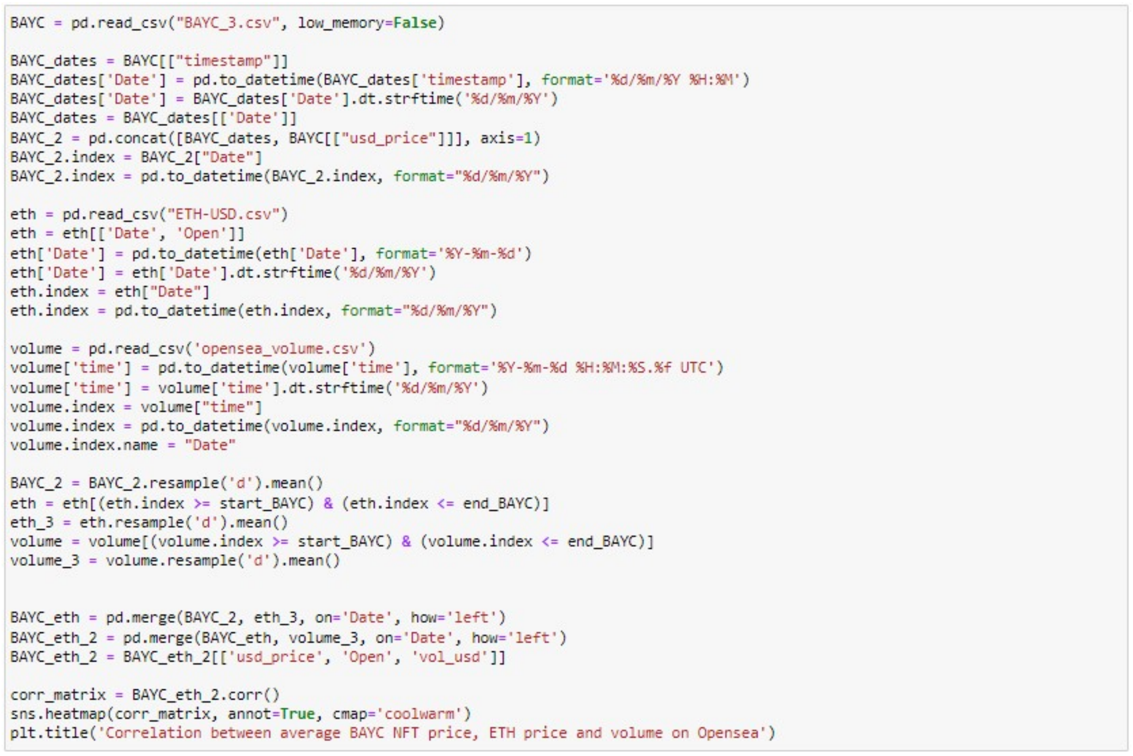
a) Code explanation

```
BAYC[['usd_price', "eth_price", 'royalty_fee', 'platform_fee']].describe()
```

This code is used to obtain the basic statistics of the transcation dataframe.

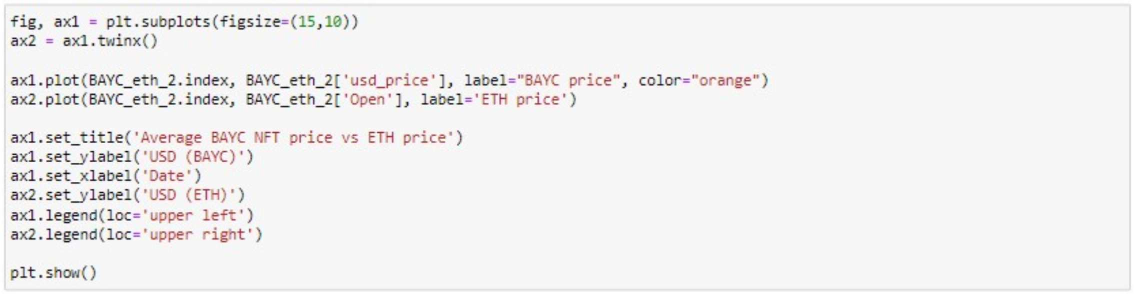
```
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)

BAYC_dates = BAYC[["timestamp"]]
BAYC_dates['Date'] = pd.to_datetime(BAYC_dates['timestamp'], format='%d/%m/%Y %H:%M')
BAYC_dates['Date'] = BAYC_dates['Date'].dt.strftime('%d/%m/%Y')
BAYC_dates = BAYC_dates[['Date']]
BAYC_2 = pd.concat([BAYC_dates, BAYC[["usd_price"]]], axis=1)
BAYC_2.index = BAYC_2["Date"]
BAYC_2.index = pd.to_datetime(BAYC_2.index, format="%d/%m/%Y")

BAYC_3 = BAYC_2.resample('w').mean()

fig, ax1 = plt.subplots(figsize=(15,10))
ax1.plot(BAYC_3.index, BAYC_3['usd_price'], color="C0")
ax1.set_title("Average selling price of NFTs (BAYC)")
ax1.set_ylabel("USD")
ax1.set_xlabel("Date")
plt.show()
```

This code groups the transactions by week and then calculates the average sale price per week. Finally, it displays the result in a graph.



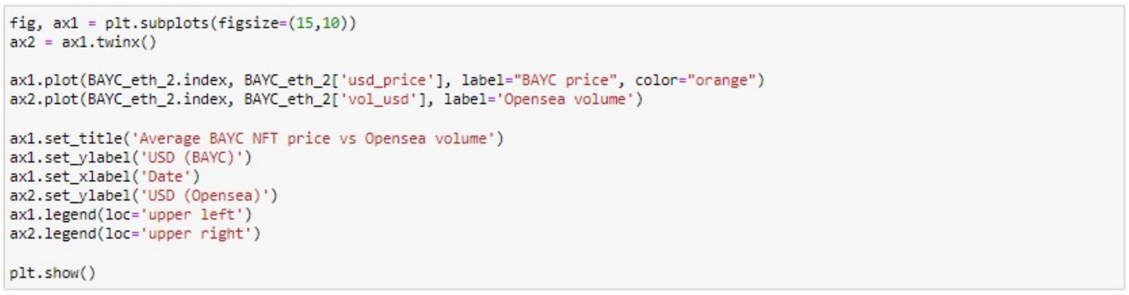
```
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)

BAYC_dates = BAYC[["timestamp"]]
BAYC_dates['Date'] = pd.to_datetime(BAYC_dates['timestamp'], format='%d/%m/%Y %H:%M')
BAYC_dates['Date'] = BAYC_dates['Date'].dt.strftime('%d/%m/%Y')
BAYC_dates = BAYC_dates[['Date']]
BAYC_2 = pd.concat([BAYC_dates, BAYC[["usd_price"]]], axis=1)
BAYC_2.index = BAYC_2["Date"]
BAYC_2.index = pd.to_datetime(BAYC_2.index, format="%d/%m/%Y")

eth = pd.read_csv("ETH-USD.csv")
eth = eth[['Date', 'Open']]
eth['Date'] = pd.to_datetime(eth['Date'], format='%Y-%m-%d')
eth['Date'] = eth['Date'].dt.strftime('%d/%m/%Y')
eth.index = eth["Date"]
eth.index = pd.to_datetime(eth.index, format="%d/%m/%Y")

volume = pd.read_csv('opensea_volume.csv')
volume['time'] = pd.to_datetime(volume['time'], format='%Y-%m-%d %H:%M:%S.%f UTC')
volume['time'] = volume['time'].dt.strftime('%d/%m/%Y')
volume.index = volume["time"]
volume.index = pd.to_datetime(volume.index, format="%d/%m/%Y")
volume.index.name = "Date"

BAYC_2 = BAYC_2.resample('d').mean()
eth = eth[(eth.index >= start_BAYC) & (eth.index <= end_BAYC)]
eth_3 = eth.resample('d').mean()
volume = volume[(volume.index >= start_BAYC) & (volume.index <= end_BAYC)]
volume_3 = volume.resample('d').mean()


BAYC_eth = pd.merge(BAYC_2, eth_3, on='Date', how='left')
BAYC_eth_2 = pd.merge(BAYC_eth, volume_3, on='Date', how='left')
BAYC_eth_2 = BAYC_eth_2[['usd_price', 'Open', 'vol_usd']]

corr_matrix = BAYC_eth_2.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation between average BAYC NFT price, ETH price and volume on Opensea')
```

This code allows to gather in the same DataFrame 'BAYC_eth_2' the daily average price of BAYC NFTs sales, the daily average price of ETH and the daily average value of the trading volume on Opensea. Finally, it performs the correlation matrix between these 3 metrics and displays the result as a heatmap.



```
fig, ax1 = plt.subplots(figsize=(15,10))
ax2 = ax1.twinx()

ax1.plot(BAYC_eth_2.index, BAYC_eth_2['usd_price'], label="BAYC price", color="orange")
ax2.plot(BAYC_eth_2.index, BAYC_eth_2['Open'], label='ETH price')

ax1.set_title('Average BAYC NFT price vs ETH price')
ax1.set_ylabel('USD (BAYC)')
ax1.set_xlabel('Date')
ax2.set_ylabel('USD (ETH)')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```

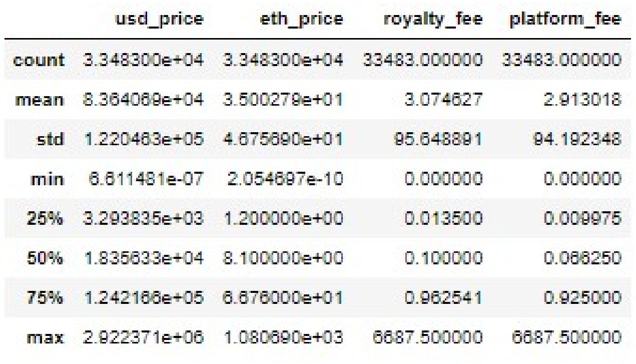
This code displays the time curves of the daily average price of BAYC NFT sales and the daily average price of ETH on the same graph.



```
fig, ax1 = plt.subplots(figsize=(15,10))
ax2 = ax1.twinx()

ax1.plot(BAYC_eth_2.index, BAYC_eth_2['usd_price'], label="BAYC price", color="orange")
ax2.plot(BAYC_eth_2.index, BAYC_eth_2['vol_usd'], label='Opensea volume')

ax1.set_title('Average BAYC NFT price vs Opensea volume')
ax1.set_ylabel('USD (BAYC)')
ax1.set_xlabel('Date')
ax2.set_ylabel('USD (Opensea)')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```

This code displays the time curves of the average daily sales price of BAYC NFTs and the average daily value of the trading volume on Opensea on the same graph.

b) Results



	usd_price	eth_price	royalty_fee	platform_fee
count	3.348300e+04	3.348300e+04	33483.000000	33483.000000
mean	8.364069e+04	3.500279e+01	3.074627	2.913018
std	1.220463e+05	4.675690e+01	95.648891	94.192348
min	6.611481e-07	2.054697e-10	0.000000	0.000000
25%	3.293835e+03	1.200000e+00	0.013500	0.009975
50%	1.835633e+04	8.100000e+00	0.100000	0.066250
75%	1.242166e+05	6.676000e+01	0.962541	0.925000
max	2.922371e+06	1.080690e+03	6687.500000	6687.500000

Figure 1

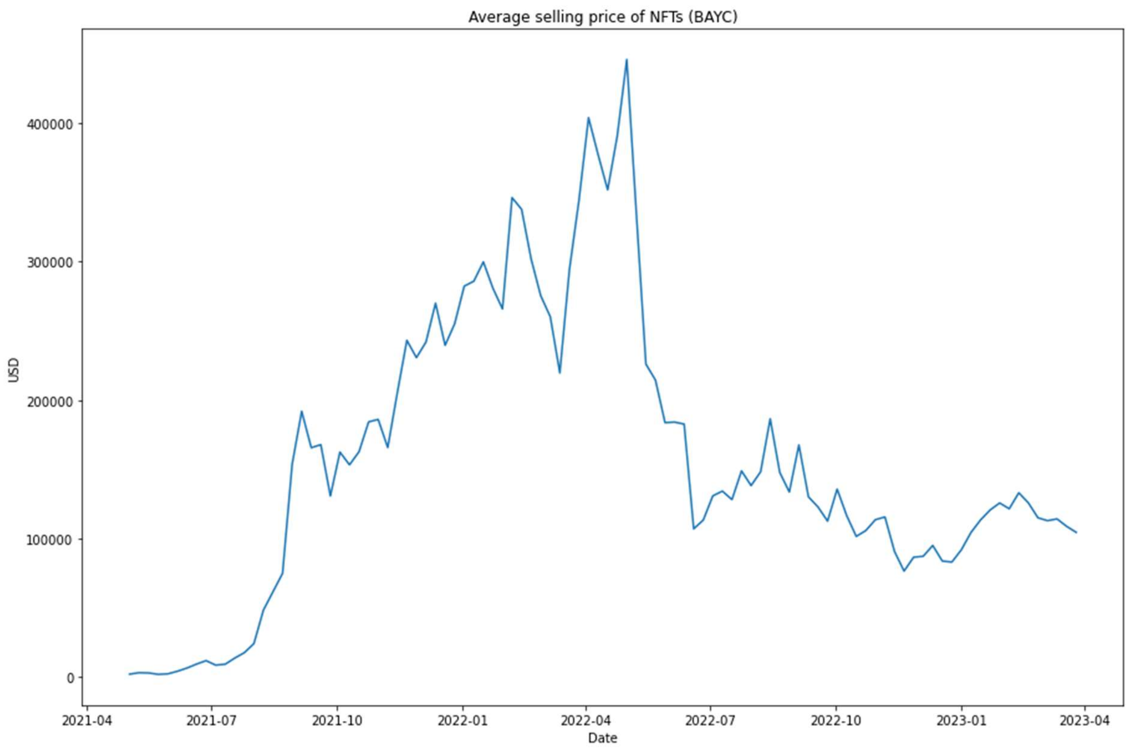


Figure 2

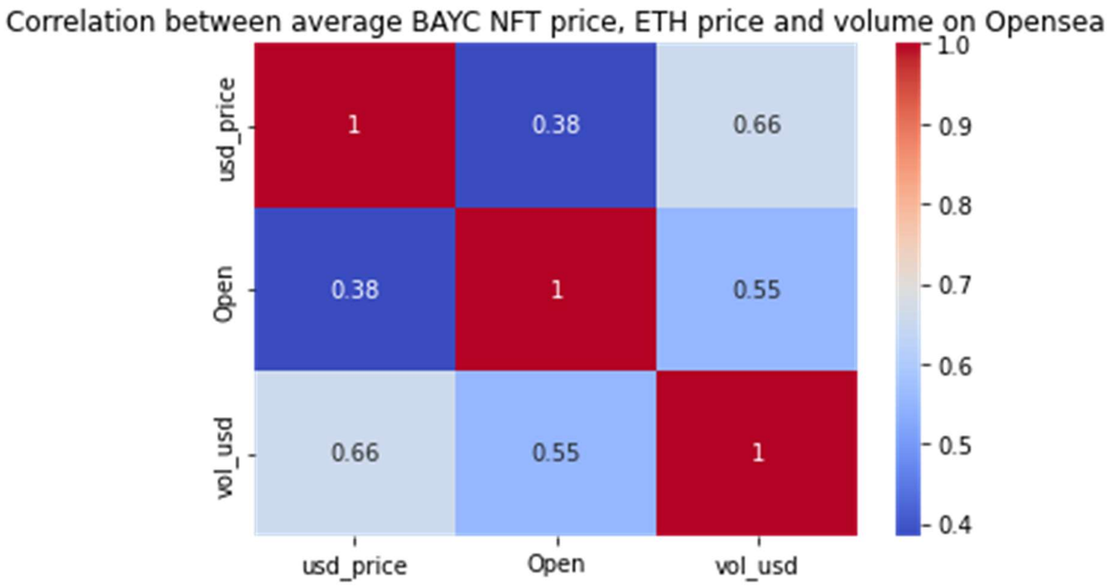


Figure 3

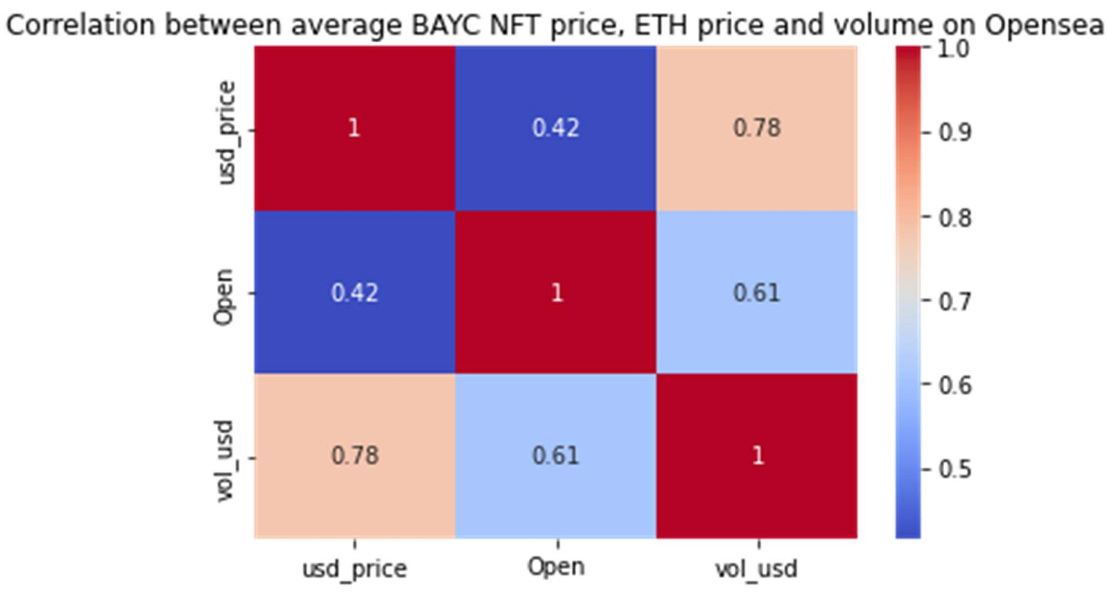


Figure 4

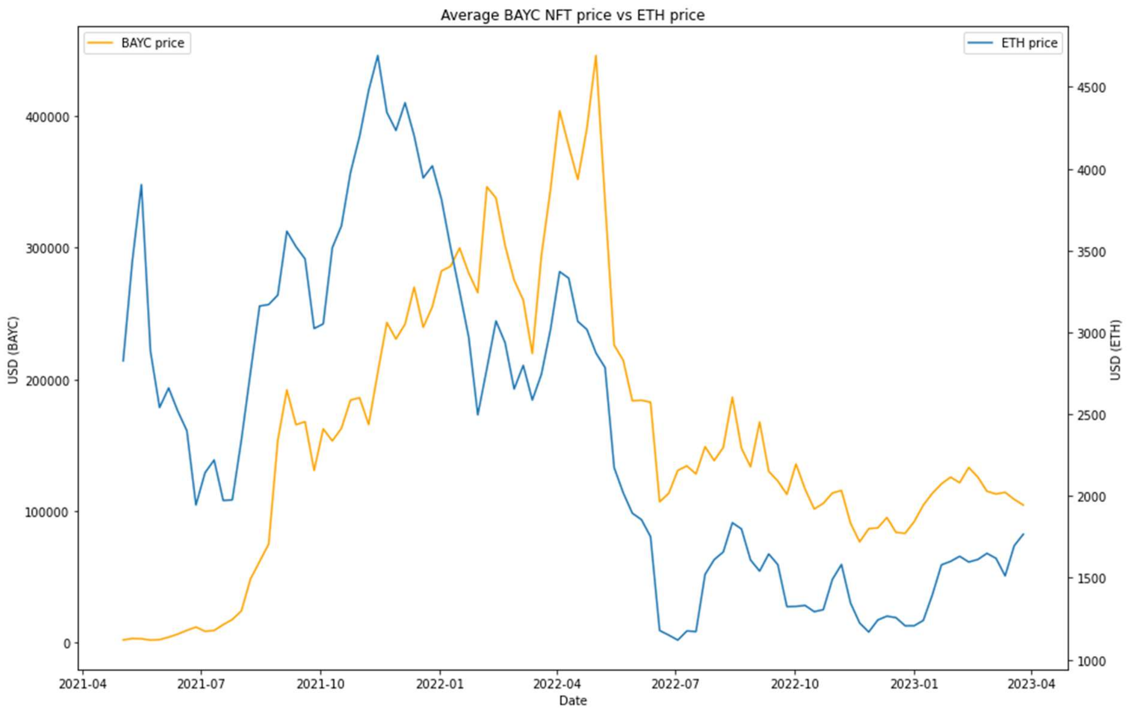


Figure 5

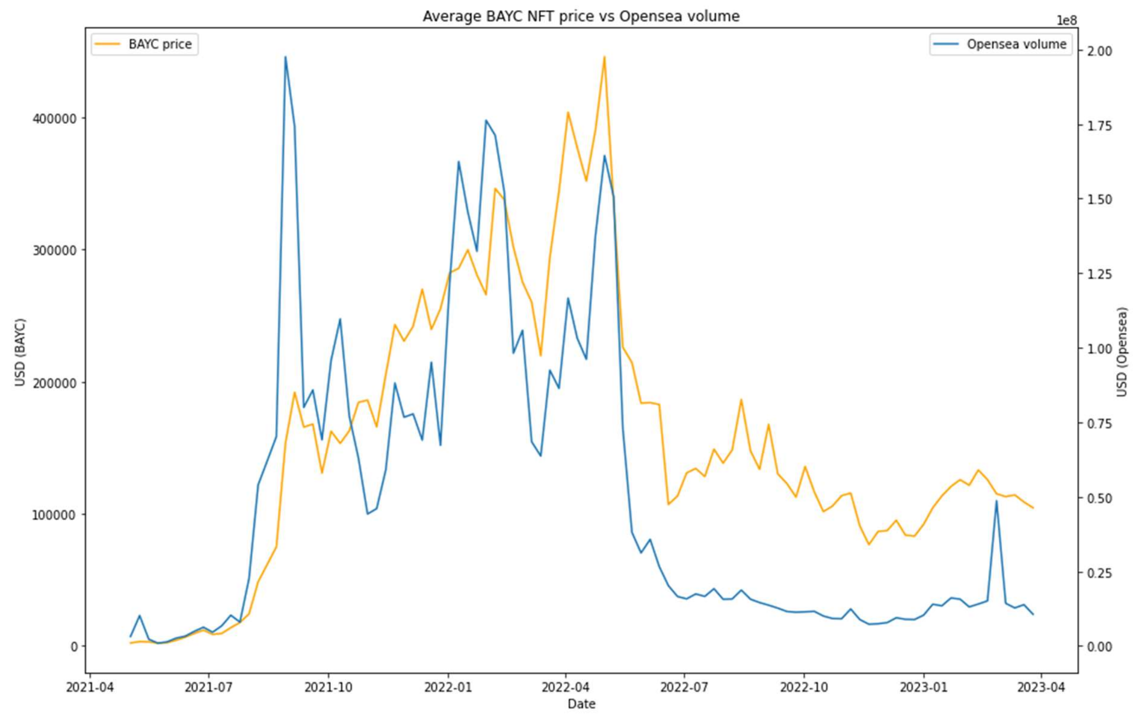


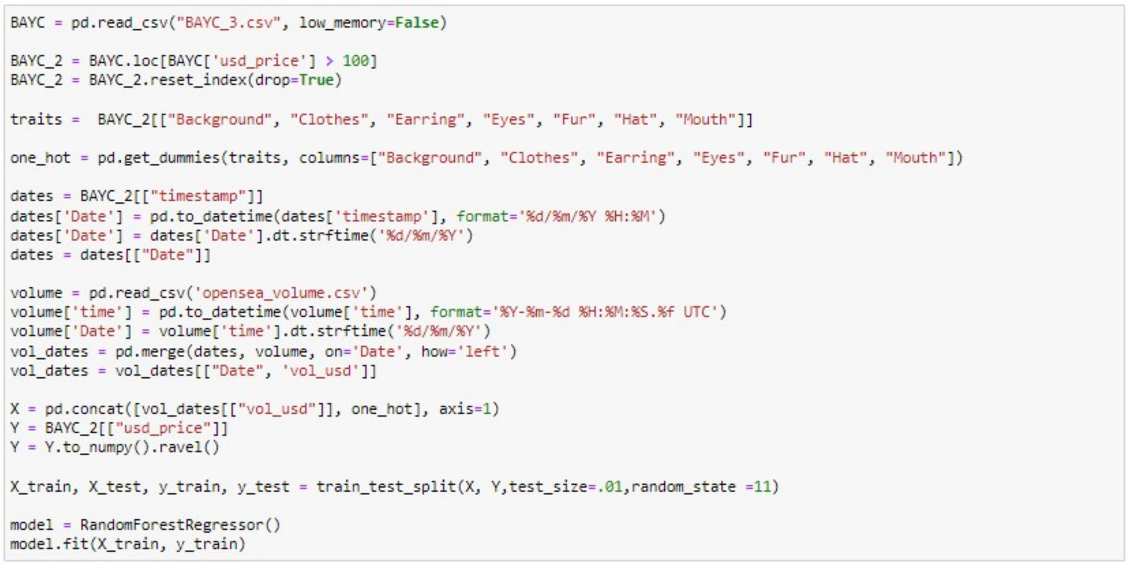
Figure 6

c) Observations

We observe here that BAYCs are sold at an average of $83,641 and that their standard deviation is extremely large at $122,046 (Figure 1). This means that there is a large disparity in the sales prices of the NFTs in the collection and this is mainly explained by the disparity in rarity between the NFTs due to their characteristics (traits) but also the volatility of this type of asset. Figure 2 shows that the collection increased in value from late 2021 to mid-2022, with a high in the week of May 1, 2022. Figures 3 and 4 are particularly interesting as they inform us of a significant positive correlation between the average price of BAYC NFTs and the average price of ETH as well as a strong positive correlation between the average price of BAYC NFTs and trading volume on Opensea. The same figures also show a positive correlation between ETH price and trading volume on Opensea. Figures 5 and 6 confirm all these observations and we can see that the curves evolve in a similar way. This information allows us to conclude that the price of ETH influences the volume of transactions on Opensea which in turn influences the price of BAYC NFTs. In the following section, we will therefore use the trading volume on Opensea to determine the fair price of the NFTs in the collection.

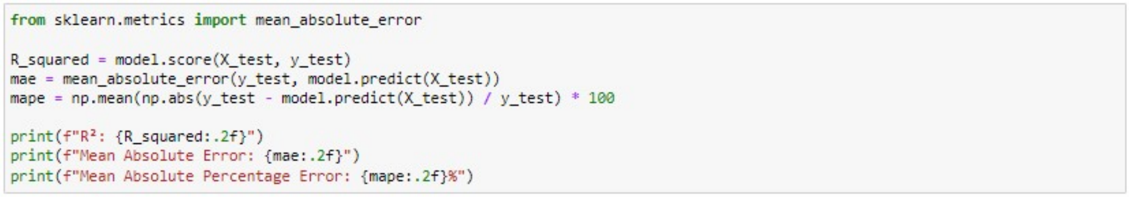
Machine learning model

a) Code explanation



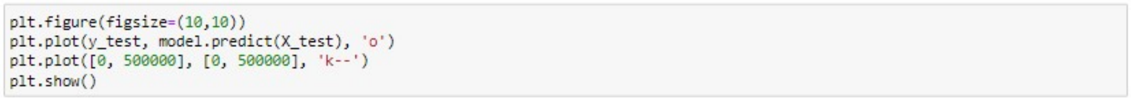
```
BAYC = pd.read_csv("BAYC_3.csv", low_memory=False)

BAYC_2 = BAYC.loc[BAYC['usd_price'] > 100]
BAYC_2 = BAYC_2.reset_index(drop=True)

traits =  BAYC_2[["Background", "Clothes", "Earring", "Eyes", "Fur", "Hat", "Mouth"]]

one_hot = pd.get_dummies(traits, columns=["Background", "Clothes", "Earring", "Eyes", "Fur", "Hat", "Mouth"])

dates = BAYC_2[["timestamp"]]
dates['Date'] = pd.to_datetime(dates['timestamp'], format='%d/%m/%Y %H:%M')
dates['Date'] = dates['Date'].dt.strftime('%d/%m/%Y')
dates = dates[["Date"]]

volume = pd.read_csv('opensea_volume.csv')
volume['time'] = pd.to_datetime(volume['time'], format='%Y-%m-%d %H:%M:%S.%f UTC')
volume['Date'] = volume['time'].dt.strftime('%d/%m/%Y')
vol_dates = pd.merge(dates, volume, on='Date', how='left')
vol_dates = vol_dates[["Date", 'vol_usd']]

X = pd.concat([vol_dates[["vol_usd"]], one_hot], axis=1)
Y = BAYC_2[["usd_price"]]
Y = Y.to_numpy().ravel()

X_train, X_test, y_train, y_test = train_test_split(X, Y,test_size=.01,random_state =11)

model = RandomForestRegressor()
model.fit(X_train, y_train)
```

This code is used to train a Random Forest regression model that returns the fair price of an NFT based on its characteristics. The model is trained with Opensea's transaction volume data (which as we have seen is correlated to the price of the NFTs) as well as with the NFTs' trait data. Transformations are made beforehand in order to transform the categorical data (here the traits) into numerical data because only this type of data can train a machine learning model. For this we use the one-hot encoder.



```
from sklearn.metrics import mean_absolute_error

R_squared = model.score(X_test, y_test)
mae = mean_absolute_error(y_test, model.predict(X_test))
mape = np.mean(np.abs(y_test - model.predict(X_test)) / y_test) * 100

print(f"R²: {R_squared:.2f}")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"Mean Absolute Percentage Error: {mape:.2f}%")
```

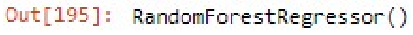
This code allows the model to be evaluated using 3 classical metrics provided for this purpose: The R^2, the mean absolute error and the mean absolute percentage error.



```
plt.figure(figsize=(10,10))
plt.plot(y_test, model.predict(X_test), 'o')
plt.plot([0, 500000], [0, 500000], 'k--')
plt.show()
```

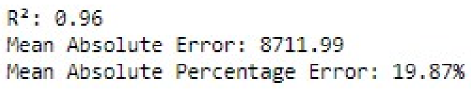
This code allows to display on the same graph the couples (true selling price, selling price prediction by the model) as well as the line of equation x=y materializing the perfect prediction.

b) Results



```
Out[195]: RandomForestRegressor()
```

Figure 1



```
R²: 0.96
Mean Absolute Error: 8711.99
Mean Absolute Percentage Error: 19.87%
```

Figure 2

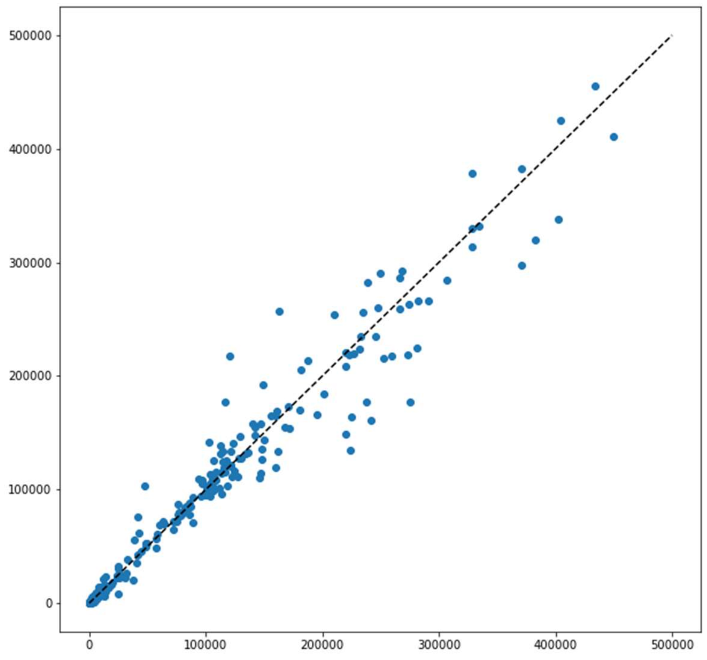


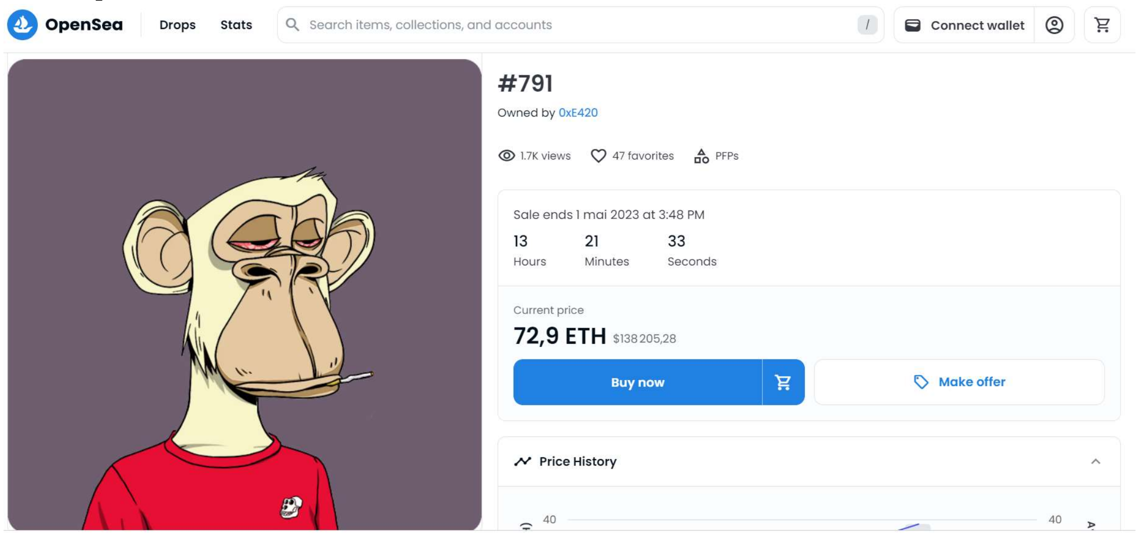
Figure 3

c) Observations

We observe here that our model predicts very well the selling prices of the NFTs of the BAYC collection. Indeed, the 3 metrics R^2, mean absolute error and absolute percentage error show that our model is very good. The R^2 of 0.96 tells us that the model explains 96% of the variance of the data which is a very good performance. The MAE of 8711.99 USD might seem to be a high error but we have to remember that the average selling price of the NFTs in the collection is 83,641 dollars, so the error is not negligible but not bad either. The MAPE of 19.87% means that, on average, the model predictions are within 19.87% of the true values which is correct for our data which has high variability. Finally, Figure 3 shows us that the model is close to a perfect model, indeed, the couples (true selling price, selling price prediction by the model) are close to the straight line of equation y=x.

Test of the model as an investor

Let's now try our model on an NFT that is currently for sale to see if it is over or under priced:



#791

Owned by 0xE420

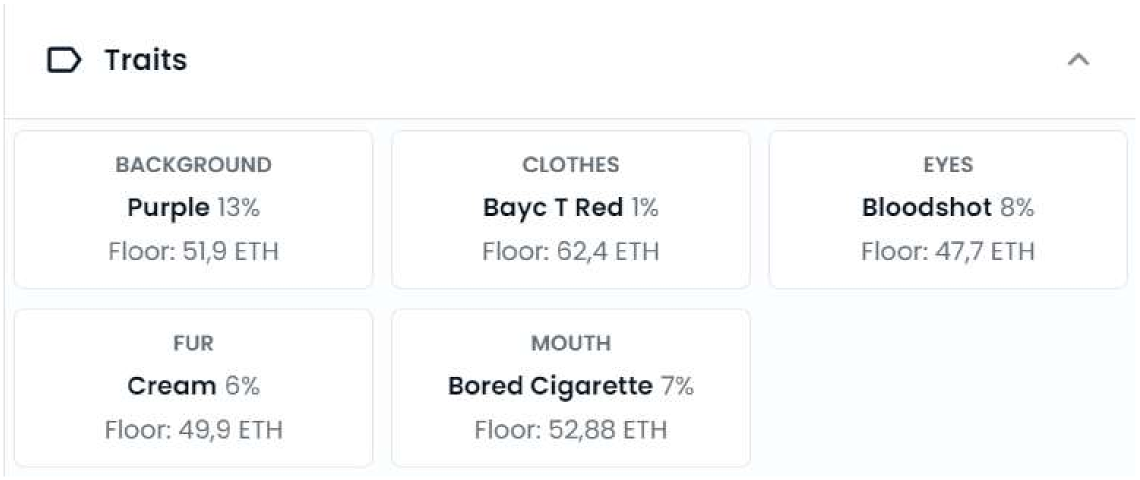
1.7K views 47 favorites PFPs

Sale ends 1 mai 2023 at 3:48 PM

13 Hours 21 Minutes 33 Seconds

Current price

72,9 ETH $138 205,28

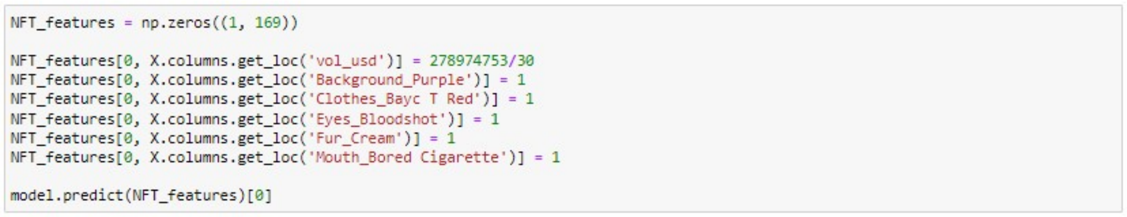


Traits

BACKGROUND Purple 13% Floor: 51,9 ETH	CLOTHES Bayc T Red 1% Floor: 62,4 ETH	EYES Bloodshot 8% Floor: 47,7 ETH
FUR Cream 6% Floor: 49,9 ETH	MOUTH Bored Cigarette 7% Floor: 52,88 ETH	

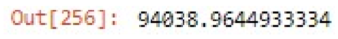
We observe that this NFT is sold at $138,205.28.

Let's now model the vector representing this NFT using its features and the average trading volume on Opensea in April 2023 ($278,974,753/30 days) :



```
NFT_features = np.zeros((1, 169))

NFT_features[0, X.columns.get_loc('vol_usd')] = 278974753/30
NFT_features[0, X.columns.get_loc('Background_Purple')] = 1
NFT_features[0, X.columns.get_loc('Clothes_Bayc T Red')] = 1
NFT_features[0, X.columns.get_loc('Eyes_Bloodshot')] = 1
NFT_features[0, X.columns.get_loc('Fur_Cream')] = 1
NFT_features[0, X.columns.get_loc('Mouth_Bored Cigarette')] = 1

model.predict(NFT_features)[0]
```

Let us now observe the output of this code corresponding to the fair price of this NFT according to our model:



```
Out[256]:  94038.9644933334
```

Our model therefore indicates that the fair price for this NFT based on its features and the current market for NFTs is $94,038.96 which is below the selling price of this NFT (138,205.28 > 94,038.96). We can therefore conclude from our model that this NFT is not a good deal and that it is overpriced.

Conclusion

The analysis of on-chain data of transactions of NFT collections allowed us to determine a high correlation between the price of NFTs in a collection, the price of ETH and the volume of transactions on Opensea.

Thanks to these findings we were able to design an accurate model that allows us to determine the fair price of an NFT based on the rarity of its traits as well as the state of the NFT market.

We can therefore conclude that data is very valuable for investors and that it can allow them to better manage their investment.