Predicting User's music listening habits using Gradient Boosted trees

Group Members: Tegan Ayers, Jiachen Liu, Yicheng Wang, Aldo Pioline Antony Charles

Summary
Several companies and services today, such as Spotify, Netflix and Amazon, rely on recommendation engines to retain customers and expose them to new content. One of the first steps in being able to provide recommendations to customers, however, is to first be able to predict their habits. As such, this project seeks to utilize various data points and metadata to predict the chances of a KKBOX user, a music service based in Asia, listening to a song more than once.

Specifically, our team utilized a Kaggle dataset that contained data such as user, song ID, genre and artist along with information about the platform and playlist that the song was played from. Moreover, metadata such as user language, city and registration data were employed as well. Each observation was labelled with a 0 or 1 indicating whether each song was played more than once within a month [1].

Due to the nature of the categorical dataset, several non-linear modeling approaches were evaluated. Specifically, our team compared the accuracy of a k-nearest neighbor (KNN) approach to several types of decision trees including a standard decision tree and two types of gradient boosted trees (XGBoost and CatBoost). Additionally, a logistic regression model was developed in order to determine how a linear model would perform with mostly nonlinear inputs. As expected, the linear logistic regression model performed the worst with an accuracy of 53.71%. Interestingly, the KNN approach performed the best with an accuracy of 66.13%. All of the tree models fell within 62.63 - 65.54% accuracy. Since this data was gathered from a Kaggle competition, we were able to see how well the first place winner of the competition performed. The top model was able to obtain 74.78% accuracy using a combination of Neural networks and LightGBM. The superior accuracy may partly be due to our team downsampling the dataset so that models could be run on our local machines in a timely manner.

Ultimately, this project allowed our team to gain exposure to cleaning messy and incomplete data, working with categorical data types and determining the advantages of various nonlinear models. While our model accuracies are not on par with the Kaggle competition winner, further exploration into feature engineering or other nonlinear models, such as support vector machines or neural networks, would be an appropriate next step to improve model performance.

Methods
Exploratory data analysis and data wrangling were done as a team so that each person was using the same dataset. Feature engineering and modeling were then performed by each individual and results were aggregated and compared for accuracy. More information on each step is provided in the subsections below.

I. Data Wrangling

After inspecting all the datasets, it was decided to only use data within a few csv's that were available (songs.csv, members.csv, train.csv). Additional data were for Kaggle competition submission purposes and didn't contain information about the target variable (sample_submission.csv, test.csv) or otherwise contained unreliable information (song_extra_info.csv) [1] and as such were eliminated. .

Included datasets were merged using a left join operation on primary keys. The combined dataset had over 7 million entries which was too much for our local machines to feasibly handle, therefore, it was decided to take a 50,000 random sample of the merged dataset. This should have preserved the characteristics of the original merged dataset.

Some features, such as gender and lyricist, had more than 40% null values while the age variable contained many outlier values and as such were removed. Remaining missing values were handled during categorical data encoding procedures. Finally, registration_date and expiration_data were replaced by registration_days (number of days as a member) as we believe this change better captured the magnitude of time. Relevant code for data wrangling is found in Appendix B.

II. Exploratory Data Analysis

Various analyses, including distribution plots, descriptive statistics and correlation matrices, were performed in order to determine relationships within the data.

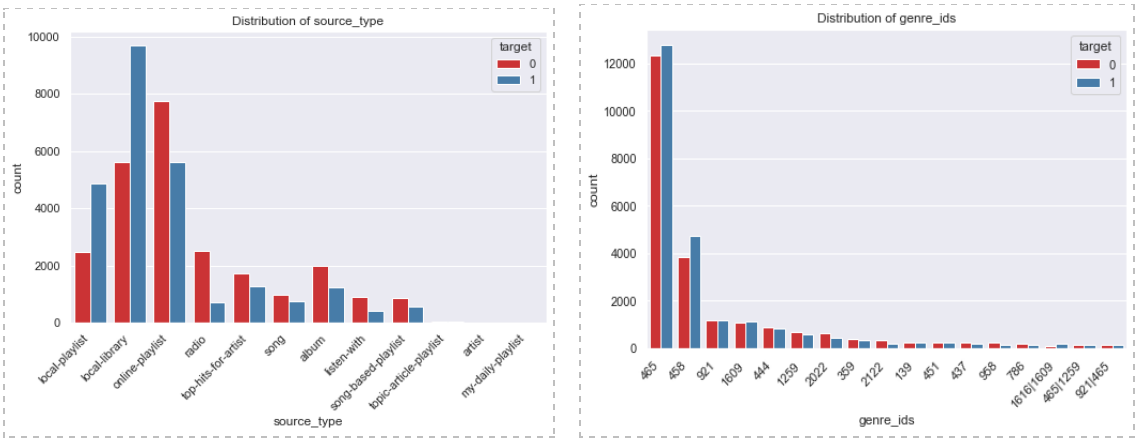


Figure 1: (left) The distribution of source type (i.e. where a song is played from) shows that when a song is played from a user's local library, as opposed to the radio, it's likely a user will listen to the song more than once. (right) In contrast, The distribution of song genre does not provide insight into whether a song will be played more than once as most genres are evenly split between songs that are played repeatedly (target = 1) and songs that are not (target - 0).

Additional EDA analyses are included within Appendix

III. Feature Engineering

Feature engineering consisted of categorical data encoding. Types of encoding explored included:

- One-hot encoding: create separate features for each class of a variable [2].
- Label encoding: encode variables with values between 1 and n_classes - 1.
- Target encoding: associates the mean of the target value to encode the class and then apply a smoothing value to help reduce overfitting [3].
- Frequency encoding: replace categorical features with the frequency of class observations in the dataset [2].

IV. Modeling

As most of the features within the dataset were categorical without ordinality, mostly nonlinear models were explored. Logistic regression and KNN models were evaluated to provide a comparison between linear and non-linear models. Then, multiple tree-based models were evaluated and compared to determine if model performance could be improved using a higher order model.

Results

A summary of each model's accuracy is provided in Table 1. The framework and parameters for each model are discussed separately in the following sections.

Table 1: Model Performances

Model	**Accuracy (%)**
Logistic Regression	53.71
K-Nearest Neighbor	66.13
Decision Tree	63.74
Gradient Boosted Tree (XGBoost)	65.54
Gradient Boosted Tree (CatBoost)	64.00

Logistic Regression

Logistic regression is a parametric linear classifier. It was expected that this classifier would not perform well on this highly categorical dataset, however, it served as a comparison against other nonlinear models. A combination of frequency encoding, which was used on high cardinality features, and one-hot encoding was used on categorical features, missing values were filled with 0s. Since logistic regression suffers from multicollinearity [5], the features "song_id", "genre_id" and "artist_name" were removed. Using a train-test split of 70%-30%, logistic regression scored an accuracy of 52.79% in cross-validation on the training set and 52.81% accuracy in the testing set.

In order to improve accuracy, chi-squared and mutual information feature selection were performed [6]. Chi-squared aims to find features that are most dependent on the target variable while mutual information uses information gain to determine feature selection[7]. Model tuning using mutual information and C = 0.001 improved our original accuracy the most by around 0.9%. As such, it was clear that logistic regression was not the appropriate model selection compared to other nonlinear models.

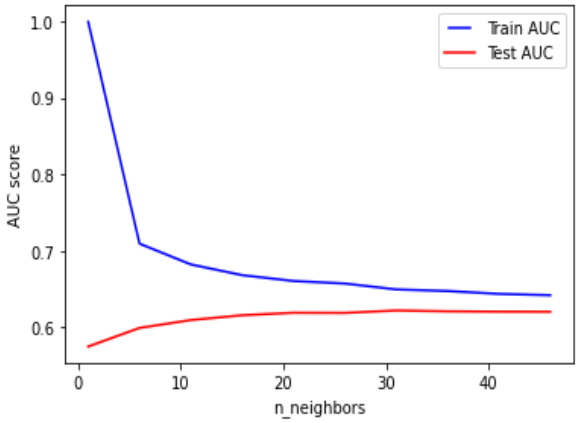
K-Nearest Neighbors (K-NN)

K-NN is a non-parametric classifier which functions by classifying an observation by a majority vote of its K nearest neighbors [8]. Normalization of feature values was performed to unify distance scale. While several categorical encoding methods were again evaluated for this model, a baseline evaluation of the model using all features and default parameters showed that target encoding outperformed other methods by a margin of 2%. Do note that missing values were handled with imputation by '-1', '0' and mean value for label, frequency and target encoding respectively. Parameter tuning was performed to extract the best parameters (Table 4 and Figure 5) which improved performance by another 2%. Mutual information selection also boosted the model accuracy by a respectable amount of 4.4%, which could be an indication that KNN is susceptible to the curse of dimensionality. Therefore, the best K-NN model accuracy produced 65.25% cross-validated score on the training set and an accuracy of 66.13% on the testing set, demonstrating the KNN is a much more appropriate model for this dataset over logistic regression.

Table 2: Parameters for K-NN

Parameter	Default	Optimized
n_neighbors	5	33
weights	uniform	uniform
algorithm	auto	auto
leaf_size	30	5
p	2(Euclidean)	1(Manhattan)

Figure 2: AUC vs n_neighbors



Decision Tree

Decision trees are a non-parametric supervised learning method which can be used in regression and classification problems. Decision trees start with observations about an item (represented in the branches) and ultimately provide predictions about the item's target value (represented in the leaves). As the dataset contained a target value of 0 or 1, the subsequent model is considered a classification decision tree.

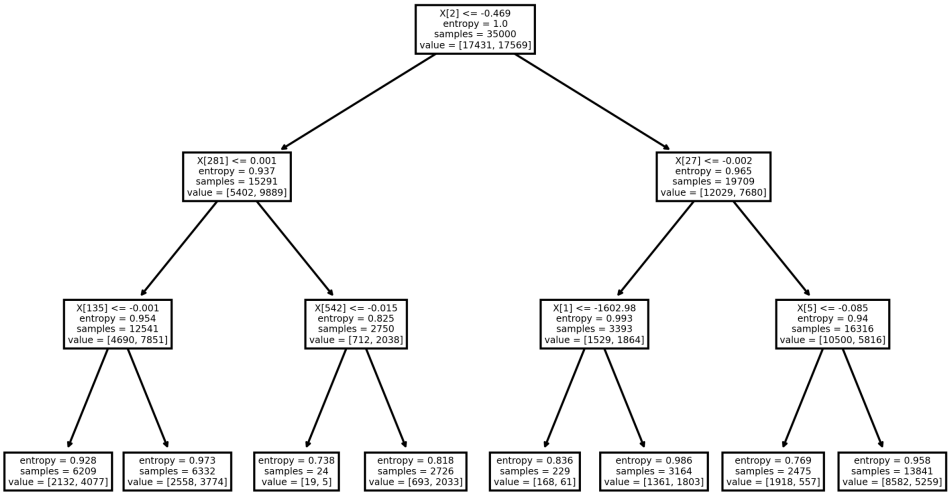
Scikit-Learn Decision Tree Classifier was used to develop a generic decision tree model using all features, default model parameters, label encoding and a 70-30 train-test split (Table 3) which

produced an accuracy of 54.31%. After calculating the missing value ratio, the variable “composer” was removed because the missing value ratio was above 20%. Then using GridSearchCV, One-Hot Encoding and PCA to select the best parameters and most important features, an accuracy of 63.74%, an improvement of 9.43% and the visualization of Decision Tree was obtained (Figure 6). Principal component analysis (PCA) was performed on the encoded dataset [4]. Due to high dimension data created by one-hot-encoding, PCA was used to project the data onto a lower dimensionality subspace. Therefore, feature selection, encoding strategy, PCA and estimator tuning drastically improved the accuracy of the Decision Tree Model.

Table 3: Default and Optimized Parameter of Decision Tree Model

Parameter	**Default**	**Optimized**
criterion	gini	entropy
max_depth	None	3
max_feature	None	None
min_samples_leaf	None	8
splitter	best	best

Figure 3: Visualization of Optimized Decision Tree



Gradient Boosted Tree (XGBoost)

XGBoost is a type of decision tree model that stands for eXtreme Gradient Boosting [9]. “Boosting” is a type of ensemble technique in which new trees are added in order to increase model performance [10]. As such, the model is built in a stage-wise fashion [11]. “Gradient

boosting” occurs when the new tree predicts the errors of prior models using the gradient descent algorithm to minimize the loss of new models.

Using the XGBoost library in Python, a model was created using the default model parameters listed in Table 2. Additionally, the model used a binary logistic regression objective function and included L2 regularization. Using a train-test split of 75-25, respectively, the test accuracy was found to be 65.54%. This was the highest accuracy of all the models when only default model parameters were considered. However, the model may have continued to improve with additional feature engineering or parameter tuning.

Table 4: Parameter Values for XGBoost Model

Parameter	**Value**
Learning Rate	0.1
Max Depth	3
Number Estimators	100
Minimum Child Weight	1

Gradient Boosted Tree (CatBoost)

CatBoost is a state-of-the-art open-source gradient boosting on decision trees. It is developed by Yandex researchers and engineers and it is the successor of the MatrixNet algorithm that is widely used within the company for ranking tasks. Similar to the previously discussed XGBoost model, CatBoost performs boosting on decision trees. But the major highlight of CatBoost is that it performs better than other models with just the default parameters without any tuning. Additionally, CatBoost is widely preferred due to its high prediction speed.

The reason Catboost outperforms other classical gradient boosting methodologies is due to its categorical feature handling mechanism which performs random permutations to prevent target leakage. By which, CatBoost uses artificial timestamps and orders the data which helps to overcome the prediction shift. The main battery of Catboost is its symmetrical tree structure. We could improve the accuracy by also selecting data points that require higher weightage. Alongside all the stated functionalities, Catboost enables staged prediction and shrinking models. It also provides information on the features important for a specific data point. But the main aspect is CatBoost eliminates the need for hyper parameter tuning.

Discussion

As specified in the Data Wrangling section, this project allowed our team to gain exposure to working with large and challenging datasets. Cautious decisions were made to ensure the final dataset contained reliable information and was a suitable size to manipulate on our local computer hardware.

The most significant portion of feature engineering was spent encoding categorical data. As discussed within the Methods section, several encoding methods were explored. Table 5 discusses the pros and cons of each method.

Table 5: Advantage and Disadvantages of Categorical Encoding

Type of Encoding	**Advantages**	**Disadvantages**
Label Encoding	Works with high cardinality data. Easy to implement.	Introduces ordinality.
Frequency Encoding	Works with high cardinality data. Good performance.	Loss of information (two classes appearing with same frequency). Multicollinearity.
Target Encoding	Works with high cardinality data. Good performance.	Tends to overfit. Smoothing is computationally expensive.
One-Hot Encoding	Preserves nominal nature well.	Drastically increases the number of features.

Ultimately, we found that frequency and target encoding performed the best with several model types. Table 6 provides a summary of logistic regression performance comparing different types of encoding.

Table 6: Comparison of Encoding Techniques

Type of Encoding	**Logistic Regression Model Accuracy (%)**
Label Encoding + One Hot	52.33
Frequency Encoding + One Hot	52.81
Target Encoding + One Hot	51.40

As most of our data was categorical and nominal in nature, our team chose to explore mostly nonlinear models. As expected the logistic regression performed the worst of all the models as this is a linear parametric model. The KNN model performed the best of these models, however, this was after extensive parameter tuning and mutual information feature selection. In contrast, the XGBoost algorithm obtained accuracy values on par with KNN while only using the default model parameters. Additional feature selection, such as PCA, or parameter tuning using GridSearchCV, would likely boost this model's performance above KNN. as expected the logistic regression performed the worst of all the models as this is a linear parametric model.

Valuable lessons were learned when comparing our results to the first place solution on Kaggle. The competition winner used a combination of neural networks and LightGBM, which is a well optimized gradient boosted decision tree model, to achieve an overall accuracy of 74% [12]. Unlike our decision tree and XGBoost models which used a combination of one-hot encoding, frequency and label encoding, LGBM takes a more balanced and efficient approach towards categorical data treatment by partitioning categories into 2 subsets and using "its accumulated values (sum_gradient / sum_hessian) to find the best split" [13]. This method may contribute to greater accuracy. The winner also conducted extensive feature engineering by including conditional probability features and single value decomposition features. Finally, the winner was able to extract song release time related information from the song_extra_info.csv dataset, a dataset that our team had chosen to dismiss.

Overall, this project taught us the importance of feature engineering and exploring multiple types of models. If given more time, our team would have liked to explore some of the feature engineering techniques mentioned above as well as additional nonlinear models such as a support vector machine or neural network.

References

1. KKBOX. (2017). WSDM - KKBox's Music Recommendation Challenge. Retrieved June 22, 2020 from https://www.kaggle.com/c/kkbox-music-recommendation-challenge
2. Shubham, Singh (2020, Feb 23). *Categorical Variable encoding techniques.* Analytics Vidhya. https://medium.com/analytics-vidhya/categorical-variable-encoding-techniques-17e607fe42f9
3. Svideloc (2020, Jan 16). *Target encoding vs one-hot-encoding with simple examples.* Analytics Vidhya. https://medium.com/analytics-vidhya/target-encoding-vs-one-hot-encoding-with-simple-examples-276a7e7b3e64
4. *What are the pros and cons of the PCA.* i2 tutorials. https://www.i2tutorials.com/what-are-the-pros-and-cons-of-the-pca/
5. *Logistic Regression.* Wikipedia. https://en.wikipedia.org/wiki/Logistic_regression#:~:text=Logistic%20regression%20is%20a%20statistical,a%20form%20of%20binary%20regression).
6. Jason Brownlee, (2020, June 30). *How to perform feature selection with categorical data*. Machine Learning Mastery https://machinelearningmastery.com/feature-selection-with-categorical-data/
7. Jason Brownlee, (2019, Nov 5). *Information gain and mutual information*. Machine Learning Mastery https://machinelearningmastery.com/information-gain-and-mutual-information/

8. *K-nearest-neighbors algorithm.* Wikipedia https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
9. *XGBoost Documentation.* https://xgboost.readthedocs.io/en/latest/index.html#
10. Brownlee, J. (2020, April 22). *A Gentle Introduction to XGBoost.* Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
11. *Gradient Boosting.* Wikipedia. https://en.wikipedia.org/wiki/Gradient_boosting
12. *A brief introduction to the 1st place solution(codes released)*. Kaggle https://www.kaggle.com/c/kkbox-music-recommendation-challenge/discussion/45942
13. *LightBGM Documentation* https://lightgbm.readthedocs.io/en/latest/Features.html?highlight=encoding#optimal-split-for-categorical-features
14. *Using pandas calculate cramer's coefficient matrix,* StackOverflow https://stackoverflow.com/questions/20892799/using-pandas-calculate-cram%C3%A9rs-coefficient-matrix

Appendix:

Appendix A: Additional EDA

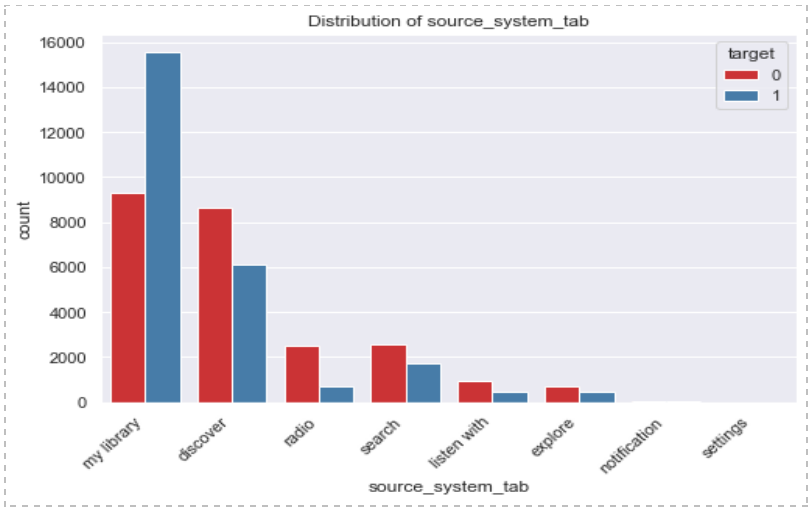
Index	ce_systen	e_screen_	ource_typ	ong_lengt	language	city	gistered_v	stration_c	msno	song_id	genre_ids	rtist_nam	composer
source_system_tab	1	0.66	0.66	0.26	0.06	0.04	0.05	0.27	0.4	0.33	0.11	0.24	0.22
source_screen_name	0.66	1	0.65	0.16	0.06	0.03	0.06	0.21	0.32	0	0.09	0.21	0.19
source_type	0.66	0.65	1	0.15	0.06	0.04	0.06	0.23	0.43	0.22	0.09	0.17	0.16
song_length	0.26	0.16	0.15	1	0.63	0	0	0	0	0.85	0.58	0.59	0.61
language	0.06	0.06	0.06	0.63	1	0.02	0.03	0.12	0.26	0.76	0.52	0.81	0.67
city	0.04	0.03	0.04	0	0.02	1	0.32	0.56	0.82	0	0.01	0.08	0.03
registered_via	0.05	0.06	0.06	0	0.03	0.32	1	0.61	0.82	0	0.02	0	0
registration_days	0.27	0.21	0.23	0	0.12	0.56	0.61	1	0.85	0	0	0	0
msno	0.4	0.32	0.43	0	0.26	0.82	0.82	0.85	1	0	0	0	0
song_id	0.33	0	0.22	0.85	0.76	0	0	0	0	1	0.76	0.81	0.83
genre_ids	0.11	0.09	0.09	0.58	0.52	0.01	0.02	0	0	0.76	1	0.75	0.62
artist_name	0.24	0.21	0.17	0.59	0.81	0.08	0	0	0	0.81	0.75	1	0.61
composer	0.22	0.19	0.16	0.61	0.67	0.03	0	0	0	0.83	0.62	0.61	1

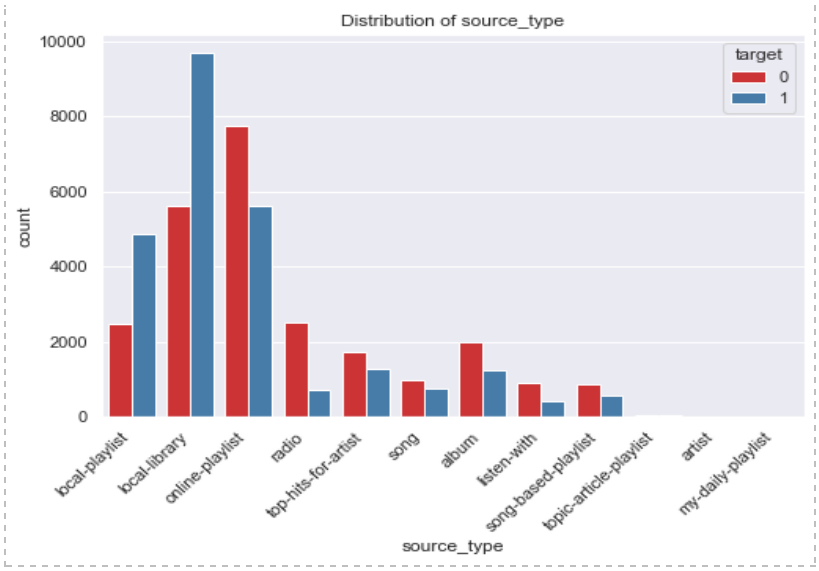
Figure 4: Cramer's V Correlation Matrix of the variables. Since the variables were mostly categorical, a traditional Pearson correlation matrix wouldn't capture proper information. Therefore, a Cramer's V correlation matrix [14] was developed using frequency encoded data. Information from the same category had higher correlation. For example, artist_name and language show high correlation because an artist will mostly use one language. The correlation matrix serves the purpose of preventing the issue of multicollinearity for certain models.

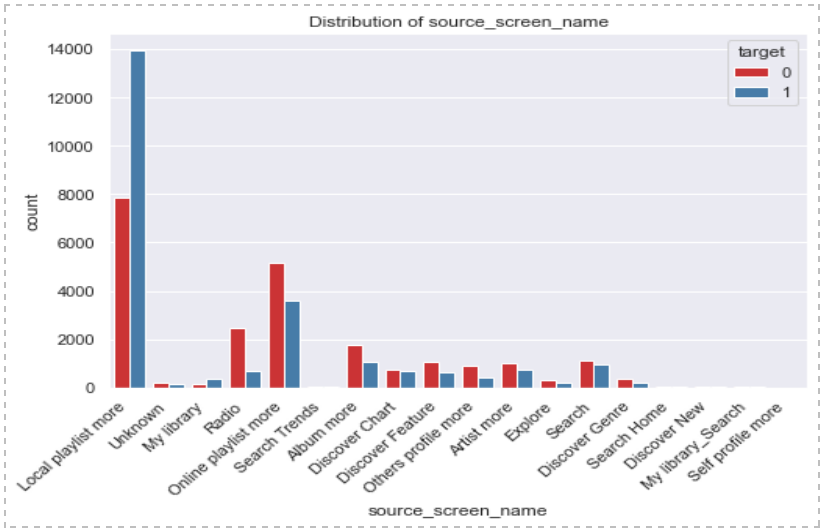
	msno	song_id	source_system_tab	source_screen_name	source_type	target	song_length	genre_ids	artist_name	composer	language	city	registered_via	registration_days
count	50000	50000	49833	47138	49857	50000.00000	50000.00000	49178	49998	38683	49997.00000	50000.00000	50000.00000	50000.00000
unique	16656	21069	8	18	12	nan	nan	208	5456	8072	10.00000	21.00000	5.00000	nan
top	MXIMDXOOj3UpaT7FvOSGW6YSzthlh+xYjTqGoUdMzEE=	reXuGcEWDDCnL0K3Th//3DFG4S1ACSpJMzA+CFipo1g=	my library	Local playlist more	local-library	nan	nan	465	Various Artists	周杰倫	3.00000	1.00000	9.00000	nan
freq	38	107	24898	21818	15282	nan	nan	25141	2173	1374	27420.00000	17854.00000	19109.00000	nan

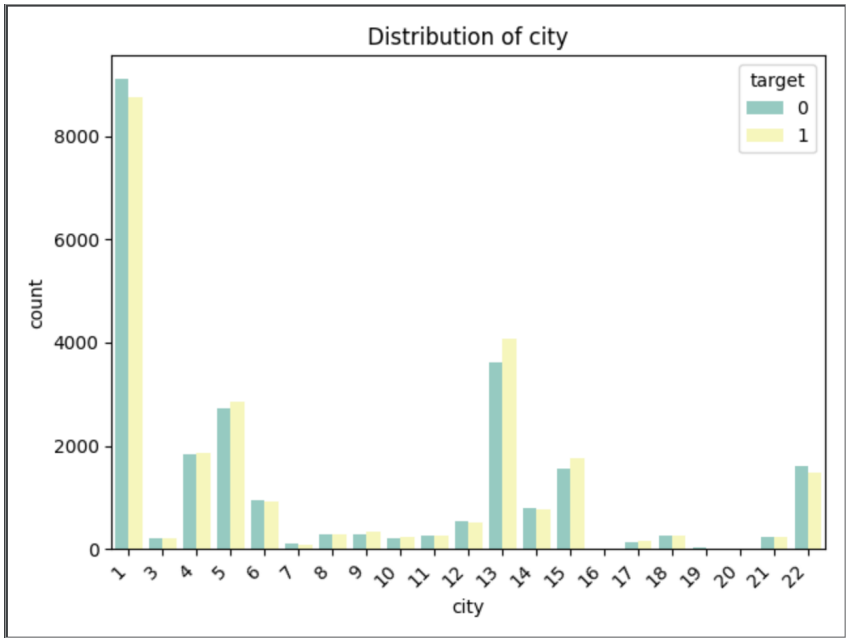
Figure 5: Summary statistics (unique, top and frequency) were calculated for each variable. From this table, it's shown that there were several thousand unique users (msno), song IDs, artists and composers making one-hot encoding a bad choice. Additionally, variables that did not have a total count of 50,000 indicate missing values, with "Composer" containing the most missing values.

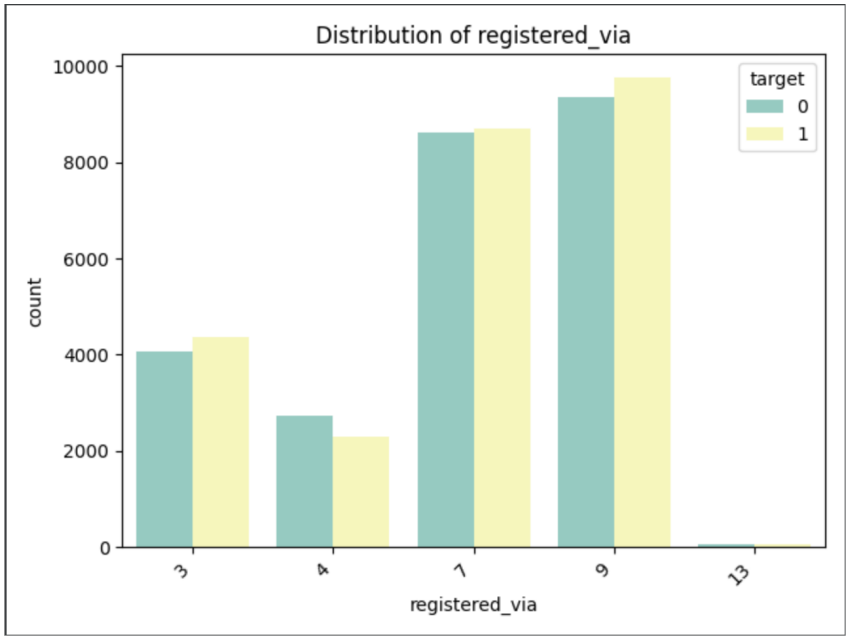
Additional Distribution Plots




Distribution of source_type
target
0
1
count
10000
8000
6000
4000
2000
0
local-playlist
local-library
online-playlist
radio
top-hits-for-artist
song
album
listen-with
song-based-playlist
topic-article-playlist
artist
my-daily-playlist
source_type


Distribution of source_screen_name
target
0
1
count
14000
12000
10000
8000
6000
4000
2000
0
Local playlist more
Unknown
My library
Radio
Online playlist more
Search Trends
Album more
Discover Chart
Discover Feature
Others profile more
Artist more
Explore
Search
Discover Genre
Search Home
Discover New
My library_Search
Self profile more
source_screen_name


Distribution of city
target
0
1
count
8000
6000
4000
2000
0
1
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
city


Distribution of registered_via
target
0
1
count
10000
8000
6000
4000
2000
0
3
4
7
9
13
registered_via