Chapter -1 Introduction

1.1 Introduction

In recent years, non-fungible tokens (NFTs) have emerged as a new frontier within the digital assets space. An NFT is a non-interchangeable unit of data stored on the blockchain. NFTs can be fundamentally seen as representing unique digital assets on the blockchain. As argued by [1] NFTs could be viewed as representing an unique digital certificate of authenticity of the asset in question. These unique and non-interchangeable digital assets have catalyzed a paradigm shift with new opportunities in market valuation practices. The NFT ecosystem has been one of the most rapidly growing segments within crypto with billions of dollars in annual trading volume for this emerging asset class. The largest portion of NFT trading volume is concentrated in collections where different NFTs with heterogeneous characteristics are released as part of an encompassing collection [1].

Popular collections include Bored Ape Yacht Club (BAYC), Crypto Punks, and Azuki. Despite the large trading volume and interest, the NFT ecosystem is in its early stages and little has been conducted around these assets. NFT holders receive a base utility from the collection while the idiosyncratic utility is captured by the NFT-specific features. NFTs within collections mainly differentiate themselves in the traits and we would argue that these represent the fundamentals in NFTs. As a result, market participants have been mainly using traits to compare and value Non-Fungible Tokens (NFTs) within collections. The assumption that market participants look at NFT traits in their fundamental valuation seems reasonable given that these are the main heterogeneous elements within collection NFTs. NFTs differentiate themselves in the traits and these represent the fundamentals in NFTs. An intriguing aspect of these traits is the element of rarity.

The NFT community has used rarity as a key metric to compare and assess NFTs, often associating it with higher desirability and value. This measure is used to compare the relative value of individual NFTs against each other within a collection. It is often assumed that rare NFTs are desirable and perceived of higher value. This has prompted the development of various methods to quantify rarity, the most prevalent of which is the rarity score. However, the question remains as to how well these metrics capture the true value of an NFT and how the market prices different levels of rarity within traits. There has been existing research on how scarcity of an asset links to its perceived value by the market, and NFTs provide an fascinating laboratory for financial-economic research in this regard. In this paper, we seek to critically analyze the role of rarity in NFT pricing, examining how market participants consider rarity in their valuation decisions.

1.2 Problem Statement

With the development of the NFT market, there was an increase in the number of artists. In the NFT market on the internet, many digital collectibles from many artists are available for cost. But the difficult job is to find the rarity and value of NFT. Also it's difficult to find things which make NFT's rare.

In the digital realm, Non-Fungible Tokens (NFTs) have become a ground-breaking asset class, providing distinct provenance and ownership for digital goods including virtual real estate, art, and collectibles. In the rapidly expanding digital collectibles market, the idea of rarity is crucial in establishing the worth and appeal of NFTs. With this project, we want to analyze NFT rarity in-depth and present the rarest tokens according to their computed rarity scores.

1.3 Objective

Our main objective is to find the Rare NFTs because it is highly important in NFTs because it determines the value and desirability of a digital asset. The rarity of an NFT is determined by factors like uniqueness, scarcity, and demand. The rarer an NFT's traits are, the more valuable it becomes. Rarity plays a crucial role in attracting collectors and investors to NFT projects. This also play a major role in the pricing.

1.4 Chapterwise Summary:

Chapter 1 provides an introduction to the research topic, outlining the significance of rarity in NFTs and identifying key research objectives.

Chapter 2 conducts a thorough literature survey, exploring existing research, methodologies, and insights related to rarity in NFTs.

Chapter 3 presents the methodology employed in the research project, detailing data collection methods, analytical frameworks, and software tools used.

Chapter 4 analyzes the results of the research, showcasing findings, insights, and implications derived from the data analysis process.

Chapter 5 concludes the research paper, summarizing key findings, discussing implications, and suggesting avenues for future research.

Chapter – 2 Literature Survey

2.1 Sections of Literature Survey

The literature survey encompasses several thematic sections, including:

- Overview of NFTs and Rarity: Examining the definition, characteristics, and significance of rarity in the context of NFTs.
- Market Dynamics: Analyzing market trends, pricing mechanisms, and demand-supply dynamics influencing rarity perception.
- Token Standards and Protocols: Exploring NFT standards (e.g., ERC-721, ERC-1155) and blockchain protocols shaping rarity assessment and token interoperability.
- User Behavior and Preferences: Investigating user preferences, collector behaviors, and psychological factors driving rarity perception and valuation.

2.2 Literature Survey

Several NFT marketplaces have emerged in recent years, including Open Sea, Rarible, Super Rare, and Nifty Gateway. Such marketplaces allow users to buy, sell, and trade NFTs using various cryptocurrencies and also offer features such as auctions, fixed-price sales, and limited edition drops. The use of NFTs has revolutionized the way digital assets were managed previously. Before NFTs, the right to ownership was not possible for digital assets. This paper also demonstrates the technologies that will be required to build a proper NFT marketplace [2].

The purpose of this paper is to provide extensive information on the NFT, including its application, method of operation, buying, creating, and selling procedures, as well as its use. The NFT when paired with Metaverse, represents a significant advancement and revolution in the realm of virtual reality and blockchain, giving artists a new avenue to express their unique and valuable work [3].

Nonfungible Tokens as Core Component of a Blockchain-based Event Ticketing Application. This paper discusses the widespread of NFTs built on the Ethereum blockchain in various fields. Also, it shows the comparison between the different NFT marketplaces that are built on the Ethereum blockchain Main network [4].

The widespread application of blockchain-based technologies, and the mechanisms in place for verifying ownership of digital assets and thus, means of securing them remained susceptible to tampering that translated into significant losses. Decades of research and advancements in blockchain led to the development of NonFungible Tokens (NFTs), which are tokens that represent digital assets and have proof of ownership embedded [5].

Blockchain, a potentially disruptive technology, advances many different applications, e.g., cryptocurrencies, supply chains, and the Internet of Things. Under the hood of blockchain, it is required to handle different kinds of digital assets and data. The next-generation blockchain ecosystem is expected to consist of numerous applications, and each application may have a distinct representation of digital assets. However, digital assets cannot be directly recorded on the blockchain, and a tokenization process is required to format these assets [6].

Investigated the sustainability of NFT marketplaces and suggested that the high energy consumption required for NFT minting and trading could be reduced by using renewable energy sources [7].

In studying the pricing behavior of NFTs, found that the price of an NFT is influenced by the observed value of the primary asset and the rarity of the NFT. Inspected the role of NFT marketplaces in supporting creative industries. He found that NFT marketplaces could provide a new revenue stream for artists and creators and could also offer opportunities for smaller artists to gain exposure and sell their work [8]. Studied the factors that influence the adoption of NFT marketplaces and found that factors such as usefulness, ease of use, and trust have a huge impact on the adoption of NFTs [9].

NFTs are transparent, traceable, and secure since they are built on blockchain technology, particularly Ethereum. Unique tokens' innovative property allowed for use cases that had never been proven, such as exclusive ownership of digital assets. Each asset's ownership may be tracked, which improves authenticity. Art collectors and fans were drawn to the concept of having total possession of an original, bought digital asset, such as photographs, gifs, films, music, etc., which spurred a quick expansion in the market [10].

NFTs may be a relatively new technology that leverages the blockchain as its foundation, but it is already sufficiently developed for practical usage. Although its future as a key component of the creative media business is already sealed, it will also have many uses in other industries. The only drawback of NFTs being bad for the environment will vanish as the world moves closer to renewable sources of energy [11].

A blockchain is essentially a distributed digital ledger of transactions [12] that encompasses the whole network of computers. It is dispersed, which means it does not require a central authority to function. Bitcoin was the first cryptocurrency to leverage blockchain technology; it was conceived in 2008 and deployed in 2009 [13].

Since then, this distributed ledger concept has attracted other initiatives from various industries; nonetheless, the financial industry is recognized as the key user. The reason seems to be that identifying the correct current owner of an asset is often not possible [14]. To verify and authenticate ownership, blockchain works in the following way: it is made up of data packages called “blocks”, which are cryptographically interconnected to one another, and by adding each additional block, it creates a chain, which is a complete digital ledger. Distributed Ledger Technology (DLT) is a decentralized database that is administered by various people. Blockchain is a sort of distributed ledger technology [15] in which transactions are stored using an irreversible cryptographic signature known as a hash, and blocks can be authenticated by the network using cryptographic means. This concept ensures the blockchain's integrity all the way to the first block.

As the hash values are unique, fraud can be detected because modifications to a block in the chain changes the hash value immediately. Because of the decentralized structure of blockchain, all transactions can be transparently viewed. The technology, on the other hand, is suitable for a wider range of applications and is being researched in a number of fields, including finance, public and social services, security and privacy, smart contracts, and IoT [13].

Ethereum is a community-run technology software platform that enables hundreds of decentralized apps to be built and deployed. Ethereum is based on blockchain technology. It is a blockchain with a built-in Turing-complete programming language. It has an abstract layer that allows anyone to define their own ownership, transaction formats, and state transition methods. This is accomplished through the use of smart contracts, which are a collection of cryptographic rules that are only performed if specific terms are satisfied [16].

In addition, such a platform serves as the foundation for a virtual currency known as Ether, which is a cryptocurrency asset used in the Ethereum blockchain. Ether is, in some ways, the gasoline for running Ethereum's distributed applications. It is possible to send money to other accounts or to machines that are doing a certain task using this currency. Ether may therefore be used to operate decentralized applications, create smart contracts, generate tokens, and make ordinary peer-to- peer payments. As a result, Ethereum is also known as “programmable currency” [17].

Ethereum consists of EOA and Contract. The EOA is controlled by a private key while Contract accounts are controlled through contract code. An account consists of four things: nonce, ether balance, contract code hash, and storage root [18].

Minting NFT is a process through which digital art becomes a part of the Ethereum Blockchain [19]. NFTs are tokens that are “minted” after they have been created, similar to how metal coins are minted and incorporated into circulation. The digital art is symbolized as an NFT, allowing it to be bought and sold on the market, as well as digitally tracked throughout the whole process [20].

The NFT market observed a sudden uprising in the second half of 2020 with an NFT art selling for USD 69 million. Furthermore, the total volume of NFT sales in 2020 was USD 2.5 billion while the total sales volume of NFTs in the first six months of 2021 surpassed USD 10.7 billion. This indicates a significant change in the growth of NFTs over a short period of time [21]. The 24-hour normal trading volume of the NFT market is USD 4 billion, while the 24-hour normal trading volume of the entire cryptographic money market is USD 341 billion [22].

Chapter – 3 Methodology

3.1 Modules of the Project

1. **Data Collection Module:**
    - This module is responsible for collecting NFT data from various sources such as blockchain explorers, NFT marketplaces, and APIs.
    - Submodules may include data scraping, API integration, and database management.
2. **Preprocessing Module:**
    - The collected data needs preprocessing to clean and prepare it for analysis.
    - Tasks within this module may include data cleaning, normalization, outlier detection, and feature engineering.
3. **Rarity Calculation Module:**
    - This module calculates the rarity scores for NFTs based on predefined criteria and algorithms.
    - It involves analyzing attributes, traits, and metadata of NFTs to determine their rarity levels.
    - Algorithms such as rarity indices or scoring systems may be implemented here.
4. **Analysis and Visualization Module:**
    - After calculating rarity scores, this module conducts analysis and generates visualizations to explore the distribution and trends of rarity in NFTs.
    - Visualization techniques such as histograms, scatter plots, and heatmaps may be utilized to visualize rarity distributions and patterns.
5. **Ranking and Identification Module:**
    - This module ranks NFTs based on their rarity scores and identifies the top rarest NFTs.
    - It involves sorting and filtering NFTs based on rarity metrics to identify the most valuable and rare assets.

These modules help structure the project workflow and facilitate the systematic development and implementation of the NFT rarity analysis system. Each module contributes to different stages of the analysis pipeline, from data collection and preprocessing to analysis, visualization, and reporting.

Architecture diagram

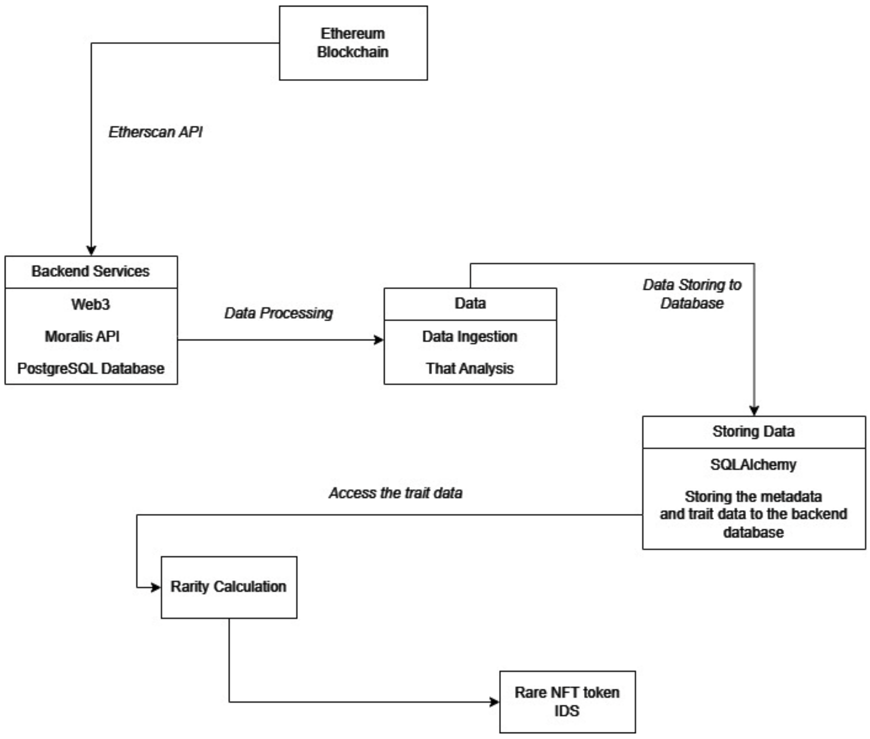


Fig 1 Architecture diagram

The project relies on multiple data sources and backend services to collect, process, and store data related to Non-Fungible Tokens (NFTs). Data is primarily sourced from the Ethereum blockchain, which serves as the origin of NFT smart contracts and token metadata. Metadata files associated with NFTs are stored in the IPFS (InterPlanetary File System) decentralized storage network.

To collect data, the project utilizes various APIs and libraries. The Etherscan API is employed to retrieve contract ABIs and token metadata URIs, crucial for interacting with smart contracts and fetching token details. Web3.py, a Python library, facilitates interaction with the Ethereum blockchain, enabling the retrieval of token metadata and contract ABIs. Additionally, an IPFS client is utilized to connect to the IPFS network and retrieve metadata files associated with NFTs.

Backend services play a crucial role in managing collected data. A PostgreSQL database is employed to store a wide range of information, including contract details, token IDs, token metadata, traits, and rarity scores. The Moralis API provides supplementary functionalities related to NFTs, such as fetching NFT owners and additional data insights.

Data processing involves several key steps, including data ingestion from Ethereum and IPFS sources, trait analysis to extract relevant traits from token metadata, and rarity calculation to determine the rarity percentages and scores for each token based on its traits.

The PostgreSQL database serves as the primary data storage solution, housing comprehensive information about collections, tokens, traits, and rarity scores. The ultimate result of the project is the identification and ranking of the rarest NFTs, providing insights into their rarity levels and associated token IDs.

Tools and platforms:

Programming language: Python and SQL

Application Programming Interface (APIs)

Etherscan API and Dapprad API

- Etherscan Api is used to fetch the collection IDs and metadata of the NFTs from the Ethereum Blockchain.
- The Dapprad Api is used to fetch the prices of the NFTs if required.
- The prices can be fetched from the top marketplace like Opensea,Rarible etc.
- In the project we have used the Opensea marketplace for getting the prices of the NFTs.

Web3

Web3 providers are objects responsible for enabling connectivity with the Ethereum network in colorful ways. Connecting your web operation to an Ethereum knot is necessary for transferring deals, querying data, and interacting with smart contracts on the network. It will provide the HTTP requests and IPFS requests.

HTTP (Hypertext Transfer Protocol) and IPFS (InterPlanetary File System) are both protocols used for transferring and accessing data over the internet, but they differ significantly in their architectures, philosophies, and use cases.

HTTP (Hypertext Transfer Protocol):

1. Architecture:

- HTTP is a protocol used for transmitting hypermedia documents, such as HTML files, over the internet.
- It operates in a client-server model, where a client (e.g., web browser) sends requests to a server, and the server responds with the requested content.

2. Statelessness:

- HTTP is stateless, meaning each request from a client to a server is treated independently, and the server does not maintain any information about previous requests.

3. Content Delivery:

- HTTP is widely used for delivering web pages, images, videos, and other media content over the World Wide Web.
- It relies on centralized servers to store and distribute content, which can lead to issues like single points of failure and data censorship.

4. Addressing:

- HTTP uses URLs (Uniform Resource Locators) to address resources on the web.
- URLs typically include the protocol (e.g., http:// or https://), domain name, and path to the resource.

IPFS (InterPlanetary File System):

1. Decentralization:

- IPFS is a decentralized protocol designed to create a peer-to-peer network for storing and sharing content.
- It aims to address some of the limitations of HTTP by providing a distributed and censorship-resistant infrastructure.

2. Content Addressing:

- In IPFS, content is addressed using cryptographic hashes derived from the content itself.
- Each piece of content has a unique hash, and nodes in the IPFS network can retrieve content by its hash without relying on central servers.

3. Content Distribution:

- IPFS uses a distributed hash table (DHT) to locate and retrieve content from peers in the network.
- When a user requests content, IPFS locates the nearest nodes containing the requested content and retrieves it through a process called content-addressed hyperlinks.

4. Offline Access:

- IPFS enables offline access to content by allowing nodes to cache and store content locally.
- This feature makes IPFS suitable for scenarios where internet connectivity is limited or intermittent.

5. Use Cases:

- IPFS is used for a variety of applications, including decentralized file storage, content distribution, version control systems, and building decentralized applications (dApps).

In summary, while HTTP remains the dominant protocol for accessing content on the web, IPFS offers a decentralized and distributed alternative that promises greater resilience, censorship resistance, and offline accessibility. The two protocols can complement each other, with IPFS providing a decentralized storage and distribution layer on top of HTTP-based web applications.

In the context of Non-Fungible Tokens (NFTs), a collection ID refers to a unique identifier associated with a specific collection of NFTs. Each NFT collection typically represents a set of digital assets that share certain characteristics, such as theme, artwork, or creator.

1. Unique Identifier: A collection ID is a unique alphanumeric code or string that distinguishes one NFT collection from another. It serves as a key attribute used to identify and categorize the NFTs belonging to the collection.

2. Metadata and Attributes: Each NFT collection may have associated metadata and attributes that provide additional information about the collection, such as its name, description, creator, total supply, royalties, and other relevant details. The collection ID serves as a reference point for accessing and managing this metadata.

3. Smart Contracts: In blockchain-based NFT ecosystems, such as Ethereum or Binance Smart Chain, NFT collections are often represented by smart contracts deployed on the respective blockchain networks. The collection ID may correspond to the contract address of the smart contract governing the NFT collection.

4. Marketplaces and Platforms: NFT marketplaces and platforms use collection IDs to organize and display NFT collections to users. Users can browse, search, and interact with specific collections based on their unique collection IDs.

5. Creator and Ownership: The creator or owner of an NFT collection has control over its collection ID and associated smart contract. They can define the characteristics, properties, and rules governing the NFTs within the collection, including how they are minted, traded, and transferred.

6. Interoperability: Collection IDs facilitate interoperability and standardization within the NFT ecosystem. They enable developers, users, and platforms to integrate and interact with NFT collections in a consistent and predictable manner across different applications and protocols.

The collection ID plays a crucial role in organizing, identifying, and managing NFT collections within blockchain ecosystems. It serves as a fundamental component for creators, users, and developers to participate in the vibrant and evolving world of digital ownership and decentralized assets.

Metadata of the collection ID

For each unique collection ID, metadata serves as a cornerstone, offering comprehensive insights into the associated smart contract's intricacies. This data, accessible through the Etherscan API via designated URLs, encapsulates a wealth of information crucial for understanding the nature and functionality of the contract. Among the key details provided in the JSON responses are the contract name, compiler version, optimization settings, and the EVM version utilized in the contract's deployment. These pieces of information collectively paint a vivid picture of the contract's specifications and operational parameters.

By harnessing the capabilities of the Etherscan API, users can efficiently retrieve and leverage contract metadata to facilitate various operational and analytical tasks. Whether conducting due diligence before engaging with a new collection or analyzing existing contracts for optimization opportunities, access to contract metadata empowers users to make informed decisions and navigate the Ethereum ecosystem with confidence.

Furthermore, the seamless integration of contract metadata retrieval into existing systems enhances workflow efficiency and data management capabilities. By automating the retrieval process and integrating metadata into databases or analytical frameworks, users can streamline operations and extract actionable insights more effectively. Ultimately, the availability of contract metadata via the Etherscan API represents a valuable resource for developers, analysts, and enthusiasts alike, fostering greater transparency and understanding within the Ethereum community.

Retrieving the metadata

The Ethereum blockchain facilitates the retrieval of metadata associated with a specific NFT token ID. The `tokenURI` function serves as a crucial mechanism in this process, as it is responsible for retrieving the metadata of an NFT linked to a given tokenId. However, it's important to note that various collections may employ different functions for handling metadata retrieval.

Therefore, before initiating a request for metadata, it is advisable to check and ensure compatibility with the specific function employed by the targeted collection. Once retrieved, the metadata typically consists of dictionary-type data, providing a structured and comprehensive set of information related to the respective NFT, contributing to a more detailed understanding of its attributes and characteristics.

Example of the metadata of a NFT token ID “343” with contract name “MutantApeYachtClub”

```
{'image': 'ipfs://QmQh5aqUEqcbbMp5Ba1FDxazzS5q8LnfhsXTEDuEpQhmFV',
 'attributes': [{'trait_type': 'Background', 'value': 'M1 Aquamarine'},
  {'trait_type': 'Fur', 'value': 'M1 Dark Brown'},
  {'trait_type': 'Eyes', 'value': 'M1 Bored'},
  {'trait_type': 'Clothes', 'value': 'M1 Leather Jacket'},
  {'trait_type': 'Hat', 'value': 'M1 Commie Hat'},
  {'trait_type': 'Mouth', 'value': 'M1 Phoneme Vuh'}]}
```

- The above metadata is used to find the rarity.

Database setup

1.Relational Databases: Relational databases like PostgreSQL, MySQL, and SQLite are commonly used to store structured data related to NFT collections, transactions, user information, and metadata. They provide a robust framework for organizing data into tables with defined relationships, constraints, and indexes.

2.NoSQL Databases: NoSQL databases such as MongoDB, Redis, and Cassandra are preferred for their flexibility and scalability, especially in scenarios where data structures may vary or evolve rapidly. NoSQL databases are suitable for storing unstructured or semi-structured data, such as NFT metadata, which may have varying attributes across different collections.

3.Blockchain Data Storage: Blockchain itself acts as a decentralized database for storing immutable transaction records and smart contract data related to NFTs. Each NFT transaction, including minting, transferring, and trading, is recorded on the blockchain, ensuring transparency, security, and tamper resistance.

4.IPFS (InterPlanetary File System): IPFS is a distributed storage protocol that enables decentralized file storage and retrieval. It is often used to store large files associated with NFTs, such as images, videos, audio files, and metadata. IPFS addresses are stored on the blockchain, providing a reliable and decentralized method for accessing NFT content.

5.Caching Mechanisms: Caching mechanisms, including in-memory databases like Redis, are employed to improve performance and reduce latency in fetching frequently accessed data, such as NFT metadata and collection information. Caching helps optimize response times and reduce the load on primary data storage systems.

6. Full-Text Search Engines: Full-text search engines like Elasticsearch and Solr are used to enable advanced search and retrieval capabilities for NFT metadata and attributes. These search engines support complex queries, faceted search, and relevance ranking, enhancing the discoverability and usability of NFT collections on marketplaces and platforms.

7. Indexing and Sharding: Indexing techniques and sharding strategies are applied to optimize database performance and scalability, especially in environments with large volumes of NFT data and concurrent user interactions. Indexes help accelerate query execution, while sharding distributes data across multiple nodes or partitions to distribute the workload and enhance throughput.

In this project we have used the PostgreSQL 16 to store and retrieve the data efficiently. There are so many ways to store and retrieve the data from the database. One of the ways is using the SQLAlchemy.

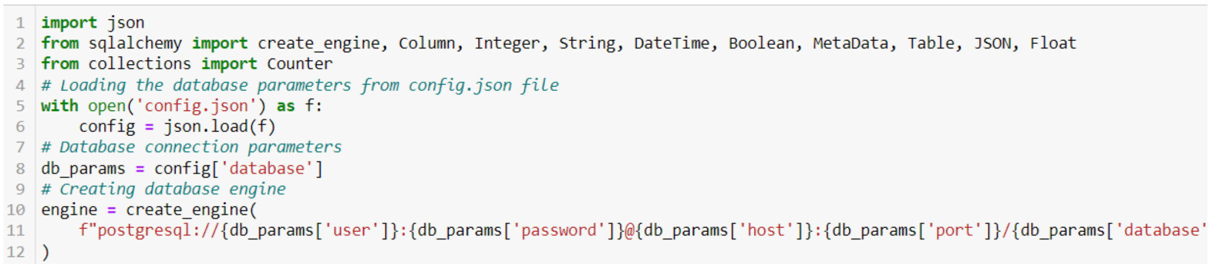
Benefits of using SQLAlchemy

SQLAlchemy offers a seamless experience for developers working across various databases, streamlining queries and providing a user-friendly ORM (Object Relational Mapping) for database interaction within Python. Its versatility simplifies the process of working with databases, enhancing efficiency and reducing complexities in database management tasks. With SQLAlchemy, developers can leverage a Pythonic approach to interact with relational databases, facilitating smoother integration and manipulation of database data. This database toolkit empowers developers with robust features and functionalities, enabling them to handle database operations with greater ease and effectiveness.

Tables in the Database

1) **Collection table** contains the collection ID and the status of the ID.
2) **Asset table** contains the metadata and the token ID of the collection ID.
3) **Trait table** contains the trait data of the NFTs.
4) **Rarity table** contains the rarity scores.

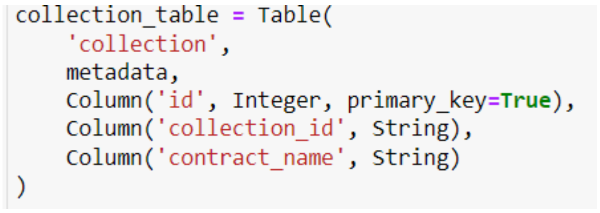
Code for establishing database connection and creating tables



```python
import json
from sqlalchemy import create_engine, Column, Integer, String, DateTime, Boolean, MetaData, Table, JSON, Float
from collections import Counter
# Loading the database parameters from config.json file
with open('config.json') as f:
    config = json.load(f)
# Database connection parameters
db_params = config['database']
# Creating database engine
engine = create_engine(
    f"postgresql://{db_params['user']}:{db_params['password']}@{db_params['host']}:{db_params['port']}/{db_params['database'
)
```

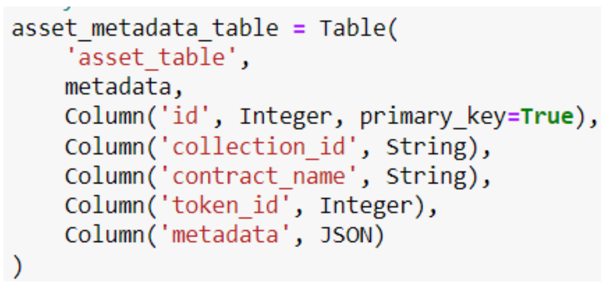
Creating tables

Collection table



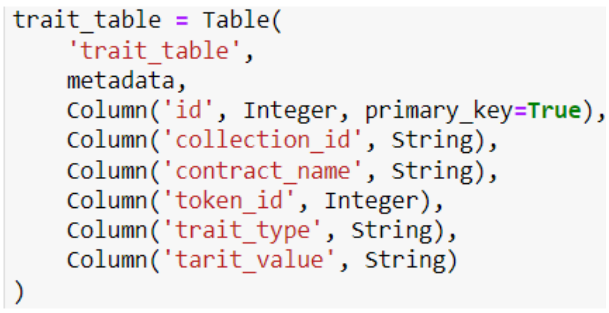
```python
collection_table = Table(
    'collection',
    metadata,
    Column('id', Integer, primary_key=True),
    Column('collection_id', String),
    Column('contract_name', String)
)
```

Asset table



```python
asset_metadata_table = Table(
    'asset_table',
    metadata,
    Column('id', Integer, primary_key=True),
    Column('collection_id', String),
    Column('contract_name', String),
    Column('token_id', Integer),
    Column('metadata', JSON)
)
```

Trait table



```python
trait_table = Table(
    'trait_table',
    metadata,
    Column('id', Integer, primary_key=True),
    Column('collection_id', String),
    Column('contract_name', String),
    Column('token_id', Integer),
    Column('trait_type', String),
    Column('tarit_value', String)
)
```

Using the above code snapshots we can establish the connection to the database using the parameters present in the config.json file we can create a engine using SQL Alchemy.

For creating the tables in the database we need to pass column name with their datatypes as the parameters to create a table. In those tables we are going to store the metadata of the NFTs using insert commands.

Cons in the database

- While storing the data to the database there may be a chance of the duplicate values or data.
- Before adding the data to the database we need to check that if the data is already present in the database ignore the new data else add the data to the database.

Factors that Determine Rarity

Rarity in NFTs is determined by various factors, including:

Uniqueness: The more unique and distinct the traits or attributes of an NFT are, the rarer it tends to be.

Scarcity: If there are fewer copies or instances of a particular NFT available, its rarity value increases.

Demand: The level of interest and demand for a specific NFT also affects its rarity. Tokens that are highly sought after by collectors tend to be rarer.

Combination of Traits: An uncommon combination of traits within an NFT collection can contribute to its rarity score.

Visual Impact: Certain eye-catching or visually striking traits may have lower visual impact, making them rarer within the context of the collection.

Legendary 1/1 Traits: Traits that exist in only one copy or instance (1/1) within an NFT project are extremely rare and often highly valued.

Rarity Calculation Methods

Determining the rarity of an NFT involves using specific methods. Here are some commonly used rarity calculation methods:

1. **Trait-Based Rarity:** This method calculates rarity based on the traits or attributes possessed by an NFT. The more unique and uncommon the combination of traits, the higher the rarity score.
2. **Scarcity-Based Rarity:** Scarcity is another factor in calculating rarity. The fewer NFTs with certain traits or attributes, the rarer they are considered to be.
3. **Demand-Based Rarity:** Rarity can also be influenced by demand. If there is high demand for a particular trait or attribute, it can contribute to the overall rarity score of an NFT.
4. **Combination-Based Rarity:** Some methods take into account both trait-based and scarcity-based factors to calculate rarity. This approach considers how different traits combine and interact with each other within a specific collection.

5. **Machine Learning-Based Rarity:** Advanced algorithms and machine learning models can be used to analyze large amounts of data and determine the rarity of NFTs based on various factors like historical sales data, market trends, and user preferences.

Rarity Calculation

In this project, the utilization of the trait rarity formula serves as a pivotal mechanism for evaluating the unique attributes and desirability of NFTs. By tapping into the metadata of each NFT, which encapsulates its distinct characteristics, the project accesses crucial information required for rarity calculations. Storing this metadata within the asset_table via SQL provides a structured and accessible repository for subsequent analysis.

The process involves a meticulous selection of distinct trait types and values across the NFT collection, reflecting the diverse attributes present within the artworks or assets. This comprehensive approach enables a granular assessment of rarity, allowing collectors and enthusiasts to discern the distinctive traits that contribute to the overall value and appeal of NFTs.

Moreover, the project offers flexibility in managing the rarity scores by providing the option to store them in a dedicated table. This feature enhances data organization and accessibility, facilitating deeper insights into the rarity dynamics within the collection.

By computing the uniqueness of each trait, the project unveils insights into the relative scarcity and significance of specific attributes. This information empowers stakeholders to make informed decisions regarding NFT acquisition, investment, and valuation, thereby contributing to the broader discourse surrounding digital asset ownership and appreciation.

Formula for trait rarity

$$Rarity\ \% = \frac{Number\ of\ items\ with\ trait}{Number\ of\ items\ in\ collection}$$

The above formula gives us the rarity percentage. It gives us the rarity of each trait type.

NFT rarity

In a NFT metadata there will be set of trait types and values in which the rarity is the sum of the rarity percentage of the each type. The rarity is inversely propostional to the uniqueness of the trait. The lesser the rarity uniqueness the higher the rare of the NFT.

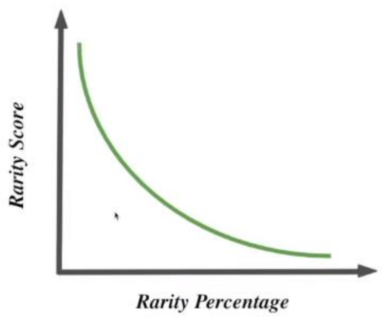


Fig 2 Rarity diagram

Rare Traits

Rare traits are an important aspect of NFT rarity strategy. These are the unique and uncommon attributes that set an NFT apart from others in the collection. Rarity is determined by factors such as scarcity, demand, and the number of similar tokens with the same traits.When calculating NFT rarity, these rare traits hold significant value and can greatly impact an NFT's overall ranking and price. The more rare traits an NFT possesses, the higher its rarity score will be.

This makes it more sought-after and valuable within the collection. By identifying and prioritizing rare traits, collectors can develop a successful strategy for maximizing their NFT's value.In the world of NFTs, rarity is not solely defined by scarcity alone. While scarcity is a fundamental component, the demand for particular traits also plays a crucial role. Certain attributes may hold more appeal to collectors, driving up the desirability and value of NFTs possessing those traits.

Moreover, the context within a collection matters greatly. A trait that might be rare in one collection could be common in another, depending on the theme, style, or narrative of the artworks or assets. Understanding the context and dynamics of the specific NFT collection is therefore paramount in assessing the rarity of its individual tokens.The rarity strategy for NFTs involves a nuanced approach that considers both quantitative and qualitative factors. Collectors and investors meticulously analyze the traits, evaluating not only their scarcity but also their aesthetic appeal, historical significance, and cultural relevance.

As the NFT market continues to evolve, so too do the strategies for assessing and maximizing rarity. Innovations in technology and methodologies for analyzing blockchain data provide collectors with increasingly sophisticated tools to identify and leverage rare traits effectively. Ultimately, in the competitive landscape of NFTs, the pursuit of rarity is a multifaceted endeavor, blending artistry, economics, and technology. By recognizing and valuing rare traits, collectors can navigate this complex terrain and position themselves to secure assets of enduring value and significance.

3.2 Software Requirements

The software requirements encompass a comprehensive set of tools, frameworks, libraries, and runtime environments essential for the development, deployment, and management of applications. Included in these requirements are development tools like Visual Studio Code and PyCharm, which serve as Integrated Development Environments (IDEs) facilitating coding tasks. Additionally, command-line interface (CLI) tools are indispensable for efficient package management and project setup processes.

For programming languages, the environment must support Python and SQL, pivotal for backend logic and database interactions, respectively. Blockchain development tools such as Web3.js or ethers.js are crucial for interfacing with Ethereum blockchain nodes and executing smart contracts seamlessly. In terms of Database Management Systems (DBMS), options like PostgreSQL, MySQL, or SQLite are indispensable for relational database management and data storage functionalities. Moreover, the inclusion of Redis for caching and session management enhances performance and scalability aspects within the application architecture. These software components collectively form the foundation for a robust and scalable development environment. Here is a general list of software requirements:

1. Development Environment:

- Integrated Development Environment (IDE) such as Visual Studio Code, PyCharm,
- Command-line interface (CLI) tools for package management and project setup.

2. Programming Languages:

- Python
- SQL

3. Blockchain Development Tools:

- Web3.js or ethers.js: JavaScript libraries for interacting with Ethereum blockchain nodes and smart contracts.

4. Database Management System (DBMS):

- PostgreSQL, MySQL, or SQLite: for relational database management and data storage.
- Redis: for caching and session management.

Chapter 4 - Result and Analysis

4.1 Implementation

- Use Python with libraries such as Web3, SQLAlchemy, and possibly others for data retrieval, analysis, and visualization.

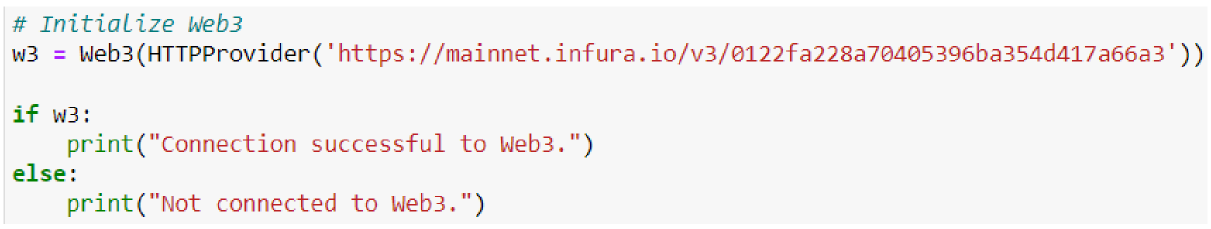
- Fetch metadata and attributes of NFT tokens using Web3 and Etherscan APIs.

- Store the data in a PostgreSQL database using SQLAlchemy.

- Perform rarity calculation and analysis based on the collected data.

- Print the top 10 NFTs with their rarity scores and their ranks.

Establishing the connection to Web3



```python
# Initialize Web3
w3 = Web3(HTTPProvider('https://mainnet.infura.io/v3/0122fa228a70405396ba354d417a66a3'))

if w3:
    print("Connection successful to Web3.")
else:
    print("Not connected to Web3.")
```

It initializes the Web3 instance to connect to the Ethereum mainnet using Infura as the provider.

Fetching collection ID data

```python
def fetch_etherscan_metadata(contract_address):
    url = "https://api.etherscan.io/api"
    params = {
        "module": "contract",
        "action": "getsourcecode",
        "address": contract_address,
        "apikey": etherscan_api_key
    }
    response = requests.get(url, params=params)

    if response.status_code == 200:
        data = response.json()
        result = data.get("result")

        if result:
            metadata = {
                "ContractName": result[0]["ContractName"],
                "CompilerVersion": result[0]["CompilerVersion"],
                "OptimizationUsed": result[0]["OptimizationUsed"],
                "Runs": result[0]["Runs"],
                "LicenseType": result[0]["LicenseType"],
                "EVMVersion": result[0]["EVMVersion"]
            }
            print(json.dumps(metadata, indent=4))
            return metadata
```

Then, it defines a function **fetch_etherscan_metadata** to retrieve metadata about a given Ethereum smart contract from Etherscan's API. The metadata includes details such as the contract name, compiler version, optimization settings, license type, and EVM version used.

Establishing database connection

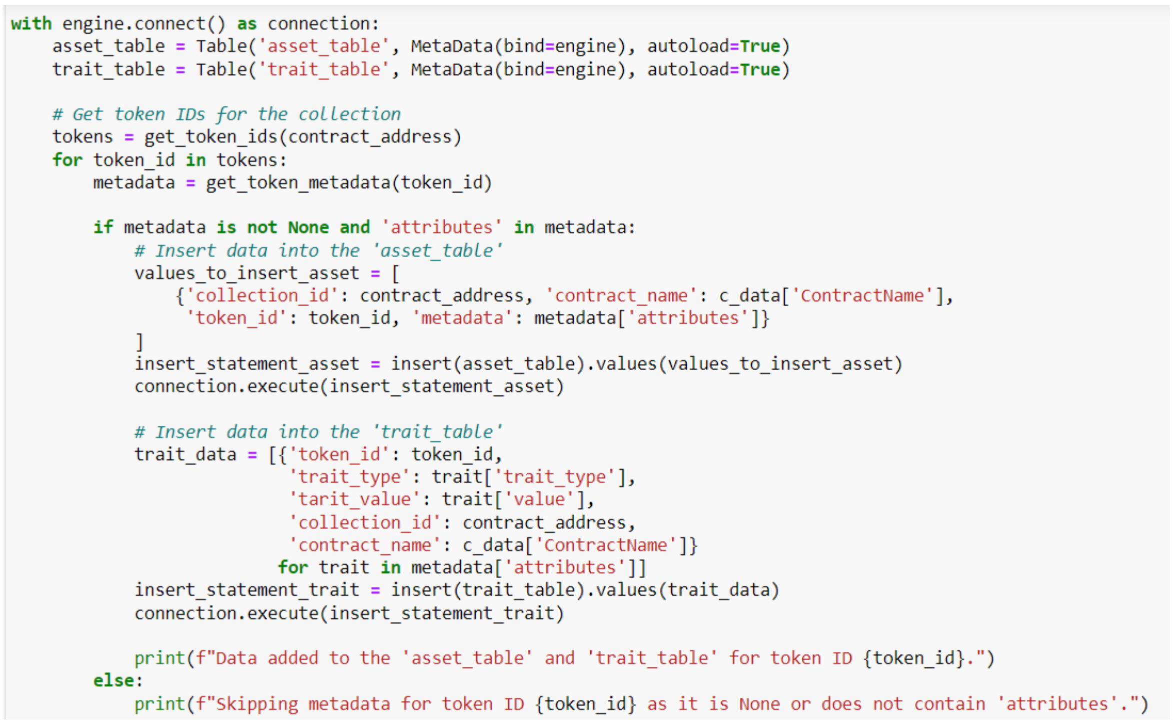
```python
with open('config.json') as f:
    config = json.load(f)
db_params = config['database']
engine = create_engine(
    f"postgresql://{db_params['user']}:{db_params['password']}@{db_params['host']}:{db_params['port']}/{db_params['database'
)
```

Fetch token IDs

```python
from moralis import evm_api

def get_token_ids(collection_id):
    api_key = "EWqWGLCnqtkZlf8x27j2Ji5axSAUtqpZt2LgthG02ZSThkJSl9URAXFcyAybLLqY"
    params = {
        "chain": "eth",
        "format": "decimal",
        "address": collection_id
    }

    token_ids = []

    # Initial request without cursor
    result = evm_api.nft.get_nft_owners(api_key=api_key, params=params)
    token_ids += [item['token_id'] for item in result.get('result', [])]

    # Check if there are more pages with cursors
    while 'cursor' in result and result['cursor'] is not None:
        cursor = result['cursor']
        params['cursor'] = cursor  # Add cursor to the parameters for the next request

        # Make the next request with the cursor
        result = evm_api.nft.get_nft_owners(api_key=api_key, params=params)
        token_ids += [item['token_id'] for item in result.get('result', [])]
        token_ids.sort()
    # Print all retrieved token IDs
    return token_ids
```

The script defines functions for fetching token IDs associated with a given collection and retrieving metadata for each token. It uses Moralis' API to fetch NFT owners and their token IDs in batches, considering pagination if there are more tokens to retrieve.

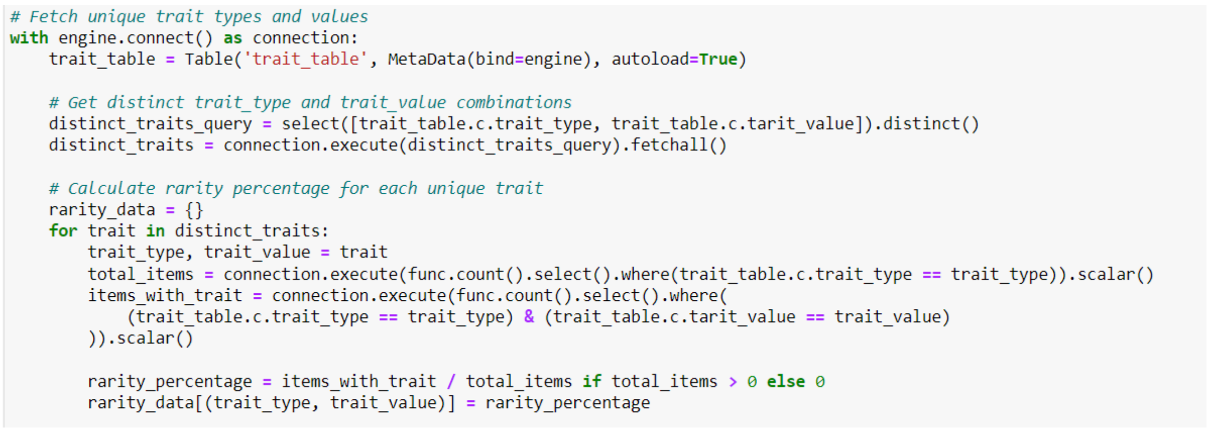
Inserting data to database



```python
with engine.connect() as connection:
    asset_table = Table('asset_table', MetaData(bind=engine), autoload=True)
    trait_table = Table('trait_table', MetaData(bind=engine), autoload=True)

    # Get token IDs for the collection
    tokens = get_token_ids(contract_address)
    for token_id in tokens:
        metadata = get_token_metadata(token_id)

        if metadata is not None and 'attributes' in metadata:
            # Insert data into the 'asset_table'
            values_to_insert_asset = [
                {'collection_id': contract_address, 'contract_name': c_data['ContractName'],
                 'token_id': token_id, 'metadata': metadata['attributes']}
            ]
            insert_statement_asset = insert(asset_table).values(values_to_insert_asset)
            connection.execute(insert_statement_asset)

            # Insert data into the 'trait_table'
            trait_data = [{'token_id': token_id,
                           'trait_type': trait['trait_type'],
                           'tarit_value': trait['value'],
                           'collection_id': contract_address,
                           'contract_name': c_data['ContractName']}
                          for trait in metadata['attributes']]
            insert_statement_trait = insert(trait_table).values(trait_data)
            connection.execute(insert_statement_trait)

            print(f"Data added to the 'asset_table' and 'trait_table' for token ID {token_id}.")
        else:
            print(f"Skipping metadata for token ID {token_id} as it is None or does not contain 'attributes'.")
```

For each token ID retrieved, the script attempts to fetch metadata associated with it. It checks whether the metadata URI points to an IPFS hash or a regular URL and fetches the metadata accordingly. Once obtained, it inserts the metadata into the PostgreSQL database, separating the asset-related information into the **asset_table** and the trait-related information into the **trait_table**.

Fetching the unique traits and calculating rarity



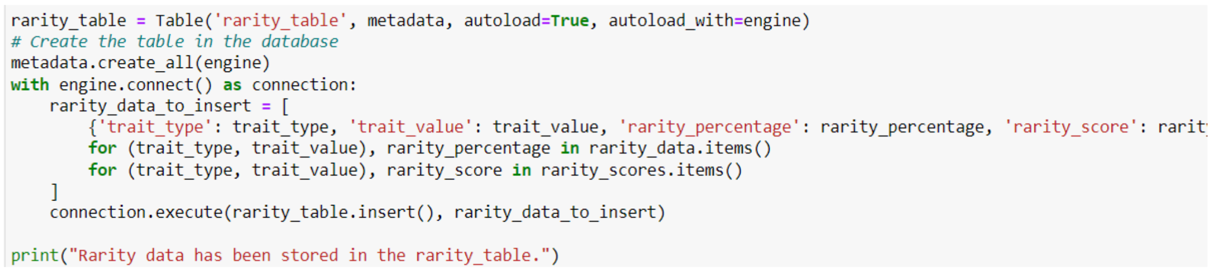
```python
# Fetch unique trait types and values
with engine.connect() as connection:
    trait_table = Table('trait_table', MetaData(bind=engine), autoload=True)

    # Get distinct trait_type and trait_value combinations
    distinct_traits_query = select([trait_table.c.trait_type, trait_table.c.tarit_value]).distinct()
    distinct_traits = connection.execute(distinct_traits_query).fetchall()

    # Calculate rarity percentage for each unique trait
    rarity_data = {}
    for trait in distinct_traits:
        trait_type, trait_value = trait
        total_items = connection.execute(func.count().select().where(trait_table.c.trait_type == trait_type)).scalar()
        items_with_trait = connection.execute(func.count().select().where(
            (trait_table.c.trait_type == trait_type) & (trait_table.c.tarit_value == trait_value)
        )).scalar()

        rarity_percentage = items_with_trait / total_items if total_items > 0 else 0
        rarity_data[(trait_type, trait_value)] = rarity_percentage
```

This portion of the Python script is responsible for calculating the rarity scores of unique traits associated with NFTs stored in a PostgreSQL database. It establishes a connection to the database using SQLAlchemy's `engine.connect()` method and retrieves the `trait_table` from the database's metadata. Once the connection is established and the table is loaded, the script constructs a SQL query to obtain distinct combinations of `trait_type` and `trait_value` from the `trait_table`. This is achieved using the `select().distinct()` method provided by SQLAlchemy.

After executing the distinct traits query, the script retrieves all distinct combinations and proceeds to calculate the rarity percentage for each unique trait. It iterates through the distinct traits, computing the total number of items associated with each trait type and the number of items with a specific trait value. For each trait, the script calculates the rarity percentage by dividing the count of items with the specific trait value by the total count of items with the same trait type. This computation is performed to determine the relative scarcity of each trait value within its trait type.

Once the rarity percentages are computed for all unique trait combinations, the script constructs a rarity score for each trait based on the inverse of its rarity percentage. If a trait's rarity percentage is greater than zero, its rarity score is set to one divided by the rarity percentage. Otherwise, if the rarity percentage is zero, indicating no occurrences of the trait, the rarity score is set to infinity. The resulting rarity scores are stored in a dictionary named `rarity_scores`, where each key-value pair represents a unique trait and its corresponding rarity score. These scores provide valuable insights into the rarity and desirability of specific traits within the NFT collection, aiding collectors and investors in assessing the value and uniqueness of individual tokens.

Storing the rarity data to the rarity table



```
rarity_table = Table('rarity_table', metadata, autoload=True, autoload_with=engine)
# Create the table in the database
metadata.create_all(engine)
with engine.connect() as connection:
    rarity_data_to_insert = [
        {'trait_type': trait_type, 'trait_value': trait_value, 'rarity_percentage': rarity_percentage, 'rarity_score': rarit
        for (trait_type, trait_value), rarity_percentage in rarity_data.items()
        for (trait_type, trait_value), rarity_score in rarity_scores.items()
    ]
    connection.execute(rarity_table.insert(), rarity_data_to_insert)

print("Rarity data has been stored in the rarity_table.")
```

This segment of the Python script pertains to the creation and population of a rarity_table within the PostgreSQL database. Initially, a SQLAlchemy. Table object named rarity_table is instantiated, utilizing the provided metadata and engine objects for table creation. The autoload=True parameter signifies that the table structure will be automatically generated based on the existing schema in the connected database, thereby aligning the table structure with any predefined schema in the database. Subsequently, the metadata.create_all(engine) statement is invoked to create the rarity_table within the database if it does not already exist. This operation ensures that the necessary table structure is established, facilitating the subsequent insertion of data.

Following the table creation, the script establishes a connection to the database using engine.connect(). Within this connection context, the script prepares the data to be inserted into the rarity_table. It constructs a list of dictionaries named rarity_data_to_insert, where each dictionary encapsulates the attributes trait_type, trait_value, rarity_percentage, and rarity_score. The data population process iterates through the rarity_data and rarity_scores dictionaries, extracting trait type-value pairs along with their respective rarity percentages and rarity scores. For each combination of trait type-value pairs, the script constructs a new dictionary entry containing these attributes, populating the rarity_data_to_insert list comprehensively.

Once the data preparation is complete, the script executes an insert operation into the rarity_table using the connection.execute() method. It inserts the preformatted data contained within the rarity_data_to_insert list, effectively storing the calculated rarity percentages and rarity scores for each trait type-value pair within the database. Upon successful execution of the insertion operation, a confirmation message is printed, indicating that the rarity data has been successfully stored in the rarity_table. This step finalizes the process of aggregating and persisting the calculated rarity metrics, providing valuable insights into the rarity distribution of traits within the NFT collection stored in the database.

```python
with engine.connect() as connection:
    # Fetch all token IDs
    token_ids = connection.execute(select([trait_table.c.token_id]).distinct()).fetchall()

    # Initialize list to store token IDs with rarity sum
    token_rarity_sums = []

    # Calculate overall rarity sum for each token_id
    for token_id in token_ids:
        token_id = token_id['token_id']

        # Fetch unique traits for the token_id
        token_traits_query = select([trait_table.c.trait_type, trait_table.c.tarit_value]).distinct().where(
            trait_table.c.token_id == token_id
        )
        token_traits = connection.execute(token_traits_query).fetchall()

        # Calculate overall rarity sum for the token_id
        rarity_sum = 0
        for trait in token_traits:
            trait_rarity_query = select([func.sum(rarity_table.c.rarity_percentage)]).where(
                (rarity_table.c.trait_type == trait['trait_type']) & (rarity_table.c.trait_value == trait['tarit_value'])
            )
            trait_rarity_sum = connection.execute(trait_rarity_query).scalar()
            if trait_rarity_sum is not None:
                rarity_sum += trait_rarity_sum
```

The provided code segment is instrumental in the NFT (Non-Fungible Token) analysis project, focusing on determining the rarity of each token ID based on its associated traits and their corresponding rarity percentages. Initially, the code establishes a connection to the database and retrieves all unique token IDs from the `trait_table`, which represent distinct NFTs within the collection. For each token ID obtained, the code iterates through its associated traits to calculate the overall rarity sum. It retrieves the unique traits for the current token ID by executing a SQL query on the `trait_table`.

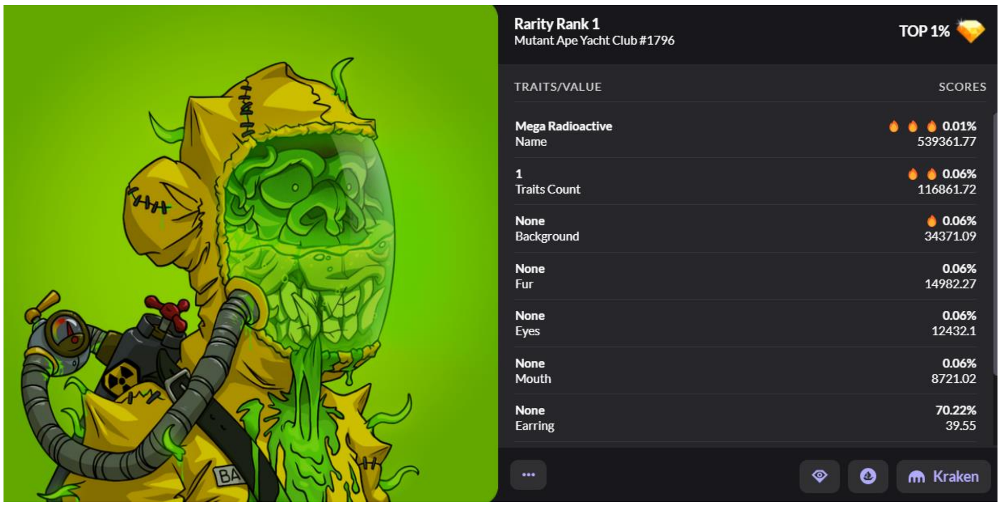
Within the trait loop, the code dynamically constructs SQL queries to obtain the rarity percentages for each trait from the `rarity_table`. It then sums up the rarity percentages obtained from the rarity table to compute the cumulative rarity sum for the token ID. During this process, if a trait's rarity sum is unavailable (i.e., None), the code skips adding it to the rarity sum for the current token ID. The calculated rarity sums are stored in a list along with their corresponding token IDs for further analysis. This data serves as the foundation for identifying and ranking the rarest NFTs within the collection based on their aggregated rarity sums, enabling valuable insights into the uniqueness and desirability of individual tokens.

4.2 Outputs and Results

The result of this project is it gives the token ID with the rarity score. In which any client can know the value of the NFT based on the rarity. We can list the top NFTs based on the client request. For example client can asks for top 10 NFTs with the rarity in the "**Mutant Ape Yacht Club".**

S.No	Token ID	Rank
1	30003	1
2	1796	1
3	30004	1
4	4318	1
5	30000	1
6	30001	1
7	9209	1
8	30002	1
9	4849	1
10	30005	1

The above table provides us the top 10 rare token IDs. The rank for every token ID is 1 because all the traits in those token IDs possess same rarity percentage. We have verified the results with the existing tool.



Rarity Rank 1
Mutant Ape Yacht Club #1796

TOP 1%

TRAITS/VALUE	SCORES
Mega Radioactive Name	**0.01%** 539361.77
1 Traits Count	**0.06%** 116861.72
None Background	**0.06%** 34371.09
None Fur	**0.06%** 14982.27
None Eyes	**0.06%** 12432.1
None Mouth	**0.06%** 8721.02
None Earring	**70.22%** 39.55

Fig 3 Validation diagram

The above image represents the result of the tools like rarity sniper tool which was build on statistical rarity model.

Chapter – 5 Conclusion

Conculsion

An additional instrument that helps investors buy or sell NFTs is the rarity score. Even if our rarity model is statistically accurate, more significant statements regarding specific qualities are occasionally made on the official website of an NFT project. For instance, Cool Cats formally declares that commonplace objects like caps and eaves urinals are worth less than uncommon objects like computer heads or ape-man attire. Furthermore, there are additional metrics, such artistic height or liquidity premium, that you can use to determine the worth of an NFT asset.

The goal of the NFT Rarity Analysis project is to provide insightful information about the dynamics of NFT token rarity within the ecosystem of digital collectibles. Collectors, investors, and enthusiasts can make well-informed decisions about asset purchase, pricing, and portfolio management in the dynamic world of digital ownership by methodically evaluating and visualizing the rarity scores of the best NFTs.

Reference

[1] Sebeom Oh, Samuel Rosen, and Anthony Lee Zhang. Investor experience matters: Evidence from generative art collections on the blockchain. Available at SSRN, 2022.

[2] Ante, Lennart, Non-fungible Token (NFT) Markets on the Ethereum Blockchain: Temporal Development, Cointegration and Interrelations August 13, 2021.

[3] F. Khan, R. Kothari, M. Patel and N. Banoth, "Enhancing Non-Fungible Tokens for the Evolution of Blockchain Technology," 2022 (ICSCDS).

[4] Regner, Ferdinand & Schweizer, André & Urbach, Nils. (2019).

[5] W. Rehman, H. e. Zainab, J. Imran and N. Z. Bawany, "NFTs: Applications and Challenges," 2021 22nd International Arab Conference on Information Technology (ACIT), 2021, pp. 1-7, doi: 10.1109/ACIT53391.2021.9677260.

[6] Wang, Gang & Nixon, Mark. (2021) SoK: Tokenization on Blockchain. 10.1145/3492323.3495577.

[7] Vitalik Buterin- The Energy of NFTs: Environmental Impacts and Mitigation Strategies

[8] Raj Kshetri- Non-Fungible Tokens and Their Potential to Support Creative Industries.

[9] Xiaohui Li and Jiahui Li- Understanding the Adoption of Non-Fungible Tokens and Non-Fungible Token Marketplaces.

[10] Rehman, Wajiha Zainab, Hijab Imran, Jaweria Bawany, Narmeen 2021/12/23NFTs: Applications and Challenges 10.1109/ACIT53391.2021.9677260.

[11] A. Mani, “A Comprehensive Study of NFTs”, International Journal for Research in Applied Science and Engineering Technology, vol. 9, no. 4, pp. 1656-1660, 2021. Available: 10.22214/ijraset.2021.34017.

[12] D. Yaga, P. Mell, N. Roby and K. Scarfone, “Blockchain Technology Overview”, arXiv: Cryptography and Security, 2018. Available: 10.6028/nist.ir.8202.

[13] H. Wang, Z. Zheng, S. Xie, H. Dai and X. Chen, “Blockchain Challenges and Opportunities: A Survey”, International Journal of Web and Grid Services, vol. 14, no. 4, p. 352, 2018. Available: 10.1504/ijwgs.2018.10016848.

[14] M. Nofer, P. Gomber, O. Hinz and D. Schiereck, “Blockchain”, Business & Information Systems Engineering, vol. 59, no. 3, pp. 183-187, 2017. Available: 10.1007/s12599-017-0467-3.

[15] S. Ølnes, J. Ubacht and M. Janssen, “Blockchain in Government: Benefits and Implications of Distributed Ledger Technology for Information Sharing”, Government Information Quarterly, vol. 34, no. 3, pp. 355-364, 2017. Available: 10.1016/j.giq.2017.09.007.

[16] D. Vujičić, D. Jagodic and S. Ranđić, “Blockchain Technology, Bitcoin and Ethereum: A Brief Overview”, 17th International Symposium Infoteh- Jahorina (INFOTEH), 2018. Available: 10.1109/infoteh.2018.8345547.

[17] S. Ferretti and G. D'Angelo, “On the Ethereum Blockchain Structure: A Complex Networks Theory Perspective”, Concurrency and Computation: Practice and Experience, vol. 32, no. 12, 2019. Available: 10.1002/cpe.5493.

[18] V. Buterin, “Ethereum: Platform Review Opportunities and Challenges for Private and Consortium Blockchains.”

[19] L. Ante, “The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum”, SSRN Electronic Journal, 2021. Available: 10.2139/ssrn.3861106.

[20] Q. Wang, R. Li, Q. Wang, and S. Chen, “Non- Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges”, 2021. Available: http://arxiv.org/abs/2105.07447

[21] L. Ante, “Non-fungible token (NFT) Markets on the Ethereum Blockchain: Temporal Development, Cointegration and Interrelations", SSRN Electronic Journal, 2021. Available: 10.2139/ssrn.3904683.

[22] A. Thilagaraj and J. Davis. “Non-Fungible Token (NFT) – The Game Changer in The Digital Art World”, Ciencia Y Sociedad, vol. 51, pp. 190-194, 2021.