NetEase Cloud Music Data

Advanced Machine Learning

Business Analytics Online

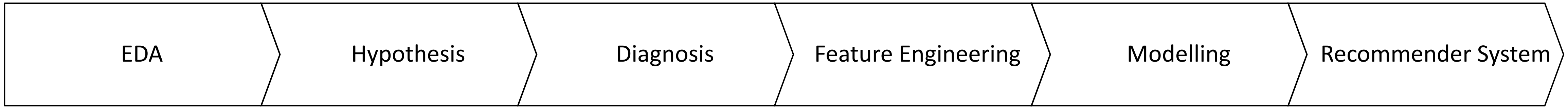
The analysis focuses on NCM’s Cloud Village tab

- NetEase Cloud Music is one of the largest free music streaming companies in China.
- The major product of NCM is a **music app** that has four main user interfaces:
  - The Main Page
  - Music Video
  - My Own Music
  - Cloud Village

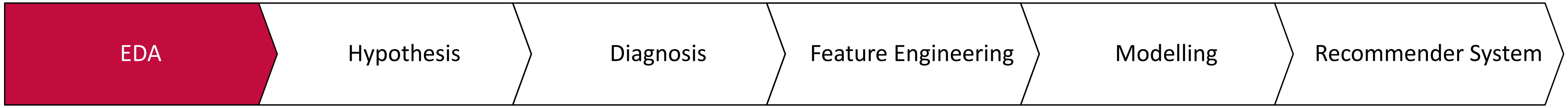
- The analysis will focus on **Discovery subtab within the Cloud Village tab**, which is an interface were the user can visualize **recommended cards** (either a short video or set of pictures with text with music background).

- The data set analysed contains more than 57 million impressions/displays of music content cards recommended to a random sample of 2 million users from **November 1st to November 30th of 2019**.
- The data set also contains information on each user, each content creator, and each content in the impression sample.

The analysis process was structured into six distinct steps



The first step is to perform EDA to identify questions to be solved through ML models



- Individually examined tables.
- Analysed variable distributions.
- Addressed missing values.
- Addressing outliers.

These primary objectives of the EDA guided the exploration of each of the six available data tables

Understand the structure of the Data	Handling Missing Values	Numerical Values Distribution	Categorical Variables Distribution	Outlier Identification & Treatment	Data Quality Checks
	by exclusion or imputation	through histograms and Box-plots	Through frequency analysis and bar charts	by exclusion, flooring and capping values	

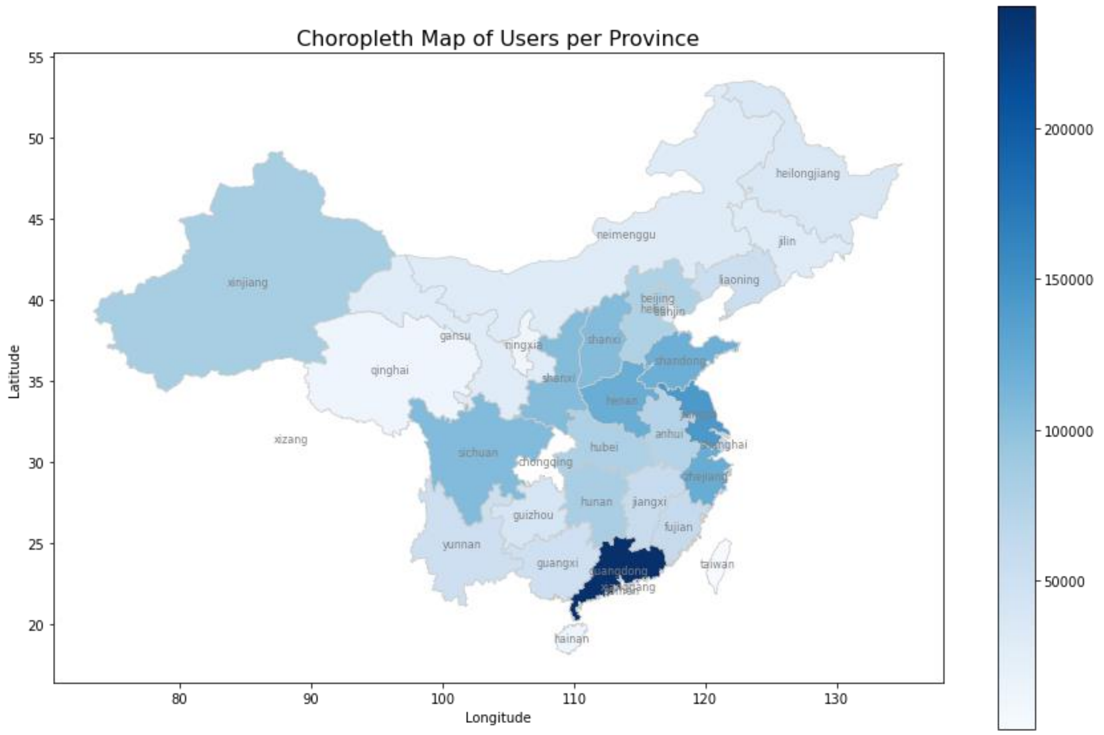
The complete EDA is included in the "Final Project EDA" Jupyter Notebook. In the next slides, some insights are highlighted.

User Demographics

Assuming that the data is representative of the population, 75% of users have registered within the past 35 months, indicating a potential influx of new users between 2017 and 2019.

Users are distributed among all provinces with the highest concentration is in Guang Dong.

A distinct age demographic is prevalent, particularly among users aged 15 to 25.



User Engagement

Users generally have high activity levels, engaging heavily with music and video content, but there's limited interaction among users on the platform seen from the relatively low average follow count.

Higher activity during weekends, with peaks on Saturdays and Sundays.

There is a strong correlation the user tenure on app and their activity level.

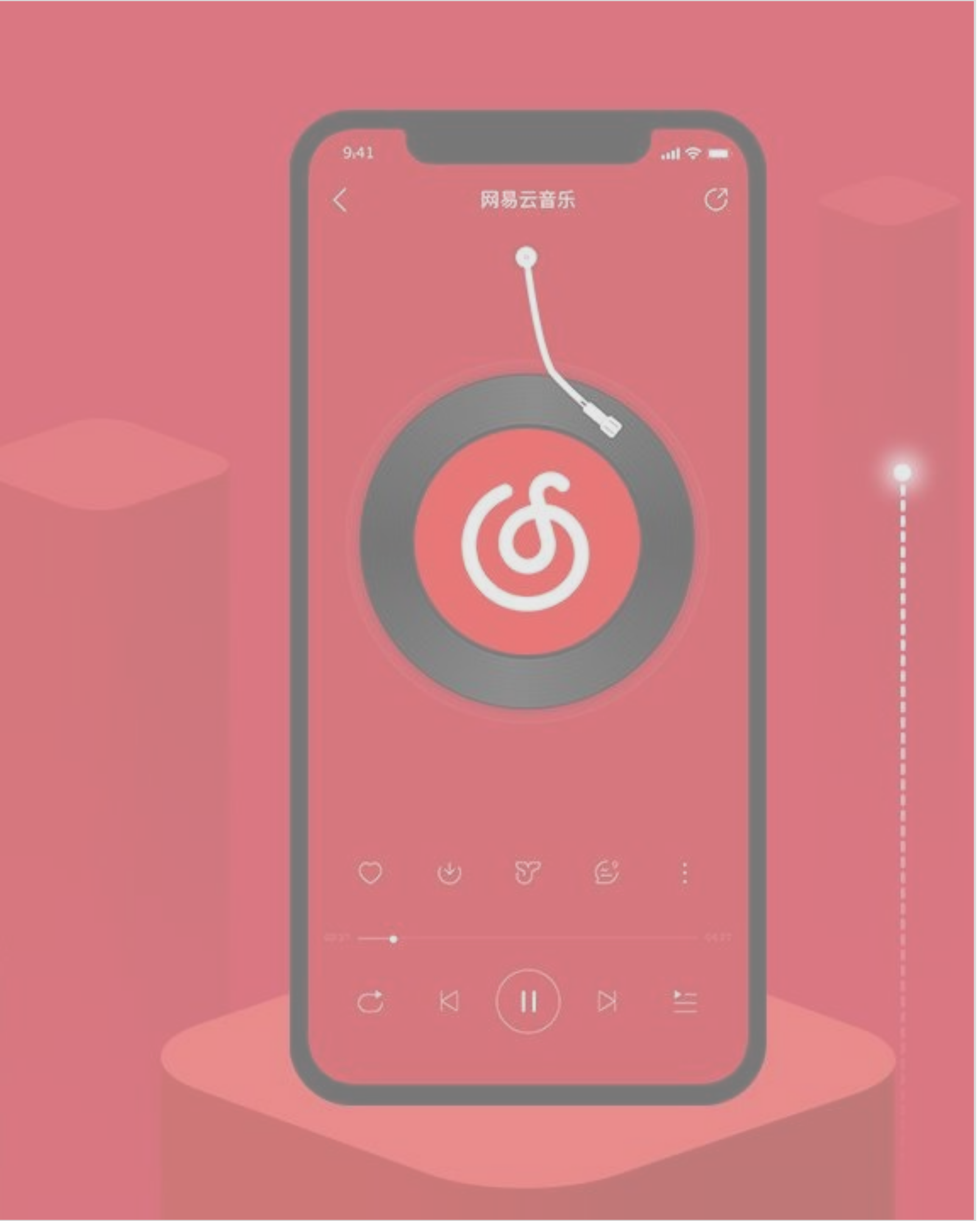
A high activity level on the platform overall does not imply that the user is active in the Discovery Subtab.

Creator Insights

Less than 1% of the users in the random sample are also creators.

Naturally, creators engagement is on average higher than the general user population.

While the creators’ follow count distribution shape mirrors that of the general users population, creators follow, on average, 3.5 times more



Card Interaction

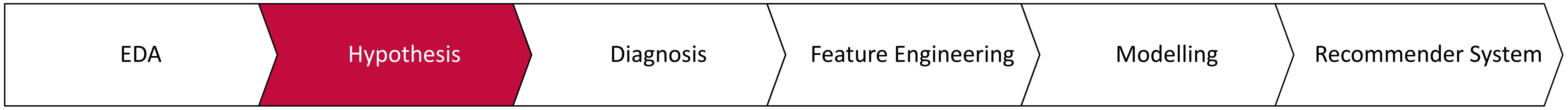
Click, like, share, comment, and creator page viewing percentages are relatively low on the Cloud Village interface, suggesting that recommended cards shown could be optimized.

A growing trend of daily displayed cards exist, yet the reactions remain consistently stable, suggesting that the recommended cards in the discovery tab may not be significantly boosting user activity

Of all the cards displayed in the Discovery Subtab only 4.97% of cards were clicked, 0.24% of cards were liked and 0.03% were shared.

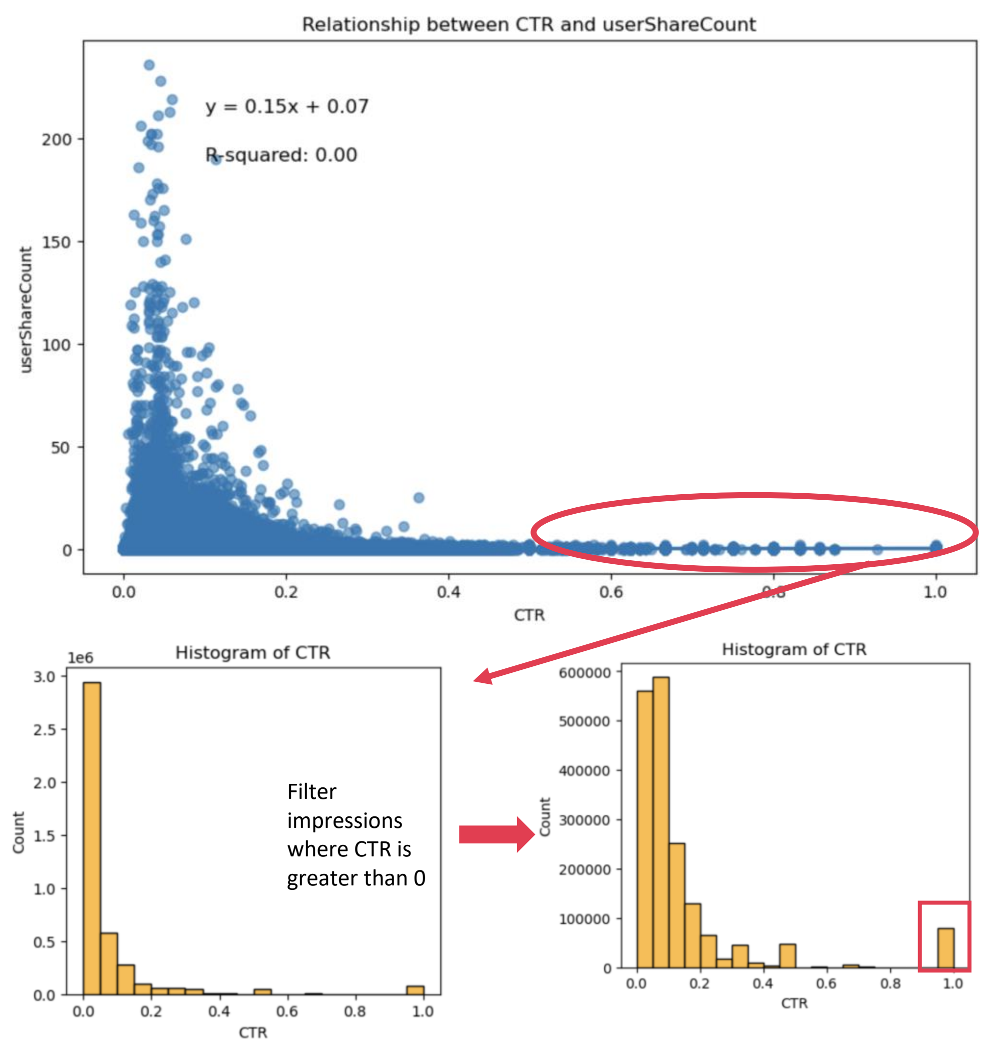
This confirms the suspicion that while users have a high activity level on the application, they are not necessarily active on the Discovery Subtab.

The subsequent step involved addressing hypotheses through data exploration.



- Explored tables collectively.
- Provided answers through detailed exploration.

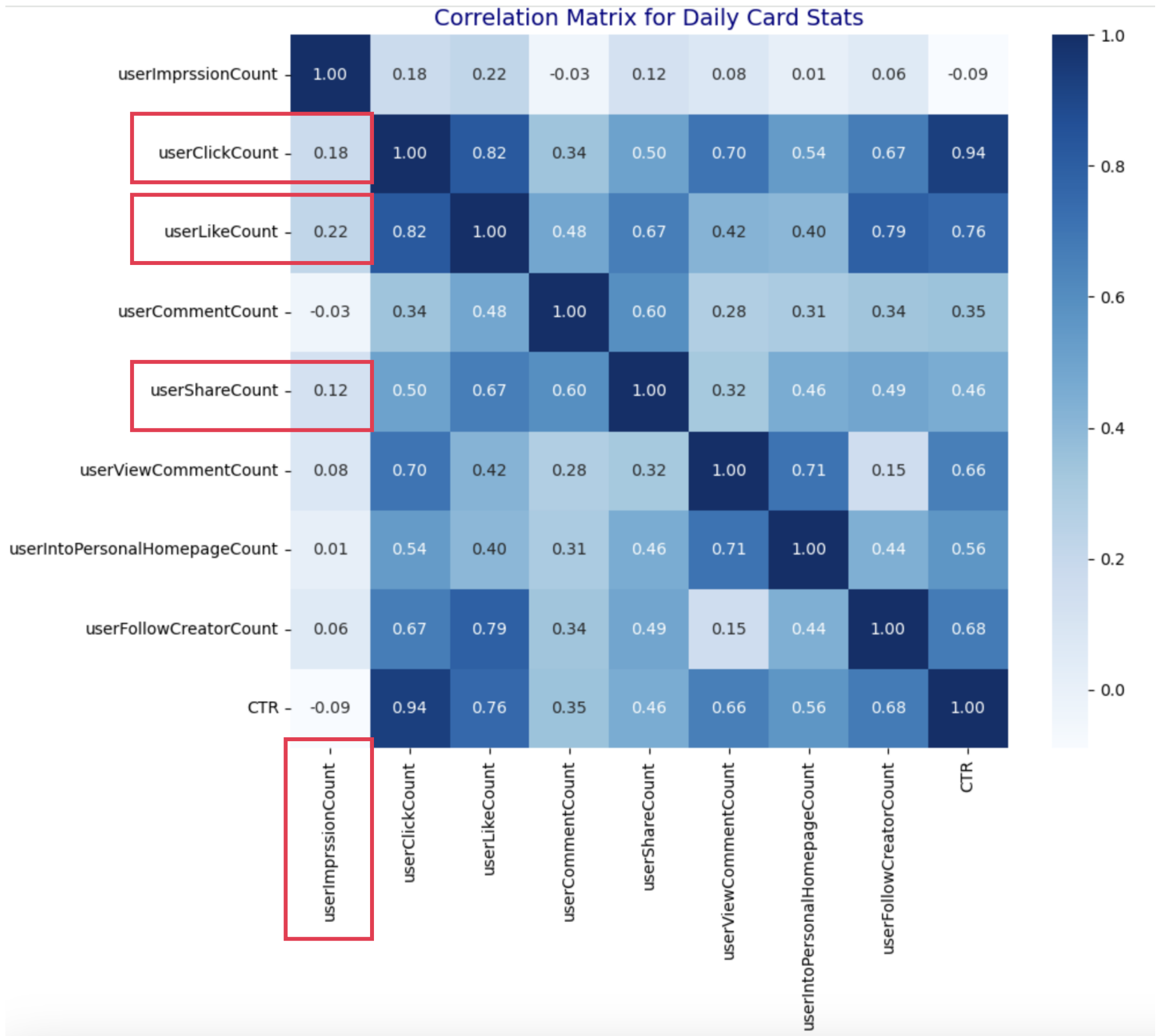
Is sharing an indicator of virality? There is no strong correlation between the share count and the card recommendation count. However, 2% of the cards have perfect CTR*.



- There is no strong correlation between share count and impression count. The same pattern was observed between click count and share count.
- Another way to approach this is to measure card success with the clicks-to-impressions ratio, aiming to recommend high-CTR cards to a broad audience.
- Surprisingly, the R-squared measure for CTR and share count relationship is very close to zero.
- However, there is a small cluster of impressions that have a high CTR.
- 2% of the cards have a perfect CTR, which is a small group in comparison to the rest of the cards, but this could provide valuable insights into user preferences, potentially serving as inputs for the recommendation system.

CTR: Click-through rate

What makes a card go viral? It appears that clicks, likes, and shares remain the primary indicators of “virality”, these counts exhibit the strongest correlation with impressions



	userImprssionCount	userClickCount	userLikeCount	userCommentCount	userShareCount	userViewCommentCount	userIntoPersonalHomepageCount	userFollowCreatorCount	CTR
userImprssionCount	1.00	0.18	0.22	-0.03	0.12	0.08	0.01	0.06	-0.09
userClickCount	0.18	1.00	0.82	0.34	0.50	0.70	0.54	0.67	0.94
userLikeCount	0.22	0.82	1.00	0.48	0.67	0.42	0.40	0.79	0.76
userCommentCount	-0.03	0.34	0.48	1.00	0.60	0.28	0.31	0.34	0.35
userShareCount	0.12	0.50	0.67	0.60	1.00	0.32	0.46	0.49	0.46
userViewCommentCount	0.08	0.70	0.42	0.28	0.32	1.00	0.71	0.15	0.66
userIntoPersonalHomepageCount	0.01	0.54	0.40	0.31	0.46	0.71	1.00	0.44	0.56
userFollowCreatorCount	0.06	0.67	0.79	0.34	0.49	0.15	0.44	1.00	0.68
CTR	-0.09	0.94	0.76	0.35	0.46	0.66	0.56	0.68	1.00

- The left-side correlation matrix was filtered to display exclusively the top 10% of cards concerning impressions.
- The majority of cards within the top 10% boast more than 400,000 impressions.
- In the correlation matrix, it appears that clicks, likes, and shares remain the primary indicators of “virality”, supported by the fact that these counts exhibit the strongest correlation with impressions in comparison to other dataset features, albeit without an extremely high correlation.
- When excluding high-impression cards, comments gain significance (Corr=0.20), just below clicks (Corr=0.35), supporting the idea that true virality relies on a higher number of shares, indicating network effects.
- For less "viral" cards, comments emerge as a potentially strong indicator.

Is the user more prone to click on a "popular" card?

It's the non-visible measure, prior clicks, that have a more significant influence on the likelihood of a user clicking on a card, rather than the more visible likes

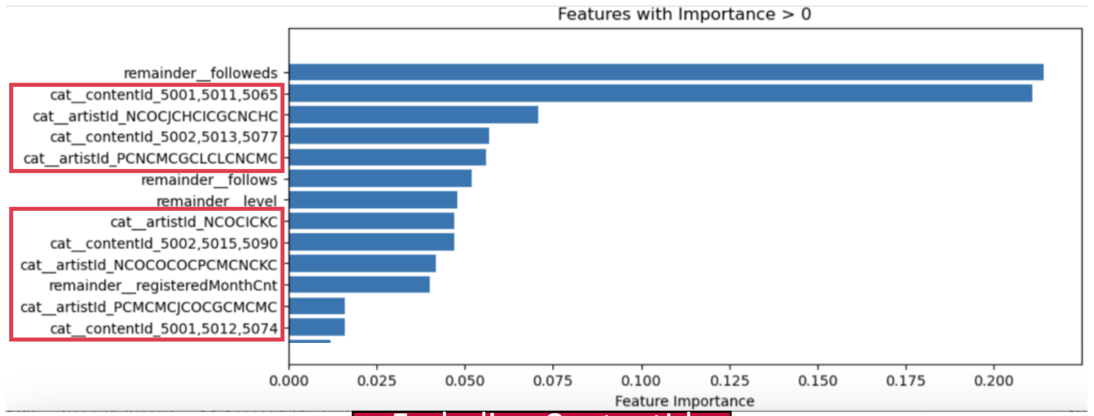
Two definitions of popularity

1. A card can be visibly popular to the user by displaying high number of likes.
2. A card can be clicked by many other users, which is not visible to the user.

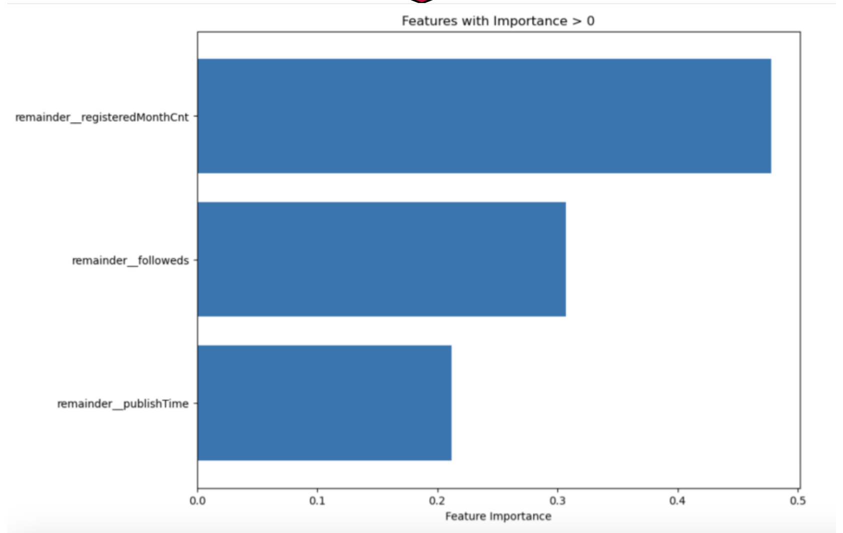
- To answer this question the running total of user interactions with cards until a given impression day was calculated. Prior probabilities of user liking or clicking on a card are derived from these running totals.
- Merging probabilities with card popularity data enables examining their influence on user clicks via simple linear regression.
- This dynamic approach aims to understand user behaviour by considering historical interactions with content.
- Non-visible measure, prior clicks, significantly influences the likelihood of a user clicking on a card (Definition 2) compared to the visible likes (Definition 1).
- Users may prefer content with quiet engagement, suggesting a deeper level of interest.
- Regression model shows a notable coefficient value for clicks, emphasizing the impact of prior clicks.
- Despite the analysis's simplicity, it provides a valuable perspective on factors capturing user attention and interest on the platform.

Which cards are recommended most frequently?

The frequency of card recommendations on the platform is notably influenced by both the Content ID and Artist ID.

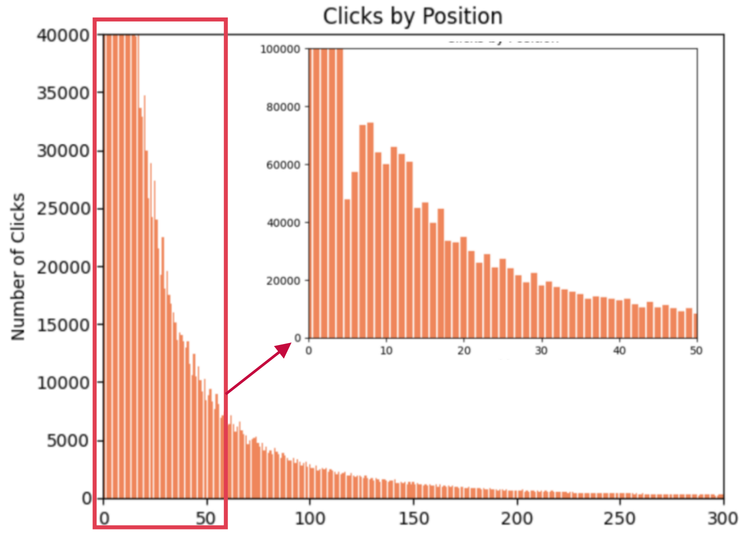


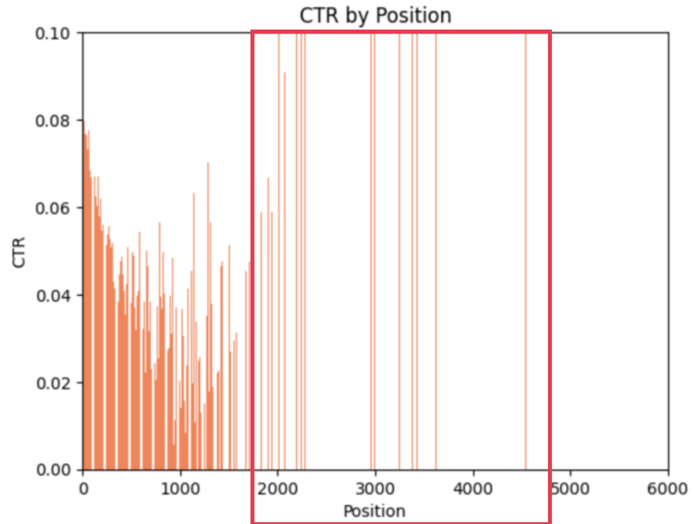
Excluding ContentId and ArtistId



- The initial step involved calculating the daily average number of impressions for each card to establish a baseline for recommendation frequency.
- A decision tree was constructed to explore attributes influencing recommendation frequency.
- The frequency of card recommendations is influenced by both Content ID and Artist ID, suggesting associations with historically popular content or artist.
- Creator popularity, measured by follower count, directly influences recommendation frequency.
- Card recency is crucial, with newer cards recommended more frequently for user engagement.
- The decision tree analysis provides insights into content type, creator popularity, card tenure, and multimedia elements influencing recommendation frequency.

Is there a discernible connection between the number of clicks and the recommended position of the card? Cards positioned from 1 to 10 exhibit a higher Click-Through Rate (CTR) compared to cards in other positions.



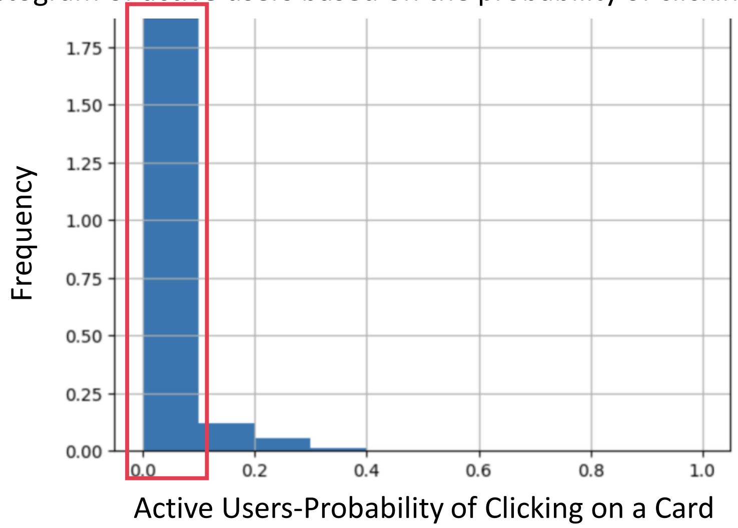


- There's a noticeable trend where the top positions in the app accumulate the highest number of clicks, implying a correlation between click count and card position. This aligns with user behaviour expectations, as users may be less inclined to scroll to lower positions and could lose interest if desired content is not found quickly. Beyond position 15, there's a significant drop in number of clicks.
- The Click-Through Rate (CTR), which factors in the number of impressions, reveals a similar trend but with nuances. While top positions still dominate, certain non-top positions exhibit unexpectedly high click rates, possibly indicating users actively searching for content they find appealing. Revealing a discrepancy, the current recommendatory system appears to be overlooking certain user preferences and failing to showcase content that aligns with their interests.
- Click-Through Rate (CTR) metric allows for a more meaningful comparison between different cards or positions, even when they have varying levels of exposure. Cards with higher CTRs are not just attracting more clicks, but they are doing so in relation to the number of times they were presented to users.

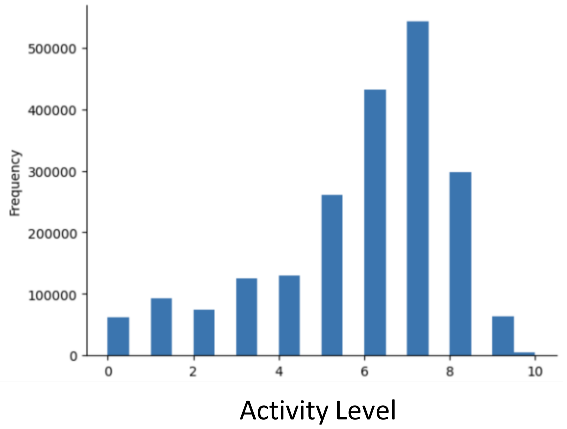
How is an active user defined?

87% of users are inactive on the cloud village section, which could indicate that the cards that are being recommended are not interesting for the users.

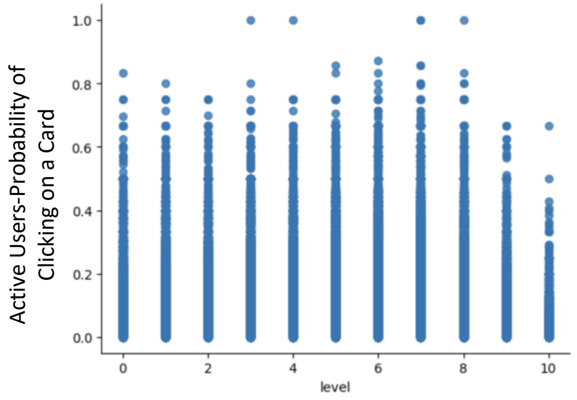
Histogram of active users based on the probability of clicking on a card



Histogram of User Activity Level on the app

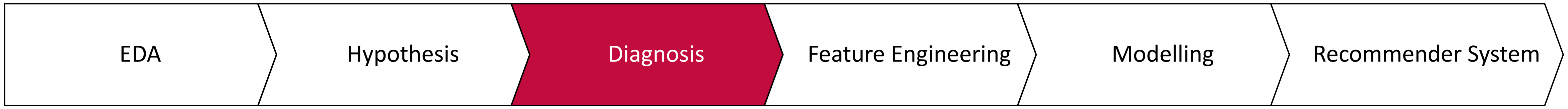


Scatter plot of activity level vs active users



- Active users, are defined by the probability of clicking on a Cloud Village card, exhibit inactivity when they have zero average probability of card clicks.
- Calculating the current click probability involves grouping users and dividing clicks by impressions; however, 87% of users are inactive on the Cloud Village, suggesting uninteresting recommended cards.
- The dataset's Activity Level variable categorizes users from 0 to 10 based on overall app activity, with the majority having a level above 5, indicating high overall app activity.
- There is no discernible correlation between the active users in the Cloud Village interface and the activity level on the entire platform, highlighting a potential area for improvement.
- Despite high overall app activity, users show low activity on the Cloud Village Tab, indicating a need for optimization in the current recommender system.

Identifying the main problem is crucial to propose a new approach for the recommendation system



- Identify the main problem.
- New focus of recommendation system

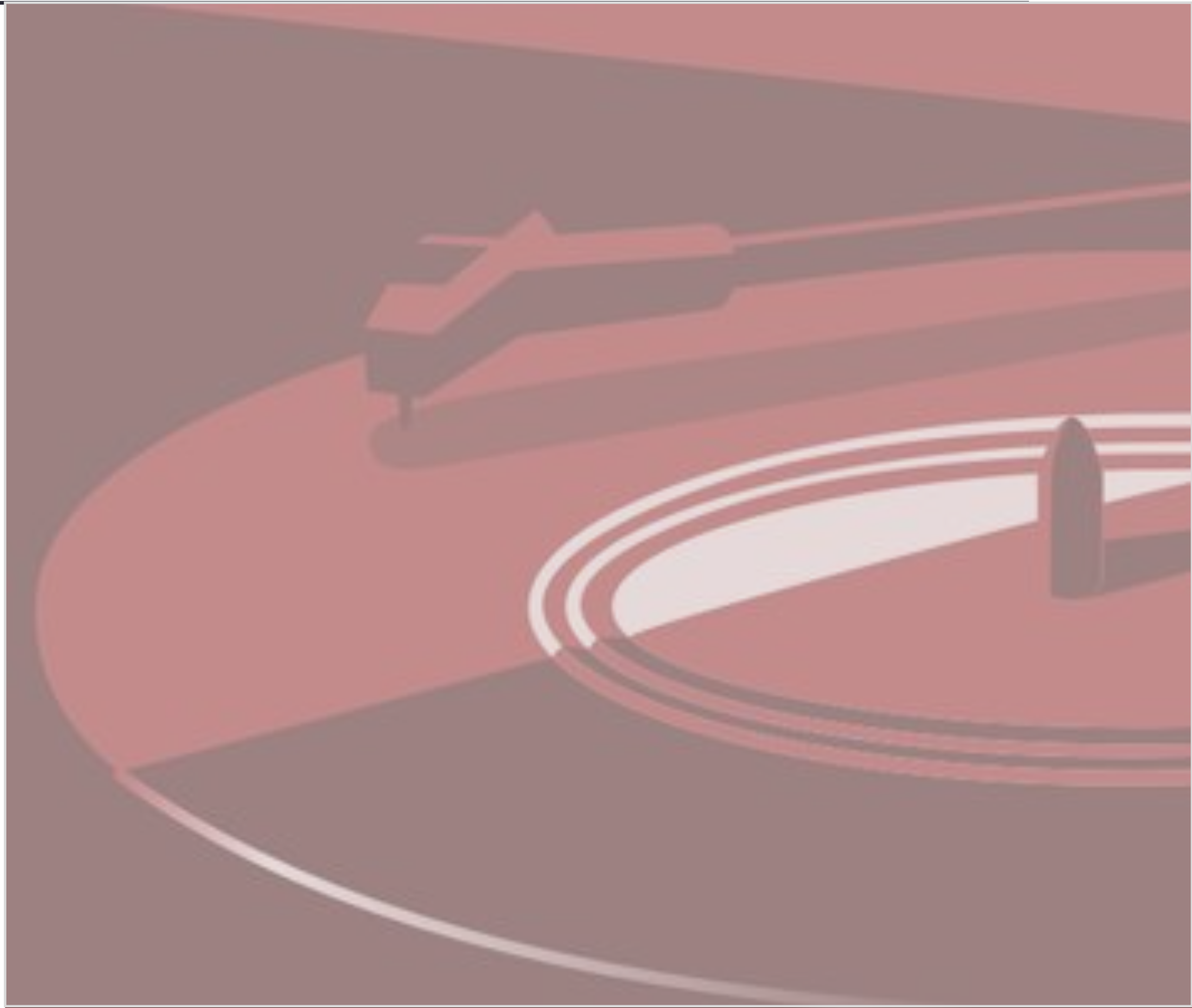
Recommender system redesign is needed with a personalized approach, emphasizing content relevance to individual musical tastes rather than prioritizing popular creator cards.

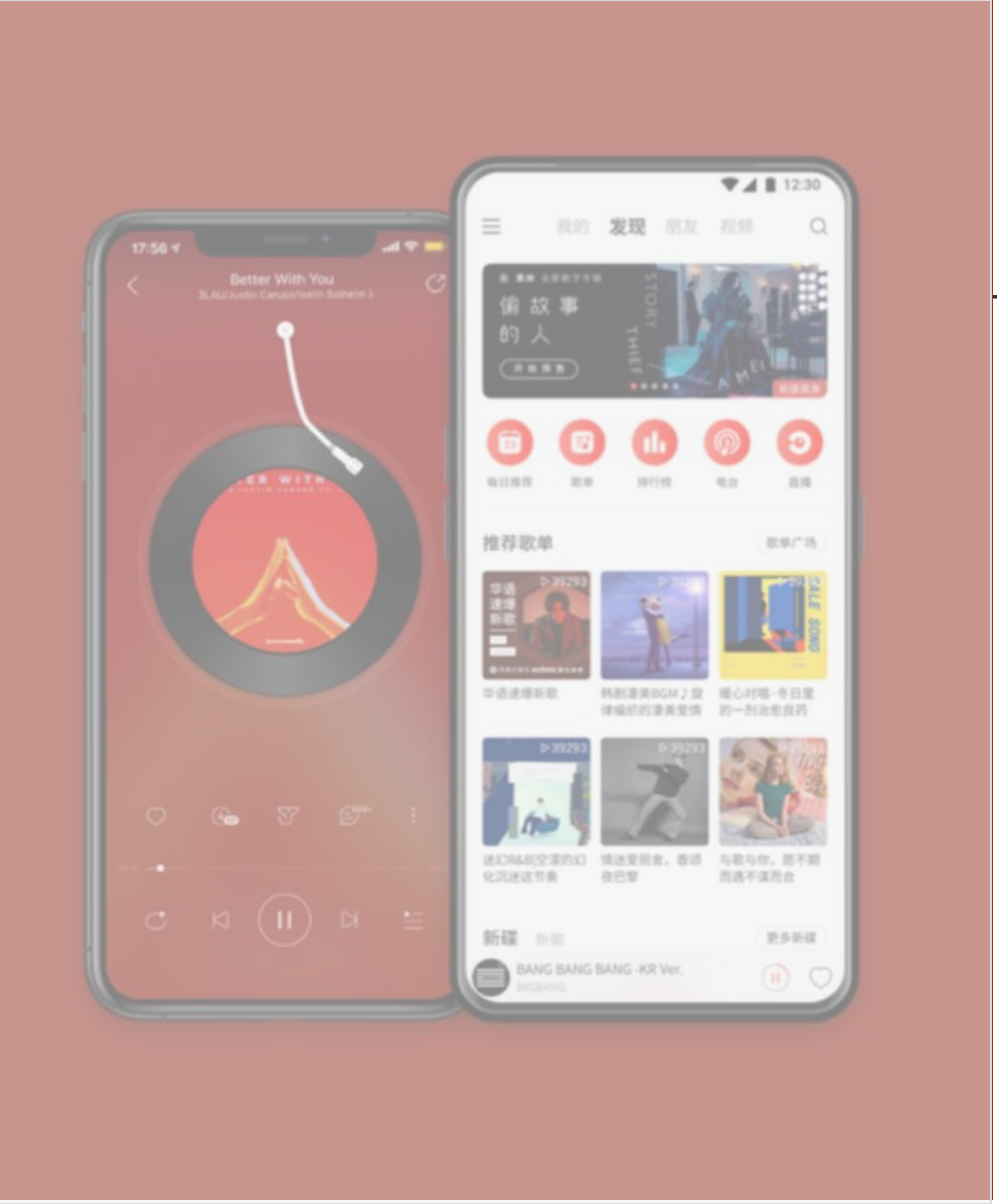
The exploration of data tables and app user activity dynamics has revealed a noticeable gap between the supply of cards recommended and the demand reflected in users' activity.

Current recommendation system appears to prioritise "popular" creator cards, resembling a social media approach rather than catering to music application goals.

Maximizing active users requires a strategic re-evaluation to encourage user interaction and provide a more personalized and engaging music discovery experience.

If the recommendation system is well-constructed, users should find satisfactory content within the top recommended positions.





The ultimate objective is to increase the number of active users in the Cloud Village Tab

Our hypothesis is that a more effective approach would involve prioritizing content relevance to individual users to boost click-through rates.

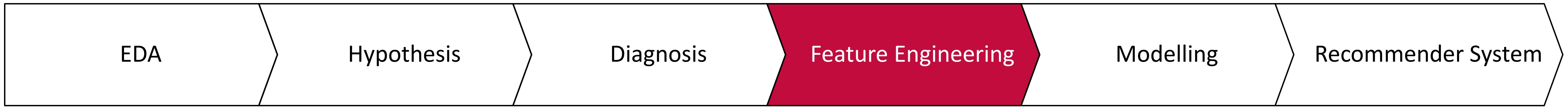
By enhancing click-through rates in the Discovery subtab, we can augment the number of active users in the Cloud Village Tab.

The focus is to rebuild a recommendatory system based on rankings

Rank the cards for each user based on the predicted probabilities in descending order. The higher the predicted probability, the higher the rank

Based on our findings in the Exploratory Data Analysis (EDA), it is advisable to select only the top 6 predicted probabilities for each user-card pair.

The feature engineering step prepares the data for modelling.



- Prepare data for modelling.
- Dimensionality Reduction.

The complete Feature Engineering is included in the “Final Project EDA” Jupyter Notebook. In the next slides, an overview is provided.

The aim of this stage is to construct a comprehensive database to use as input for recommender system

Constructing the master database

Impressions

- Impression Day
- Impression Hour
- Impression Position

User & Creator Demographics

- User Gender
- User Province
- User Tenure
- User Age
- User is Creator
- Creator Gender
- Creator Tenure
- Creator Level
- Creator Follows
- Creator Followeds

Card Demographics

- Content Id
- Tallk Id
- Type
- Publish Time

Past performance/preferences

- Prior User Probability Click
- Prior User Probability Like
- Prior User Probability Share
- Prior User Probability Comment
- Prior User Probability Swipe
- Prior User Probability Check Creator
- Prior User Average View Duration
- Prior Card Probability Click
- Prior Card Probability Like
- Past Preferred Artist Probability Click
- Past Preferred Song Probability Click
- Past Preferred Creator Probability Click

- Numerical Variables
- Categorical Variables

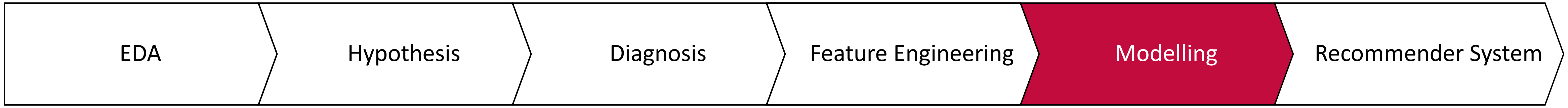
Transforming variables to calculate Past Performance and Preferences

- **Historical Engagement Tracking**: Assessed user engagement by tracking historical interactions-clicks, likes, shares, and views-with each content card. Calculated 'prior_probClick' and 'prior_probLike' by dividing accumulated interactions by the total number of card impressions up to the day before the current impression, indicating past card performance.
- **Historical Preferences Tracking**: Similarly, the prior probability of click towards card contents, artists and other songs, was calculated to represent the users' past preferences.

- **Integrating with Impressions Dataset:** Merged these historical engagement and preference probabilities with the 'impressions' dataset. This integration provides a comprehensive view of each card's performance, which is essential for understanding patterns in user behaviour and preferences over time.

- **Feature Set Enhancement for Recommendation System:** Integrated the new metrics into the recommendation system. These metrics enables the system to leverage historical user behaviour and preferences to anticipate and enhance future user engagement.

Building upon the insights gained from the EDA and using the master database, we embark on the modelling

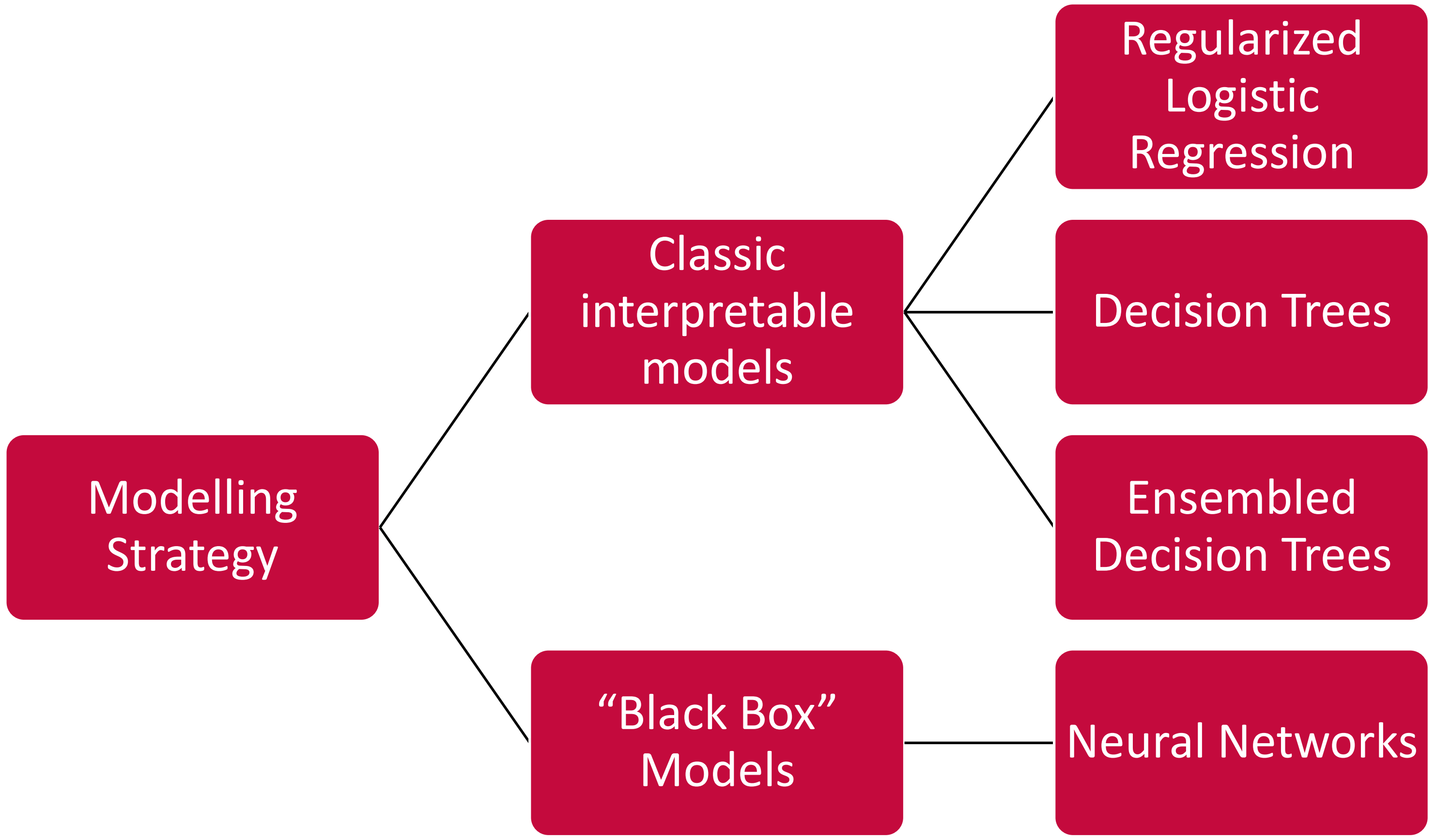


- Propose an analytical approach.
- Build ML models

The modelling phase is useful to unravel the factors influencing user activity on the Discovery subtab

Definition for Active	• User interaction with a card from the Discovery subtab, specifically through a click
Unit of Analysis	• Impressions on the Discovery Subtab within the Cloud Village Tab, with underlying user, card and creator demographics
Target Variable	• The binary **isClick** variable extracted from the impressions dataset
Output of the Models	• The propensity of a user to click on a card from the Discovery subtab, which is used to predict the binary isClick target variable

The goal is to establish the infrastructure for a recommendation system that tailors and prioritizes card recommendations based on the user's likelihood to click on a given card.



<u>Model Selection</u>

- Interpretability and effectiveness in conveying insights to the NCM management team.

- Chosen as a benchmark for its prowess in handling classification problems and capturing intricate relationships in high-dimensional spaces

Pre-Modelling Setup: Three main measures were taken: a random sub-sample was used, variables were treated, and the data set was balanced regarding the target variable.

Random sub-sample

- **Random sub-sample of 5 million impressions is considered:** Due to the computational complexity of processing the entire impressions dataset on personal laptops, a strategic decision was made to extract a sub-sample.

- **Sub-sample was validated for representativeness** to ensure balancing between computational feasibility and maintaining data integrity for meaningful analysis. A comparison of summary statistics between the original dataset and the sub-sample.

Treating Variables

Categorical and Numerical Variables were distinctly identified.

- **Categorical Variables**: To mitigate noise and computational challenges associated with a high number of distinct categories, categories representing less than 0.5% of the total sample data were grouped. Specifically, talkId, contentId, and user province were re-categorized.

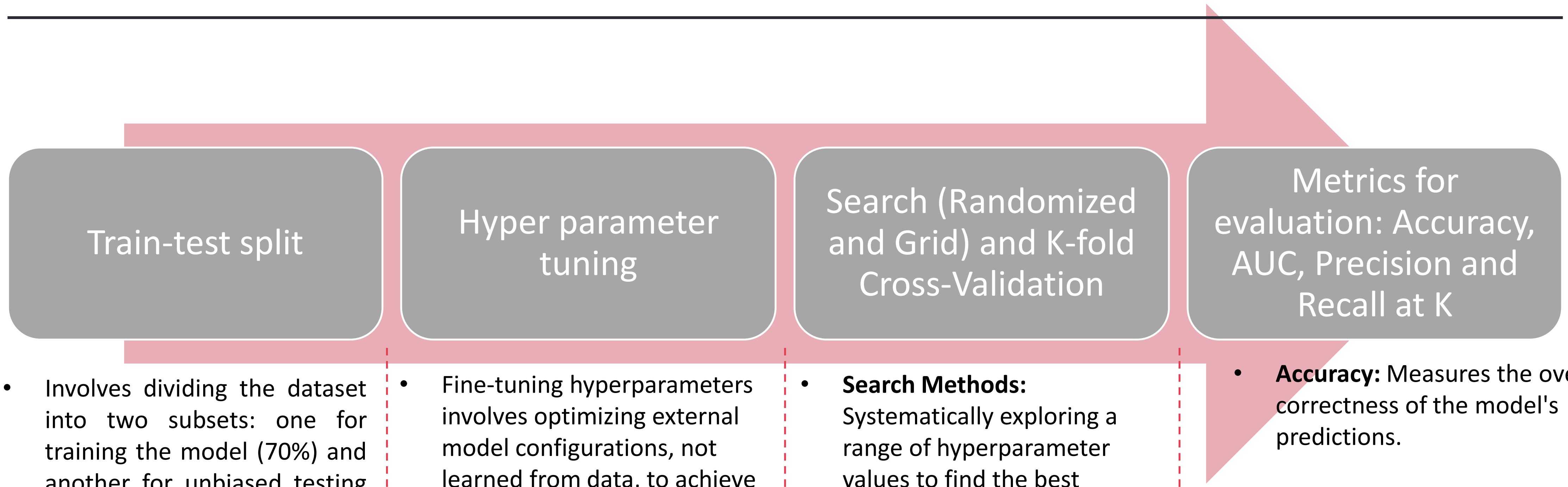
- **Numerical Variables:** Normalization was performed on numerical variables to avoid skewing results towards the higher scaled variables

Balancing Train Data

Two approaches were considered to mitigate the minority class (isClick=1) representing <5%.

- **Oversampling**: Done by duplicating random samples of the minority class entries in the train data
- **Re-weighting:** Adjusts the influence of each class during training, assigning higher weights to the minority class to address imbalances and enhance model performance. Implemented using the built-in functionality (such as 'class_weight' parameter) in Python algorithms/

Analytics Approach :The four models were executed employing a consistent methodology


Train-test split
Hyper parameter tuning
Search (Randomized and Grid) and K-fold Cross-Validation
Metrics for evaluation: Accuracy, AUC, Precision and Recall at K

- Involves dividing the dataset into two subsets: one for training the model (70%) and another for unbiased testing (30%) of its performance.
- It helps assess how well the model generalizes to new, unseen data by providing a separate dataset for evaluation, avoiding overfitting.

- Fine-tuning hyperparameters involves optimizing external model configurations, not learned from data, to achieve the best performance.
- It ensures a balance between underfitting and overfitting by adjusting settings like step size or tree depths.

- **Search Methods:** Systematically exploring a range of hyperparameter values to find the best combination
- **Cross-Validation:** Dividing the train dataset into 'K' subsets, training the model on 'K-1' subsets, and validating on the remaining one, repeating this process 'K' times.

- **Accuracy:** Measures the overall correctness of the model's predictions.
- **AUC (Area Under the Curve):** Evaluates the model's ability to distinguish between positive and negative examples, particularly useful for binary classification problems.
- **Precision and Recall at K:** Focuses on the model performance among the top 'K' recommended items.

Neural Networks

Hyperparameter Tuning & Fitting: Done in 'Neural Network Hyperparameter Tuning - Resampling' and 'Neural Network Hyperparameter Tuning – Reweighing' Jupyter Notebooks

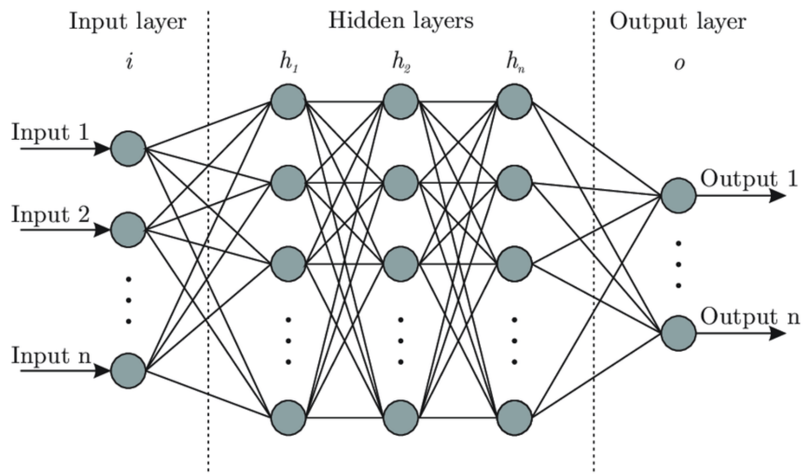


Figure Source: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed

Neural Networks include many hyperparameters to tune. In our analysis we searched for the optimal combination of:

- Number of Hidden Layers: 1, 2, and 3
- Number of Neurons/Nodes per Layer: 5, 10, and 15
- Batches Size: 100K, 250K, and 500K
- Epoch/Iterations: 20, 40 and 60

Loss Function: Binary Cross Entropy
Optimiser: Stochastic Gradient Descent

A **grid search** and **5-fold cross-validation** was utilised to find the set of parameters that optimised AUC and accuracy, with higher emphasis on optimising AUC.

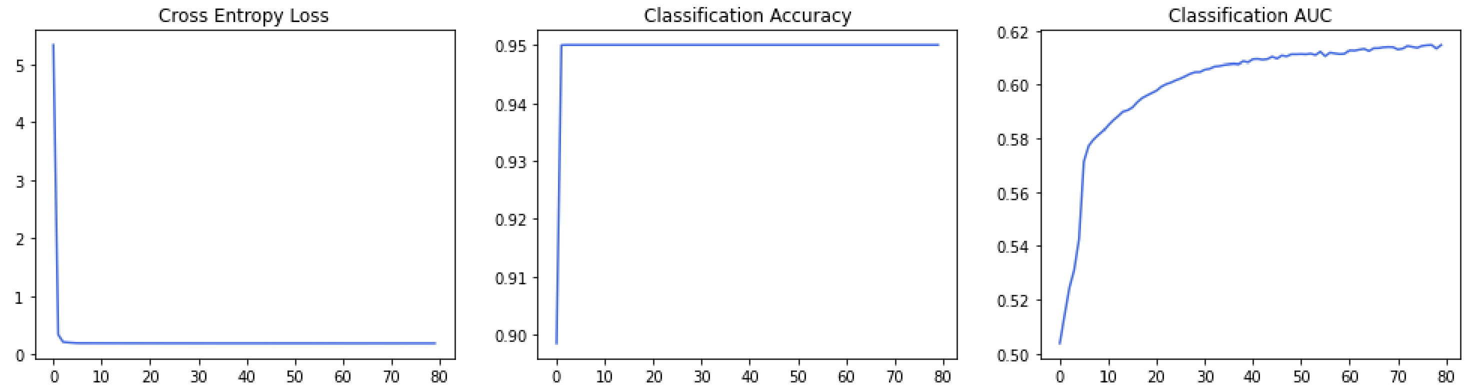
Neural Networks were the first model in our testing, and during the hyperparameter tuning phase, we utilized the two candidate data balancing techniques: re-sampling and re-weighting. This allowed us to gauge the impact of these techniques on both model performance and computational efficiency. In doing so, we saw no material difference in performance but significant increase in computational complexity (run time) associated with re-sampling given that the data size had increased with oversampling.

Neural Networks

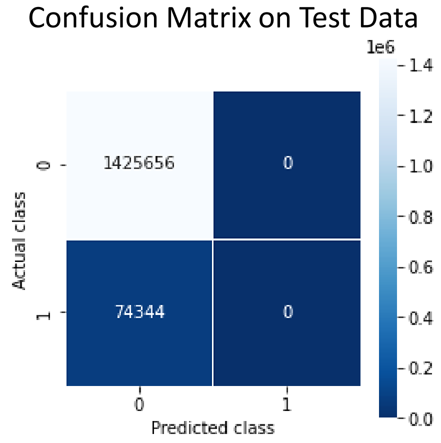
Results using Reweighing Balancing Technique: Presented in 'Final Project Modelling' Jupyter Notebook

Optimal Parameters: {'batch_size': 100000, 'epochs': 80, 'layers': 1, 'learning_rate': 0.01, 'neurons': 15}

Examining the following chart, exhibiting the model performance over the iterations/epochs, the loss function and accuracy converge quickly, while the AUC takes longer to approach convergence



Despite the initial promise high accuracy ratio and relatively good AUC, deeper analysis of the confusion matrices on the full-train and test datasets uncover concerning patterns on the performance of the Neural Network.



The model exhibits a **complete inability to accurately predict instances of the "positive" class**, where isClick equals 1. Instead, it uniformly assigns all instances to the majority class (isClick equals 0), resulting in 0 positive recall.

Note: This was the case using both data balancing techniques. Testing of allowing higher epochs was conducted with no improvement.

Regularized Logistic Regression

Hyperparameter Tuning & Fitting: Done in 'Logistic Regression Hyperparameter Tuning - Reweighing' Jupyter Notebook

Re-weighting Balancing Technique was used:

- Primarily to reduce computational complexity, since the process of duplicating entries during oversampling significantly increases the sample size, thus significantly increasing computational complexity as was observed in training the Neural Networks
- Secondly, we believe that oversampling, by its nature, might result in a loss of unique information from the original minority class samples, potentially compromising the model's ability to accurately capture the genuine underlying patterns.

Regularization Penalty: L2 Penalty

Hyperparameter to tune: Regularization Parameter (C)
A range of logspace(-2, 1, 15) was tested, which includes 15 values between 0.01 and 10.

A **randomized search** was used instead of grid search to improve computational complexity, accompanied with **5-fold cross validation** was utilised.

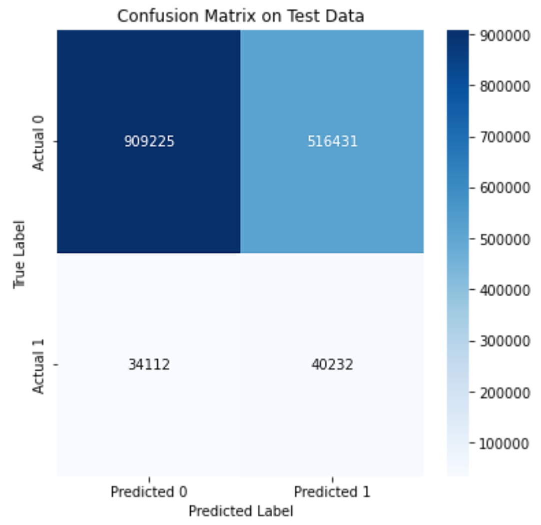
Optimal C = 0.02682695795279726

	Full Train Data	Test Data
Accuracy	63.35%	63.30%
AUC	63.12%	63.11%

✓ **Nearly identical accuracy and AUC scores** on train and test.

✓ **Ability to distinguish between positive class and negative class.**

Commendable Recall scores, striking a good balance between sensitivity (54%) and specificity (64%), on both train and test datasets.



Regularized Logistic Regression

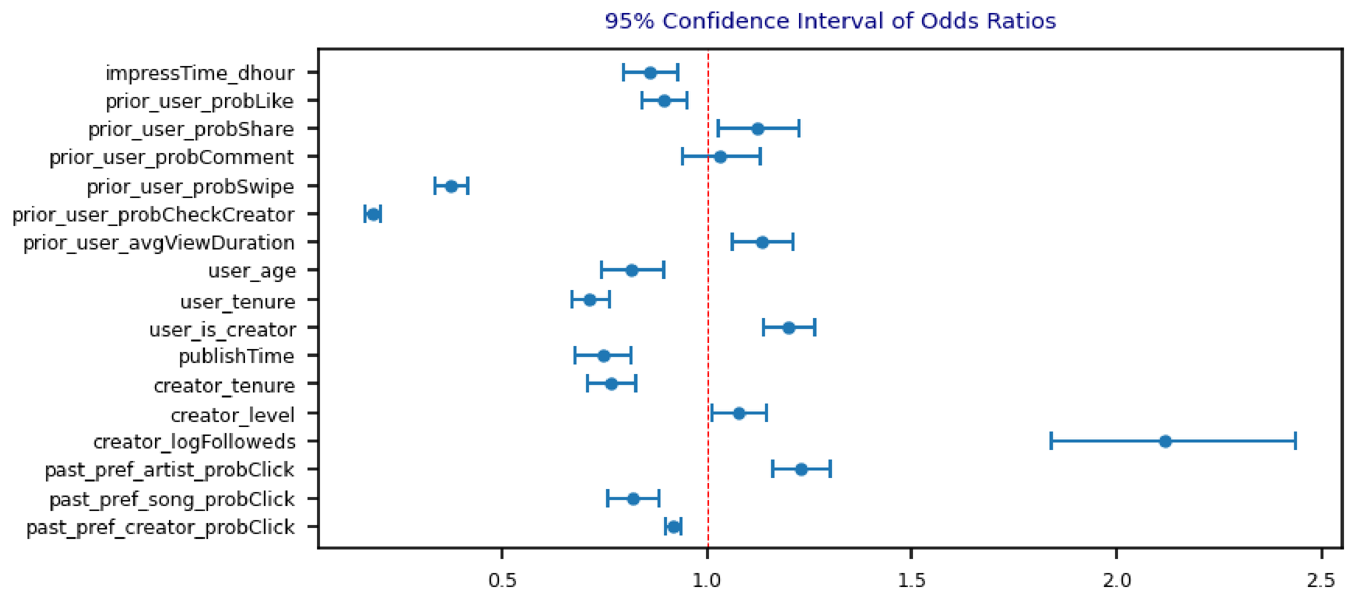
Exploring Feature Importance: Presented in 'Final Project Modelling' Jupyter Notebook

The logistic regression model not only demonstrates strong performance but also offers a window into understanding features influence.

Given the dataset's significant dimensionality, statistically significant features were identified using a criterion: *if the p-value is less than or equal to 0.05, the feature is considered statistically significant.*

An odds ratio (exponential of the coefficient) was calculated to aid in analyzing whether a feature would increase (odd ratio >1) or decrease (odd ratio <1) the click probability. To gain insights, a graph illustrating the confidence interval of each feature's odds ratio was employed.

A sample of the graph featuring numerical and binary variables is presented below.



Regularized Logistic Regression

Insights

Users with a **longer average view duration** on previous impressions are more likely to click.

Age Influence
Younger users exhibit a higher probability of clicking on a card.

Content Genre Importance
Content genre, represented by Content ID and Talk ID, strongly influences click probability. Some IDs positively reinforce click probability, while others have a negative impact. Further analysis of these important features is recommended for NetEase to understand user preferences.

Cards created by **more active and popular creators** attract more clicks, indicating the **influence of experienced creators.**

New Users Engagement
Users who recently joined the application are more prone to clicking on a card

Users who are also creators are more likely to click, suggesting they may seek inspiration or explore the platform

Card Age Impact
Older cards receive fewer clicks compared to new cards

Peculiar Observations:

Not all prior user activities behave similarly. For instance, high prior activity in sharing and commenting correlates with a higher probability of clicking, while the impact is opposite for liking and commenting.
Collinearity or a shift in user behavior towards exploring preferred content in the Follow subtab could explain these variations.

Decision Tree

Hyperparameter Tuning & Fitting: Done in 'Decision Tree Hyperparameter Tuning - Reweighing' Jupyter Notebook

Re-weighting Balancing Technique → Same as used in the logistic regression

Information Criterion: Entropy

Hyperparameter to tune:

1. Maximum tree depth (max_depth): A range between the array np.arange(5, 30, 5) was tested. The final optimal value was 5
2. Minimum samples to split (min_samples_split): The values 50, 100, 200 and 500 were tested. 200 had the best performance

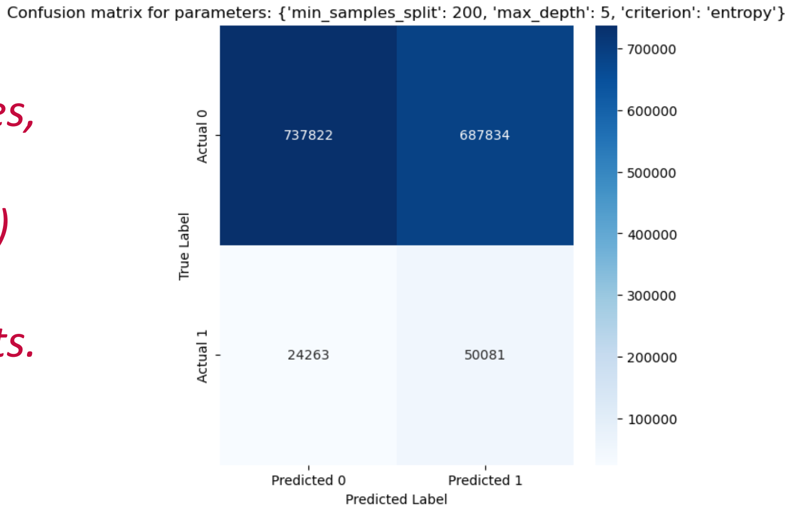
A **randomized search** was used instead of grid search to improve computational complexity, accompanied with **5-fold cross validation** was utilised.

Optimal max_depth = 5; min_samples_split=200, criterion = entropy

	Full Train Data	Test Data
Accuracy	52.47%	53%
AUC	64.10%	64%

- ✓ **Again, similar level between training and test scores**
- ✓ **High level of interpretability of feature importance. Fastest model to train**
- ✓ **Ability to distinguish between positive class and negative class.**
- ‼ **Accuracy decreased in comparison with the logistic regression**

Commendable Recall scores, striking a good balance between sensitivity (52%) and specificity (67%), on both train and test datasets.



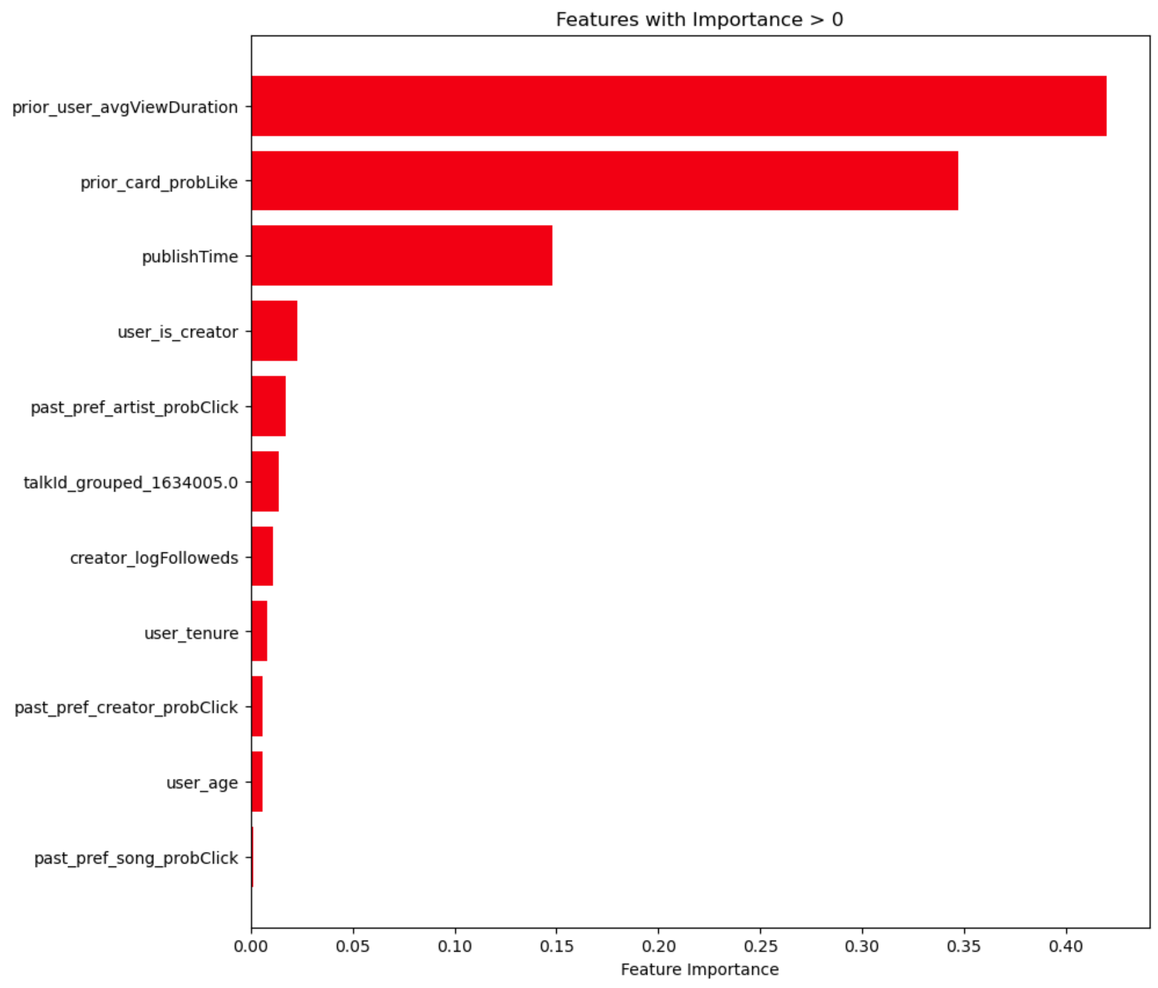
Decision Tree

Exploring Feature Importance: Presented in 'Final Project Modelling' Jupyter Notebook

The Decision tree model performed very similarly to the logistic regression model. The accuracy was considerably lower, while the AUC was only slightly superior.

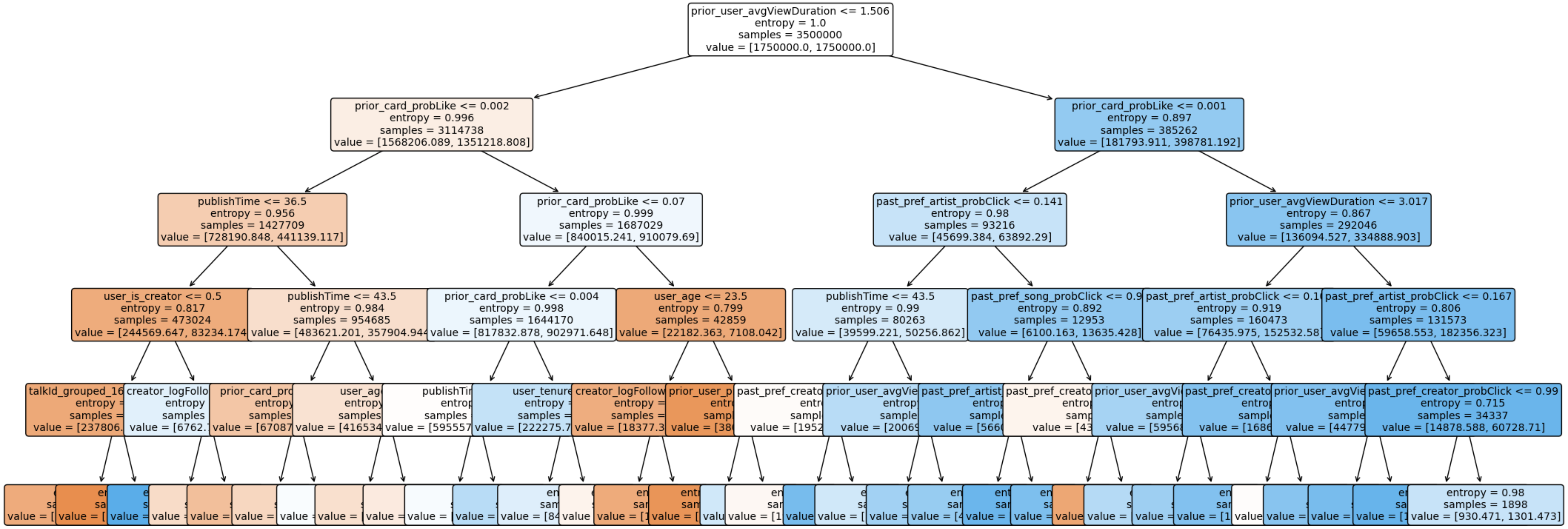
The final best tree was one with a low maximum depth, meaning that the simpler trees were favored in the validation stage. This is a reasonable result given the high level of complexity of the dataset.

The features that were considered most important are shown in the image on the right. The three top features are very predominant over the rest. We can also gain some insight on how the feature are influencing the probability of the click by inspecting the top splits of the decision tree



Decision Tree

Exploring Feature Importance: Presented in 'Final Project Modelling' Jupyter Notebook


prior_user_avgViewDuration <= 1.506
entropy = 1.0
samples = 3500000
value = [1750000.0, 1750000.0]
prior_card_probLike <= 0.002
entropy = 0.996
samples = 3114738
value = [1568206.089, 1351218.808]
prior_card_probLike <= 0.001
entropy = 0.897
samples = 385262
value = [181793.911, 398781.192]
publishTime <= 36.5
entropy = 0.956
samples = 1427709
value = [728190.848, 441139.117]
prior_card_probLike <= 0.07
entropy = 0.999
samples = 1687029
value = [840015.241, 910079.69]
past_pref_artist_probClick <= 0.141
entropy = 0.98
samples = 93216
value = [45699.384, 63892.29]
prior_user_avgViewDuration <= 3.017
entropy = 0.867
samples = 292046
value = [136094.527, 334888.903]
user_is_creator <= 0.5
entropy = 0.817
samples = 473024
value = [244569.647, 83234.174
publishTime <= 43.5
entropy = 0.984
samples = 954685
value = [483621.201, 357904.944
prior_card_probLike <= 0.004
entropy = 0.998
samples = 1644170
value = [817832.878, 902971.648]
user_age <= 23.5
entropy = 0.799
samples = 42859
value = [22182.363, 7108.042]
publishTime <= 43.5
entropy = 0.99
samples = 80263
value = [39599.221, 50256.862]
past_pref_song_probClick <= 0.9
entropy = 0.892
samples = 12953
value = [6100.163, 13635.428]
past_pref_artist_probClick <= 0.1
entropy = 0.919
samples = 160473
value = [76435.975, 152532.58
past_pref_artist_probClick <= 0.167
entropy = 0.806
samples = 131573
value = [59658.553, 182356.323]
past_pref_creator_probClick <= 0.99
entropy = 0.715
samples = 34337
value = [14878.588, 60728.71]
entropy = 0.98
samples = 1898
value = [930.471, 1301.473]

Decision Tree
Insights

Again, users with a **longer average view duration** on previous impressions are more likely to click.

Age Influence
Although age is a factor here, now it's not as relevant as it was with logistic regression.

New Users Engagement
This feature was also relevant for the decision tree. This speaks to the importance of engaging users early on!

Content Genre Importance
Only one talkId was a relevant factor for the decision tree.

Cards created by **more active and popular creators** attract more clicks, indicating the **influence of experienced creators.**

Again, users who are also creators are more likely to click.

Card Age Impact
Older cards receive fewer clicks compared to new cards

Peculiar Observations:

It is interesting how some features that were highly important in the logistic regression model were not considered in the tree. This will be explored further in the next slides.

Random Forest Classifier

Hyperparameter Tuning & Fitting: Done in 'Random Forest Hyperparameter Tuning - Reweighing' Jupyter Notebook

Re-weighting Balancing Technique → Same as used in the logistic regression

Hyperparameter to tune:

1. Number of estimators (n_estimators): Two values were tested: 50 and 100
2. Maximum tree depth (max_depth): 5, 10 and 20 were tested

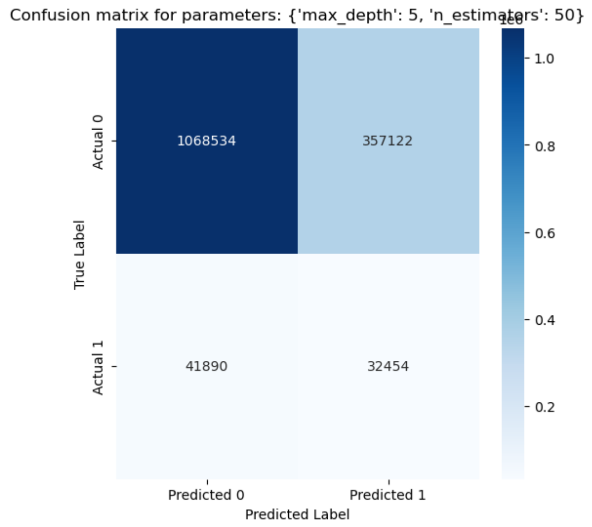
A **randomized search** was used instead of grid search to improve computational complexity, accompanied with **4-fold cross validation** was utilised. Here the computational limitations were very constraining, which forced us to limit our search space significantly.

Optimal max_depth = 5; min_samples_split=200, criterion = entropy

	Full Train Data	Test Data
Accuracy	73.34%	73%
AUC	64.59%	65%

✓ **High level of interpretability of feature importance.**
✓ **Ability to distinguish between positive class and negative class.**
‼ **Model took many hours to run with simple parameters**
‼ **Although it has the best accuracy and AUC, it also achieves that by the higher prediction of class 0, not the class of interest**

This model has the worse balance between sensitivity (75%) and specificity (44%), of the tree models tested

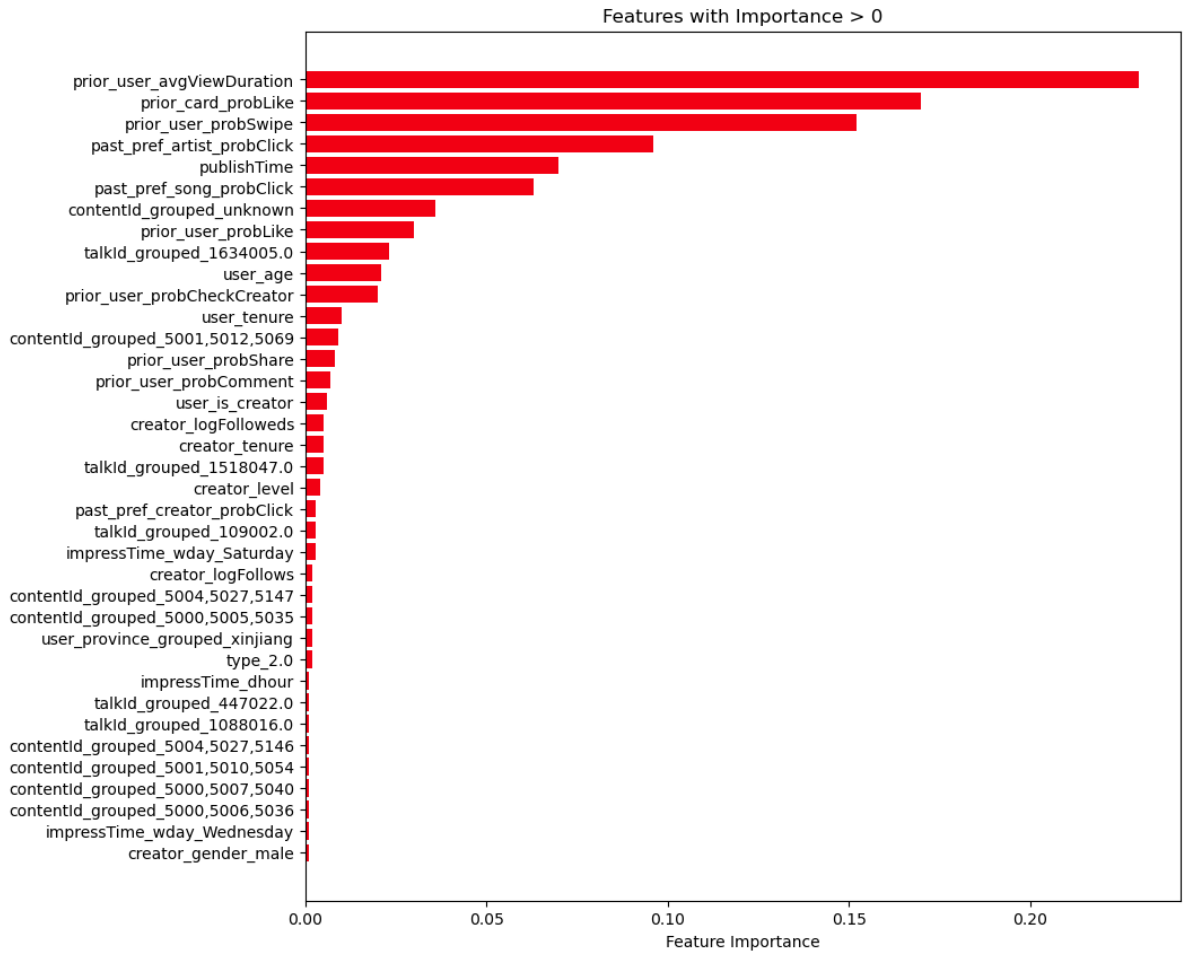


Random Forest Classifier

Exploring Feature Importance: Presented in ‘Final Project Modelling’ Jupyter Notebook

The Random forest classifier had a slightly better performance then the other models in terms of accuracy and AUC. However, as it is expected it goes over more features through the bagging process, and thus has more features with an importance > 0.

Unlike what we observed with the decision tree, there is no predominance of a few features, and the top features also got slightly shuffled. We will deepen the analysis of the features importance in the upcoming slides



Random Forest Classifier

Insights

Again, users with a **longer average view duration** on previous impressions are the most likely to click.

Age Influence
Although age is a factor here, it is again not as relevant as it was with logistic regression.

Swiping probability
In this model, the swiping probability appears as one of the most important features. This feature did not come up at all with the decision tree

Content Genre Importance
Unlike with the decision tree, and similar to Logistic Regression, now several talkId groups are relevant.

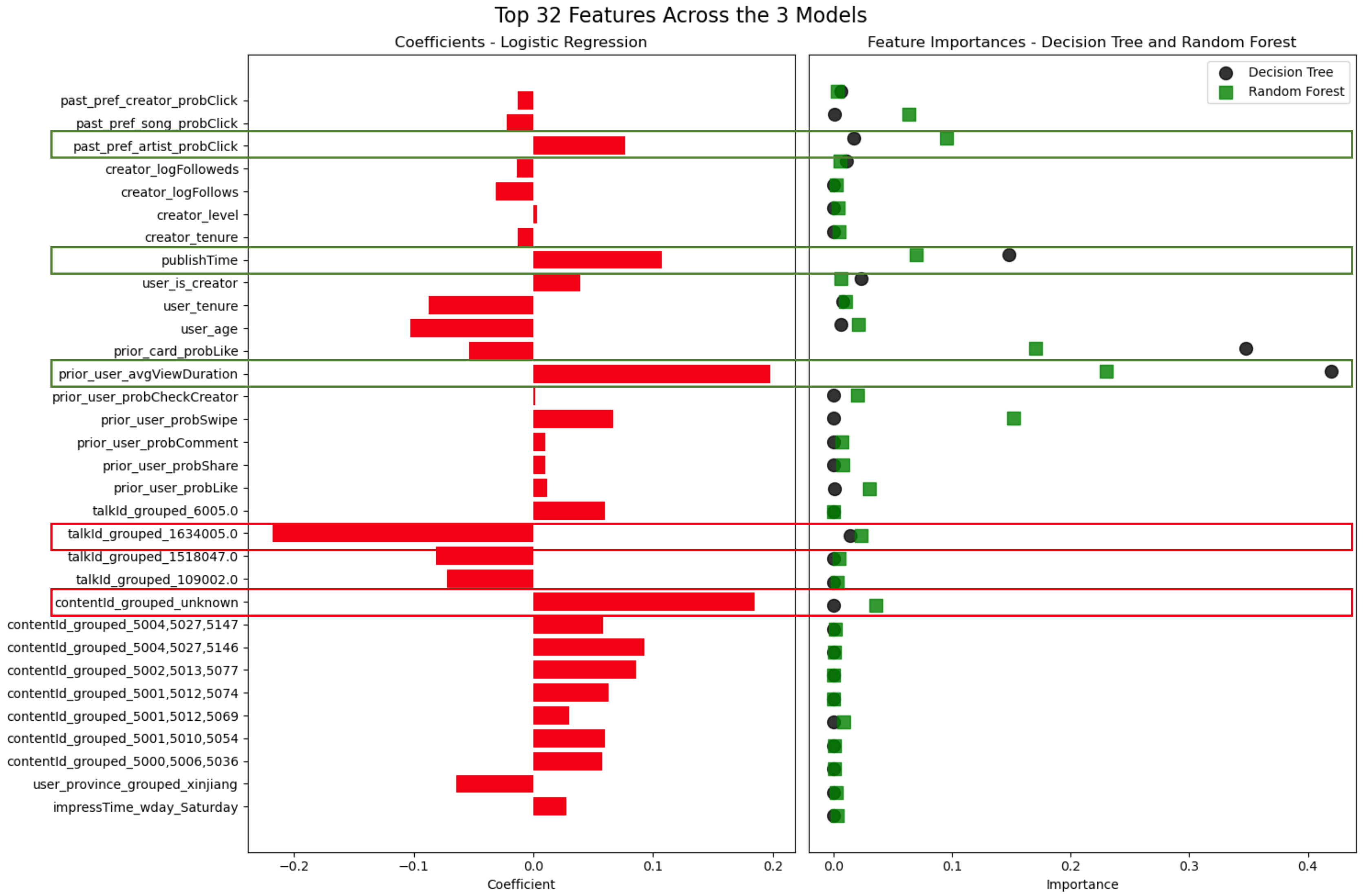
Cards created by **more active and popular creators** attract more clicks, indicating the **influence of experienced creators.**

Card Age Impact
Older cards receive fewer clicks compared to new cards. This could potentially be a bias of the current recommendation system, if it's pushing the new cards to the top positions

Peculiar Observations:

This model had the best performance in terms of accuracy and AUC, but it sacrificed a lot of sensitivity for that. For this application, we don't really have any serious risks with a low precision or recall. Furthermore, given the fact that the 0 class is overrepresented, the better accuracy achieved by predicting the 0 class better is not something really relevant.

Card publishing times, past artist preferences, and prior user average time views are key factors influencing user engagement across all the models.



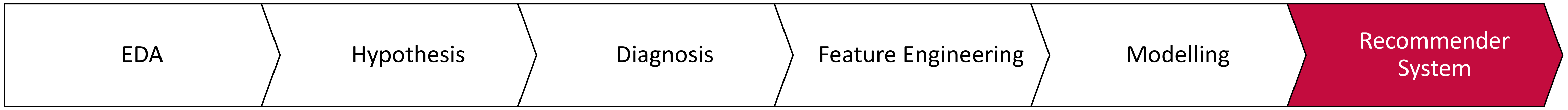
Similar importance across models

Publishing Time: Recommends creating a backlog of cards for simultaneous publication at optimal times or encouraging creators to align their card releases with peak engagement periods.
Past Preferred Artist: Suggests that users' historical preferences for specific artists significantly influence their likelihood to click, providing groundwork for developing a content-based filtering system in the future.
Prior User Average Time View: Emerges as a critical feature consistently impactful across all three models, underscoring its significance in predicting user engagement.

Difference in importance between models

Differences in **talkId_grouped_1634005** and **contentId_grouped_unknown** across models can be attributed to modelling characteristics. Logistic regression emphasizes linear relationships, while decision trees capture complex patterns and interactions. Sensitivity to multicollinearity and distinct criteria for splitting in decision trees contribute to variations in variable importance.

The conclusive phase involves presenting a recommendation system proposal while delving into its limitations and exploring avenues for further enhancements.



- Propose a Recommender System
- Limitations
- Improvements

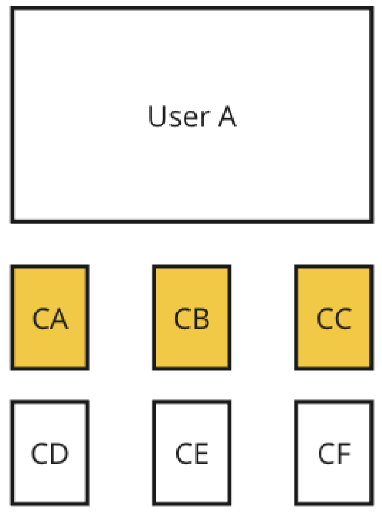
The proposed recommendation system operates akin to a preference-based matching network

The proposed recommendation system functions as a preference-based matching network, where user-card rankings are determined by predicted click probabilities from our most accurate predictive model. These rankings dictate the positions of card recommendations on the Discovery Subtab within the Cloud Village Tab.

Users	Card ID	Probability to Click
User A	Card A	0.92
User A	Card B	0.87
User A	Card C	0.80
User A	Card D	0.74
User A	Card E	0.61
User A	Card F	0.55
User A	Card G	0.50
User A	Card 8	0.49

*Based on the EDA, we selected the top 6 positions.

Recommender System Implemented Mockup



*The yellow cards are the cards that the user clicked in reality.

Model Selection

Precision at K and Recall at K: Computed and Presented in 'Final Project Recommender System Assessment' Jupyter Notebook

Precision at K and Recall at K are extensions to traditional binary classification metrics, precision, and recall. They become especially valuable in contexts where a user is provided with a list of K recommended items, and the objective is to assess how effectively these recommendations align with the user's preferences. Precision emphasizes the accuracy of the recommendations, emphasizing how many of the top recommendations are genuinely relevant. In contrast, recall emphasizes the system's capability to retrieve all relevant items, irrespective of the total number of retrieved items.

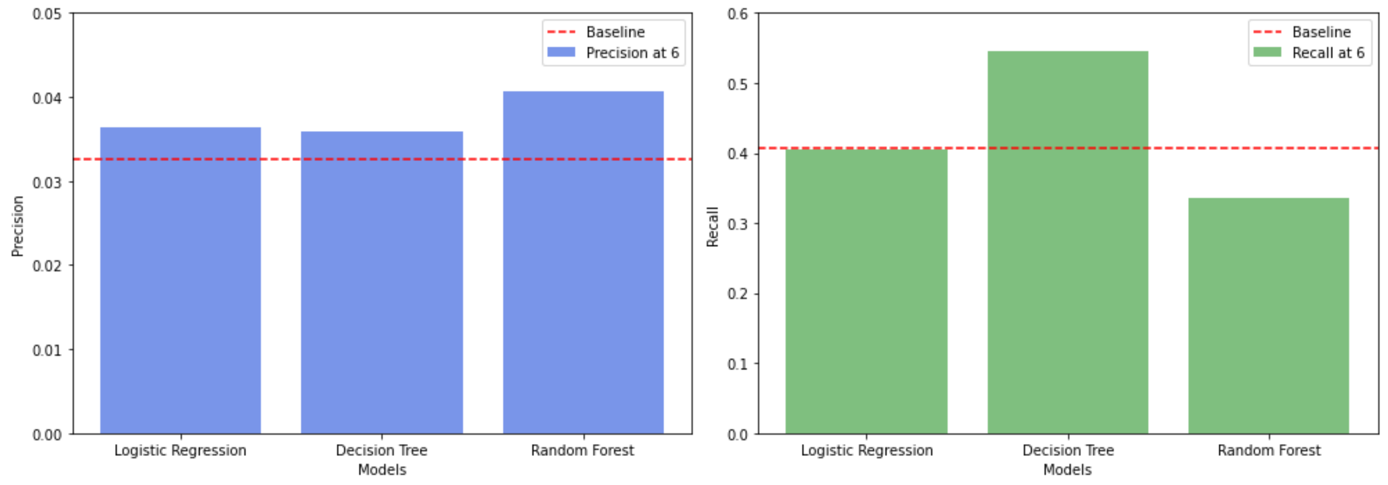
$$Recall\ at\ K = \#\ of\ recommended\ items\ at\ k\ that\ are\ relevant\ to\ the\ user / Total\ \#\ of\ relevant\ items\ to\ the\ user$$

$$Precision\ at\ K = \#\ of\ recommended\ items\ at\ k\ that\ are\ relevant\ to\ the\ user / \#\ of\ recommended\ items\ at\ k$$

Our goal is to identify the machine learning model that, if implemented as a recommendation system, would excel in accurately predicting the user's preferences for the top-recommended items.

Precision at K and Recall at K were calculated for the existing recommender system (baseline) based on the impressPosition feature, presented by the red horizontal lines.

Decision tree seems to strike the best balance between recall and precision, **surpassing the baseline performance** in both. Random Forests and Logistic Regression fall short in terms of Recall.



Useful Resource: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54

Limitations and Future Strategies

Limitations

- **Computational Constraints:** Limited capacity to explore certain models like SVM and to extensively search for optimal hyperparameters.
- **Data Scope and Sample Limitation:** Analysis restricted to a random sample of users and only one month of impressions data, constraining the depth and breadth of data transformation and historical analysis.
- **Data Quality Issues:** Encountered challenges with data quality, particularly with variables having predominantly unknown values, which might have affected the model's accuracy. Additionally, significant missing data in key variables like 'age and 'gender' necessitated imputation strategies, potentially impacting the accuracy of demographic analyses.
- **Anonymity of Key Identifiers:** The anonymity of content ID and talk ID limited the ability to perform a more in-depth analysis of model results and user-content interactions. This was compounded by the challenges in interpreting complex, anonymized, and categorical variables.

Future Strategies

- **Content-Based Filtering Enhancements:** Consider implementing content-based filtering methods that reduce the need for extensive user scrolling, similar to vertical scrolling interfaces like Spotify. This approach could enhance user experience and increase engagement with a wider variety of content.
- **Expanding Data Collection:** Broaden the dataset to include a more extended time frame and a larger, more representative user sample.
- **Overcoming Computational Barriers:** Explore ways to enhance computational capacity for experimenting with more complex models and extensive hyperparameter tuning.
- **Mitigating Recommendation Bias:** Plan to implement methods for unbiased data collection and algorithmic updates to capture a broader spectrum of user behaviors and preferences, counteracting the skew introduced by the existing recommendation system and enabling a more diverse and adaptive content discovery experience.