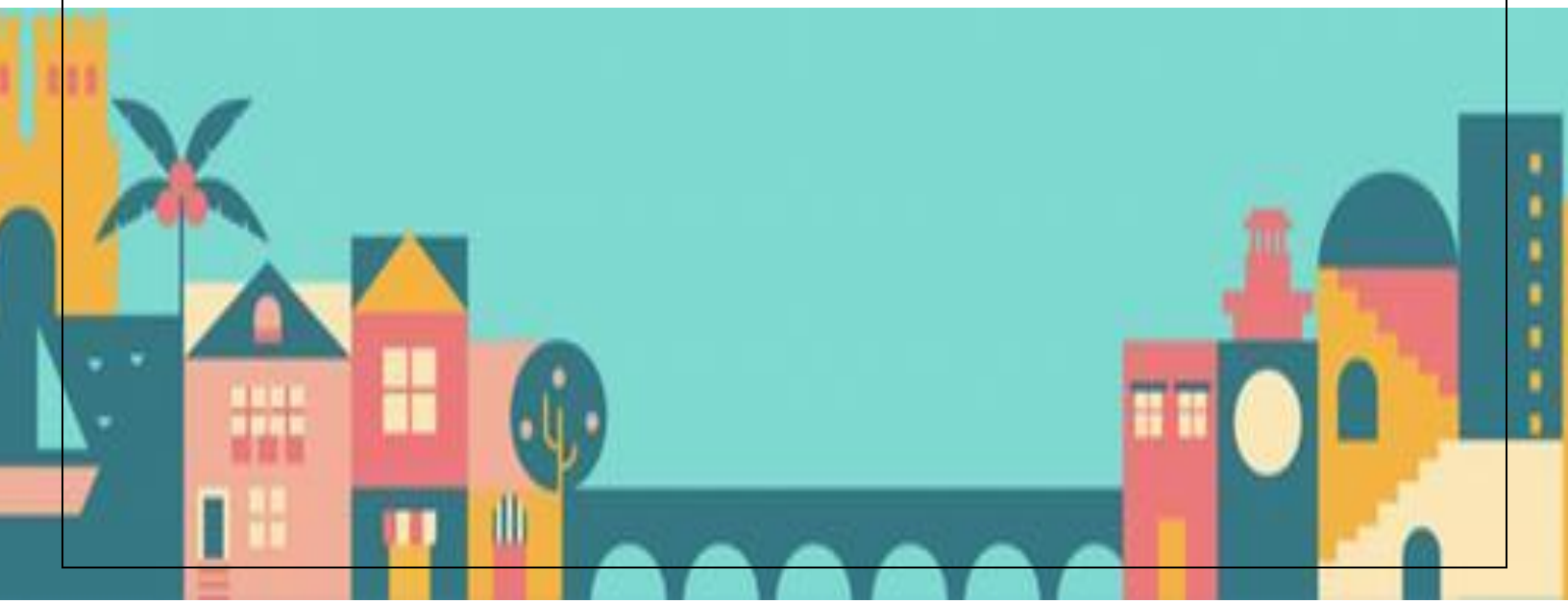
AIRBNB – New User Bookings

WHERE WILL A NEW USER BOOK HIS/HER FIRST TRAVEL DESTINATION?

Priyanka G Kalmane | Technology Fundamentals for Business Analytics | 12/5/2016



Contents

Executive Summary

BACKGROUND

Airbnb is a homestay network and an online marketplace where travelers find the most suitable accommodation where the cost of such accommodation is set by the user. Depending on the prior bookings, Airbnb customizes the account to make it more intuitive to its users, thus making the booking an enjoyable experience.

PROBLEM STATEMENT

In case of new users, it is generally difficult to provide a customized experience and thus Airbnb wants us to predict a list of 5 possible countries the user may book his/her first travel experience.

WHY IS THIS IMPORTANT?

Using a list of variables provided by Airbnb, it is possible to closely predict the possible destination countries. By doing this, it is possible to provide the user with a well-focused, personalized experience. Good user experience = Happy customers. Happy customers = Higher Revenue.

SOLUTION

The data provided by Airbnb for this problem is broken down into several components, data is preprocessed and cleaned. Missing data is handled and 3 different classification machine learning algorithms are applied to train the model on the training data set. This model is later fit to the test data and is submitted to Kaggle, where the competition was hosted earlier in the year 2016. 3 different accuracy values are obtained.

CONCLUSION

The accuracy differs with different algorithms used. Preprocessing, cleaning and imputing missing values must be given initial priority. Using the evaluation criteria and optimizing it in order to obtain high accuracy. It is also important to be careful not to overfit the training data which affects the accuracy. Choice of the most relevant variables is important. All of these factors collectively contribute to high accuracy.

Introduction of Problem and Context

Predictive analysis is a technique that is nowadays being employed by many digital marketing, ecommerce companies in order minimize risk and reduce uncertainty. Predictive analytics is highly useful for better customer experience thus enabling companies obtain higher revenue.

Airbnb expects us to employ the same technique for a more customized, personalized experience for the new users who want to book their first travel experience. Out of 10 countries and 2 other classifications (other and NDF – No destination found) given, we are supposed to find 5 best possible destinations the new users any choose to go to, based on various factors.

With the help of this, Airbnb hopes to achieve a better customer engagement, better target promotions, optimized pricing and a smoother, predictive search for the users. Happy users equals happy customers. Happy customers equals higher revenue.

Since we have 12 different classifiers, this a classification-prediction problem. We can use multiple classification machine learning algorithm packages to achieve this from Python or R.

CHALLENGES OF PREDICTIVE ANALYTICS

Predictive Analytics does not guarantee an outcome. Also, with the given training data set, there is a possibility of overfitting the data and hence get a low accuracy for the test data. Thus, using sophisticated machine learning algorithm may not be the solution. Observing, preprocessing, cleaning of the training data and identifying the correct patterns is of utmost importance.

Data description and initial pre-processing

The data is given to us in 5 different csv files.

Train data – 213451 observations. Consists of variables such as id, date of the account created, timestamp first active, date of the first booking, gender, age, signup method, signup app, language, type of marketing, digital marketing provider, first device type, first browser, country destination.

Test data – 62096 observations. Consists of all the variables in the train data set except the country destination.

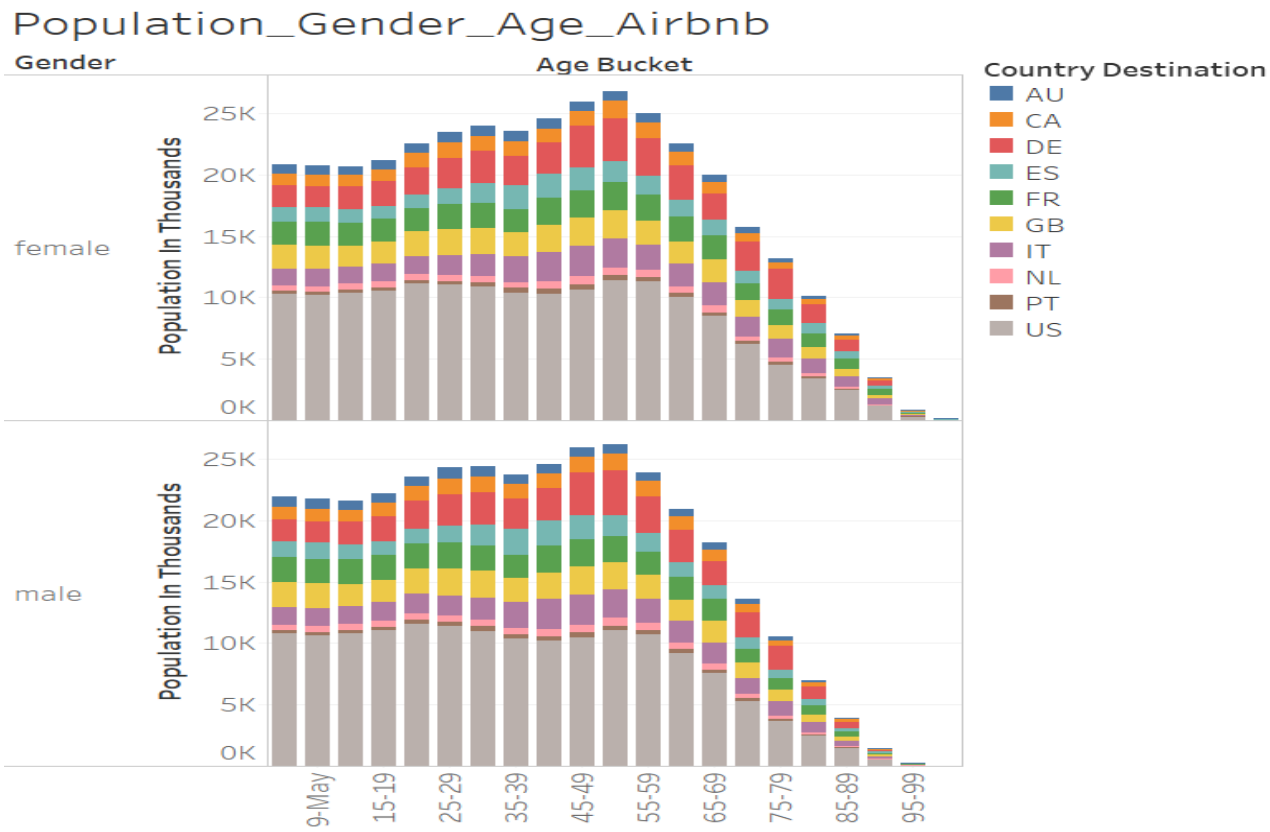
Sessions – 1048575 observations. Contains the sessions data of the users. Variables include user id (which may be used to merge with train data), action, action detail, device type and seconds elapsed.

There are 2 reference files. These 2 files are used to make the initial visualizations that help in understanding the patterns in the data better.

Age gender buckets – 420 observations. Consists of variables such as age bucket, country, gender, population, year.

Countries – 10 observations. Consists of countries related data. Latitude, Longitude distance. Also consists of the distance in kms, where distance from US is considered to be zero.

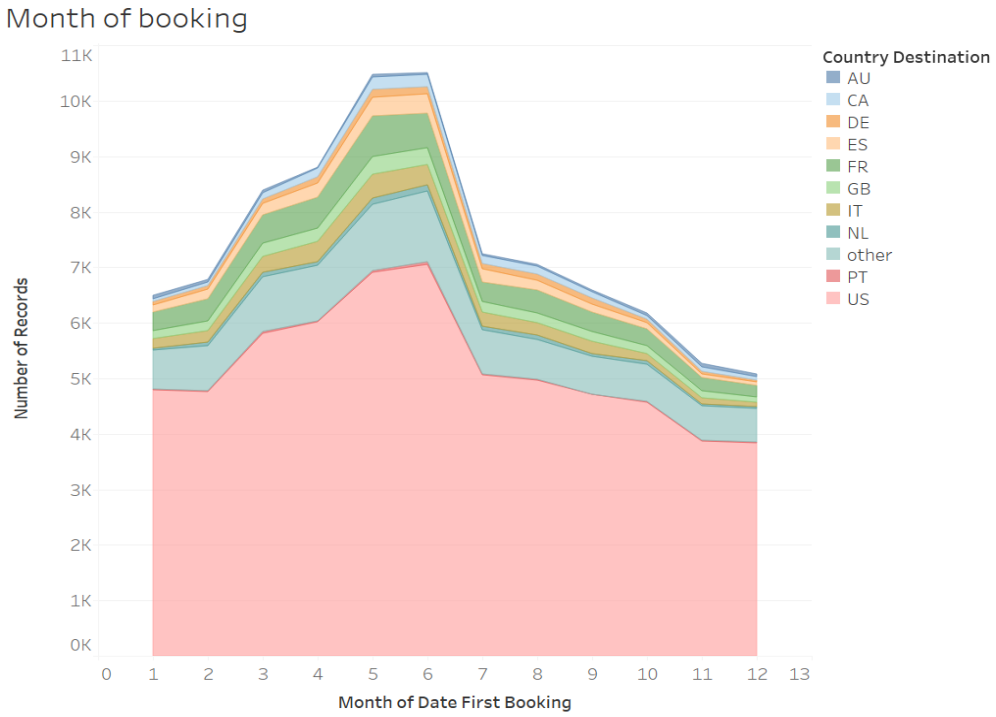
VISUALIZATIONS, INTUITION AND DATA UNDERSTANDING



Sum of Population In Thousands for each Age Bucket broken down by Gender. Color shows details about Country Destination.

General Intuition:

Age, Gender influence the destination country to a larger extent. We see that female travelers between 50-55 travel the most and choose US as their country of destination. It is also interesting to see how "Denmark" is also a popular country of destination among the travelers. Male travelers between the age of 45-50 and 50-55 travel the most compared to the other age groups. US is again their favorite country of destination.

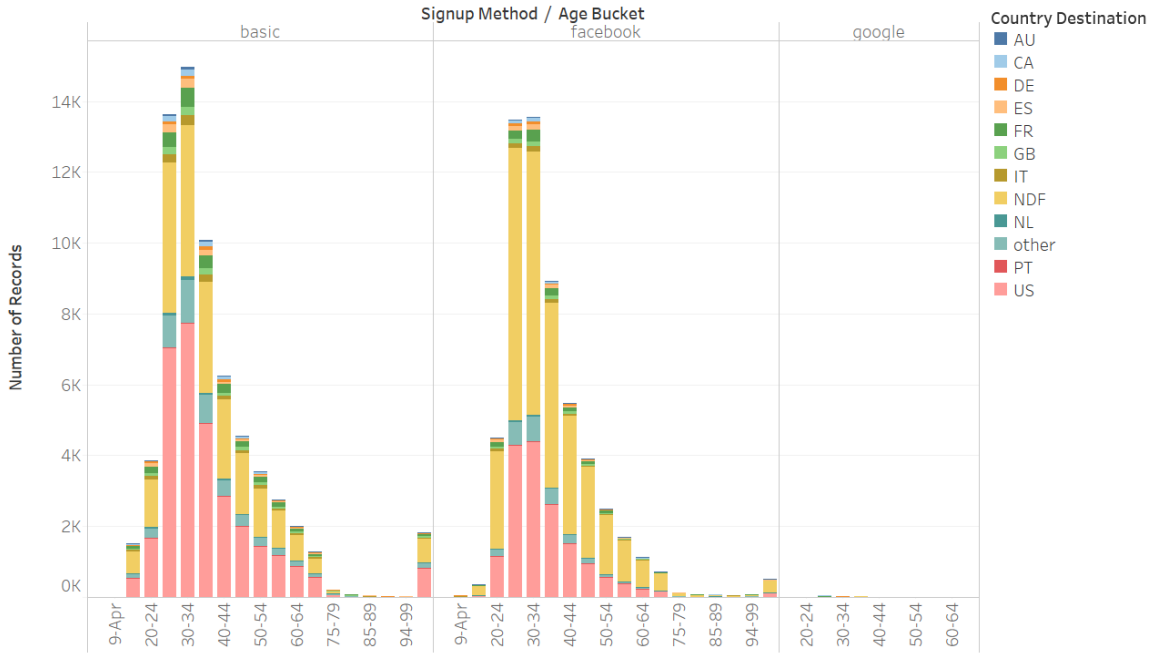


The plot of sum of Number of Records for Date First Booking Month. Color shows details about Country Destination. The data is filtered on Date First Booking Month, which has multiple members selected.

General Intuition:

Month of booking which is extracted from the Date of First booking in the train data set drives the analysis to a great degree too. It is observed from the analysis that in the month of June the highest amount of traveling happens and we may see hoe the number of gradually picks up during Spring up until Summer. We can also see how it drops from there on.

Signup method against age group



Sum of Number of Records for each Age Bucket broken down by Signup Method. Color shows details about Country Destination. The view is filtered on Age Bucket, which has multiple members selected.

General Intuition:

Signup method, Age buckets play a deciding role in deciding the country to a certain extent. We see from the above visualizations how through the "basic" method of booking, most bookings happen to US as the destination country. However, we see that while the number of bookings for the age group 20-24 is higher with "facebook" as a general signup method, the destination country when the signup method as "facebook" is mostly undecided or none. (NDF-No destination found)

Modeling

MODELS USED

3 different models were used apart from the Baseline model. They are:

- Naïve Bayes
- Random Forest
- Extreme Gradient Boosting

NAIVE BAYES ALGORITHM:

This consists of 2 types of probabilities that is calculated using the training data. Probability of each class and the conditional probability for each class for a given x value.

In this given problem, we use "gender","age","language","date_first_booking_month","timestamp_first_active_day","timestamp_first_active_year","timestamp_first_active_month","first_device_type","signup_method","signup_app" as the independent variables to build the model. We train the training data on this model.

After submission to Kaggle, we get an accuracy of 25.15%. This is not great compared to the accuracy that we obtain on a baseline model where the default country is "US". The accuracy of the baseline model is 29%.

The disadvantage of Naïve Bayes Algorithm is that it assumes each input variable as an independent variable. This assumption is too strong for a real world problem which is why it may not be useful to use this algorithm here.

RANDOM FOREST ALGORITHM:

Random forest algorithm is one of the most powerful Bootstrap aggregation or Bagging algorithms. Multiple samples are taken from the given training data and the models are built for each of these samples. In this case, multiple decision trees are created by introducing splits through randomness. The advantage of using a model such as this is that all the trees created are different, unique and accurate in their own sense. Thus, through aggregation, we get a better estimate of the true value.

In this given problem, we use "gender","language","age","first_device_type","signup_method","signup_app","date_first_booking_month",'affiliate_channel', 'affiliate_provider' as the variables to build the model.

After submission to Kaggle, we get an accuracy of 45.048% which is again quite low. This may be due to the outliers in Age which could have been cleaned earlier during the pre-processing stage.

EXTREME GRADIENT BOOSTING ALGORITHM

Extreme gradient boosting is more efficient than gradient boosting framework. It is almost 10 times faster and consists of both linear model solver and tree learning algorithms – thus giving us the best of both worlds. Extreme Gradient Boosting works only with numeric vectors and thus data preprocessing is of utmost importance for the implementation of this model. It also consists of One-Hot-Encoding features which is an alternate method of feature creation/feature engineering.

In this problem, we include the Evaluation criteria given to us. Normalized Discount Cumulative Gain. NDCG basically measures the performance of a recommendation system. This measurement is based on its graded relevance of recommended entities.

After submission to Kaggle, we get an accuracy of 84.94%. Again, by recoding the Age outliers, we could have got a higher accuracy.

Evaluation and Summary

From the above described models we see that Extreme Gradient Boosting provides us with the highest accuracy of 84.94%. The accuracy of all the three models applied here fair as below:

Naïve Bayes < Random Forest < Extreme Gradient Boosting.

MEASURES FOR A HIGHER ACCURACY:

The reason for Extreme Gradient Boosting working better than the other two was due to the underlying machine learning algorithm in its implementation and also the inclusion of the Evaluation criteria during the execution. Also, if multiple layers of machine learning algorithm were used, it would have resulted in a higher accuracy.

LEARNINGS FROM THE CHALLENGE:

- Visualizations greatly assisted in the right choice of predicting variables.
- It is always not important to impute missing values. Sometimes, there may be a reason for the missing values and we may unnecessarily create a bias by imputation.
- Using multiple layer of algorithms, which works on the previous model built is currently my new area of interest.

CONCLUSION

Overall, from this challenge we learned that Age, Gender, Month of booking, Signup App and Signup Method greatly decide the choice of the destination country for the new users. While sessions data and the user data have very weak correlation, it may be worth taking a dive in it to understand how it affects the trends.

Code

```
#DATA EXPLORATION AND CLEANING

#Reading the train file
train <- read.csv('C:/Users/USER/Documents/Tech fundamentals
assignments/Project/train_users_2.csv')
test <- read.csv('C:/Users/USER/Documents/Tech fundamentals
assignments/Project/test_users.csv')
head(train)
head(test)
#Reading the sessions file
sessions<-read.csv('C:/Users/USER/Documents/Tech fundamentals
assignments/Project/sessions.csv')
head(sessions)
#Identifying only unique ids from sessions to check the number of times they have
occured
length(unique(sessions$user_id))
length(train$id)
library(plyr)
#Reorganizing the sessions frame with only 2 columns (user id and the number of
occurences)
sessions_1<-ddply(sessions,.(user_id),nrow)
sessions_1
#Transferring it to a fresh file
write.csv(sessions_1,file="C:/Users/USER/Documents/Tech fundamentals
assignments/Project/sessions_1.csv")

#Renaming the label in sessions from "user_id" to "id"
colnames(sessions_1)[which(names(sessions_1) == "user_id")] <- "id"
colnames(sessions_1)[which(names(sessions_1) == "V1")] <- "Occurence"
head(sessions_1)
#Combining both frames and writing to a new file
train_combined<-merge(train,sessions_1,by="id",all.x=TRUE)
test_combined<-merge(test,sessions_1,by="id",all.x=TRUE)
write.csv(train_combined,file="C:/Users/USER/Documents/Tech fundamentals
assignments/Project/train_combined.csv")
write.csv(test_combined,file="C:/Users/USER/Documents/Tech fundamentals
assignments/Project/test_combined.csv")
```

```
#Breaking down the "date_account_created" and "date_first_booking" into year and
month
date_account_created_year<-as.numeric(substr(train_combined$date_account_created, 7,
8))
date_account_created_month<-as.numeric(substr(train_combined$date_account_created,
1, 2))
date_first_booking_year<-as.numeric(substr(train_combined$date_first_booking, 7, 8))
date_first_booking_month<-as.numeric(substr(train_combined$date_first_booking, 1, 2))

date_account_created_year_test<-
as.numeric(substr(test_combined$date_account_created, 7, 8))
date_account_created_month_test<-
as.numeric(substr(test_combined$date_account_created, 1, 2))
date_first_booking_year_test<-as.numeric(substr(test_combined$date_first_booking, 7,
8))
date_first_booking_month_test<-as.numeric(substr(test_combined$date_first_booking, 1,
2))

timestamp_first_active_year =
as.numeric(substr(as.character(train_combined$timestamp_first_active), 1, 4))
timestamp_first_active_month=
as.numeric(substr(as.character(train_combined$timestamp_first_active), 5, 6))
timestamp_first_active_day =
as.numeric(substr(as.character(train_combined$timestamp_first_active), 7, 8))

timestamp_first_active_year_test =
as.numeric(substr(as.character(test_combined$timestamp_first_active), 1, 4))
timestamp_first_active_month_test=
as.numeric(substr(as.character(test_combined$timestamp_first_active), 5, 6))
timestamp_first_active_day_test =
as.numeric(substr(as.character(test_combined$timestamp_first_active), 7, 8))

write.csv(train_combined,file="C:/Users/USER/Documents/Tech fundamentals
assignments/Project/train_combined.csv")
write.csv(test_combined,file="C:/Users/USER/Documents/Tech fundamentals
assignments/Project/test_combined.csv")

#Saving it as a new column in train_combined
train_combined$date_first_booking_year<-date_first_booking_year
train_combined$date_first_booking_month<-date_first_booking_month
test_combined$date_first_booking_year<-date_first_booking_year_test
test_combined$date_first_booking_month<-date_first_booking_month_test
```

```
train_combined$date_account_created_year<-date_account_created_year
train_combined$date_account_created_month<-date_account_created_month
test_combined$date_account_created_year<-date_account_created_year_test
test_combined$date_account_created_month<-date_account_created_month_test

train_combined$timestamp_first_active_year<-timestamp_first_active_year
train_combined$timestamp_first_active_month<-timestamp_first_active_month
train_combined$timestamp_first_active_day<-timestamp_first_active_day

test_combined$timestamp_first_active_year<-timestamp_first_active_year_test
test_combined$timestamp_first_active_month<-timestamp_first_active_month_test
test_combined$timestamp_first_active_day<-timestamp_first_active_day_test

#MODELING AND CHECKING ACCURACY BY DIVIDING TRAIN INTO "trainAirbnb"
AND "testAirbnb"

set.seed(1234)
ind <- sample(2, nrow(train_combined), replace=TRUE, prob=c(0.7, 0.3))
#Select our training data
trainAirbnb <- train_combined[ind==1,]
testAirbnb<-train_combined[ind==2,]
#Selecting the independent variables
myvars<-
c("gender","age","language","date_first_booking_month","timestamp_first_active_day","ti
mestamp_first_active_year","timestamp_first_active_month","first_device_type","signup_
method","signup_app")
X<-trainAirbnb[myvars]
Y<-trainAirbnb$country_destination

#This is for training the entire set that will be used to fit for the test_sers csv
X_final<-train_combined[myvars]
Y_final<-train_combined$country_destination

library(e1071)
#Model number 1 : Applying the naive bayes model
fit<-naiveBayes(Y~.,data=X)
predicted=predict(fit,testAirbnb)
predicted
predicted_nb=predict(fit,testAirbnb,type="raw")
head(predicted_nb)
```

```
testAirbnb$pred_results<-predicted
count<-0
for(i in 1:nrow(testAirbnb)){
  if(testAirbnb$pred_results[i]==testAirbnb$country_destination[i])
    count=count+1
  else{}
}
count
nrow(testAirbnb)
accuracy_train = (count/nrow(testAirbnb))*100
accuracy_train

#Predicting values for test_users.csv
fit<-naiveBayes(Y_final~.,data=X_final)
predicted_nb=predict(fit,test_combined,type="raw")
predictions_nb <- as.data.frame(matrix(predicted_nb, nrow=12))
rownames(predictions_nb) <- c('AU','CA','DE','ES','FR','GB','IT','NDF','NL','other','PT','US')
predictions_top5 <- as.vector(apply(predictions_nb, 2, function(x) names(sort(x)[12:8])))
ids<-test_combined$id
id_mtx <-  matrix(ids, 1)[rep(1,5), ]
id_nb<-c(id_mtx)
submission <- NULL
submission$id <- id_nb
submission$country <- predictions_top5
submission <- as.data.frame(submission)
head(submission)
write.csv(submission, "C:/Users/USER/Documents/Tech fundamentals
assignments/Project/submission_NB_test.csv", quote=FALSE, row.names = FALSE)

#End of Naive Bayes.
#Model number 2 : Applying Random forest model

train_combined <- read.csv('C:/Users/USER/Documents/Tech fundamentals
assignments/Project/train_combined.csv')
set.seed(1234)
ind <- sample(2, nrow(train_combined), replace=TRUE, prob=c(0.7, 0.3))
#Select our training data
trainAirbnb <- train_combined[ind==1,]
testAirbnb<-train_combined[ind==2,]
head(trainAirbnb)

#Selecting the independent variables
```

```
myvars_rf<-
c("gender","language","age","first_device_type","signup_method","signup_app","date_first_
booking_month",'affiliate_channel', 'affiliate_provider')

M<-trainAirbnb[myvars_rf]
N<-trainAirbnb$country_destination
fit_rf1<-randomForest(N~.,data=M)
predicted_rf<-predict(fit_rf1,final_test,type="prob")
testAirbnb$pred_results_rf<-predicted_rf
count_rf<-0
for(i in 1:nrow(testAirbnb)){
 if(testAirbnb$pred_results_rf[i]==testAirbnb$country_destination[i])
   count_rf=count_rf+1
 else{}
}
count_rf
nrow(testAirbnb)
accuracy_train_rf = (count_rf/nrow(testAirbnb))*100
accuracy_train_rf


#Prediction of test_users.csv
test_combined <- read.csv('C:/Users/USER/Documents/Tech fundamentals
assignments/Project/test_combined.csv')
test_combined$is_test<-1
train_combined$is_test<-0
test_combined$country_destination<- -1

set.seed(1234)
ind <- sample(2, nrow(train_combined), replace=TRUE, prob=c(0.7, 0.3))
#Select our training data
trainAirbnb <- train_combined[ind==1,]
testAirbnb<-train_combined[ind==2,]

#Ensuring both test and train have the same levels in factor variables
total_combined<-rbind(train_combined,test_combined)
test_combined[is.na(test_combined)]<-1

for (f in 1:length(names(total_combined))) {
 levels(train_combined[, f]) <- levels(total_combined[, f])
}
head(total_combined)
```

```
M<-trainAirbnb[myvars_rf]
N<-trainAirbnb$country_destination
M[is.na(M)]<- -1

fit_rf_final<-randomForest(N~.,data=M)
final_test<-subset(total_combined,is_test==1)
final_test[is.na(final_test)]<- -1
head(final_test)
final_test$date_first_booking=NULL

predicted_rf_final<-predict(fit_rf_final,final_test,type="prob")
predicted_rf_final

predictions <- as.data.frame(matrix(predicted_rf_final, nrow=12))

rownames(predictions) <- c('AU','CA','DE','ES','FR','GB','IT','NDF','NL','other','PT','US')
ids<-NULL
#Random forest algorithm and extracting top 5 predictions
predictions_top5 <- as.vector(apply(predictions, 2, function(x) names(sort(x)[12:8])))
ids<-final_test$id
id_mts <-  matrix(ids, 1)[rep(1,5), ]
idf<-c(id_mts)
submission<-NULL
submission$id<-idf
submission$country<-predictions_top5
submission <- as.data.frame(submission)
head(submission)
write.csv(submission, "C:/Users/USER/Documents/Tech fundamentals
assignments/Project/submission_RF_test.csv", quote=FALSE, row.names = FALSE)

#End of random forest model.
#Model number 3 : Applying Xgboost model

# load libraries
library(xgboost)
library(readr)
library(stringr)
library(caret)
library(car)

set.seed(1)
```

```
# load data
df_train = read.csv("C:/Users/USER/Documents/Tech fundamentals
assignments/Project/train_users_2.csv")
df_test = read.csv("C:/Users/USER/Documents/Tech fundamentals
assignments/Project/test_users.csv")
labels = df_train['country_destination']
df_train = df_train[-grep('country_destination', colnames(df_train))]
df_train[is.na(df_train)]<--1
df_test[is.na(df_test)]<--1

ndcg <- function(preds, dtrain) {

  labels <- getinfo(dtrain,"label")
  num.class = 12
  pred <- matrix(preds, nrow = num.class)
  top <- t(apply(pred, 2, function(y) order(y)[num.class:(num.class-4)]-1))

  x <- ifelse(top==labels,1,0)
  dcg <- function(y) sum((2^y - 1)/log(2:(length(y)+1), base = 2))
  ndcg <- mean(apply(x,1,dcg))
  return(list(metric = "ndcg5", value = ndcg))
}

labels
# combine train and test data
final_all = rbind(df_train,df_test)
# remove date_first_booking
final_all = final_all[-c(which(colnames(final_all) %in% c('date_first_booking')))]
# replace missing values
final_all[is.na(final_all)] <- -1
is.na(final_all)
# This R script is based on  indradenbakker's R script
# I customized eval_metric ndcg5 so that it is much easier to monitor ndcg value.

# load libraries
set.seed(1)



# split date_account_created in year, month and day
dac = as.data.frame(str_split_fixed(final_all$date_account_created, '-', 3))
final_all['dac_year'] = dac[,1]
```

```
final_all['dac_month'] = dac[,2]
final_all['dac_day'] = dac[,3]
final_all = final_all[,-c(which(colnames(final_all) %in% c('date_account_created')))]

# split timestamp_first_active in year, month and day
final_all[,'tfa_year'] = substring(as.character(final_all[,'timestamp_first_active']), 1, 4)
final_all['tfa_month'] = substring(as.character(final_all['timestamp_first_active']), 5, 6)
final_all['tfa_day'] = substring(as.character(final_all['timestamp_first_active']), 7, 8)
final_all = final_all[,-c(which(colnames(final_all) %in% c('timestamp_first_active')))]

# clean Age by removing values
#final_all[final_all$age < 14 | final_all$age > 100,'age'] <- -1


# one-hot-encoding features
features = c('gender', 'signup_method',  'language', 'signup_app', 'first_device_type')
dummies <- dummyVars(~ gender + signup_method + language  + signup_app +
first_device_type , data = final_all)
final_all_ohe <- as.data.frame(predict(dummies, newdata = final_all))
final_all_combined <- cbind(final_all[,-c(which(colnames(final_all) %in%
features))],final_all_ohe)

# split train and test
X = final_all_combined[final_all_combined$id %in% df_train$id,]
data.matrix(X[!is.finite(X)])<--1
X[is.na(X)]<--1
y <- recode(labels$country_destination,"'NDF'=0; 'US'=1; 'other'=2; 'FR'=3; 'CA'=4; 'GB'=5;
'ES'=6; 'IT'=7; 'PT'=8; 'NL'=9; 'DE'=10; 'AU'=11")
X_test <- final_all_combined[final_all_combined$id %in% df_test$id,]
y


set.seed(1)
# train xgboost
xgb <- xgboost(data = data.matrix(X[-1]),
            label = as.numeric(as.character(y)),
            eta = 0.1,
             missing=NaN,
            max_depth = 8,
            nround=20,
            subsample = 0.7,
            colsample_bytree = 0.8,
```

```
            seed = 1,
            eval_metric = ndcg5,
            objective = "multi:softprob",
            num_class = 12,
            nthread = 3
)

y_pred <- predict(xgb, missing=NaN,data.matrix(X_test))

# extract the 5 classes with highest probabilities
predictions <- as.data.frame(matrix(y_pred, nrow=12))
rownames(predictions) <- c('NDF','US','other','FR','CA','GB','ES','IT','PT','NL','DE','AU')

predictions_top5 <- as.vector(apply(predictions, 2, function(x) names(sort(x)[12:8])))
ids<-final_test$id
id_mts <-  matrix(ids, 1)[rep(1,5), ]
idf<-c(id_mts)
submission<-NULL
submission$id<-idf
submission$country<-predictions_top5
submission <- as.data.frame(submission)

# generate submission file
submission <- as.data.frame(submission)
write.csv(submission, "C:/Users/USER/Documents/Tech fundamentals
assignments/Project/submission_xgboost.csv", quote=FALSE, row.names = FALSE)
```