KKBOX'S MUSIC RECOMMENDATION

Yunru Huang[1], Mengyu Li[2], Yun Wu[3]
Department of [1]Management Science and Engineering, [2]Statistics. [3] SCPD, Stanford University, USA

Overview

- **Task**: build a **music recommendation system** based on user behaviors and song features.
- We built models to predict the chances of a user listening to a song repetitively after the first observable listening event within a time window, providing a binary prediction.

Data Set

- 3,076 users, 113,750 of songs, and over 700 thousands records on user-user interactions.
- Text data only, no audio features.
- Balanced Class labels (50.9% Positive, 49.1% Negative).
- 70% as training set, 15% as validation set and 15% as test set.
- **Challenge:** Large Amount of Categorical Features (with high cardinality)

Feature Engineering

- **Data Cleaning**: Inspect and remove a small portion of rows missing important features or even label
- **Feature Generation**: Explicitly inject time index as feature
- **K-Modes Clustering**: Cluster Categorical Data, Generate Song Features and UI features.
- **MCA**: Multiple Correspondence Analysis to develop feature association between user and songs.
- **One-Hot Encoder**: For Categorical Features, using Label encoder and one hot encoder.
- **Standardization**: For numerical Data, standardized to have mean 0 and variance 1.

Methodology

Diverse Gradient Boosting Decision Tree

XGBoost:
- Sparsity-aware algorithm for sparse data
- Weighted quantile sketch for approximate tree learning

CatBoost:
- Ordered boosting for Categorical Features

LightGBM:
- Gradient-based One-Side Sampling: Exclude a significant proportion of data instances with small gradients, and only use the rest to capture the most important information gain.
- Exclusive Feature Bundling: bundle mutually exclusive features to reduce the number of features.
- Efficient Leaf-Wise Search

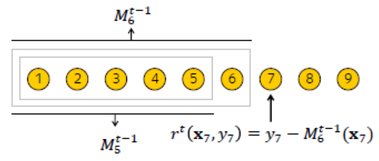
Other Explorations
1. Linear SVM
2. Logistic Regression
3. Decision Tree
4. Fully-Connected Neural Network

Fine Tuning
1. Randomized Search Cross Validation

Algorithm : Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ *to* m **do**
  // enumerate missing value goto right
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I_k,$ *ascent order by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // enumerate missing value goto left
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*$(I_k,$ *descent order by* $\mathbf{x}_{jk})$ **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

ordered boosting example

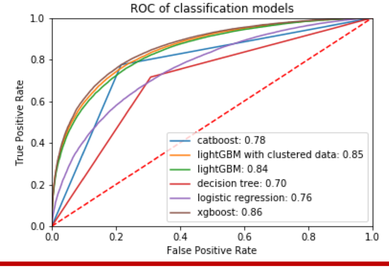


Results

From our experiments, we can see that **boosting models fit our dataset best.** This is probably because our data is full of categorical features, and boosting methods can handle the categorical features as well as the sparsity of the data very well. Our best single model is achieved by XGBoost with the ROC-AUC Score of 0.859. LightGBM also has a comparable result of 0.853. Catboost doesn't perform as well as the other 2 methods, with the score of 0.779.

Neural Network doesn't perform well on our dataset due to the sparsity. After adding the clustering features, we observe a significant improvement of 10%. Dropout Layer also improves the performance by 2%.

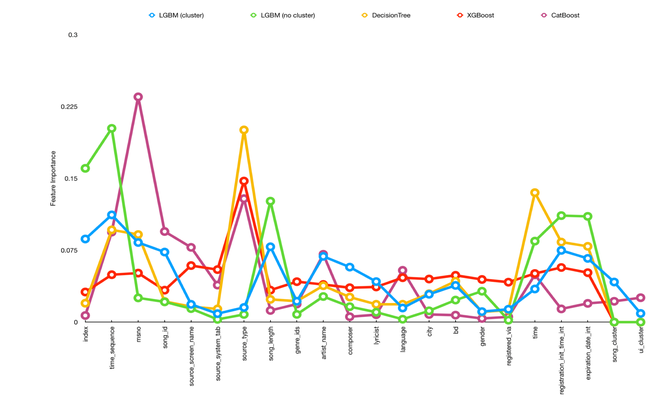
Model	Roc-Auc Score
SVM	0.687
Logistic	0.773
3-Layer FC NN	0.51
3-Layer FC NN + Cluster	0.625
3-Layer FC NN + Dropout	0.641
Decision Tree	0.703
CatBoost	0.779
LightGBM	0.853
XGBoost	0.859
Ensemble (Diverse Boosting)	**0.861**



Discussion

Different Boosting Models have different weights for features due to the algorithm emphasis.

For LightGBM, the top three most important features are **the index of user's listening record that includes timestamp information, msno (user id), and user registration time.** This intuitively suggests that user identity decides if they listen the song once again. We then see that artist name ranking one of the top five features, way more important than genre, language, lyricist, etc. This is again intuitive in the sense that artist name carries more specific "genre" information.



CatBoost lay more emphasis on categorical features. It's top 3 important features include msno(user id), source type and song id. Noticing that CatBoost doesn't capture the information of time series indicator of timestamp and registration time well, which may explain its worse performance compared to 2 other features.

XGBoost weights features evenly overall and only gives high weights to the time series indicator.

Future work

Modeling
- Improved Random forest
- Stacking with intentionally diversified models

Data & Feature Engineering
- Categorical feature cardinality reduction or conversion to numerical values- Data transformation for heavy tail features
- Outlier detection and aggressive removal for untypical user that fails multiple model predictions