NFT Machine Learning (ML) Recommender System

- Non Fungible Tokens (NFTs) are digital assets that have become extremely popular & desirable
- Analysis of 2,800+ NFT wallets suggests 5 distinct buyer types; topic-modeling on 32,000+ NFT descriptions identifies “unique” as the most salient term & significant interest in NFT media mgmt.
- We introduce an ML recommender to navigate the large, complex & expanding NFT marketplace

---

PROJECT INTRODUCTION

Non Fungible Tokens (NFTs) are digital assets stored on the blockchain which have gained in popularity along with the rise of cryptocurrencies. Our project objective is to build a recommender system which aims to address: “What NFTs would a given wallet holder be interested in?”. We find this problem particularly interesting as desirability and demand for digital assets -unlike physical assets- is less well understood.

More holistically, along with the recommender system, we plan to develop and deploy a website front-end where a user can enter their NFT wallet ID, and the website will fetch and analyze their current possessions and return NFT recommendations that the user is likely to be interested in. Additionally, for users that do not have NFTs, the website will allow for the creation of a “paper wallet” to be used for simulating an NFT purchase(s) and in turn serve recommendations based on simulated wallet holdings.

DATASET DISCUSSION

During project initiation we identified OpenSea (OS) marketplace as the dominant platform for buying and selling NFTs; according to some estimates, OS accounts for 97% of the overall NFT market share. Therefore, the OS API is where we begin our data discovery and collection[1].

Three datasets -sourced from the OS API- were curated to build the unsupervised ML recommender system; these include: transactional NFT event data, NFT wallet data and NFT asset data. We begin with a data discussion and conclude with a short overview on the pipeline details including the data collection technique and database technologies used.

Transactional NFT Data	
Asset events represent state changes that occur for assets. This includes putting them on sale, bidding on them, selling them, cancelling sales, transferring them, and more.	
asset_id	Unique identifier for the asset
event_type	State change for the asset (sale, bid, cancelled, transfer, etc.)
created_date	Timestamp of the when the event was recorded
from_account	wallet_id for the transacted party transferring the asset
to_account	wallet_id for the transacted party receiving the asset
payment_symbol	Symbol of currency (crypto) used for payment
quantity	The amount of the item that was sold
total_price	Total price the asset was purchased for

Dataset Size: 1,003,500 rows x 109 columns

NFT Wallet Data	
Compilation of asset data and associated transcational data for a discrete NFT Wallet on the OpenSea platform.	
wallet_id	Unique identifier of the wallet
asset_data	Dictionary containing above NFT asset data
transactional_data	Dictionary containing above transaction data
timestmp	Datetime at which the wallet data was retrieved

Dataset Size: 563,665 rows x 102 columns

NFT Asset Data	
The primary object in the OpenSea API is the asset, which represents a unique digital item whose ownership is managed by the blockchain.	
asset_id	Unique identifier for the asset
image_url	An image for the NFT
name	Unique name of the NFT
asset_description	Text description of the NFT
owner	Dictionary data on the owner (wallet address, purchase price, etc.)
traits	A list of traits associated with the item / describing the item
creator_username	OpenSea usename of the NFT creator
payment_symbol	Symbol of currency (crypto) used for payment
num_sales	Number of times the asset had been sold
collection_name	Collection for which the asset belongs
timestmp	Datetime at which the asset data was retrieved

Dataset Size: 47,047 rows x 15 columns

[1] Source: https://tokenist.com/how-opensea-captured-97-of-the-nft-market/

Table-01: NFT recommender datasets – represents key data attributes used in building the recommender system (non-exhaustive list of data elements)

Transactional NFT Event Data

Transactional data was collected as a pseudo-random sampling of activity occurring on the OS platform from mid-December '21 through March '22. Each record represents an NFT "event" on the platform -listing, sale, bid, removal- and contains the associated asset meta-data (collection, description, etc.) as well as the respective parties, closing price (if sold) and contract details. This dataset enabled us to harvest a large number of unique NFT wallet ID's (buyers and/or sellers) which were later used to collect NFT wallet data used to build the recommender system.

NFT Wallet Data

Each unique wallet address gathered from the NFT event dataset was passed back into the OS API to collect that respective wallet's asset holdings as well as the transactions for which that wallet had been associated with at that point in time, represented with a timestamp. As of March 2022, we've collected data for 2,813 unique NFT wallets accounting for 500,000+ total NFT records (with duplication). Based on the timing of the API call, it is possible for an NFT to exist in multiple wallets in our dataset albeit very infrequently.

NFT Asset Data

At its core, an NFT is a digital asset that represents ownership of digital items (such as an image file) as well as potential privileges that asset confers on its owner (such as membership in a virtual club). Most NFT's belong to a higher-level parent collection characterizing those NFT's. With that in mind, returning to the NFT asset dataset, the core variables of interest for the recommender system include the asset description, traits (uniform categorical features of an asset), the url of the raw image file, the collection name which the asset belongs to, creator name, and historical asset transactional data.

NFT Wallet Data Pipeline Details

From a technical perspective, to compile these datasets required a pipeline connecting OS API calls to a database with tables designed for storing the API response JSON data. Additionally, due to rate-limitations this pipeline required a scheduled interval to ensure a fault-tolerant design if we hit the rate-limit (which we did not).

The API calls are fairly "garden-variety" REST types requiring authentication and can be written in a number of languages, including Python. Each dataset mentioned above has its own end-points, parameters and JSON response payload. From there, our tested and validated API scripts -written in Python- were deployed on an Azure instance and scheduled as CRON jobs. As a final step, the API payload was inserted into structured postgres database tables, also stored in Azure. More details of the OS datasets and APIs can be found in the docs[2].

NFT WALLET CLUSTERING

To build a high-quality recommender system requires an intuition of NFT wallet holders as well as understanding NFT asset characteristics. We begin by exploring NFT wallet data in an attempt to build a wallet-level classifier.

NFT Wallet EDA

[2] Source: https://docs.opensea.io/reference/api-overview

From our NFT market research we came to learn there are different segments of NFT buyers, ranging from well funded Venture Capitalists and Investors, to collectors genuinely interested in accumulating NFTs, to individual speculators motivated to make a “quick buck”. Based on this subjective understanding, we thought it’d be good to start by looking at distributions for wallet’s NFT transaction count, number of NFTs held and the variety of unique collections held within the wallet.

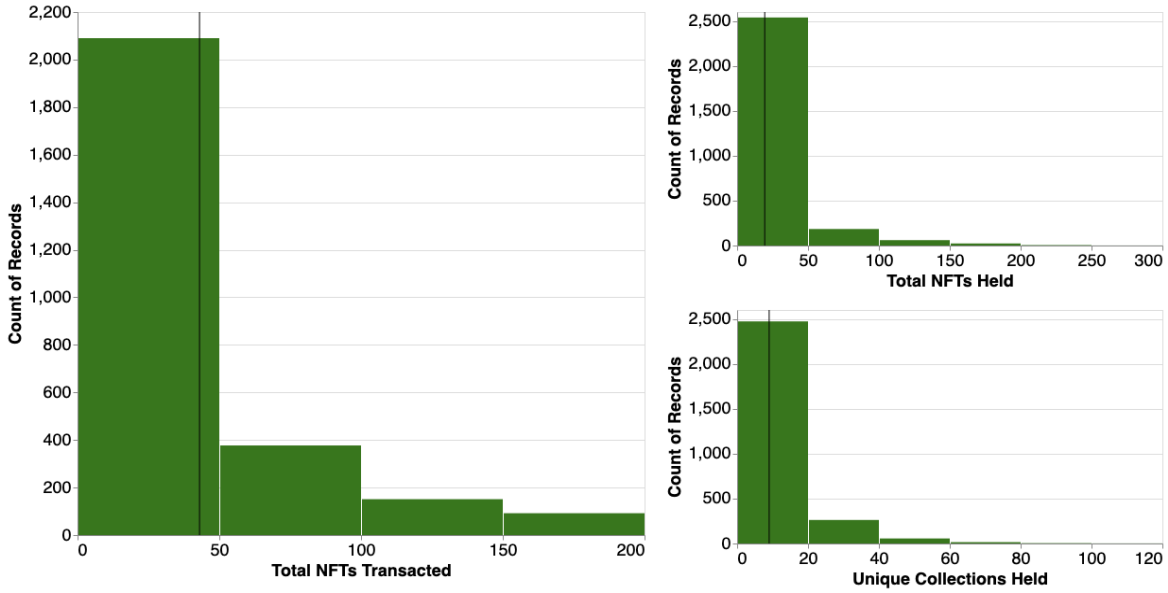


Figure-01: Distributions of the number of NFT transactions per wallet, total NFTs held and unique collections held

Analyzing NFT transaction count and holdings across 2,800+ unique wallet’s, we observe right-tail skewness suggesting a proportionately small number of wallets engage in a very large number of transactions and hold many NFTs and collections. These properties lead us to hypothesize an NFTs collection matters less to wallets holding many NFTs vs those holding a lesser number representing “ordinary” collectors.

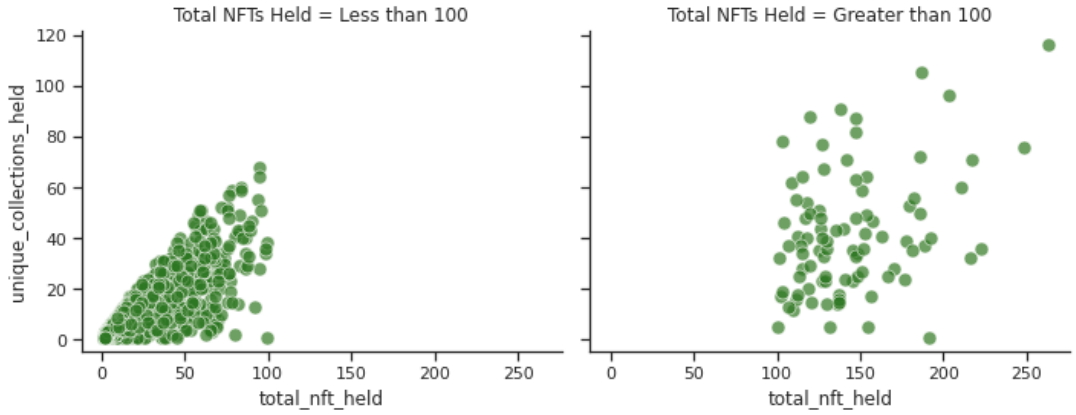


Figure-02: Total NFT holdings vs unique collection holdings for wallets with less than 100 NFTs and greater than 100 NFTs

Looking at the relationship between number of NFTs in a wallet and count of unique collections in that wallet suggests that “ordinary” collectors (less than 100 NFT holdings) are likely to purchase from different collections, whereas higher-volume wallets tend to be more concentrated on their collection holdings. This relationship is represented by a positive correlation coefficient of 0.69 compared to 0.11 for wallets containing greater than 100 NFTs.

These findings imply that a recommender system could benefit from discriminating on NFTs held when determining which collections to recommend NFTs from. It could, for instance, recommend more concentrated collections for wallets with high holdings, representing “exploitative” tendencies, versus more diverse collections for wallets with lesser holdings representing “explorative” tendencies.

With a basic intuition of NFT wallet characteristics, we now look to build a wallet classifier using unsupervised ML to cluster wallets with similar properties in an effort to generate more meaningful recommendations based on a unique wallet archetype.

NFT Wallet Classifier Feature Selection

In addition to the above features discussed -total wallet NFTs, NFTs transacted, unique collection holdings- we’ve also found each wallet’s total NFTs sold and average volume of the NFTs held to have discriminatory power. Additionally, we engineered a wallet feature representing the amount of churn defined as the ratio of NFTs transacted by total (current) NFT holdings. The motivation behind this feature is higher churn reflects higher relative transaction velocity, versus a “buy and hold” strategy with lower transaction velocity. Wallet holders with different strategies and objectives (i.e. traders vs collectors) might have different NFT recommendation preferences.

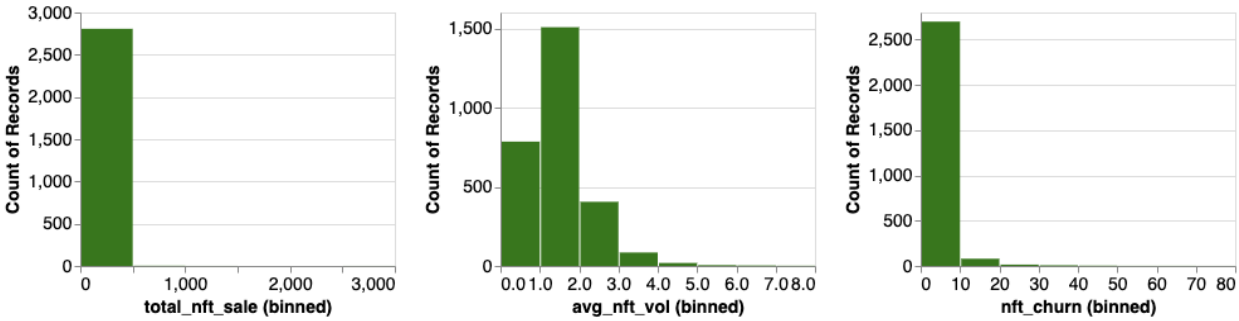


Figure-03: Distributions of the number of NFT sale transactions per wallet, average volume for wallet NFTs and NFT churn rate (lower = less relative transaction velocity)

Classifier Build & Evaluation

We begin by fitting a basic k-means classifier (scikit-learn) with default parameters, max iterations set to 100 and a single run with centroid seeds on the aforementioned feature space. With this fitted model we’ve iterated over a range of cluster sizes from 2 to 20 and evaluated the inertia score for each. The inertia score measures the distance between each point and its centroid with lower scores representing greater density and cluster “compactness” – desirable properties. Applying the “elbow method” to the inertia plot is a subjective heuristic for balancing low inertia scores with a relatively lower cluster count. As we see in Figure-04 below, a cluster size of 5 seems to generate a favorable inertia score with diminishing returns as we scale beyond 5 clusters.

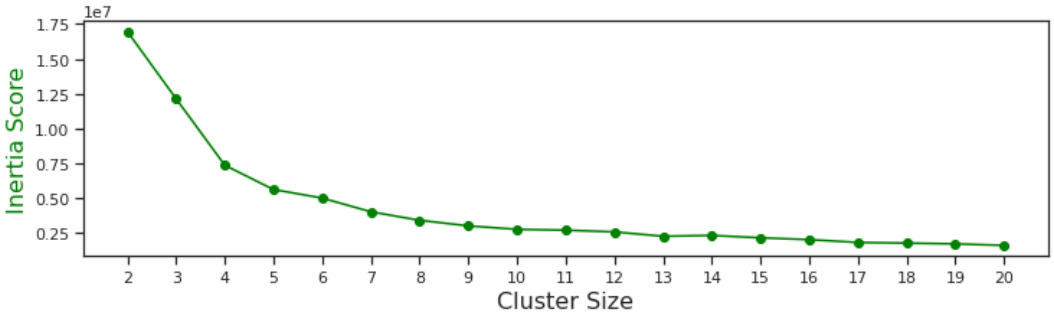


Figure-04: K-mean classifier evaluation metric (inertia score) as a function of cluster size

With the number of clusters set to 5, we use the fitted k-means model to predict a class label for each wallet and review the predictions across the range of input features to qualitatively assess the “goodness” of the model. Figure-05 below provides a scatter plot matrix of the class predictions relative to the input feature space.

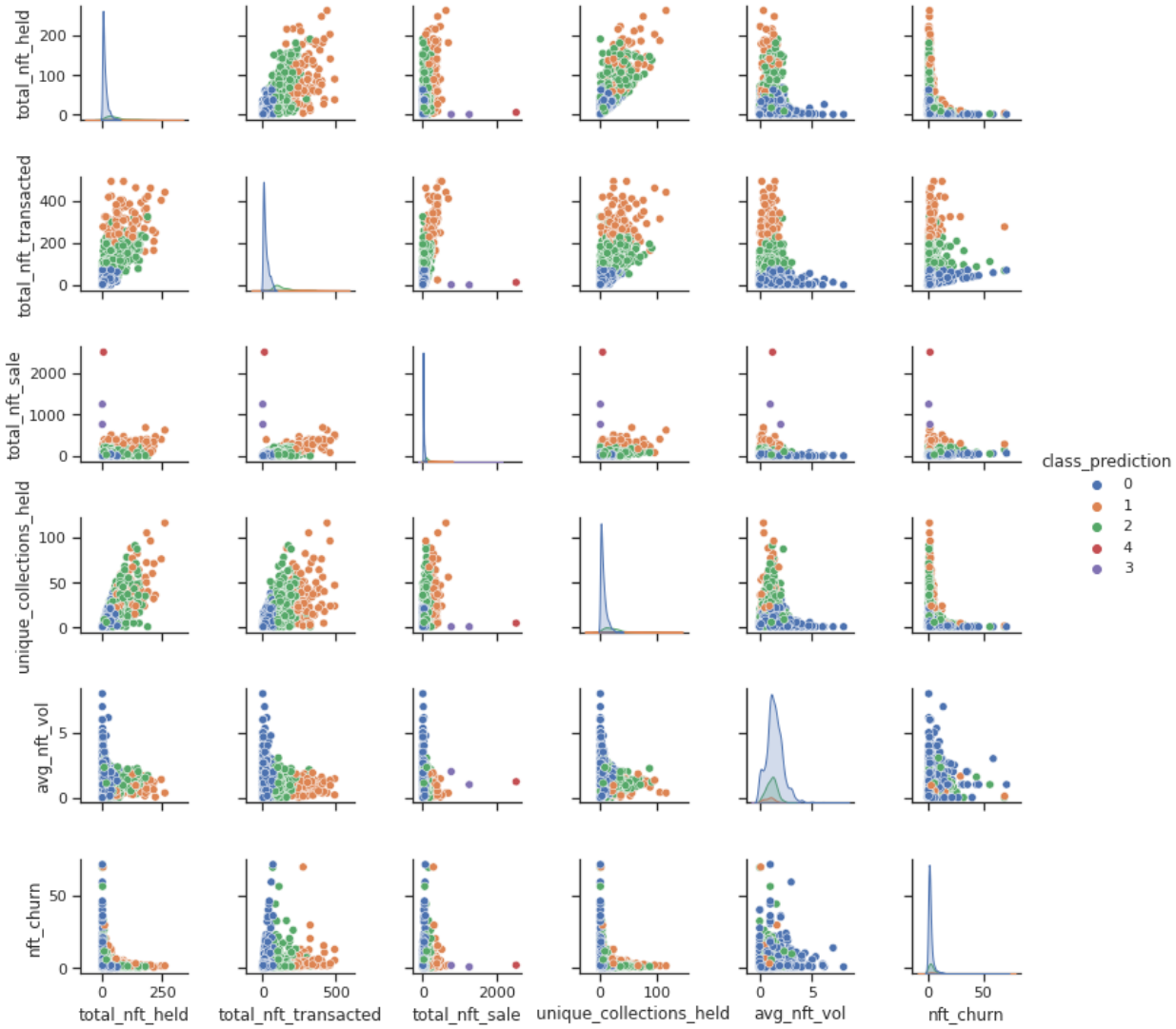


Figure-05: Scatterplot matrix of feature variables color encoded by class prediction for each wallet

Analyzing Figure-05, there are a couple findings that stand out. First, it appears that the classifier does a reasonable job at segregating the wallets into classes based on observed boundaries of relative class color encoding across the pair-plots. Secondly, those boundaries appear to be especially pronounced for total NFTs held and NFTs transacted which validate our intuition around market segments and findings from the wallet-level EDA.

NFT Wallet Classifier Next Steps

This concludes our initial modeling and analysis of an unsupervised ML NFT wallet classifier. In relation to the broader project objective, next steps for this component would be to insert it into the pipeline to classify the end-users wallet and have custom recommender systems for each wallet class as a means of generating higher-quality recommendations.

NFT TOPIC MODELING

To further improve the recommender result set, we turn to NFT topic modeling in an effort to measure the similarity of the NFT recommendations relative to each other as well as those held within the wallet. More specifically, we developed Latent Dirichlet Allocation (LDA) and Latent Semantic Indexing (LSI) models and fitted them to the pre-processed, vectorized description string for each NFT. Before digging into the models, we first start with NFT description exploratory data analysis.

NFT Text Description EDA

We begin by unpacking all of the NFTs from their respective wallets, leaving us with 32,873 unique NFTs each of which contain some form of text description. From there, we split each NFT description into "tokens" (i.e. tokenize) and remove commonly used stopwords, as defined in the stopword corpus contained in the gensim library[3]. Figure-06 below provides summary statistics on the resulting tokenized dataset. As you can see, there are a total of 47,649 distinct tokens with a token count average of 14.

```
count    47649.000000
mean        14.441730
std        106.040427
min          1.000000
25%          1.000000
50%          2.000000
75%          5.000000
max      10524.000000
Name: token_count, dtype: float64
```

Figure-06: Summary statistics of the total NFT Description tokenized dataset

Reviewing Figure-06, it appears additional string cleaning is required based on the max, min and interquartile range values. To better understand the tokenized data, we looked at the token count frequency for the top 50 most frequent words; Figure-07 below captures this frequency count.

[3] Source: https://radimrehurek.com/gensim/parsing/preprocessing.html

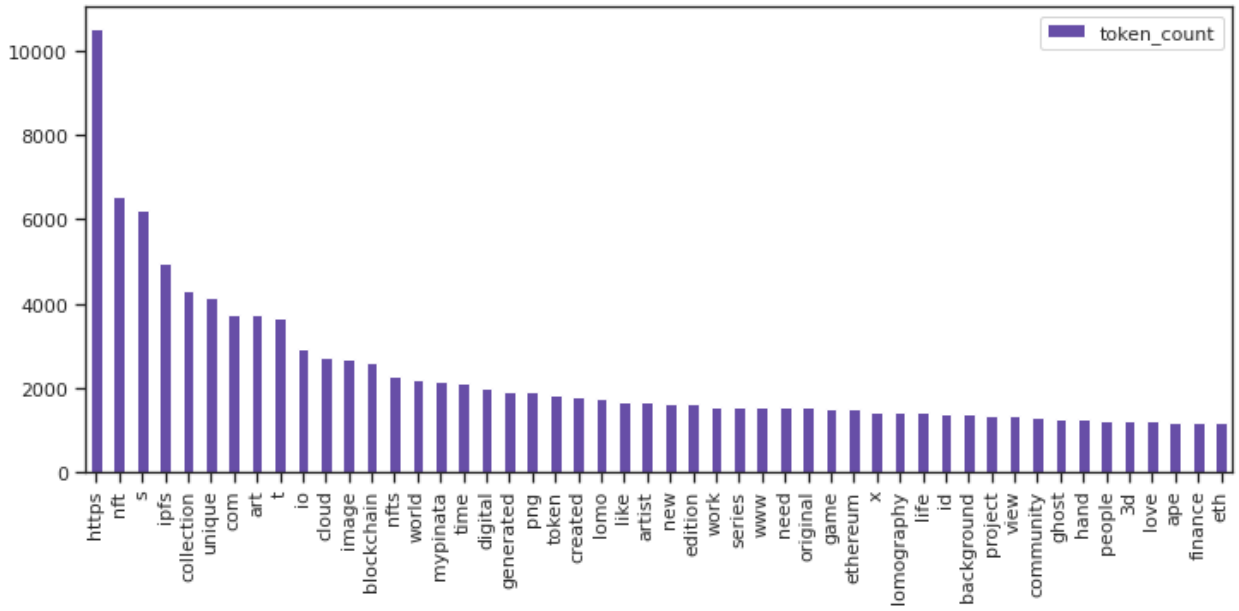


Figure-07: Count frequency for the top-50 most common tokenized words in NFT descriptions

Analyzing the token count frequency, we first observe words that are not very descriptive nor expressive and thus contain low-information gain; a few examples: “https”, “com”, “nft”, “eth”. The “https” and “com” suggest URLs are contained in many of the NFT descriptions. Additionally, there are several single-character tokens like “s”, “t” and “x” that do not add-value. Based on these insights, we append the stopword corpus with these tokens, and perform additional string cleaning such as constraining the token character length to greater than 3 and less than 15. Furthermore, we chose to exclude tokens with a frequency count below 10 or above 10,000. Finally, with the dataset prepared, we convert the documents into a bag-of-words (BOW) format using gensim corpora dictionary module and vectorize the BOW with a term frequency-inverse document frequency (TF-IDF) vectorizer from gensim models module.

LSI + LDA Model Build & Evaluation

The topic models were built using the gensim models library. Starting with LSI, this gensim model “implements fast truncated SVD (Singular Value Decomposition) … [which] can be updated with new observations at any time, for an incremental, memory efficient training”[4]. LDA, on the other hand, “trains a model from a training corpus and inference of topic distribution on new, unseen documents.”[5] Similar to the k-means wallet clustering model, both LSI and LDA are unsupervised ML techniques and require a pre-specified number of topics (conceptually similar to number of clusters). Therefore, we start by looping over an array of topic numbers, ranging from 5 to 100 topics in increments of 5, generate an LSI and LDA model using the corpus (TF-IDF for LSI) and compute coherence scores for each model based on the varying number of topics. Figure-08 captures the coherence scores for each model with different topic numbers.

[4] Source: https://radimrehurek.com/gensim/models/lsimodel.html
[5] Source: https://radimrehurek.com/gensim/models/ldamodel.html

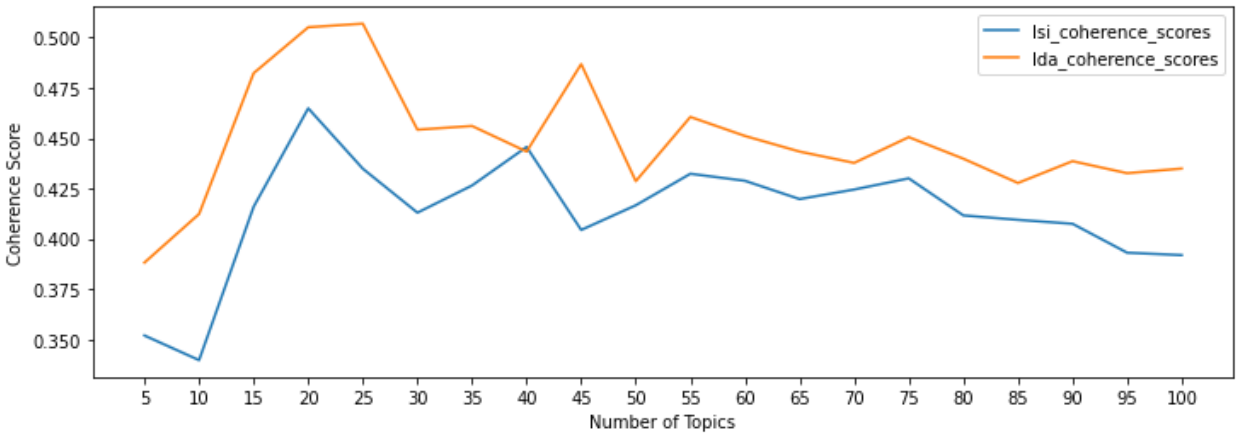


Figure-08 NFT description topic modeling coherence scores for LSI and LDA models

Coherence scores measure the degree of semantic similarity between words in a topic – higher scores represent higher levels of similarity and are thus more desirable. Analyzing the LSI and LDA coherence scores we see LDA marginally outperforms LSI at generating semantic similarity, with 25 topics resulting in the largest coherence score of ~0.5. Therefore, our final topic model is an LDA model with the number of topics parameter set to 25.

We conclude our topic modeling analysis with a visualization of the model results leveraging the pyLDAvis package[6], as reflected in Figure-09.

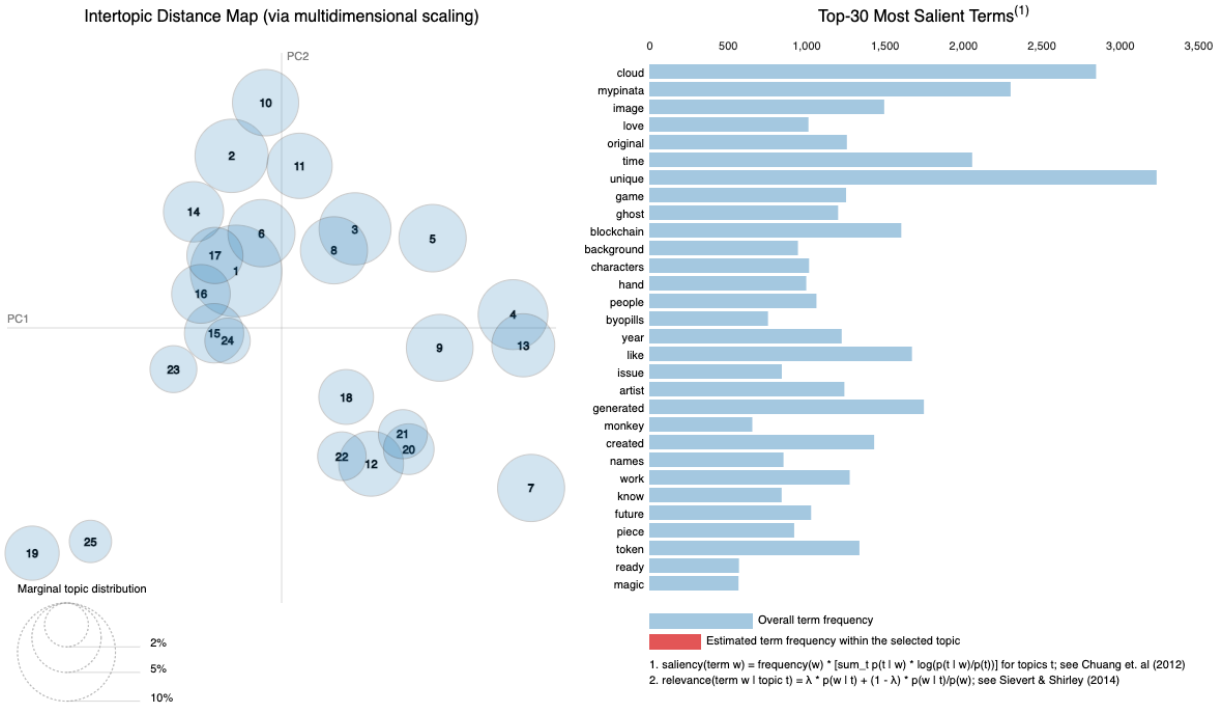


Figure-09 Intertopic distance map and top-30 most salient terms produced by the LDA model

[6] Source: https://pyldavis.readthedocs.io/en/latest/readme.html

Overall we see the 25 topics are mostly homogeneous across 2 principle components, with the top-5 most salient terms being “unique”, “cloud”, “mypinata”, “time” and “generated”. The terms “cloud’ and “mypinata” in particular stood out to us and after further research we came to learn these are in reference to an NFT media management platform known as Pinata[7].

NFT Topic Modeling Next Steps

We’ve completed development of a first-pass initial topic model. As a next step we look to integrate the model into the NFT recommender to evaluate NFT topics within a wallet and prioritize NFT topic homogeneity in the recommender result set, overlapping those already held. We believe such capability can further enhance the quality of recommendations served.

NFT RECOMMENDER SYSTEM

Choice of Recommender

Since our goal was to create a live web application, we needed a recommender that could provide speed, modularity, and good results:

- Speed and modularity were necessary in the sense that new users needed to be added to our recommender model and be able to receive recommendations as quickly as possible.
- In order to facilitate recommender quality, we started with a history of transactions and developed and evaluated models against this transaction history, using previously bought NFTs as our relevant documents when assessing precision.

Prior work[8] in the area informed our decision to proceed with a collaborative filtering model. We considered the following recommenders by testing them with a training set before deploying to the app:

- Full Matrix Memory-Based Collaborative Filtering[9]
    - We used cosine similarity between users to generate predictions for all assets in our training set. See Evaluation for a discussion of the strengths and weaknesses of this model.
- Community Matrix Memory-Based Collaborative Filtering
    - We applied the same prediction method as the Full Matrix Memory-Based Collaborative Filtering, but reduced the number of users in consideration via “Community” or “Profile Space” filtering.
    - Community/profile space filtering involves limiting the users and assets to the community of users that own an asset the query user owns, and the assets owned by the query user and that community of users. Precedence for this model is using cosine similarities to only show the top N most-similar users[10]. Our method is different in that the community size (N) is variable, and based on the natural connectivity of the user.
        - This decreases the time to predict and the accuracy, as shown below.
        - This also results in null recommendations for users without a sufficient community, which is a problem in environments with sparsely connected assets. This is a problem that is addressed in future enhancements.

[7] Source: https://www.pinata.cloud/
[8] Source: https://blog.opensea.io/analysis/predict-and-recommend-nfts/
[9] Source: https://developers.google.com/machine-learning/recommendation/collaborative/basics
[10] Source: https://medium.com/sfu-cspmp/recommendation-systems-user-based-collaborative-filtering-using-n-nearest-neighbors-bf7361dc24e0

    - The most important feature this model has is that the number of users and assets grows more slowly than the full matrix solution, allowing for some future-proofing of our app to prevent excessive recommendation times (or even out-of-memory events).
    - See Evaluation for a further discussion of the strengths and weaknesses of this model.
- Full Matrix SVD-Based Collaborative Filtering[11]
    - Here, we implemented what turned out to be a model that performed poorly on precision-based metrics but exceptionally on RMSE-based metrics using matrix factorization. See Evaluation for a further discussion of the strengths and weaknesses of this model.

Recommender Evaluation

Model	*Precision @1000[1]*	*RMSE @1000[1]*	*Time to query model for new user (seconds)[2]*	*Average Rate of RMSE Degradation per Additional Asset[3]*
Full Matrix Memory-Based	0.858	0.217	170-180+	0.138
Community Matrix Memory-Based	0.568	0.367	1-3+	0.130
Full Matrix SVD-Based (topics=50)	0.004	0.320	1-2+	0.036
Full Matrix SVD-Based (topics=500)	0.060	0.247	24+	0.014

[1] Averaged across a sample of 1M user profiles utilizing 1000 retrieved documents
[2] Based on non-cached predictions, not including data handling, using 1M rows of context data
[3] Fit to average RMSE score by number of assets (from 1 to 9) in user wallet series

Table-02 Recommender model Precision, RMSE scores and performance metrics

Referring to Table-02 above, we will explain point by point why the Community Matrix Memory-Based model is the one chosen for deployment:

1. Precision
    - In this round, Full Matrix SVD-based solutions were rejected due to their very low precisions. Since the SVD-based solutions compress the data, the time to query a new user is faster, but the quality of the results is lower.
2. RMSE
    - The RMSE of both memory-based models are similar, with the Full Matrix version being the best of all models since it uses the full information of the matrix for every prediction.
3. Time to query model for new user
    - The Full Matrix version performs well on precision and RMSE, but fails when it comes to updatability. The Full Matrix version needs to process the entire matrix of user-nft interactions each time a new user is added to it. The Community Matrix version is much faster, being able to limit its users and assets under consideration based on the community filtering.
4. Rate of RMSE degradation per additional asset
    - The rate of RMSE degradation is also an important factor to take into account, because we want a model whose prediction quality decreases as slowly as possible as more assets are added into the mix. In order to find this value, we fit a simple linear regression

[11] Source: https://developers.google.com/machine-learning/recommendation/collaborative/matrix

model on the RMSEs obtained from our dataset of 1M users to estimate the increase in this error metric for each model.

- As reflected in the table above and figure below, the SVD-based models had very low RMSE-degradation over the number of additional assets.
- Though the Community Matrix never quite catches up with the Full Matrix Memory-based model, the difference in RMSEs quickly gets very small, with the difference in RMSE between the two models collapsing to 5-6% after a user owns 4 NFTs.

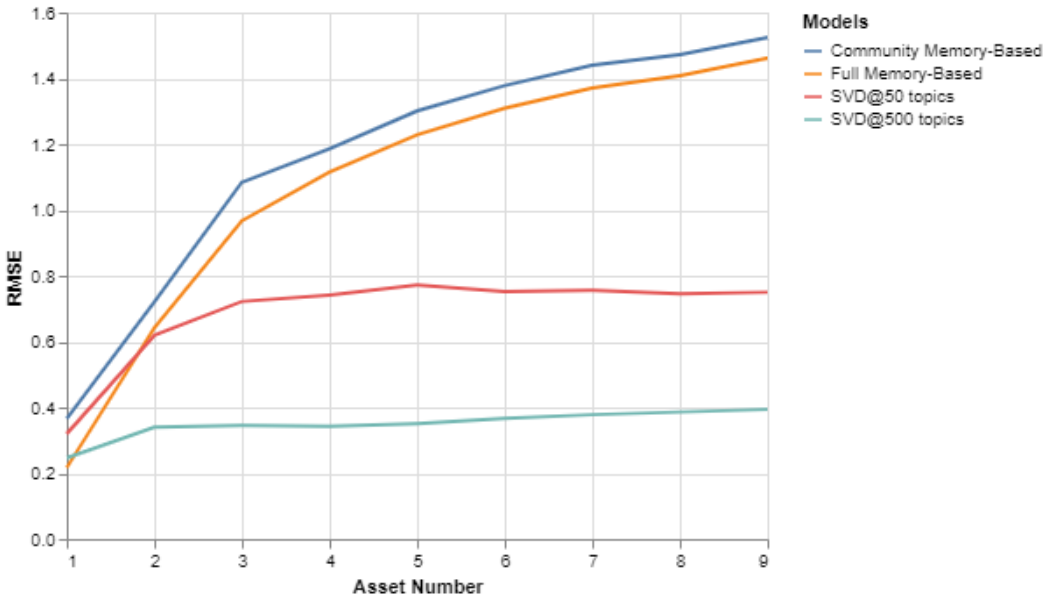


Figure-10 RMSE scores for different numbers of assets owned, by recommender model technique

Recommender Deployment

We ultimately deployed the Community Matrix Memory-Based model due to its high speed, relatively high precision, and close RMSE performance with the Full Matrix Memory-Based model. We used an ownership context database and a coupled data handler to manage the creation of the community for the query user.

The ownership context database stores all assets “owned” by a user and any feedback that user has given on an asset. This provides a clean storage space for all of the information necessary to generate recommendations in our community-based model. This database will grow larger as more users, assets, and user interactions on the website are added.

Recommender Feedback

We also revised the deployed Community Matrix-based model to take into account positive and negative feedback from users on the website by assigning positive values to all positively-rated community assets and assigning negative values the same for negatively-rated community assets.

If a user uses the positive feedback (thumbs up) button on the recommendations section of the website, it will add the asset to their simulated wallet. If a user uses the negative feedback (thumbs down) button, it will add the asset to their ownership context with a negative amount, but not the wallet, since the user doesn’t like it enough to buy. Increasing the default amounts for positive and negative feedback respectively increase and decrease the spread of predicted ratings results for the user.

Increasing the positive feedback number can quickly drown out negative feedback however, so it is necessary to carefully balance prediction spread and the impact of negative feedback.

Recommender Future Enhancements

Based on properties above, the evaluation of our models, and the goal of our application, we have several areas to make enhancements:

- Poorly-connected users
    - Most users do not own many assets, and owe the size of their community either to a handful of other smaller users or to one user with a large amount of assets.

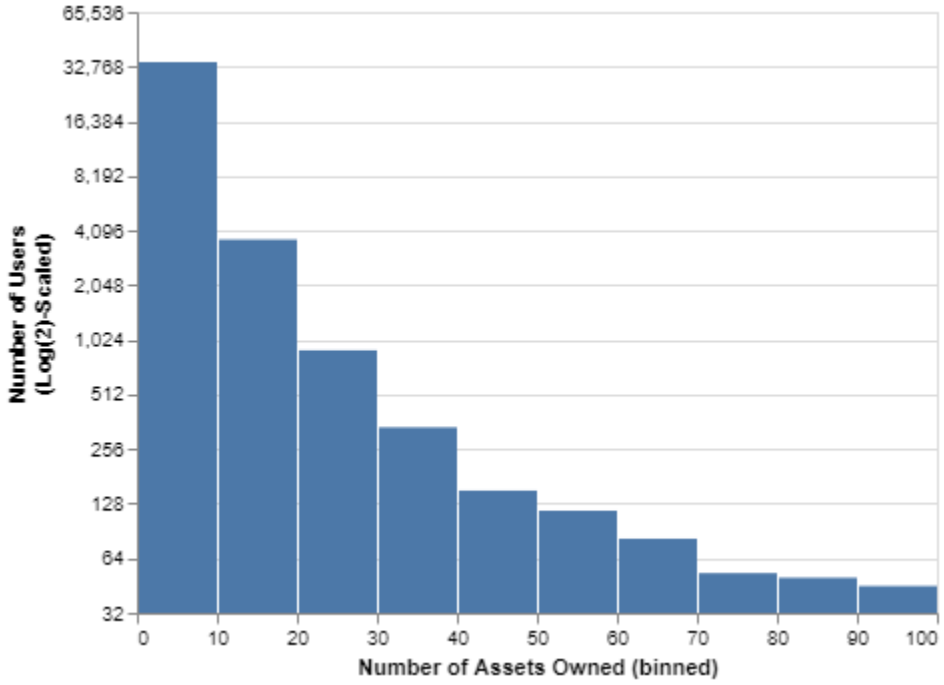


Figure-11 Distribution of NFT users and assets owned in their wallet

    - This sometimes results in very poorly-connected users who may not have any assets in common with anyone else. We currently return a random set of recommendations for these kinds of users.
- Model Precision and RMSE Degradation
    - Model precision could be higher, with just over half of relevant documents showing up in every large query.
    - For the Memory-based models, RMSE increases at a high rate for every new asset added to a user's wallet. It would be more desirable to have a minimal increase in the RMSE per new asset, as in the SVD-based models.
- Content-based filtering
    - Content-based filtering is another family of recommender models that seeks to use features built from information the users and assets to create recommendations.
    - In our case, we can use image and text data associated with the NFTs to engineer features for each NFT, and then use this information to characterize users.

    - There are many approaches where existing collaborative filtering models are extended using content-based filtering[12].

Recommender Next Steps
In order to make improvements along the lines previously suggested, a model was proposed but not developed. This model would make use of our unsupervised research and involve transforming image and text data.

The model would cluster users and assets via LSI/LDA or other unsupervised clustering methods, and then allow either a collaborative or content-based filtering model to be built on top of this user type-asset type matrix. This has a few benefits:

- Ensured community
    - Assigning every new user to a type regardless of the number of assets they hold will ensure that there is a set of NFTs that can stand in for or augment the user's potentially sparse community. A user with very few assets will still be able to receive recommendations that make sense.
- Generalized model information
    - Encoding information from a larger scope than the user level into our model will help ensure that RMSE is not as subject to random user-localized variations in assets and does not degrade as quickly over time.
- Cross-validation
    - We can combine information about the assets in a type cluster and past information about the user's visual and topic preferences to make direct comparisons of the assets being considered in a recommendation, in order to cross-validate and remove or promote assets based on their visual and topical similarity to the assets in the type cluster.

WEB SERVICES AND DEPLOYMENT

As we are building a system to recommend NFTs, we wanted to provide an easy way for users to try our system without actually needing to spend money and purchase NFTs. For this, we created a simulator that can:

- Create simulated wallet IDs
- Provide an asset marketplace where users can purchase assets for their simulator wallets using paper money
- Users can find more details about an NFT they are interested in, and be able to open them in OpenSea
- Users can get a recommendation view where they can control the number of recommendations they see
- Users have the ability to provide feedback on the recommendations shown, which affects the future recommendations that they receive

The simulator encompasses 3 aspects:

1. Web Frontend
2. Web Backend
3. Web Hosting / Infrastructure

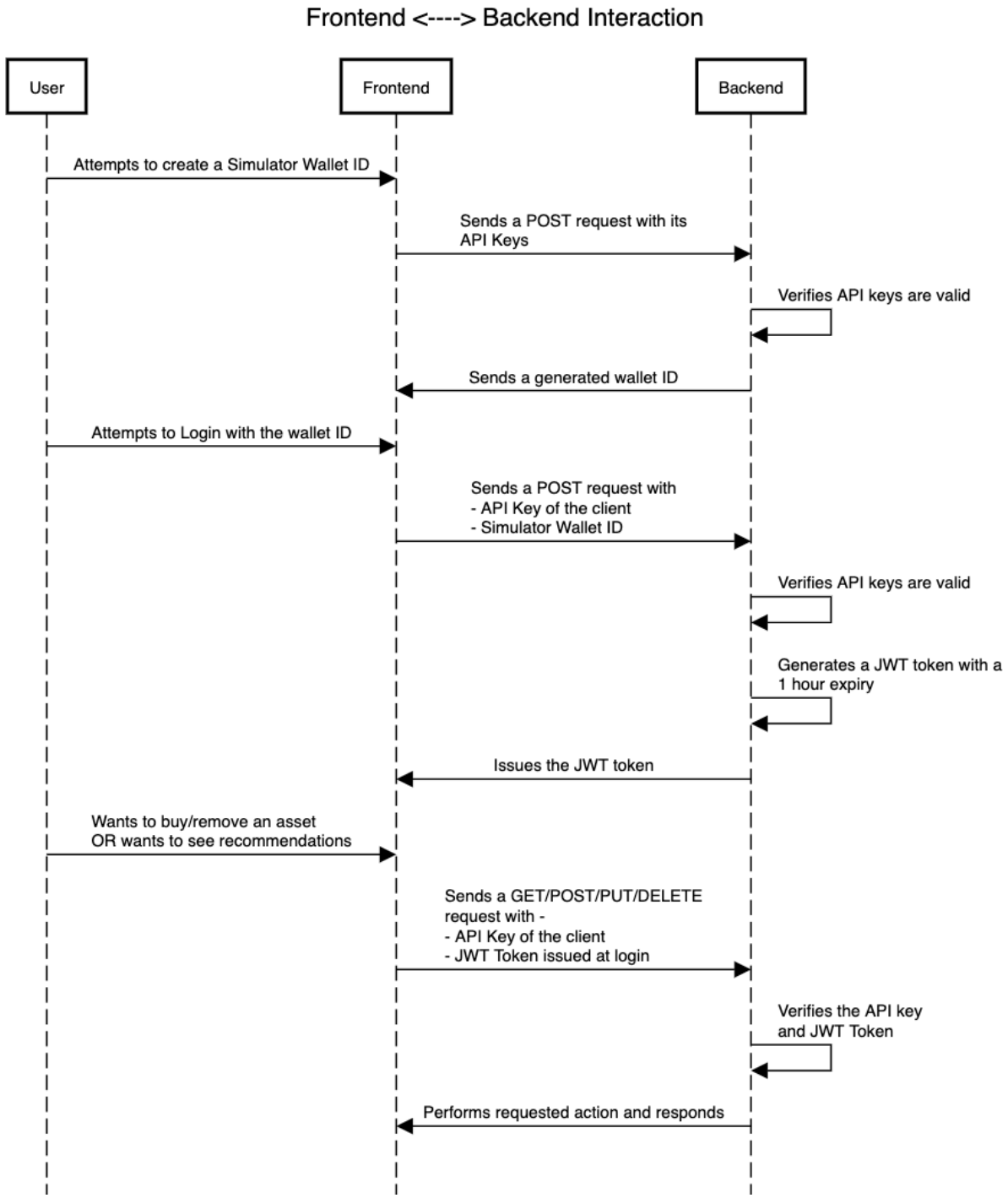
Below, we detail each of these aspects.

[12] Source: https://github.com/lyst/lightfm/tree/master/lightfm

1. Web Frontend

The frontend is written in ReactJS and leverages open source frontend libraries such as Bootstrap / CoreUI / others. It is hosted in Microsoft Azure and we have purchased a domain through GoDaddy for the website (www.nft-recs.com).

The sequence flow for the website is as follows:



2. Web Backend

The backend consists of two components:

- RESTful API Server
- PostgreSQL DB

The API Server is written in Python using Flask as the web framework, which made it easy to integrate with our recommender model which is also written in Python. The API Server uses a PostgreSQL DB as its persistent store. This database consists of our main dataset as well as stateful data that supports operations of the project.

The sequence flow for the backend is illustrated in the above Figure.

3. Web Hosting / Infrastructure

For hosting the website and the infrastructural components, we decided to use Microsoft Azure due to the following reasons:

- Microsoft Azure offers $100 free credit with lifetime hosting for free for student accounts (even after the credits expire)
- We started working on our project early in the course before instructions on availing AWS credits were conveyed

Using Azure, we set up a CD (Continuous Deployment) pipeline with our Github repository for the project. That way, any of us could commit code to the main branch which would trigger an update to the server or the website.

STATEMENT OF WORK

Project Lighthouse was produced by Sahil Pujari, Joseph Hardy and Mike Clark. Mike led the dataset collection, NFT wallet clustering and NFT topic modeling. Joseph drove the research and development of the recommender system including supporting production deployment. Sahil took the lead on the UI web-application development including full stack end-to-end integration of the requisite technologies. Each team member contributed to the production of this written report for their respective focus area.