Содержание

Содержательная постановка задачи. Описание данных

Одна очень известная международная компания X, занимающаяся инвестициями в криптовалюту и блокчейн решила попробовать заняться инвестициями в набирающую популярность область невзаимозаменяемых токенов (NFT), и она требует выделить разные группы NFT, которые имеют большой шанс на успех, и наименее интересные наборы NFT.

Полученные данные были взяты с сайта coinmarketcap.com, и были собраны в один датасет и выложены в общий доступ на kaggel.com. Формат данных *.csv. Набор исследуемых данных, без предварительной обработки составляет 587 строк и 17 колонок, содержащие всю необходимую информацию для оценки, например, название, объем торгов, капитализация, продажи, количество NFT в коллекции и другие, более подробная информация с описанием данных находится ниже.

- Index: Индекс в файле;
- Name: Имя NFT коллекции;
- Volume: Объем продаж из NFT коллекции в Solana (SOL);
- Volume_USD: Объем продаж из NFT коллекции в Долларах США (USD).
- Market_Cap: Рыночная капитализация – итоговое количество предметов NFT находящихся в коллекции, которые находятся в обороте – в Solana (SOL).
- MarketCapUSD: Рыночная капитализация – итоговое количество предметов NFT находящихся в коллекции, которые находятся в обороте – в Долларах США (USD);
- Sales: Количество продаж из коллекции NFT;
- Floor_Price: Минимальная цена за любое NFT из коллекции в Solana (SOL);
- FloorPriceUSD: Минимальная цена за любое NFT из коллекции в Долларах США (USD);

- Average_Price: Средняя цена за любое NFT из коллекции в Solana (SOL);
- AveragePriceUSD: Средняя цена за любое NFT из коллекции в Долларах США (USD);
- Owners: Количество владельцев NFT в коллекции;
- Assets: Количество предметов в коллекции;
- OwnerAssetRatio: Процент владения всеми предметами в коллекции;
- Category: Категория NFT коллекции;
- Website: Вебсайт NFT коллекции;
- Logo: Изображение NFT коллекии.

После предварительной обработки, были убраны строки, не содержащие данных или те строки, которые имели ошибку, их число значительно уменьшилось, до 238, это мало, но уже хоть что-то.

	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

Рисунок 1 – набор данных после отчистки

Для обработки данных использовался язык программирования Python, в силу обширного набора библиотек и документации. Библиотека Pandas позволяет обработать данные и подготовить их для анализа, Matplotlib необходим для отображения данных при получении результатов , полученных при помощи визуальных инструментов, также при обработке и для достижения необходимых для исследования результатов были использованы другие библиотеки.

Из дальнейших исследований, при первичной обработки данных были убраны такие признаки, как Volume, Market_Cap, Average_Price, Floor_Price, так как они лишь дублируют такие же показатели, только в Долларах США, и для удобства были оставлены именно они, Website, Logo, Category, были убраны потому что они нам не нужны для анализа, это не числовые показатели и ценности для анализа никакого не представляют.

Далее были нормированы признаки Sales, Average_Price_USD, Floor_Price_USD, Owners, Assets, Owners_Assets_Ratio, для того, чтобы в дальнейшем составить рейтинг NFT относительно Owners_Assets_Ratio построить топ лучших NFT.

```
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()
```

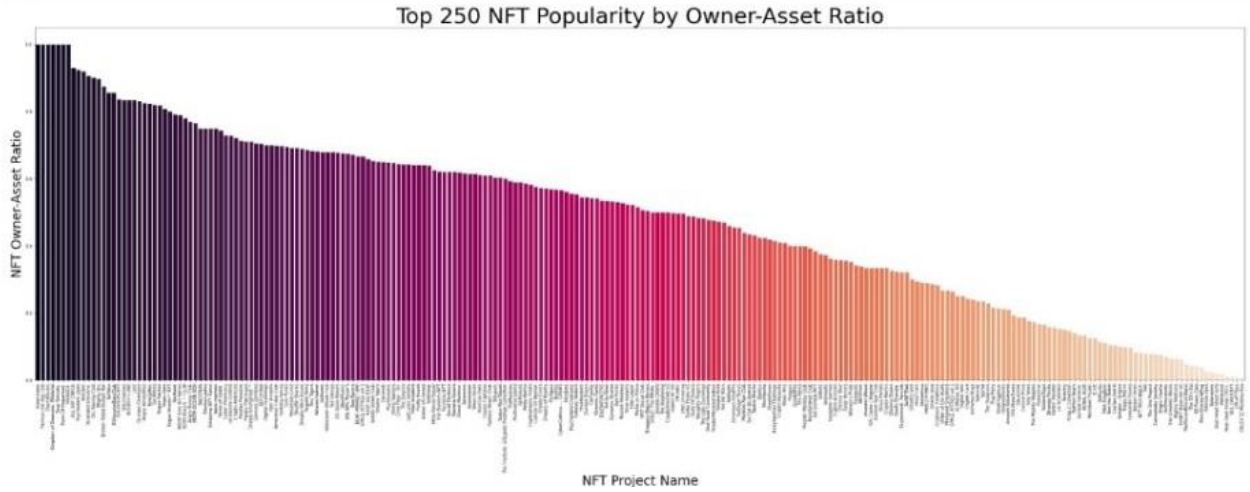


Рисунок 2 – топ лучших NFT по выбору людей

Относительно этого посмотрим на таблицу корреляции данных и построим тепловую карту, чтоб видеть все более наглядно.

```
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
```

Рисунок 3 – таблица корреляции

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

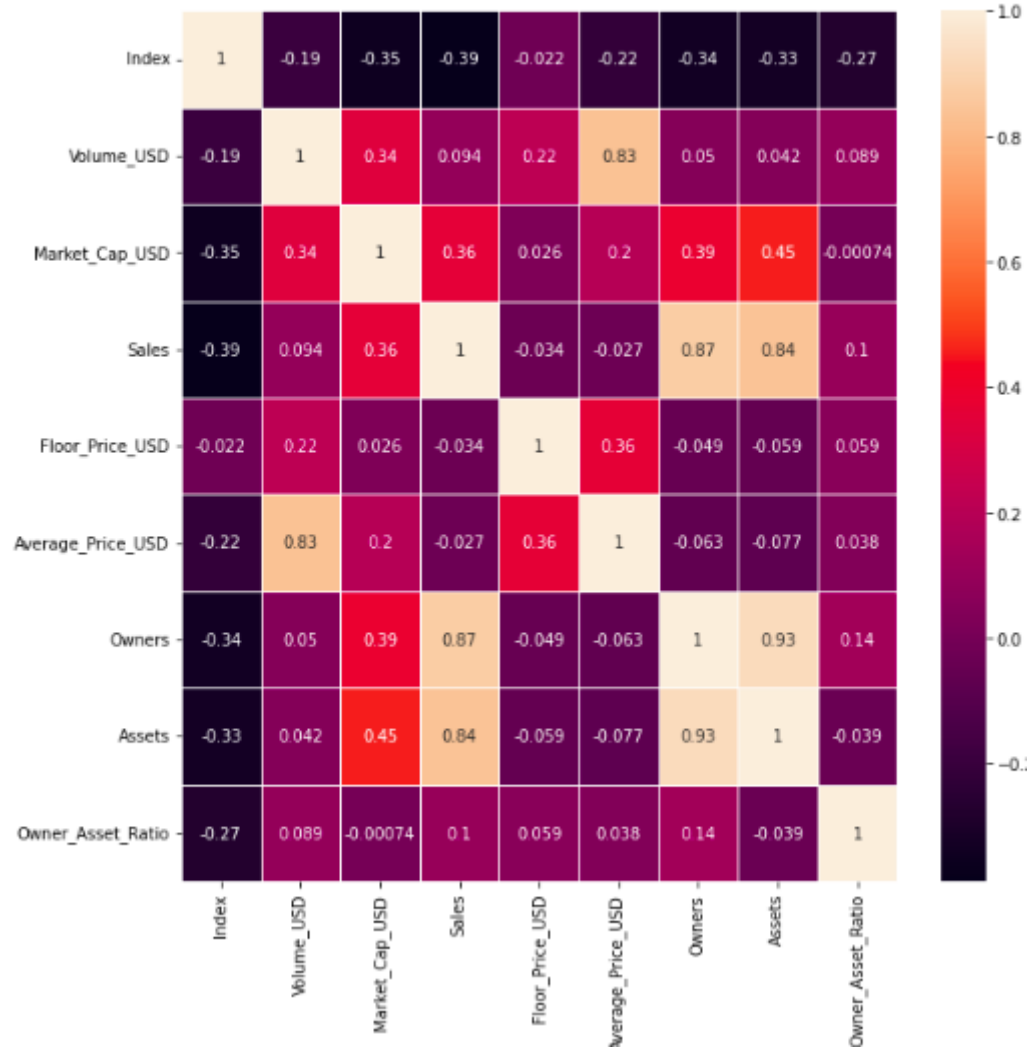


Рисунок 4 – тепловая карта

По тепловой карте все видно гораздо лучше и гораздо понятней. Признак Index в дальнейшем удалим, он нам не нужен. Признаки Assets и Owners

между собой связаны и сильно коррелируют, и чисто по описанию этих признаков, поэтому убирать один из этих признаков не нужно. Average_Price и Market_Cap_USD также связаны по своему смыслу, капитализация зависит от средней цены по определению. Sales вместе с Owners и Assets также связаны по определению, и убирать какой-то из этих признаков не нужно. Соответственно, все остальные данные коррелируют более-менее в адекватных пределах, что говорит о том, что можно провести кластеризацию и каких-то непонятных ошибок не будет, все объясняют данные. Но для того чтобы привести данные в нормальный для кластеризации вид, посмотрим на их графики нормального распределения и выбросы, если таковые имеются. Из предварительного анализа, было установлено, что некоторые признаки, а именно, Market_Cap_USD, Volume_USD, Sales, Average_Price_USD, Floor_Price_USD, Owners, Assets, необходимо прологарифмировать, потому что их графики не подчиняются закону нормального распределения. Соответственно, графики по выбросам и графики нормального распределения ниже.

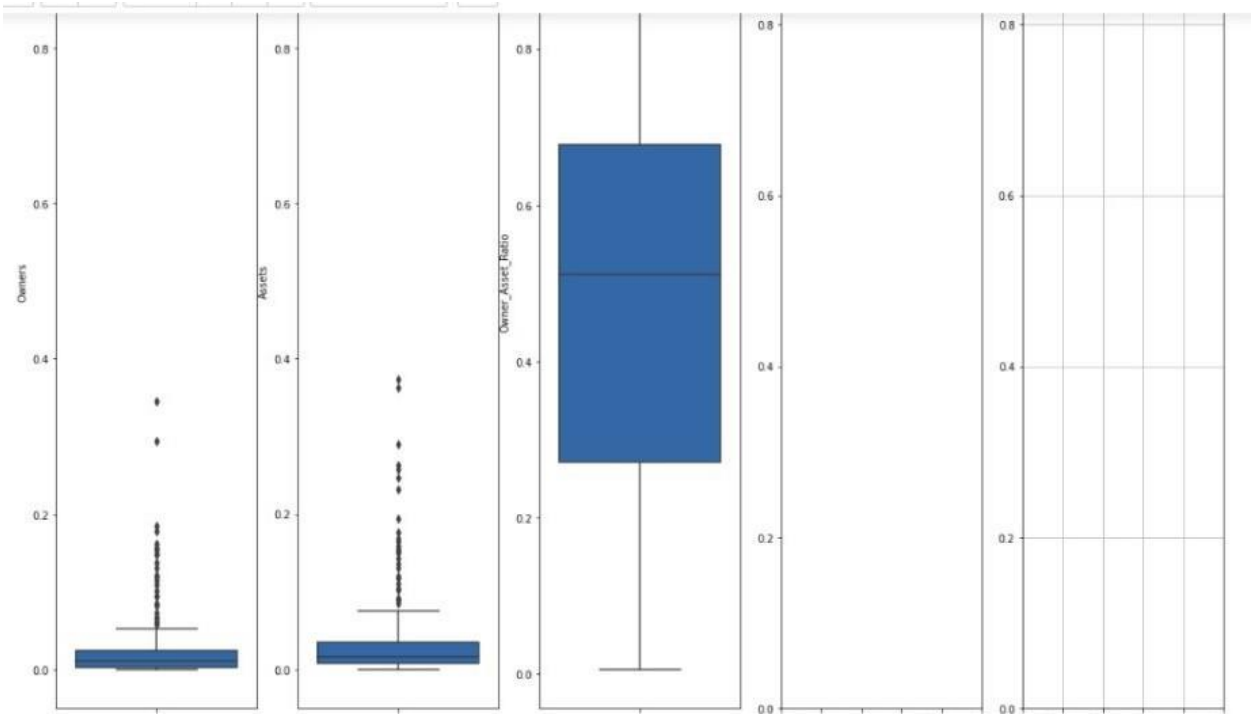


Рисунок 5 – График «ящик с усами» для просмотра выбросов 1

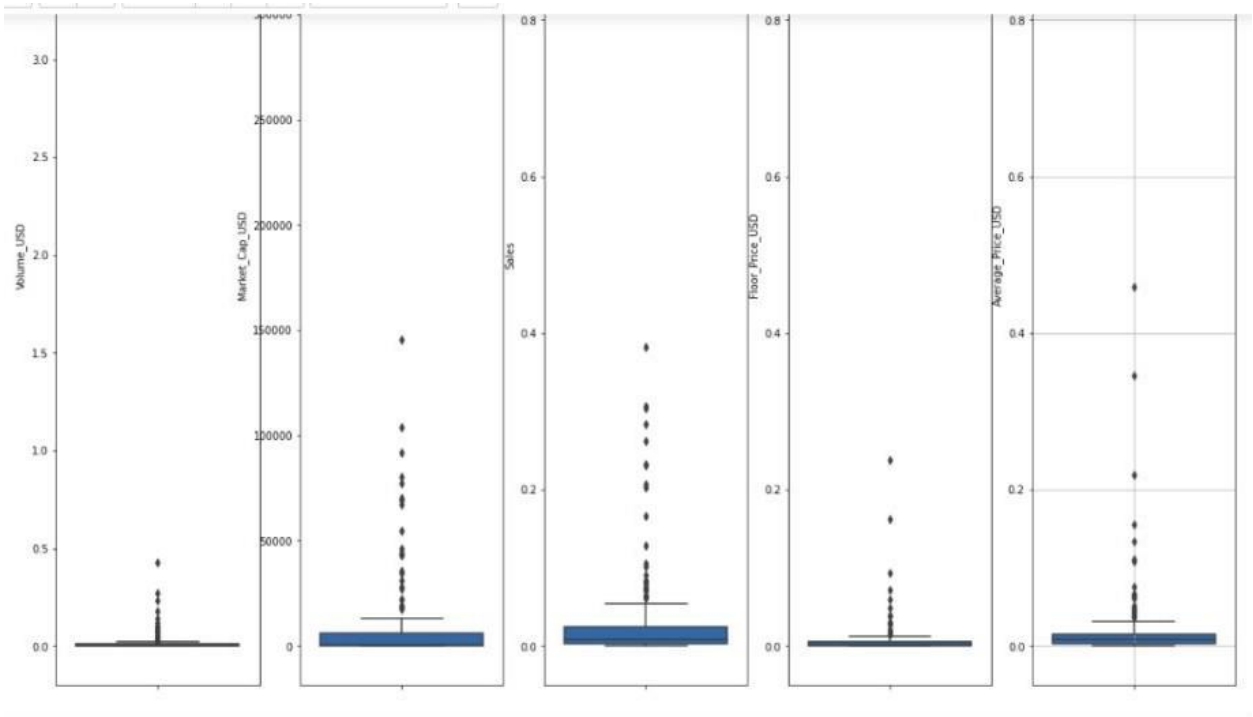
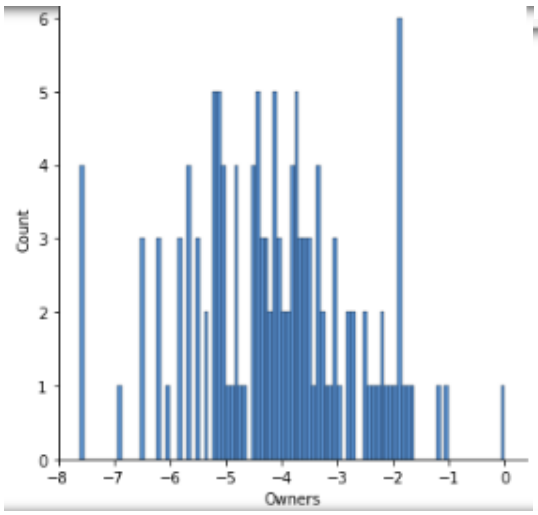
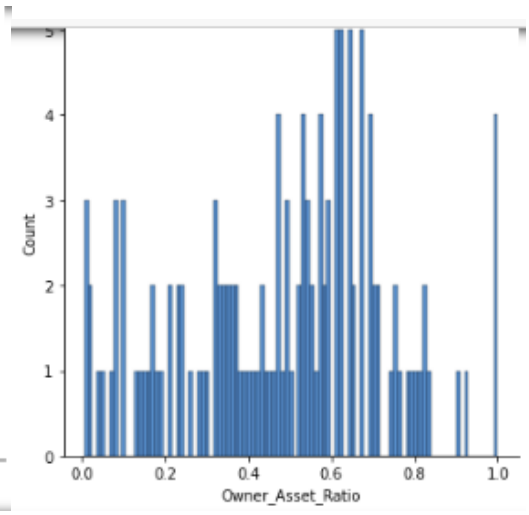


Рисунок 6 – График «ящик с усами» для просмотра выбросов 2

Такой вид графиков объясняется тем, что данных очень мало, очень много значений находятся у какой-то точки, и лишь небольшая, малая из часть сильно отличается от средних значений признаков. В мире блокчейна, криптовалют и NFT это можно считать нормой, так как эти рынки имеют большую волатильность, и учитывая особенности NFT (токены не взаимозаменяемы) это условно можно считать за норму.





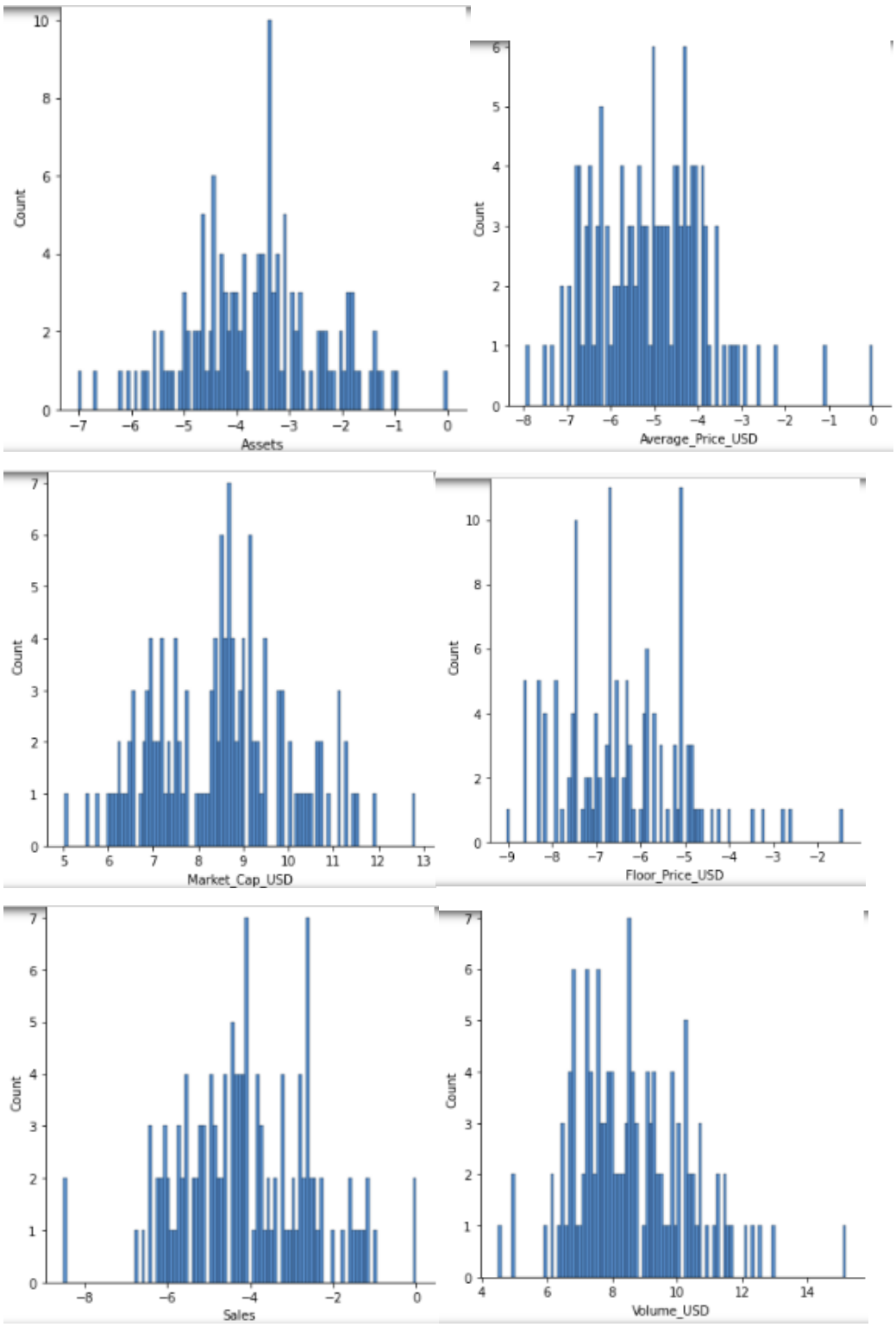


Рисунок 7 – распределение признаков

Как можно предположить, логарифмирование выбранных мной признаков помогло, и теперь признаки подчиняются закону нормального распределения.

Матрица Евклидовых расстояний.

Для того чтобы построить матрицу евклидовых расстояний, уберем все нечисловые признаки Name и колонку Index, в нашем датасете, и строим эту самую матрицу.

```
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
```

Рисунок 8 – Евклидово расстояние

И немного теории, что такое это Евклидово расстояние? Это кратчайшая прямая между двумя точками в евклидовом пространстве, а в нашем случае, между признаками.

Вообще говоря, евклидово расстояние широко используется в разработке 3D-миров, а также алгоритмов машинного обучения, которые включают в себя метрики расстояния, такие как К-ближайшие соседи. Как правило, евклидово расстояние будет представлять, насколько похожи две точки данных, предполагая, что некоторая кластеризация на основе других данных уже была выполнена.

Математическая формула для расчета расстояния между двумя точками в 2D пространстве:

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

Общая формула может быть упрощена до такого вида:

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

Ещё, можно сказать, что эта формула очень сильно похожа с теоремой Пифагора:

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

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

И ведь на самом деле, между ними существует связь, евклидово расстояние рассчитывается с помощью теоремы Пифагора, учитывая декартовы координаты двух точек.

Описание выбранного алгоритма

В качестве алгоритма кластеризации, заказчик заранее поставил условие, что должен быть применен метод BIRCH.

Полное название BIRCH – сбалансированное итеративное сокращение и кластеризация с использованием иерархий. Здесь используется иерархический метод для кластеризации и уменьшения данных вверх, и методу необходимо осуществить всего лишь один проход для выполнения кластеризации.

Алгоритм BIRCH использует древовидную структуру, которая помогает нам быстро кластеризоваться. Эта числовая структура похожа на сбалансированное дерево B +. Его обычно называют деревом функций кластеризации (CF Tree). Каждый узел этого дерева состоит из нескольких функций кластеризации (CF). На рисунке ниже мы можем увидеть, как выглядит дерево функций кластеризации: каждый узел, включая конечные узлы, имеет несколько CF, а CF внутренних узлов имеют указатели на дочерние узлы, а все конечные узлы связаны двусвязным списком.

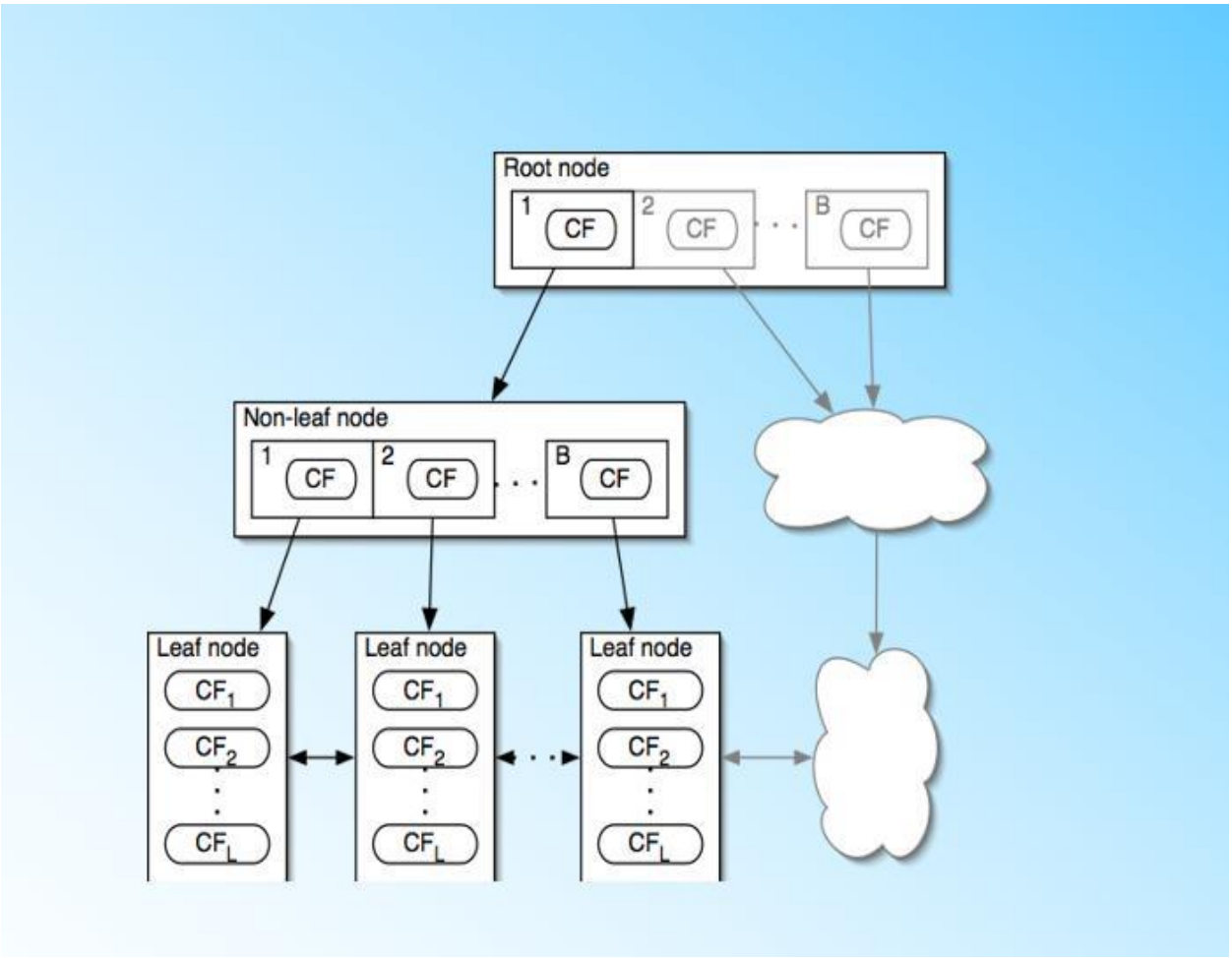


Рисунок 9 – древо функции кластеризации

Если говорить об алгоритме кратко, то Алгоритм BIRCH не требует ввода значения K номера категории, которое отличается от K-средних и мини-пакетных K-средних. Если значение K не введено, количество последних кортежей CF будет окончательным K, в противном случае кортежи CF будут объединены в соответствии с расстоянием согласно входному значению K.

Вообще говоря, алгоритм BIRCH подходит для ситуаций с большим размером выборки, который аналогичен K-средним минипакетам, но BIRCH подходит для ситуаций, когда количество категорий относительно велико, в то время как K-средние минипакетные обычно используются. для средних или относительно больших категорий. Когда меньше. Помимо кластеризации, BIRCH также может выполнять дополнительное обнаружение выбросов и предварительную обработку данных в соответствии со спецификациями категорий. Однако, если размер признака данных очень велик, например,

больше 20, BIRCH не подходит. В настоящее время K-средние Mini Batch работают лучше.

Для настройки параметров BIRCH более сложен, чем K-Means и Mini Batch K-Means, потому что ему необходимо настроить несколько ключевых параметров дерева CF, которые имеют большое влияние на окончательную форму дерева CF.

В заключение резюмируя достоинства и недостатки алгоритма БЕРЕЗА: Основными преимуществами алгоритма БЕРЕЗА являются:

1) Сохранение в память, все образцы находятся на диске, в дереве CF хранятся только узлы CF и соответствующие указатели.

2) Скорость кластеризации высока, и дерево CF может быть создано путем сканирования обучающего набора только один раз. Добавление, удаление и изменение дерева CF выполняются быстро.

3) Может определять точки шума, а также может предварительно обрабатывать набор данных для предварительной классификации

Основными недостатками алгоритма БЕРЕЗА являются:

1) Поскольку дерево CF имеет ограничение на количество CF для каждого узла, результат кластеризации может отличаться от реального распределения категорий.

2) Эффект кластеризации высокоразмерных пространственных данных не очень хорош. В настоящее время вы можете выбрать Mini Batch K-Means

3) Если кластер распределения набора данных не похож на гиперсферу или не является выпуклым, эффект кластеризации плохой.

Результаты выполнения алгоритма. Выводы

```
# 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>
```

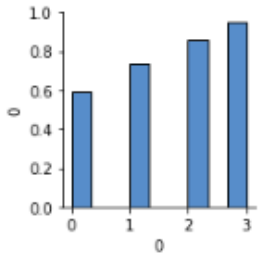


Рисунок 10 – результаты выполнения кластеризации

В результате выполнения алгоритма, мной было принято решение разбить данные на 4 кластера, потому что данный алгоритм кластеризации не подходит для такого малого количества данных, он более хоро и эффективен, когда есть большая выборка данных. При выборе 2 кластеров, получится не пойми что, и при этом это не будет удовлетворять постановке задачи, При выборе 3 кластеров, они распределяются неравномерно, и в одно кластере

почти нет объектов, а 4 кластера, наиболее оптимальный вариант, это опять же можно увидеть на рисунке 10.

В ходе выполнения кластеризации выбранным методом BIRCH, получилось 4 кластера, в первом 25 объектов, во втором, 31, в третьем 36, в четвертом 40.

Наиболее интересными кластерами для компании X считаем первый и второй кластер, первый, так как в этом кластере наиболее дорогие и наиболее популярные NFT, а второй, так как там не самые дорогие NFT, в которые можно вложиться.

Если рассматривать NFT из второго кластера, то это наиболее желательные вещи к покупке, чтобы заработать в средний или короткий срок.

NFT из третьего кластера это NFT, которые очень рискованно покупать для инвестиций, так как высок шанс того, что большинство из них не представляют никакой ценности.

NFT из четвертого кластера к покупке не рекомендуются, их лучше обходить стороной, слишком маленький интерес, капитализация и прочие признаки, при их покупке, лучше смотреть на отдельные экземпляры коллекции, именно они могут представлять хоть какую-то ценность.

Как итог, кластеризация методом BIRCH на данном датасете не является репрезентативной, этот метод подойдет для более большой выборки, чего нельзя сказать про данную мне выборку. Данный метод хорош будет лишь в том случае, если мы точно знаем, сколько кластеров у нас в датасете, здесь мы лишь предположили, что будет 4 кластера. Для получения достоверного и правильного результата кластеризации, в данном случае, необходим другой метод, например K-means, либо же, необходимо собрать больше данных о различных коллекциях NFT, пользуясь не только coinmarketcap.com, но и другими социальными сетями или другими источниками информации.