<u>Comparison between long-term and short-term users of Supersense application</u>

Group: AC

<u>Github</u>: *https://github.com/avielbst/adv_programming_project/*

<u>Definitions</u>

In our research we will use the following definitions:
VI - Visual Impairments
Supersense - An application that supports people with VI by using camera and sounds
Long-term users - Users that used Supersense more than 6 times or more
Short-term users - Users that used Supersense 1-5 times

We decided to define long term users as 20 or more times after trying for 3, 5, 10, 15, 18
and realized that 20 sessions maximizes the variance between the two type of users
The data covers 25 days of usage and therefore 20 sessions seemed reasonable.

<u>Introduction</u>

The question we are trying to investigate is what is the difference between long-term users and short-term users in terms of quality of service from Supersense?
In our research we will compare the quality of service that long-term and short-term users had while using Supersense.

Over 440 million people worldwide face visual impairments and most of the tools
that offer help are expensive which makes them less viable to people with VI
Nowadays, companies are trying to develop new tools to help people with VI face daily tasks with more ease.
Supersense is an application made by Mediate and is supported by leading organizations like NSF and more.
Supersense is using a unique AI technology and a accessible user interface
We have a dataset that contains data about Supersense's users such as actions performed, usage time, results and more.

This analysis supposed to contribute in two main areas:

1. The development of the application; To understand better what features are most desired within their users.
2. The blind people by adding the features that they look for to help them face daily difficulties

By helping Supersense understand how to improve we can contribute to reduced inequalities (SDG 10)

This can be hard because of:

1. We suspect that some users in the data do not have VI and don't represent properly the designated people.
2. Given a successful detection by the application, we don't know for sure if this was the correct detection (detect object as A while it is B)

The closest prior research was predicting churn by using AI neural networks (Mostly in the telecom industry). “ because of its nested nonlinear structure, most deep learning models, including our model, are black boxes that, despite their good performance, do not provide information regarding the basis of

the predictions” (<a href="https://www.jmir.org/2021/1/e22184/"> Lifelog Data-Based Prediction Model of Digital Health Care App Customer Churn: Retrospective Observational Study </a>)
Also, their data was very different from ours.

To approach the problem, we first explored the data by using plots and statistics that helped us define what user is "Exprienced" and to distinguish between the two types of users. Afterwards we used features that seemed reasonable to fit a classifier model that predicts the whether a user is experienced or not. And tried to understand which features affecting the type of users the most.

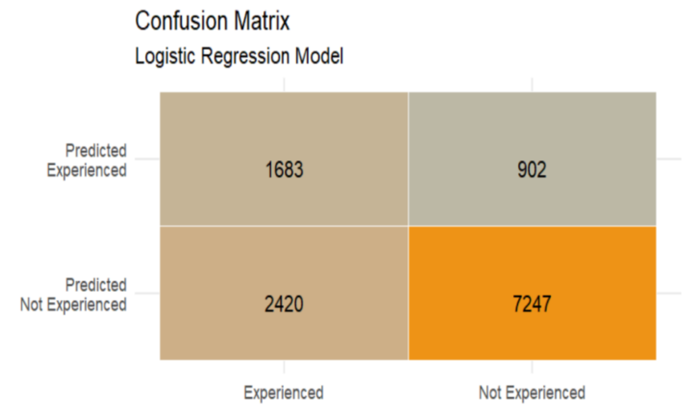
Data Overview

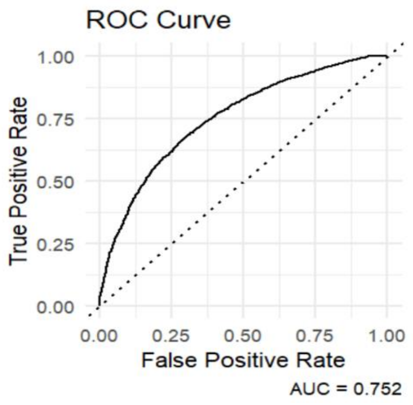
The dataset contains data from Supersense's log of users across 127 different countries that used the application between 21/01/06 to 21/02/01.
A session is a series of events that happened since the application opened and until it was closed (sessions included are 1 to 1200 seconds long).
An event is defined to be a specific action the user has initiated in the application.

Variables
event_name - `char` Defines the specific action used (session_start, explore, explore_end, select_object_event, detection_success, speech_recognition, speech_event, find_useful)
user_psuedo_id - `dbl` Unique id of the user.
mobile_brand_name - `char` Mobile brand used to run Supersense application
continent - `char` Continent in which the user has been while activating the application
answer - `char` Users' answer ("Yes"/"No"/"No idea") to the question "Was the search useful"
session_id - `char` Unique id for every session (from when the app starts running until it closes)
experienced - `char` User type. experienced if used 20 or more sessions
session_length - `dbl` Seconds passed from session start until it ended
speak - `char` The language the user is speaking (only 2 first letters of language column)

Methods and results

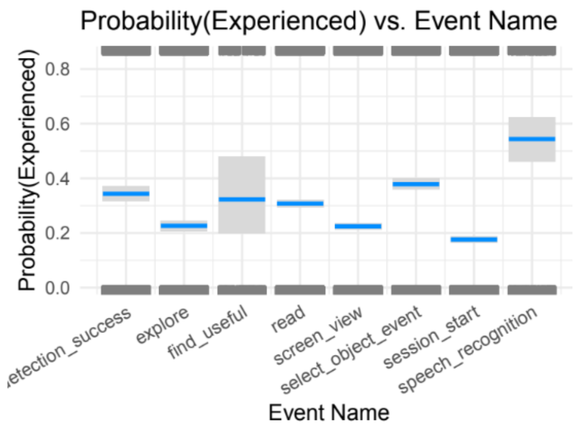
Through our analysis, we used a logistic regression model to predict if a user is experienced or not by using event_name, mobile_brand_name, answer, session_length and speak and analyzed its performance by using confusion matrix, ROC and plots.
We chose logistic regression because we wanted to classify the type of the user while understanding which of the features affected the prediction the most.

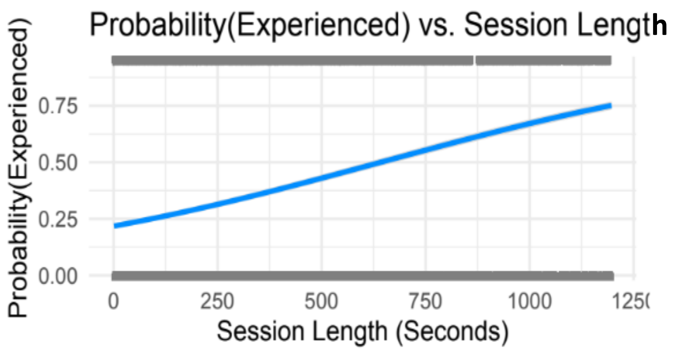


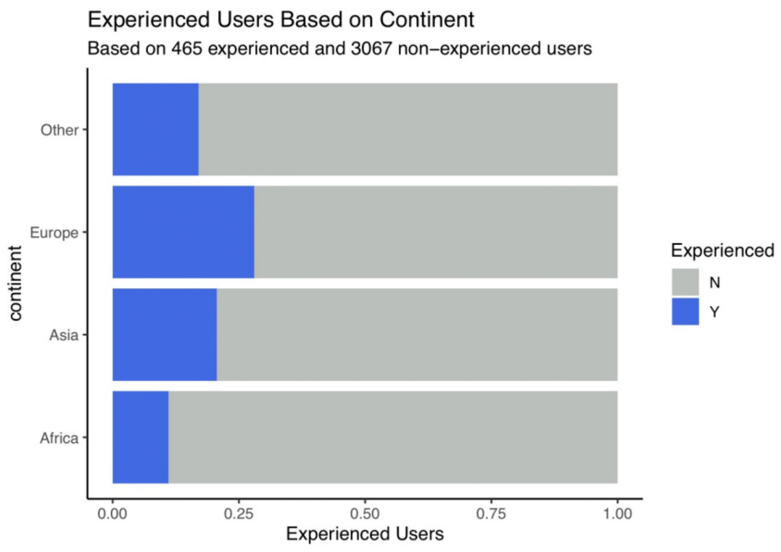


```
"Accuracy:0.729 , Precision: 0.410,  Recall: 0.651"
```

As we can see our model's precision is only at 41% which is not as we expected but still good considering the unbalance between experienced users compared to non-experienced. Also, the ROC indicates that our model's performance is good and most of its Positive predictions are true.
Below we can see that Speech Recognition event increases the probability that the action was made by an experienced user.







We found out the using "Speech Recognition" feature, using the application for long sessions or being in Europe positively affecting the probability of being an experienced user, although, since the amount of experienced users is very low compared to inexperienced ones it's still more likely the model will predict non-experienced even with these features.

<u>Limitations and Future Work</u>

During our research we faced two things that affected our results; Data covers only 25 days and most users are short-term with 1-2 sessions only. In future work, we recommend using data that covers longer period of time that includes more users – to reduce the bias. Also, the data is related to Android users only – expanding it to Apple devices can help achieving better understanding and results. Another limitation is that we can't know if a detection was successful or not.

Source code

```r
knitr::opts_chunk$set(echo = TRUE)
# install.packages("visreg")
library(visreg)
library(tidyverse)
library(tidymodels)
supersense <- read_csv("C:/Users/aavvi/Desktop/Adv_Programming/FinalProject/proposal/supersense_with_s

supersense <- supersense %>%
  select(-1,-6,-8,-9,-13,-14,-16,-18) %>%
  filter(session_length > 1) %>%
  filter(session_length < 1200)
# >>>> Create 'experienced' column <<<<
exp <- supersense %>%
    mutate(exp = 0) %>%
    group_by(session_id, user_pseudo_id) %>%
    count(user_pseudo_id) %>%
```

```r
func <- function(x) {
  output = "Y"
  if(x %in% exp$user_pseudo_id) {
    return(output)
  }
  else{
    output <- "N"
    return(output)
  }
}
func_V <- Vectorize(func)

supersense_exp <- supersense %>%
  mutate(experienced = func_V(user_pseudo_id))
# Fill missing values in 'answer'
supersense_exp$answer[is.na(supersense_exp$answer)] <- "not sure"

# Add column for speaking language
supersense_exp <- supersense_exp %>%
  mutate(speak = substr(language, 1,2))
# Split into test, train
data_for_reg <- supersense_exp %>%
  select("answer", "mobile_brand_name", "speak", "event_name", "session_length", "experienced") %>%
  filter(!(event_name %in% c("detection_end", "read_end", "session_end", "explore_end", "speech_event")

## 75% of the sample size
set.seed(123)
split <- initial_split(data_for_reg, prop = 0.70)

train_df <- training(split)
test_df <- testing(split)
# Logistic Regression
# Target > experienced
# Features > answer, mobile_brand_name, event_name, session_length, "speak"
model1 <- logistic_reg() %>%
  set_engine("glm")

rec <- recipe(
  experienced ~ answer + mobile_brand_name + event_name + session_length + speak,
  data=train_df)

exp_wf <- workflow() %>%
  add_model(model1) %>%
  add_recipe(rec)

exp_fit <- exp_wf %>%
fit(data = train_df)
```

```r
# Prediction
pred_test <- predict(exp_fit, test_df, type="prob") %>%
  bind_cols(test_df)

# >>> Confusion Matrix <<<<
```

```r
res <- pred_test %>%
  select(experienced, .pred_N, .pred_Y)

TP <- res %>%
  filter(experienced == 'Y' & .pred_Y > 0.5) %>%
  nrow()

TN <- res %>%
  filter(experienced == 'N' & .pred_N > 0.5) %>%
  nrow()

FP <- res %>%
  filter(experienced == 'N' & .pred_Y > 0.5) %>%
  nrow()

FN <- res %>%
  filter(experienced == 'Y' & .pred_N > 0.5) %>%
  nrow()

Real <- factor(c(0, 1, 0, 1))
Predicted <- factor(c(1, 1, 0, 0))
Y      <- c(TP, FP, FN, TN)
cm <- data.frame(Real, Predicted, Y)

ggplot(data =  cm, mapping = aes(x = Real, y = Predicted)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#BCB8A5", high = "#EE9316") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(
    title="Confusion Matrix",
    subtitle="Logistic Regression Model",
    x="",
    y="") +
  scale_x_discrete(labels=c("0"="Experienced", "1" = "Not Experienced")) +
  scale_y_discrete(labels=c("0"="Predicted\nNot Experienced", "1" = "Predicted\nExperienced"))
# Confusion matrix metrics:
accuracy <- (TP+TN)/(TP+TN+FP+FN)
precision <- TP/(TP+FN)
recall <- TP/(TP+FP)
F1 <- (2*TP)/(2*TP+FP+FN)
sprintf("Accuracy:%.3f , Precision: %.3f,  Recall: %.3f", accuracy, precision, recall)
# AUC calculation
AUC <- pred_test %>%
    roc_auc(
      truth = as.factor(experienced),
      .pred_Y,
      event_level = "second"
    )

auc <- sprintf("AUC = %.3f", AUC$.estimate)
# Plot ROC curve with AUC calculation on bottom right
```

```r
pred_test %>%
  roc_curve(
    truth = as.factor(experienced),
    .pred_Y,
    event_level = "second") %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal() +
  theme_minimal() +
  labs(
    title="ROC Curve",
    x = "False Positive Rate",
    y = "True Positive Rate",
    caption = auc
  )
# Create the addional model with 'speak' column as well
model2 <- glm (as.factor(experienced) ~ ., data = train_df, family = binomial)
# Plot probability for experienced user based on event name
visreg(model2, "event_name", gg=TRUE, scale="response") +
  labs(
    title = "Probability(Experienced) vs. Event Name",
    x = "Event Name",
    y = "Probability(Experienced)"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, vjust = 1, hjust=1))
visreg(model2, "session_length", gg=TRUE, scale="response") +
  labs(
    title = "Probability(Experienced) vs. Session Length",
    x = "Session Length (Seconds)",
    y = "Probability(Experienced)"
  ) +
  theme_minimal()
generalize_continents <- function(x){
  if(grepl("Asia", x)){
    return("Asia")
  }
  else if(grepl("Europe", x)){
    return("Europe")
  }
  else if(grepl("Africa", x)){
    return("Africa")
  }
  else{
    return("Other")
  }
}
generalize_continents <- Vectorize(generalize_continents)

tmp <- supersense_exp %>%
  distinct(session_id, session_length, sub_continent, experienced) %>%
  filter(!(sub_continent == "(not set)")) %>%
  filter(!(sub_continent %in% c("Australasia", "Carribean"))) %>%
  mutate(continent = generalize_continents(sub_continent))
cat(readLines("C:/Users/aavvi/Desktop/Adv_Programming/FinalProject/proposal/README.txt"), sep = '\n')
```