Content

Content statement of the problem. Data Description

One very well-known international cryptocurrency and blockchain investment company X has decided to try to invest in the growing area of non-fungible tokens (NFTs), and it claims to distinguish between different groups of NFTs that have a high chance of success and the least interesting sets of NFTs.

The data obtained was taken from the coinmarketcap.com website and was collected into one dataset and made publicly available on kaggel.com. Data format *.csv. The set of researched data, without pre-processing, is 587 rows and 17 columns, containing all the necessary information for evaluation, for example, the name, trading volume, capitalization, sales, the number of NFTs in the collection and others, more detailed information with a description of the data is below.

- Index: Index in the file;
- Name: The name of the NFT collection;
- Volume: Sales volume from the NFT collection in Solana (SOL);
- Volume_USD: Sales volume from the NFT collection in US Dollars (USD).
- Market_Cap: Market cap - the total number of NFT items in the collection that are in circulation - in Solana (SOL).
- MarketCapUSD: Market cap - the total number of NFT items in the collection that are in circulation - in United States Dollars (USD);
- Sales: Number of sales from the NFT collection;
- Floor_Price: The minimum price for any NFT from the collection in Solana (SOL);
- FloorPriceUSD: The minimum price for any NFT from the collection in US Dollars (USD);
- Average_Price: Average price for any NFT from the collection in Solana (SOL);
- AveragePriceUSD: The average price for any NFT from the collection in US Dollars (USD);
- Owners: Number of NFT owners in the collection;
- Assets: The number of items in the collection;

• OwnerAssetRatio: Percentage of ownership of all items in the collection;

• Category: Category of the NFT collection;

• Website: Website of the NFT collection;

• Logo: An image of the NFT collection.

After pre-processing, lines that did not contain data or those lines that had an error were removed, their number decreased significantly, to 238, which is not enough, but at least something.

	Index	Name	Volume	Volume_USD	Market_Cap	Market_Cap_USD	Sales	Floor_Price	Floor_Price_USD	Average_Price	Average_Price_USD
0	0	basis.markets	27256.63	4.001818e+06	708.145455	103969.915600	0.073494	39.50	0.237866	74.471667	1.000000
5	5	I'M AIKO	2904.70	4.264681e+05	2530.877143	371583.382100	0.283333	1.20	0.007226	2.058611	0.027643
9	9	Meta Waifus	1844.59	2.708227e+05	989.491267	145277.107800	0.232129	1.27	0.007648	1.595666	0.021426
12	12	Hot Bunnies NFT	1590.89	2.335745e+05	527.850000	77498.937000	0.082731	1.00	0.006022	3.861383	0.051850
13	13	BOSS BULLS ™ CLUB	1236.33	1.815180e+05	244.686667	35924.896400	0.074297	1.49	0.008973	3.341432	0.044869
...	...	...	...	...	...	...	...	...	...	...	...
569	575	META OCEAN BOX	2.65	3.890730e+02	4.095455	601.294636	0.002209	0.20	0.001204	0.240909	0.003235
578	584	Cannababy Society	1.00	1.468200e+02	13.000000	1908.660000	0.000201	1.00	0.006022	1.000000	0.013428
579	585	Mountain Lionz	1.00	1.468200e+02	5.916667	868.685000	0.002410	0.09	0.000542	0.083333	0.001119
580	586	AI Motion Art	0.70	1.027740e+02	0.000000	0.000000	0.000201	0.50	0.003011	0.700000	0.009400
581	587	SolNFTPad	0.60	8.809200e+01	123.000000	18058.860000	0.000201	0.31	0.001867	0.600000	0.008057

238 rows × 17 columns

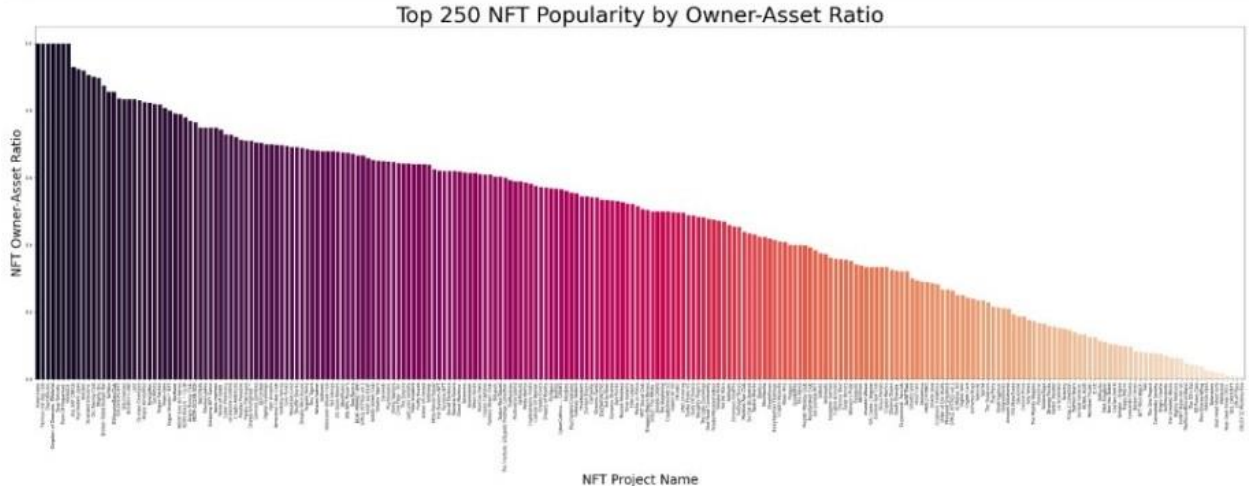
Picture 1 – data set after cleaning

For data processing, the Python programming language was used, due to the extensive set of libraries and documentation. The Pandas library allows you to process data and prepare them for analysis, Matplotlib is necessary to display data when obtaining results obtained using visual tools, and other libraries were used during processing and to achieve the results necessary for the study.

From further research, during the initial data processing, such signs as Volume, Market_Cap, Average_Price, Floor_Price were removed, since they only duplicate the same indicators, only in US dollars, and for convenience, they were left, Website, Logo, Category, were removed because we do not need them for analysis, these are not numerical indicators and do not represent any value for analysis.

Next, the features Sales, Average_Price_USD, Floor_Price_USD, Owners, Assets, Owners_Assets_Ratio were normalized in order to further rank NFTs relative to Owners_Assets_Ratio and build the top best NFTs.

```
plt.subplots(figsize=(50,15))
sns.barplot(x=top_nft.Name,y=top_nft.Owner_Asset_Ratio.sort_values(),palette = "rocket")
plt.xticks(rotation = 90)
plt.xlabel("NFT Project Name",fontsize =30)
plt.ylabel("NFT Owner-Asset Ratio",fontsize =30)
plt.title("Top 250 NFT Popularity by Owner-Asset Ratio",fontsize =50)
plt.show()
```



Picture 2 – top best NFTs by people's choice

Regarding this, let's look at the data correlation table and build a heat map to see everything more clearly.

```
nft.corr()
```

	Index	Volume_USD	Market_Cap_USD	Sales	Floor_Price_USD	Average_Price_USD	Owners	Assets	Owner_Asset_Ratio
Index	1.000000	-0.194681	-0.352059	-0.387925	-0.021775	-0.222767	-0.336481	-0.333560	-0.274713
Volume_USD	-0.194681	1.000000	0.336854	0.094460	0.217398	0.833787	0.050087	0.041804	0.089185
Market_Cap_USD	-0.352059	0.336854	1.000000	0.357250	0.025849	0.197482	0.387028	0.453841	-0.000741
Sales	-0.387925	0.094460	0.357250	1.000000	-0.034031	-0.026963	0.874261	0.840765	0.102533
Floor_Price_USD	-0.021775	0.217398	0.025849	-0.034031	1.000000	0.363359	-0.049177	-0.059221	0.059223
Average_Price_USD	-0.222767	0.833787	0.197482	-0.026963	0.363359	1.000000	-0.062952	-0.077243	0.038352
Owners	-0.336481	0.050087	0.387028	0.874261	-0.049177	-0.062952	1.000000	0.927723	0.137130
Assets	-0.333560	0.041804	0.453841	0.840765	-0.059221	-0.077243	0.927723	1.000000	-0.038661
Owner_Asset_Ratio	-0.274713	0.089185	-0.000741	0.102533	0.059223	0.038352	0.137130	-0.038661	1.000000

```
# Смотрим на корреляцию
```

Picture 3 – correlation table

```
plt.subplots(figsize=(10,10))
sns.heatmap(nft.corr(),annot=True,linewidths = 1)
plt.show()
```

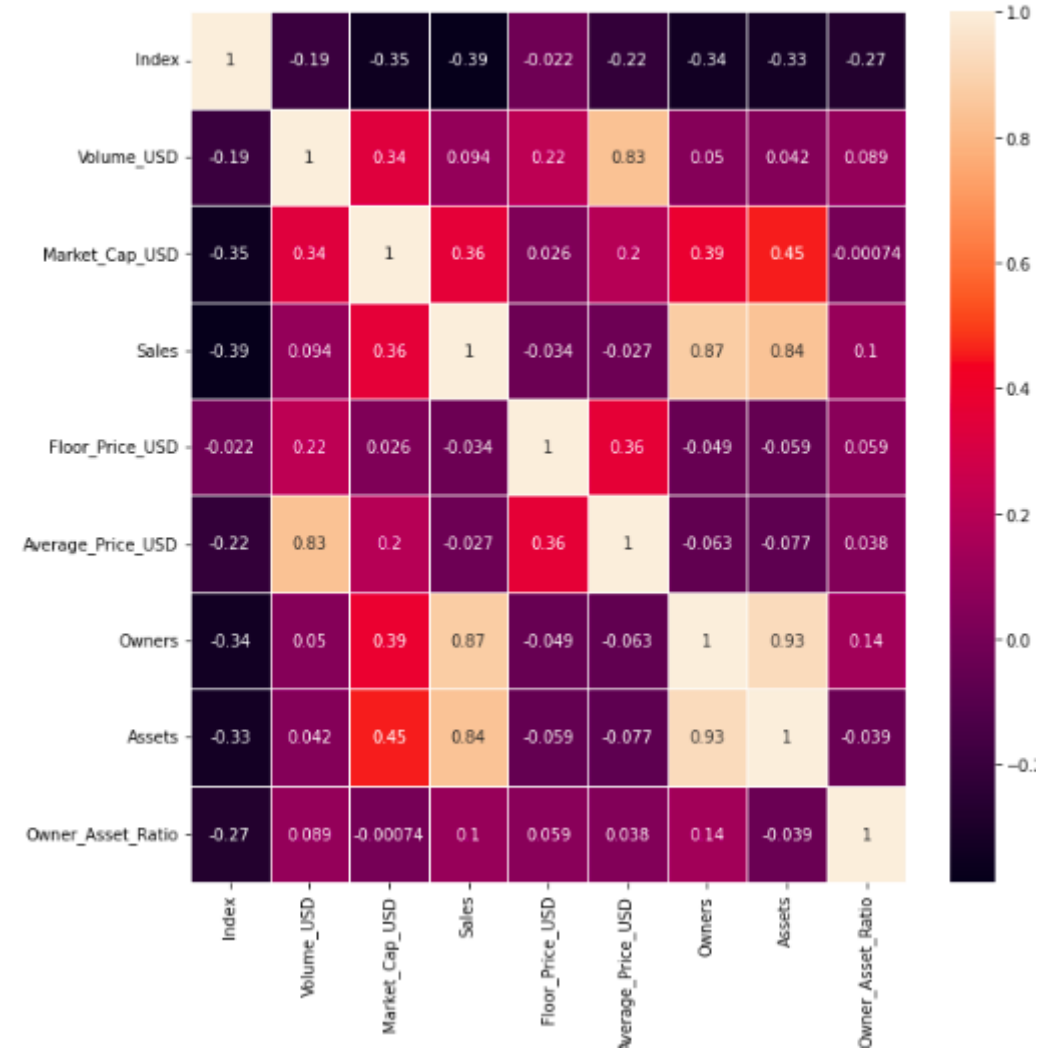
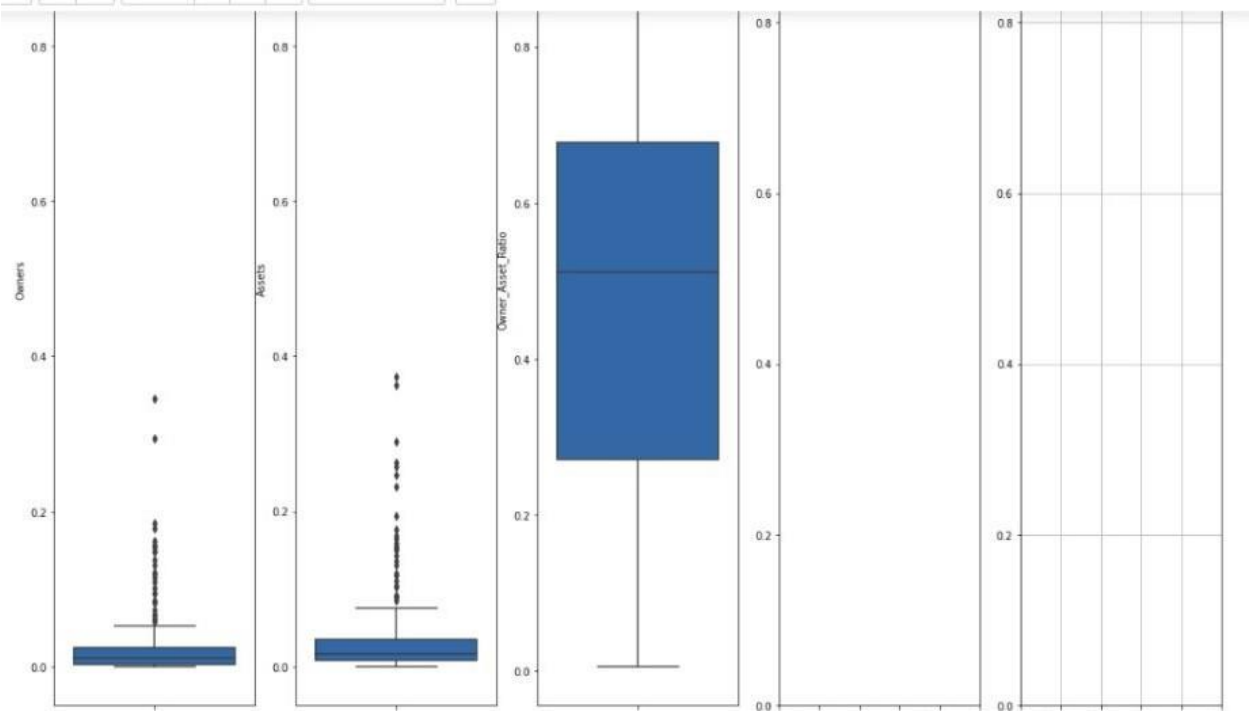


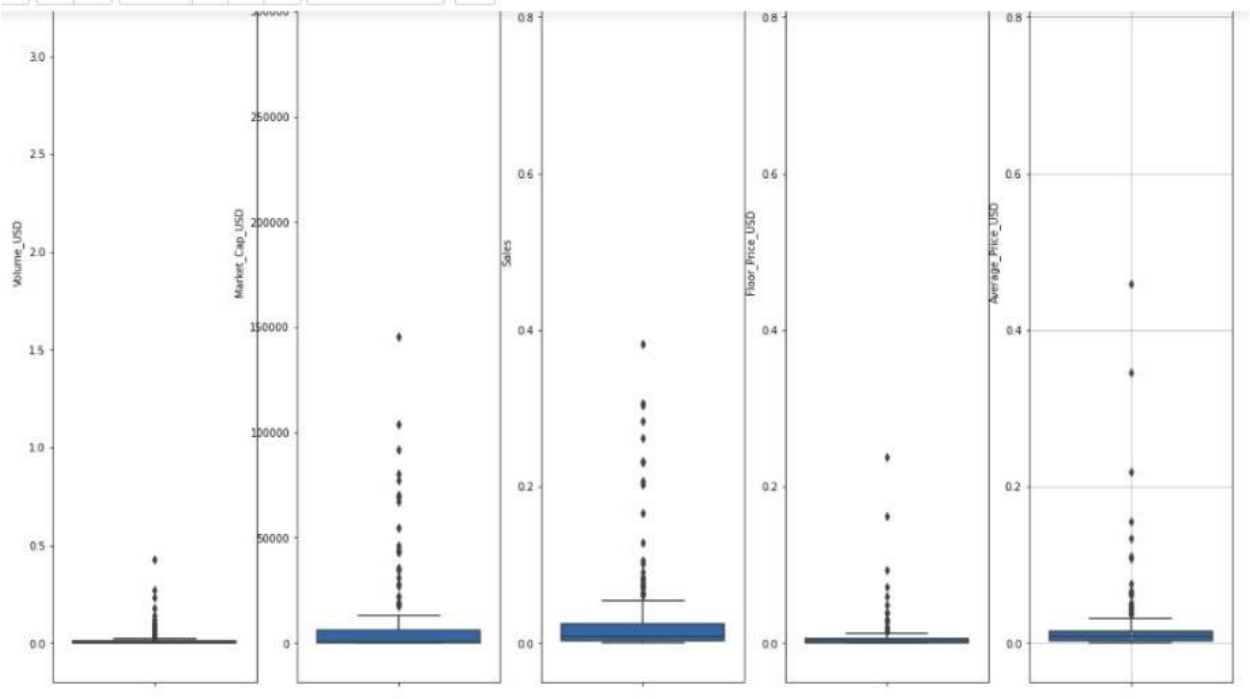
Рисунок 4 – heat map

On the heat map, everything can be seen much better and much clearer. We will delete the Index sign in the future, we do not need it. The Assets and Owners features are interconnected and highly correlated, and purely by the description of these features, so there is no need to remove one of these features. Average_Price and Market_Cap_USD are also related in their meaning, capitalization depends on the average price by definition. Sales, along with Owners and Assets, are also related by definition, and there is no need to remove any of these features. Accordingly, all other data correlate more or less within adequate limits, which suggests that clustering can be carried out and there will be no incomprehensible errors, everything explains the data. But in order to bring the data into a normal form for

clustering, let's look at their normal distribution plots and outliers, if any. From a preliminary analysis, it was found that some features, namely, Market_Cap_USD, Volume_USD, Sales, Average_Price_USD, Floor_Price_USD, Owners, Assets, need to be logarithmic, because their charts do not follow the normal distribution law. Accordingly, the outlier plots and normal distribution plots are below.

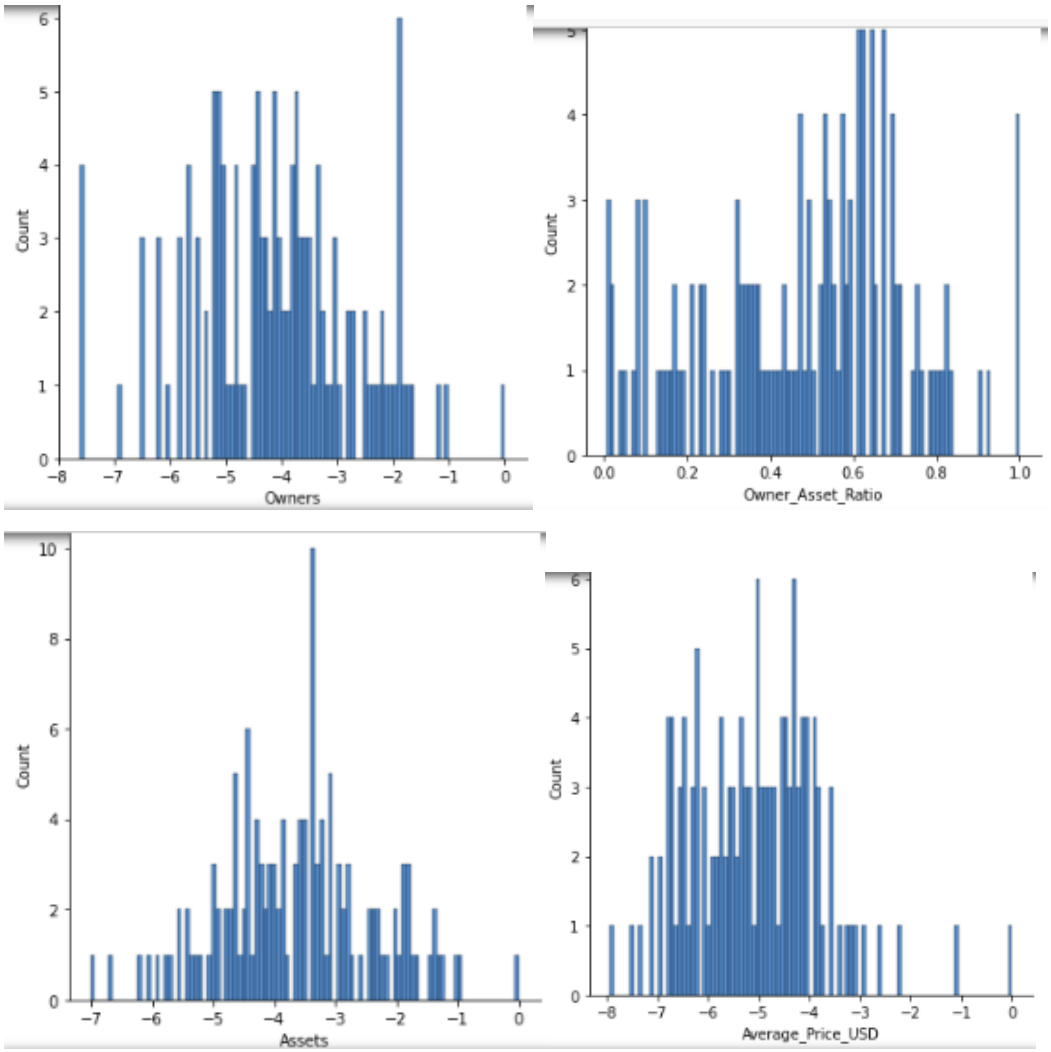


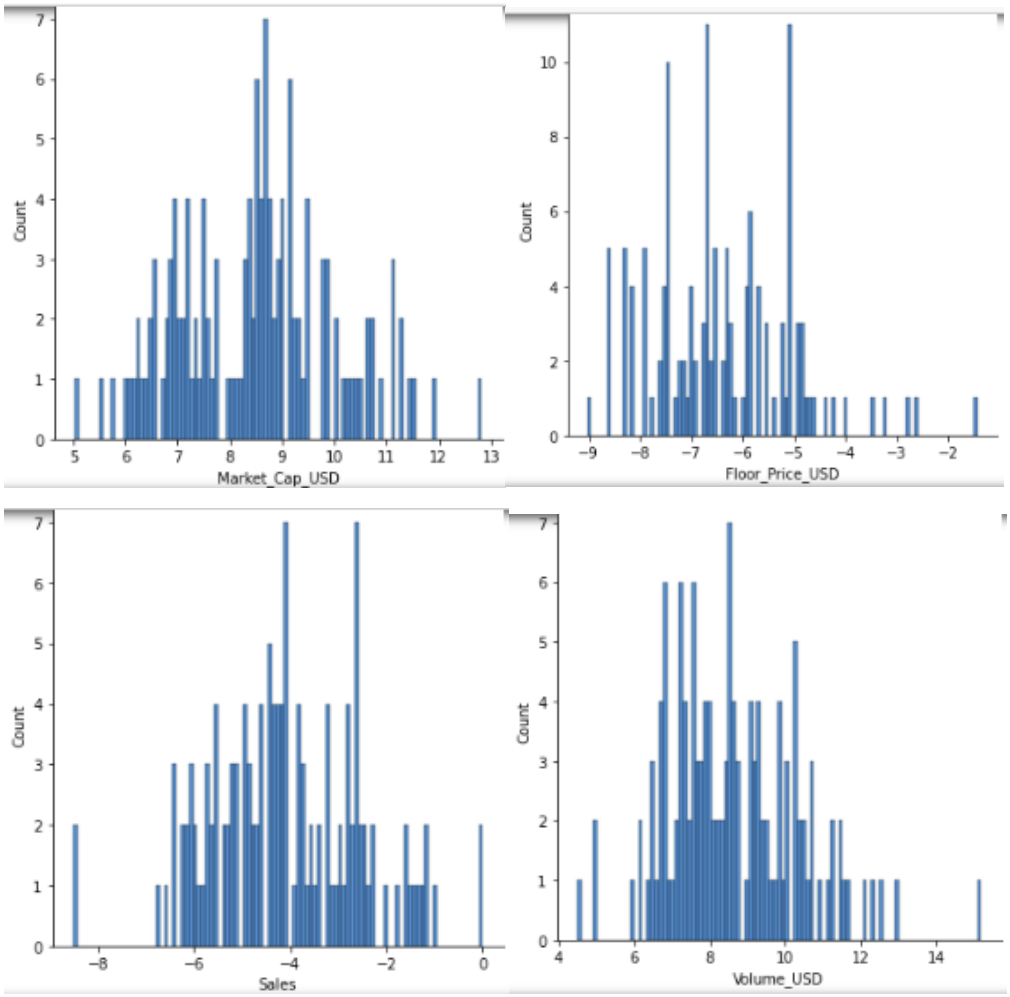
Picture 5 – Box plot to view outliers 1



Picture 6 – Box plot to view outliers 2

This type of graphs is explained by the fact that there is very little data, a lot of values are at some point, and only a small, small part is very different from the average values of the features. In the world of blockchain, cryptocurrencies and NFTs, this can be considered the norm, since these markets have a lot of volatility, and given the features of NFTs (tokens are not fungible), this can be conditionally considered the norm.





Picture 7 – feature distribution

As you can imagine, the logarithm of the features I chose helped, and now the features obey the normal distribution law.

Matrix of Euclidean distances.

In order to build a matrix of Euclidean distances, we remove all non-numeric Name features and the Index column in our dataset, and build this very matrix.

```
nft = nft.drop(columns = ["Name"],axis =1)
```

```
nft = nft.drop(columns = ["Index"],axis =1)
```

```
euclide = euclidean_distances(nft,nft)
```

```
# Расчитываем евклидово расстояние
```

```
df = pd.DataFrame(euclide)
df
```

	0	1	2	3	4	5	6	7	8	9	...	122	123	124	
0	0.000000	8.220669	7.797892	7.051633	6.310266	11.257290	8.083051	12.685423	8.729753	7.128480	...	13.061947	13.225932	13.318906	12.933
1	8.220669	0.000000	1.347643	2.619554	4.080226	4.251217	4.663315	5.347480	3.121791	5.785319	...	11.945317	11.355184	11.717735	10.068
2	7.797892	1.347643	0.000000	1.660908	2.903683	3.905457	3.374208	5.184753	2.282798	4.490720	...	10.746623	10.169863	10.516543	9.036
3	7.051633	2.619554	1.660908	0.000000	1.715224	4.853835	3.023835	6.065123	1.912447	3.566770	...	9.769529	9.385861	9.682711	8.264
4	6.310266	4.080226	2.903683	1.715224	0.000000	5.655206	2.278120	6.997720	2.819968	2.027657	...	8.729786	8.447281	8.691262	7.688
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
127	13.179416	11.603717	10.432414	9.543731	8.592211	10.452248	7.876116	11.098360	8.801205	7.030872	...	0.965402	1.193034	1.049180	2.436
128	13.601811	12.432815	11.232256	10.313933	9.271889	11.256483	8.554719	11.907316	9.632797	7.622937	...	0.821914	1.708059	1.262073	3.398
129	13.924617	14.356803	13.265466	12.173725	11.076780	14.186685	11.040627	15.049699	11.865022	9.666807	...	4.517204	5.411139	5.182395	6.154
130	15.042613	12.423313	11.329814	10.630731	9.932135	10.739532	8.978577	11.090065	9.613192	8.518847	...	2.917133	2.399905	2.482076	2.706
131	14.465243	12.046523	11.221901	10.371010	9.984348	11.688644	9.931778	12.144540	9.570646	9.215492	...	5.468908	5.576446	5.729582	3.753

132 rows × 132 columns

```
# Выводим в нормальном виде
```

Picture 8 – Euclidean distance

And a bit of theory, what is this Euclidean distance? This is the shortest straight line between two points in Euclidean space, and in our case, between features.

Generally speaking, Euclidean distance is widely used in the development of 3D worlds, as well as machine learning algorithms that include distance metrics such as K-nearest neighbors. Generally, the Euclidean distance will represent how similar two data points are, assuming some clustering based on other data has already been done.

The mathematical formula for calculating the distance between two points in 2D space is:

$$d(p,q) = \sqrt[2]{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

The general formula can be simplified to this form:

$$d(p,q) = \sqrt[2]{(q_1 - p_1)^2 + \ldots + (q_n - p_n)^2}$$

Also, we can say that this formula is very similar to the Pythagorean theorem:

$$C^2 = A^2 + B^2$$

$$d(p,q)^2 = (q_1 - p_1)^2 + (q_2 - p_2)^2$$

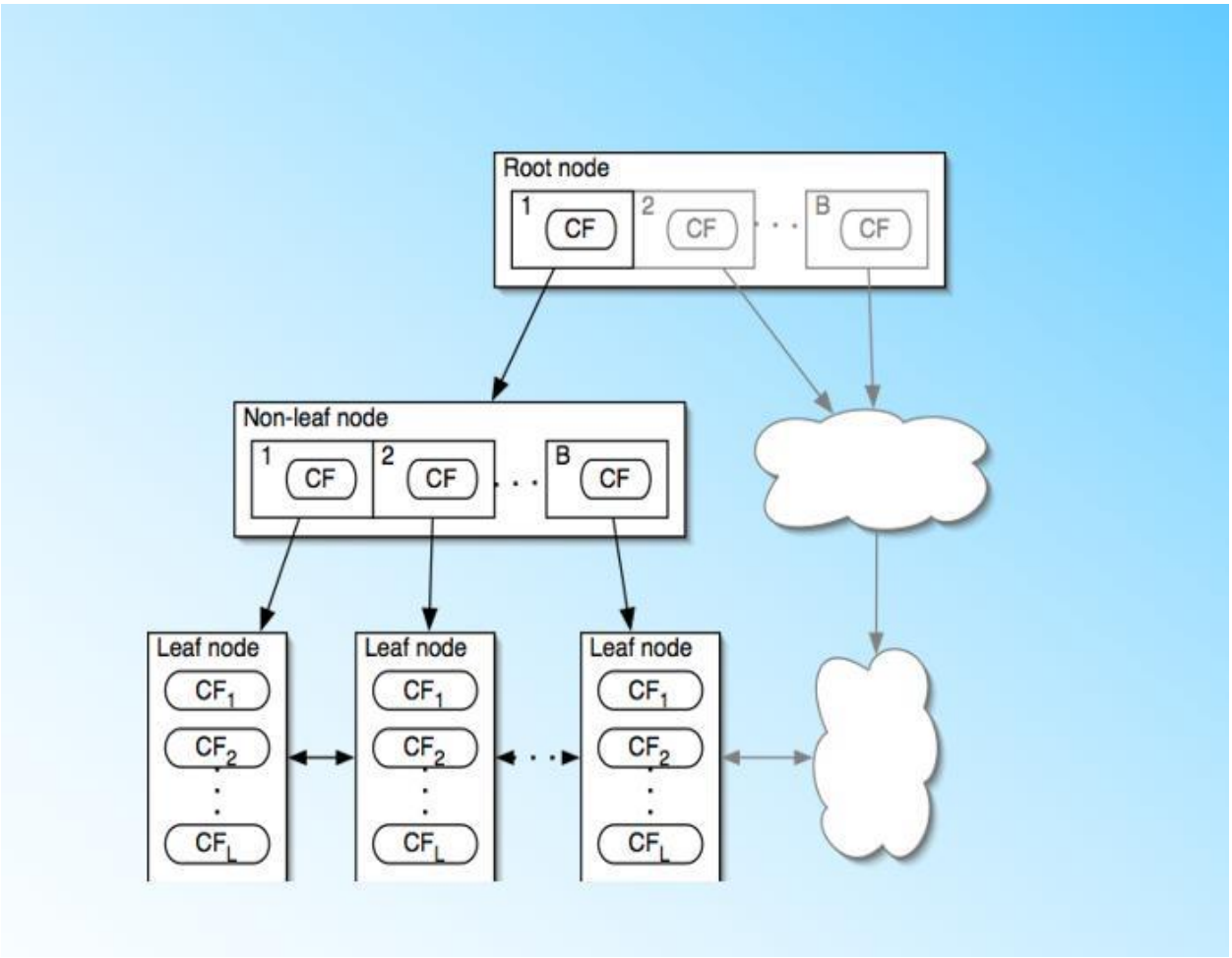
And in fact, there is a connection between them, the Euclidean distance is calculated using the Pythagorean theorem, given the Cartesian coordinates of two points.

Description of the selected algorithm

As a clustering algorithm, the customer has set a condition in advance that the BIRCH method should be applied.

The full name of BIRCH is balanced iterative reduction and clustering using hierarchies. Here, a hierarchical method is used to cluster and reduce the data upwards, and the method needs only one pass to perform the clustering.

The BIRCH algorithm uses a tree structure that helps us cluster quickly. This numerical structure is similar to a balanced B+ tree. It is commonly referred to as the Clustering Feature Tree (CF Tree). Each node of this tree consists of several clustering functions (CFs). In the figure below, we can see what the clustering feature tree looks like: each node, including leaf nodes, has multiple CFs, and the CFs of internal nodes have pointers to child nodes, and all leaf nodes are linked by a doubly linked list.



Picture 9 – clustering function tree

In a nutshell, the BIRCH Algorithm does not require the entry of a category number K value, which is different from K-means and mini-batch K-means. If no K value is entered, the number of last CF tuples will be the final K, otherwise the CF tuples will be concatenated according to the distance according to the input K value.

Generally speaking, the BIRCH algorithm is suitable for situations with a large sample size, which is similar to K-means mini-batch, but BIRCH is suitable for situations where the number of categories is relatively large, while K-means mini-batch is commonly used. for medium or relatively large categories. When less. In addition to clustering, BIRCH can also perform additional outlier detection and data preprocessing according to category specifications. However, if the size of the data feature is very large, such as greater than 20, BIRCH is not appropriate. Currently K-means Mini Batch works better.

To tune the parameters, BIRCH is more complicated than K-Means and Mini Batch K-Means because it needs to tune a few key CF tree parameters that have a big impact on the final shape of the CF tree.

In conclusion, summarizing the advantages and disadvantages of the BIRCH algorithm:

The main advantages of the BIRCH algorithm are:

1) Save to memory, all samples are on disk, only CF nodes and corresponding pointers are stored in the CF tree.

2) The clustering speed is high and the CF tree can be created by scanning the training set only once. Adding, removing, and modifying a CF tree is fast.

3) Can identify noise points and also can pre-process dataset for pre-classification

The main disadvantages of the BIRCH algorithm are:

1) Since the CF tree has a limit on the number of CFs for each node, the result of clustering may differ from the actual distribution of categories.

2) The clustering effect of high-dimensional spatial data is not very good. Currently you can choose Mini Batch K-Means

3) If the dataset distribution cluster is not like a hypersphere or is not convex, the clustering effect is bad.

Algorithm execution results. Conclusions

```
# Кластеризация методом BIRCH (Береза)
```

```
model = Birch(n_clusters = 4)
```

```
model.fit(nft)
pred = model.predict(nft)
```

```
pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 0, 0, 0, 0, 1, 3, 0, 0, 3, 0, 0,
       1, 0, 3, 3, 1, 0, 3, 0, 3, 3, 3, 3, 3, 1, 0, 0, 3, 3, 3, 3, 3, 3,
       1, 3, 3, 3, 0, 3, 0, 3, 1, 3, 3, 3, 3, 1, 2, 3, 3, 3, 3, 1, 3, 3,
       1, 0, 2, 1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 3, 3, 2, 3, 1, 2, 2, 2,
       1, 2, 2, 2, 2, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 2, 1, 2, 2, 1,
       2, 2, 2, 1, 2, 2, 3, 1, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1])
```

```
np.unique(pred, return_counts = True)
```

```
(array([0, 1, 2, 3]), array([25, 31, 36, 40]))
```

```
pred = pd.DataFrame(pred)
pred
```

	0
0	0
1	0
2	0
3	0
4	0
...	...
127	2
128	2
129	1
130	2
131	1

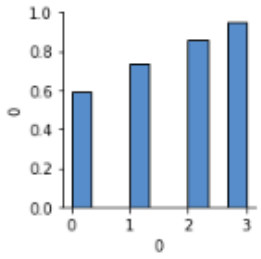
132 rows × 1 columns

```
pred.describe()
```

	0
count	132.000000
mean	1.689394
std	1.099150
min	0.000000
25%	1.000000
50%	2.000000
75%	3.000000
max	3.000000

```
sns.pairplot(pred)
```

```
<seaborn.axisgrid.PairGrid at 0x7f84be6bfc40>
```



Picture 10 – clustering results

As a result of the algorithm, I decided to split the data into 4 clusters, because this clustering algorithm is not suitable for such a small amount of data, it is more good and efficient when there is a large data sample. When choosing 2 clusters, it will turn out not to understand what, and at the same time it will not satisfy the problem statement, When choosing 3 clusters, they are distributed unevenly, and there are almost no objects in one cluster, and 4 clusters, the most optimal option, this again can be seen in figure 10.

In the course of performing clustering by the selected BIRCH method, 4 clusters were obtained, in the first 25 objects, in the second 31, in the third 36, in the fourth 40.

We consider the first and second clusters to be the most interesting clusters for company X, the first, since this cluster has the most expensive and most popular NFTs, and the second, since there are not the most expensive NFTs that you can invest in.

If we consider NFTs from the second cluster, then these are the most desirable things to buy in order to earn in the medium or short term.

NFTs from the third cluster are NFTs that are very risky to buy for investment, as there is a high chance that most of them do not represent any value.

NFTs from the fourth cluster are not recommended for purchase, it is better to bypass them, too little interest, capitalization and other signs, when buying them, it is better to look at individual copies of the collection, they can be of at least some value.

As a result, BIRCH clustering on this dataset is not representative, this method is suitable for a larger sample, which cannot be said about the sample given to me. This method will be good only if we know exactly how many clusters we have in the dataset, here we only assumed that there would be 4 clusters. To obtain a reliable and correct clustering result, in this case, another method is needed, such as K-means, or, it is necessary to collect more data on various NFT collections, using not only coinmarketcap.com, but also other social networks or other sources of information.