User Engagement Analysis

Hans Peter Ndeffo

Showwcase

September 14, 2020

Abstract

In this project, we aim at **investigating criterias** and **defining metrics to measure user engagement**. In general, there are several components involve when assessing user involvement in a platform but according to the dataset that this project is based on they are: **session_id, customer_id, login_date, projects_added, likes_given, comment_given, inactive_status, bug_occured, session_projects_added, session_likes_given, session_comments_given, inactive_duration, bugs_in_session, session_duration**. However, not all of them are important as we will see in this analysis. The following process will be used in order to filter non-relevant variables when trying to identify highly or lowly active users using the following steps:

- Import showwcase sessions' dataset.
- Display an overview of the dataset to find missing values and check spelling mistakes. Also, because the dataset has about 300 rows, there is no need to use Apache Spark which is a big data computational tool.
- Replace TRUE and FALSE with 1s and 0s. This will help when counting and achieving other computations.
- Simplify the model. We will assume that session_id, login_date, session_likes_given are irrelevant for this analysis. Also, we will create a new variable called active_session_duration which is the difference between the total session duration and inactive_duration. Therefore, delete session duration and inactive_duration will be deleted.
- Determine thresholds for session duration and login frequency. These decision boundaries will help decide whether a user is engaged or not.
- Setup the random seed for sampling. The dataset will be split into 80% for training and 20% for testing.
- Use Random Forest, Logistic Regression, KNN to predict user engagement.
- Identify trends by determining important features
- Determine model effectiveness and predicting power by computing training error and testing error.
- Compare the different models

Introduction

In this age where technology dominates everything, understanding customers involvement in any given platform has become a real challenge. **What are the most frequent features by which we can help label highly and least active users**? Solving this problem can have significant economic benefits through better-targeted ads campaigns and positive feedbacks from users. Also, this might demonstrate that are certain patterns through which we can determine which customers are more likely to be engaged. In this situation, we will first use association rules to evaluate what are the most frequent patterns. While looking for the right algorithm to use, there are certain procedures to follow to remain consistent and precise in any analysis. The outcome of those procedures will determine which machine learning algorithm suit best the problem.

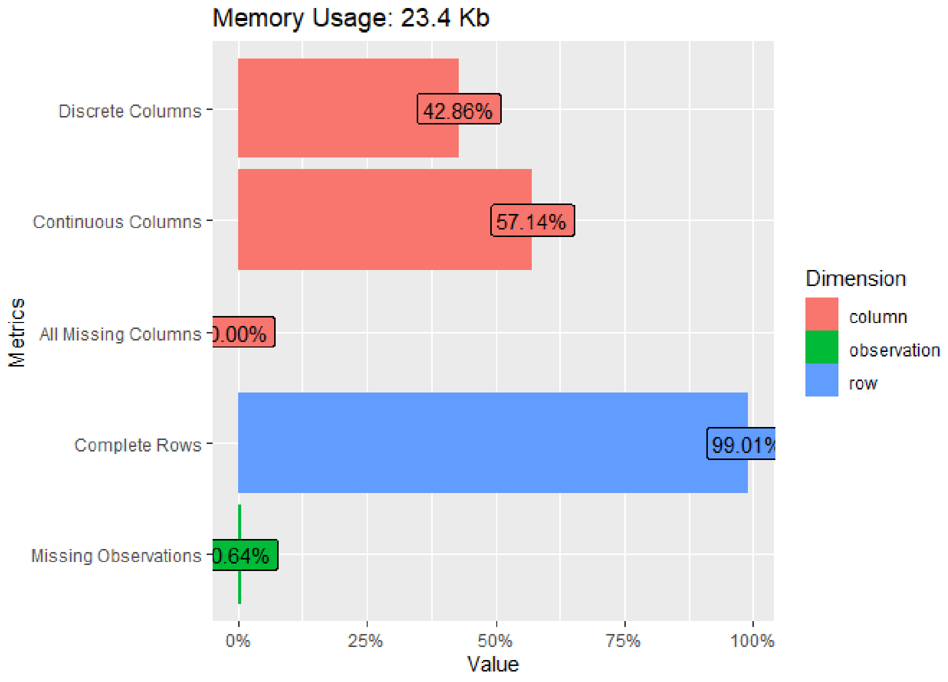
Pre-requisites

Step 1: Read customers dataset' file and load the required libraries

Step 2: Show an overview and a sample of the dataset

Having an overview of the dataset structure is extremely important because this could show the data distribution and missing values. For example, to evaluate whether the dataset is balanced or imbalanced. A dataset is balanced if it follows the following criterias: If there is n distinct preferably discrete output, Then the data should be evenly distributed among each output. In that case, the probability p of any output i for example to occur is approximatively p =$\frac{1}{n}$. However, in this case, we do not know how many users can be labeled as engaged and not. This scenario implies only 2 outcomes because we assume in general that users can be engaged or not. This assumption will be carried on throughout this analysis to simplify the model.

The plot below count the missing values and display how many discrete and continuous columns.



As we can see there some missing values about 0.64% in data. Show all missing data

```
users_data[rowSums(is.na(users_data)) > 0,]
```

```
##     ï..session_id customer_id login_date projects_added likes_given
## 36         862128       23404   10/26/19           TRUE       FALSE
## 301            NA          NA                        NA          NA
## 302            NA          NA                        NA          NA
##     comment_given inactive_status bug_occured session_projects_added
## 36           TRUE            TRUE       FALSE                      2
## 301            NA              NA          NA                     NA
## 302            NA              NA          NA                     NA
##     session_likes_given session_comments_given inactive_duration
## 36                   NA                      2              1120
## 301                  NA                     NA                NA
## 302                  NA                     NA                NA
##     bugs_in_session session_duration
## 36                0               95
## 301              NA               NA
## 302              NA               NA
```

Since missing data at row 301 and 302 have all columns empty and are the end of the file, we can just delete those rows from the dataset. However, at row 36 we only see one missing value located at **session_likes_given**. That value can be replaced by 0 because any given user has likes or not.

Check the spelling mistake in the column names

```
names(users_data)
```

```
##  [1] "ï..session_id"          "customer_id"            "login_date"
##  [4] "projects_added"         "likes_given"            "comment_given"
##  [4] "projects_added"         "likes_given"              "comment_given"
##  [7] "inactive_status"        "bug_occured"              "session_projects_added"
## [10] "session_likes_given"    "session_comments_given"  "inactive_duration"
## [13] "bugs_in_session"        "session_duration"
```

Fix session id name

```
names(users_data)[names(users_data)== names(users_data)[1]] <- "session_id"
```

After the data are cleaned we need to change the TRUE and FALSE to 1 and 0. This will help during computations because characters values such as “TRUE” can not be computed.

Check if TRUE and FALSE have been replaced by 1s and 0s

session_id	customer_id	login_date	projects_added	likes_given	comment_given	inactive_status	bug_occured	session_projects_added	session_likes_given	session_comments_given	inactive_duration	bugs_in_session	session_duration
624205	80746	10/30/19	0	1	1	1	0	0	24	3	1146	0	1564
624241	24520	10/30/19	1	1	1	1	0	2	3	5	133	0	1766
111002	32047	10/30/19	1	1	1	1	0	1	5	5	1571	0	2230
545113	23404	10/30/19	1	1	1	0	0	1	10	21	0	0	633
750269	40235	10/30/19	1	1	0	1	0	3	16	0	1405	0	1679

Step 3: Simplify the model

If the **inactive duration** is greater than session duration for some cases that implies that inactive duration **included** a measure of **log out + login time**. Which might be caused by bug or connexion issues. On the other hand session duration only measures the login time.So the difference measures the active time.

The plot below better illustrates those difference between active session, session duration and inactive duration over time

Add active session duration to the dataset

Remove session duration and inactive duration

Show a sample after binding and deletion of certain columns

customer_id	projects_added	likes_given	comment_given	inactive_status	bug_occured	session_projects_added	session_likes_given	session_comments_given	bugs_in_session	active_session_duration
80746	0	1	1	1	0	0	24	3	0	418
24520	1	1	1	1	0	2	3	5	0	1633
32047	1	1	1	1	0	1	5	5	0	659

Step 4: Determining a threshold for active users

We need now decide a threshold for session duration that will define is the user engaged or not. We will use the **average duration** as a threshold for active and not engaged users. Lets declare a variable called user_engagement. That variable reflects whether or not an user is engaged if his active session duration is above the threshold. **This approach assumes** for a given session that the **active session duration is the most determinant factor to evaluate an user engagement**.

```
# This variable has TRUE or FALSE values which will be represented by 1s and 0s
user_engagement <- c()
```

Add user engagement values to the dataset

Step 5: Determining other relevant factors to predict user engagement

Here we will **ignore the active session duration as a variable** and **try to find other important factors**. In this dataset, since there is no obvious relationship between data, we will use several machine learning algorithm and plot their perfomance. The dependent variable to predict will be user_engagement and all the other variables will be considered as independent.

Split the dataset into 80% training and 20% testing

Step 6: Predict test and train data to test model effectiveness

Random Forest

The model effectiveness is 65% accuracy

Get important features from the random forest model

```
##                        MeanDecreaseGini
## customer_id                   23.518306
## projects_added                 2.743130
## likes_given                    3.231945
## session_projects_added         9.410683
## inactive_status               25.107729
## comment_given                  2.987632
## bug_occured                    5.084175
## session_likes_given           19.031958
## session_comments_given        12.630227
```

The important features are:

- **customer_id** counts for about **23.51%**
- **inactive_status** use by the customer counts for **25.10%**
- **session_likes_given** influence the model for about **19.03%**
- **session_comment_given** influence the model for about **12.63%**
- **session_project_added** influence the model for about **9.41%**
- **bug_occured** influence the model for about **5.08%**

Any feature that influences the model by less than 5% will be rejected because p value = 5%.

Determining important features using a logistic regression model

The assumption using this model is the user engagement is linearly depend on all the other independent variables such customer_id, inactive_status. This model might suit this problem because we face only 2 outcomes(an user can be engaged or not). However, because we assume we do not know exactly which feature is important that we will have to add more variables(higher dimensionality) to the model. This might reduce the model accuracy. The logistic regression will be define as follows

Fixing a threshold for logistic regression.

We expect the logistic regression model to perform potentially poorly because of there are too many variables involves and also due the fact we have to choose manually a random threshold. In this case, we will use the mean.

Computing testing error for logistic regression

The **testing error for the logistic regression** is **30%**

Using the KNN neighbors

Knn can be tested because this analysis uses supervised learning. We will compute knn with different parameters k and select the one that gives the highest accuracy. The maximum value of k has to be less than the number of observations or rows in this dataset which is about 300.

The code below defines different values of k to be tested

This function computes the accuracy of each model

```
accuracy = function(actual, predicted) {
  mean(actual == predicted)
}
```

use a matrix store the model by saving its accuracy and k

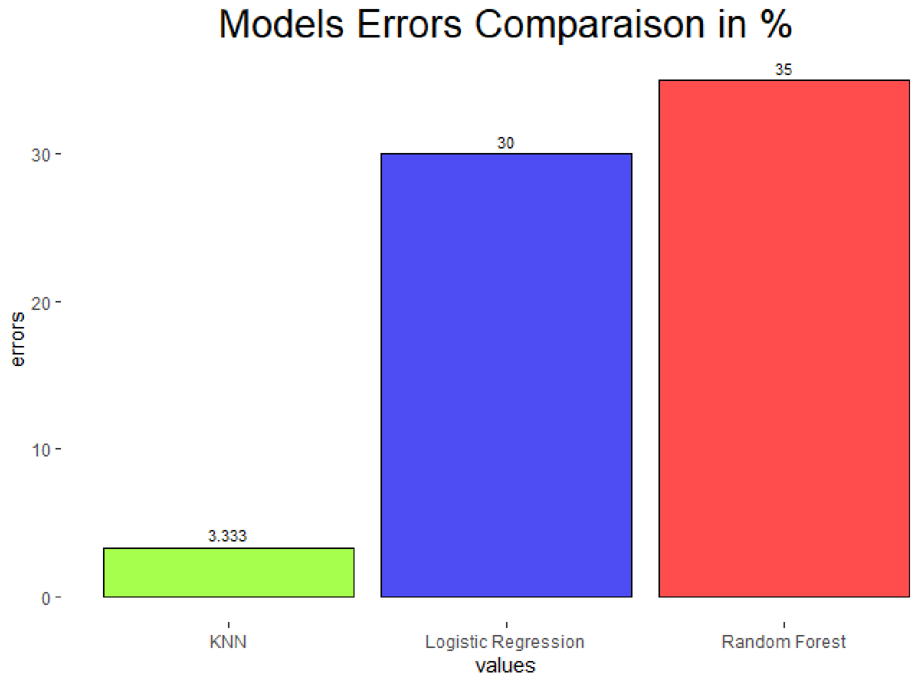
For all k, compute the predicted value

Now we find the model with highest accuracy

With **Knn we get an acccuracy of 96.66% with k = 1**. Therefore, the error is:

Step 7: Compare the different models

```
## No summary function supplied, defaulting to `mean_se()`
```



Conclusion

In this project, **KNN was the best model in this analysis with an accuracy of 96.66%**. However, due to time constraints when using KNN, we are not able to determine exactly what are the most significant features and the correlations between them. Nonetheless, **we are sure** about **67.5% based on the average accuracy** from the **random forest** and **logistic models** that the **most relevant features** are **customer_id**, **inactive_status**, **session_likes_given**, **session_comment_given**, **session_project_added**, **bug_occured**. Those variables define **“engaged customers”**. For instance, with customer_id, we can track the frequency at which a customer login overtime during different sessions. This implies we can increase the predicting power of the random forest and logistic models given a set of input features such as **customer_id**, **session_likes_given** by 32.5% in the ideal situation by building a much more complex model. However, the tradeoffs can be huge for such an approach which might not only cause **overfitting** but raise the computational cost and training time. In addition, a such model is likely to have less predicting power on new datasets.