WSDM KKBOX MUSIC RECOMMENDATION SYSTEM

BAN 671 Data Analytics with R

Team Project

by

Sona Nathani

Tanvi Kopardekar

Neha Gupta

Navneet Chauhan

Saif ur Rahman Mohammed

Manga Lakshmi Prasanna Chitti

WSDM KKBOX MUSIC RECOMMENDATION SYSTEM

ABSTRACT

KKBOX, Asia's leading music streaming service, holding the world's most comprehensive Asia-Pop music library with over 30 million tracks. KKBOX lets you enjoy music anytime, anywhere, whether on a mobile phone or computer, it even makes it possible to interact with friends and artists via music.

The main goal of this project is to fetch the dataset provided by KKBOX on Kaggle community and build a better music recommendation system. They currently use a collaborative filtering based algorithm with matrix factorization and word embedding in their recommendation system but believe new techniques could lead to better results.

I. INTRODUCTION

The 11th ACM International Conference on Web Search and Data Mining (WSDM 2018) challenges to build a better music recommendation system using a donated dataset from KKBOX. WSDM (pronounced "wisdom") is one of the the premier conferences on web inspired research involving search and data mining. They're committed to publishing original, high quality papers and presentations, with an emphasis on practical but principled novel models.

Not many years ago, it was inconceivable that the same person would listen to the Beatles, Vivaldi, and Lady Gaga on their morning commute. But, the glory days of Radio DJs have passed, and musical gatekeepers have been replaced with personalizing algorithms and unlimited streaming services.

While the public's now listening to all kinds of music, algorithms still struggle in key areas. Without enough historical data, how would an algorithm know if listeners will like a new song or a new artist? And, how would it know what songs to recommend brand new users?

WSDM has challenged the Kaggle ML community to help solve these problems and build a better music recommendation system.

II. PROJECT DESIGN

To investigate the impact of using different classification algorithms, we compared the accuracy of each on a variety of factors like variance, bias and accuracy. We used ensemble learning classifier – Random Forest, because it had low variance high accuracy.

Since this is a huge dataset that has many non-linearities, we also built a Neural Network using keras to better identify the patterns in which users chose their songs, in order to make predictions using similar data in the future.

Further, to boost the model performance, we used XGboost-which combines a group of weak classifiers to create a strong classifier and found out the accuracy of the model increased by around 6-7%

III. IMPLEMENTATION OF THE PROJECT

We have two files to do the exploratory analysis and run the prediction (recommendation) model. Datasets provide training and test data which are selected from users listening history for a time period. Datasets are chosen based on time, basis on the split of public/private which are based on unique user/song pairs.

Training Dataset: train.csv

msno: userid

song_id: songid

source_system_tab: Event was triggered or not. System tab categorize KKBOX mobile apps functions. Tab my library contains functions to manipulate the local storage, and tab search contains functions relating to search.

source_screen_name: name of the layout a user observe.

source_type: An entry point a user first plays music on mobile apps. For e.g. Album, Online-playlist, Songs etc.

target: "1" means recurring listening event triggered within a month after the user's very first observable listening event "0" otherwise.

Test Datasets: test.csv

msno: userid

song_id: songid

source_system_tab: Event was triggered or not. System tab categorize KKBOX mobile apps functions. Tab my library contains functions to manipulate the local storage, and tab search contains functions relating to search.

source_screen_name: name of the layout a user observe.

source_type: An entry point a user first plays music on mobile apps. For e.g. Album, Online-playlist, Songs etc.

Songs Datasets: songs.csv

song_id: songid

song_length: in ms

genre_ids: multiple genre category separated by " | "

artist_name: Name of the Artist

composer: composer name

lyricist: lyricist name

language: language name

Members datasets: members.csv

msno: unique members tag

city: members associated with city

bd: Age. (Contains outliers)

gender: male/female (contain missing values too)

register_via: method of registration

registration_init_time: Date in %Y%M%D format

expiration_date: expiration date in %Y%M%D format

Extra Info: song_extra_info.csv

song_id: song id

song_name: name of the song

isrc: International Standard Recording Code. It's an identification for the code (Note: reference code and reference year can be misleading and incorrect)

3.1. XGboost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient**,** flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples. This method is used by our team to boost the model performance. It combines a group of weak classifiers to create a strong classifier and found out the accuracy of the model to be increased by around 6-7%.

3.2. Neural Network using keras

Keras is a high-level neural networks API, capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. By using neural network using keras, we are trying to increase the capacity by altering the hyper parameters and observing the results.

3.3. Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In this, we have used "Ranger" package to increase the accuracy by 66.95%.

IV. DATA ANALYSIS AND INFERENCE

This dataset consists of information of the first observable listening event for each unique user-song pair within a specific time duration. Metadata of each unique user and song pair is also provided.
We are predicting the chances of a user listening to a song repetitively after the first observable listening event within a time window was triggered.

The KKBOX Music box data has information about their users as shown below:

- The attributes **msno** and **songid** pair together make each record distinct from one another.They are 30755 unique users and 359966 unique songs which each of these users listen on a daily basis.
- **source_system_tab**: the name of the tab where the event was triggered. System tabs are used to categorize KKBOX mobile apps functions.The source tab can be one of those following values they are explore , my library ,search, discover ,radio, listen with ,notification null , settings.
- We can observe the influence of the source_system tab on our target variable as follows:

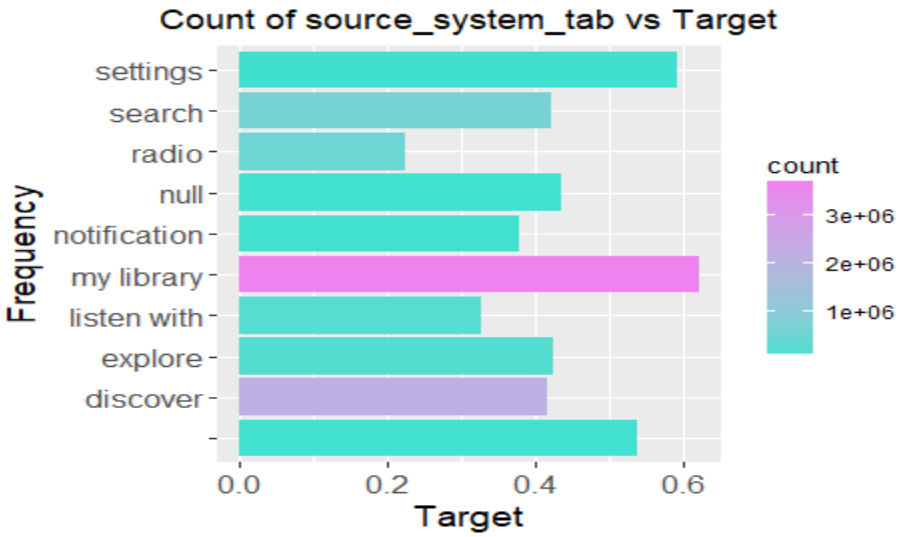
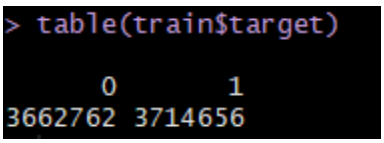


FIGURE 1

We can see that my library has the most count and radio has the least count in the data set. It looks like songs are played mostly through my library, search and discover. One interesting thing is that, if the song is from my library then it is more likely to be replayed within a month and if it is from radio then it is less likely. My library is where the user stores their songs locally and hence they really love that song, leading to high mean_target. On contrary, radio is a random shuffle of songs and hence the user likeability is not predefined leading to low mean_target.

- **source_screen_name**: name of the layout a user sees. This attribute has 21 unique different values from where a user sees the song which can be "Album more" "Artist more" "Concert" , "Discover Chart", "Discover Feature", "Discover Genre" , "Discover New", "Explore" , "Local playlist more" ,"My library", "My library_Search" ,"Online playlist more", "Others profile more" , "Payment" ,"Radio"," Search" , "Search Home" , "Search Trends" ,"Self profile more" or "Unknown".

- **Target:** We have 3662762 occurences of target as 0 recurring listening event not triggered and 3714656 as target 1 recurring listening event triggered



Song id and user id pair are unique in train data set. Songs are grouped together and their count is checked against the target variable. The count of a song present in the train data set is almost linearly associated with the mean_target. Assuming the train data set is randomly drawn from the population, the more the song occurs the more it is discoverable by the user. This plots shows the relationship between discoverability vs mean_target. You could see that there are 166766 songs that are appearing only once and has a lower mean_target and a single song that is appearing 13293 time that has a higher mean_target.

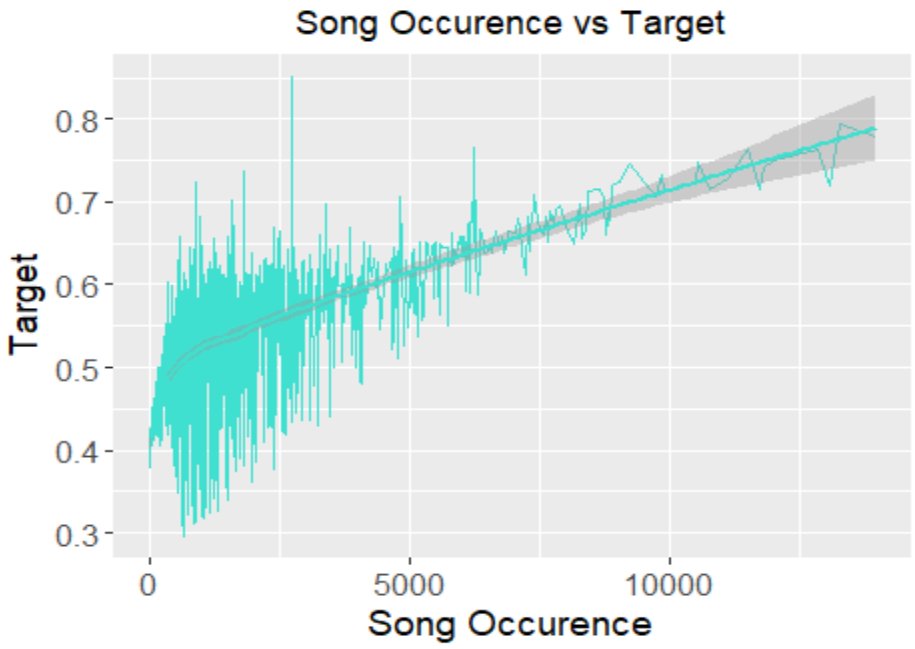


FIGURE 2

The training data set additionally also has member information like city, bd, gender, registered via are categorical and registration init and expiration date are dates for all their users. As mentioned in the data dictionary there seems to be outliers in the age field. There are negative values as well as values above 1000.Sorted bd vs Frequency is shown in the tibble as well as the graph. There are 19932 records with 0 as age. This could be either outliers or missing values. Plotting in the age range 1 -100 to show the real distribution. From the below plot we can conclude that our data mostly consists of users age group between 20 and 30.

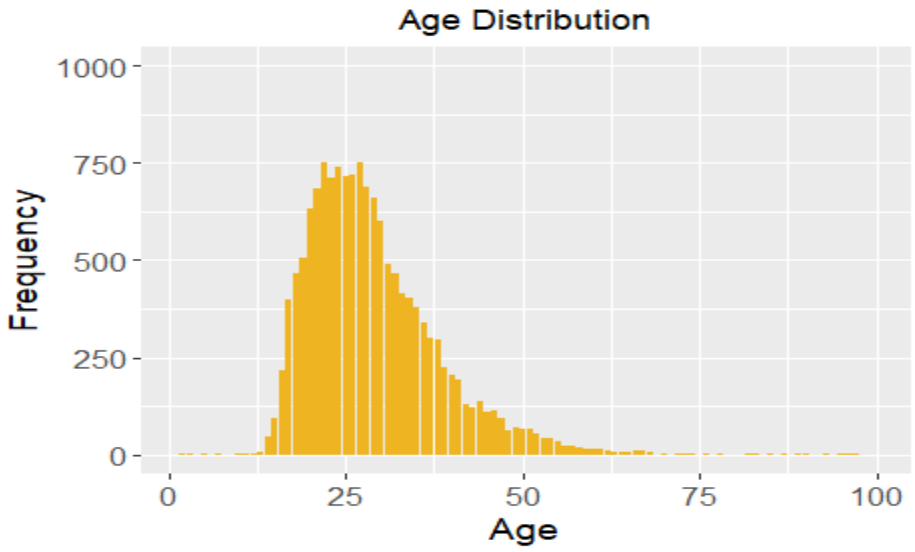


FIGURE 3 – STAR RATINGS

For the gender attribute we have 7096 female users and 7405 male users for our dataset as show below:

```
female  7096
male    7405
```

The City distribution shows that City 1 seems to be the highly dominating and lots of members belong to City 1.

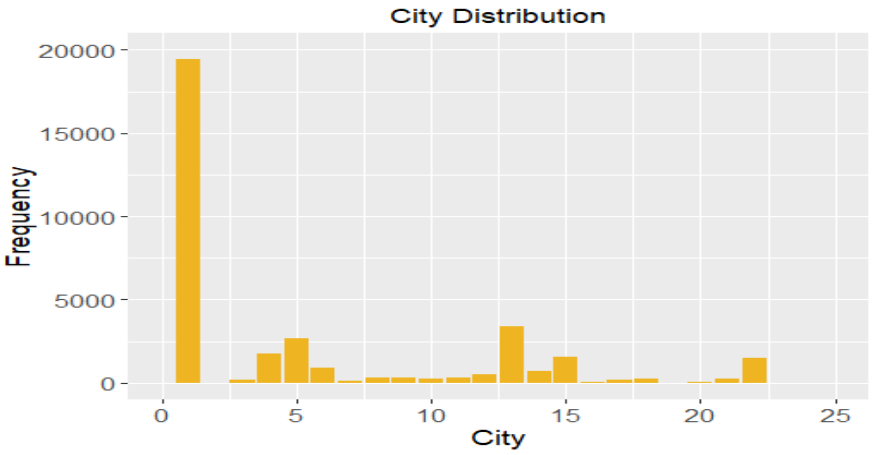


FIGURE 4

V. RESULTS AND DISCUSSION

Observation	Random Forest	Neural Network
Package	**"Ranger"**	**Keras in Keras Tensorflow module**
Accuracy	**66.95%**	**54.12%**
AUC	**0.7**	**0.49**
RMSE	**0.46**	**0.54**

By applying these two methods in our project, we came to a comparison shown in table above and the conclusions is as follows:

- Accuracy by using RandomForest is coming out to be 66.95% , whereas accuracy by using Neural network is coming out to be 54.12%
- We performed Linear combination of the predictors for the ensemble model that maximizes the AUC score.
- RMSE is 46% for RandomForest and 54% by using Neural Network.

```r
library("xgboost")


param = list(
  objective="binary:logistic",
  eval_metric= "auc",
  subsample= 0.95,
  colsample_bytree=0.45,
  max_depth= 10,
  min_child= 6,
  tree_method= "approx",
  eta  = 0.9 ,
  nthreads = 8
)
x_train <- xgb.DMatrix(
  as.matrix(train_df[subs,]),
  label = y[subs],
  missing=-1)
x_val <- xgb.DMatrix(
  as.matrix(ens_val),
  label = y_ens_val, missing=-1)
x_test <- xgb.DMatrix(as.matrix(test_df), missing= -1)

model <- xgb.train(
  data = x_train,
  nrounds = 100,
  params = param,
  maximize= T,
  watchlist = list(val = x_val),
  print_every_n = 50
)
```

```r
devtools::install_github("rstudio/keras")
library(keras)

install_keras()

B_Size= 2^15
x_train<- as.matrix(train_df)
y_train<- as.matrix(data.frame(p=1-y, q=y))

model <- keras_model_sequential()
model %>%
  layer_dense(
    units= 128,
    input_shape = c(ncol(x_train)),
    kernel_initializer='he_normal' #,
  ) %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate= 0.1) %>%

  layer_dense(
    units= 512,
    kernel_initializer='he_normal' #,
  ) %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate= 0.3) %>%

  layer_dense(
    units= 64,
    kernel_initializer='he_normal' #,
  ) %>%
  layer_activation("relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate= 0.1) %>%

  layer_dense(2) %>%
  layer_activation("softmax")

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c("accuracy")
)
```

```r
library(ranger)
set.seed(3141569)
E_SIZE <- 0.185
h<- sample(nrow(train_df), E_SIZE*nrow(train_df))
ens_val <- train_df[h, ]
y_ens_val <- y[h]
train_df <- train_df[-h, ]
y <- y[-h]

# now only _after_ this split we can scale the columns
to_do <- names(train_df)
for (f in to_do){
  mm<- mean(train_df[[f]])
  ss<- sd(train_df[[f]])
  train_df[[f]] <- (train_df[[f]] -mm)/ss
  test_df[[f]] <- (test_df[[f]] -mm)/ss
  ens_val[[f]] <- (ens_val[[f]] -mm)/ss
}

# shrink? Then use this subset
subs <- sample(nrow(train_df), 0.3 *nrow(train_df))


rf <- ranger(y[subs] ~ . , data = train_df[subs,], num.trees = 12
             , verbose= FALSE)

pred_1_e<-predict(rf, ens_val, type = "response")
pred_1_e <- pred_1_e$predictions
pred_1_t<-predict(rf, test_df, type = "response")
pred_1_t <- pred_1_t$predictions

install.packages("Metrics")
library(Metrics)

diagnosis(y_ens_val, pred_1_e, title="ranger")


install.packages("magrittr")
library(magrittr)
```

References

1) Electronic Media
Dataset:
https://www.kaggle.com/c/kkbox-music-recommendation-challenge

2) Books
Data Manipulation with R by Phil Spector
Learning R by Richard Cotton