Homework 6

Group-12

Tushar Jayendra Mhatre, Roshini Talluru, Rahul Kataram

#Library imports

```
library(mice)

library(party)

library(tidyverse)

library(VIM)

library(pls)

library(glmnet)

library(caret)

library(earth)

library(car)

library(partykit)
```

1 Online retail sales prediction

(a) Preparation and modeling.

1. (a). (i) Data understanding: Generate a Data Quality Report. Also, choose at least two meaningful visualizations and/or analyses and explain their relevance.

#Reading Train and Test Datasets

```
Train<-read.csv("C:/Users/Tushar/Downloads/Train.csv/Train.csv",na.strings =
c("","NA"))
Test<-read.csv("C:/Users/Tushar/Downloads/Train.csv/Test.csv",na.strings =
c("","NA"))
```

#Splitting Train Data into numeric and Discrete

```
TrainNumeric <- Train %>%
  select_if(is.numeric)


TrainDiscrete <- Train[, sapply(Train, function(x) !is.numeric(x))]%>%
  select(-date)

#For Test Data
TestNumeric <- Test %>%
```

```
  select_if(is.numeric)

TestDiscrete <- Test[, sapply(Test, function(x) !is.numeric(x))]%>%
  #select_if(~ !is.numeric(.) && . != "date")
  select(-date)

#TestDiscrete <- Test %>%
#transmute_if(is.character, as_factor)
#select_if(is.character)

#glimpse(TrainDiscrete)
```

#Quantile Functions

```
Q1<-function(x,na.rm=TRUE) {
  quantile(x,na.rm=na.rm)[2]
}
Q3<-function(x,na.rm=TRUE) {
  quantile(x,na.rm=na.rm)[4]
}
```

#Train Nummeric Summary

```
TrainNumericSummary <- function(x){
  c(length(x), n_distinct(x),((n_distinct(x)/length(x))*100),
sum(is.na(x)),((sum(is.na(x))/length(x))*100), mean(x, na.rm=TRUE),
    min(x,na.rm=TRUE), Q1(x,na.rm=TRUE), median(x,na.rm=TRUE),
Q3(x,na.rm=TRUE),
    max(x,na.rm=TRUE), sd(x,na.rm=TRUE))
}

TrainNumericTableSummary <- TrainNumeric %>%
  summarize(across(everything(), TrainNumericSummary))

## Warning: Returning more (or less) than 1 row per `summarise()` group was
deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that
`reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

#Test

```
TestNumericSummary <- function(x){
  c(length(x), n_distinct(x),((n_distinct(x)/length(x))*100),
sum(is.na(x)),((sum(is.na(x))/length(x))*100), mean(x, na.rm=TRUE),
    min(x,na.rm=TRUE), Q1(x,na.rm=TRUE), median(x,na.rm=TRUE),
Q3(x,na.rm=TRUE),
```

```
    max(x,na.rm=TRUE), sd(x,na.rm=TRUE))
}

TestNumericTableSummary <- TestNumeric %>%
  summarize(across(everything(), TestNumericSummary))
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was
deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that
`reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

#Train

```
# View the structure of 'numericSummary'
#glimpse(numericSummary)
TrainNumericTableSummary <-cbind(
  stat=c("n","unique","Unique_percentage","missing","missing_Percentage",
"mean","min","Q1","median","Q3","max","sd"),
  TrainNumericTableSummary)
#glimpse(TrainNumericTableSummary)
```

#Test

```
TestNumericTableSummary <-cbind(
  stat=c("n","unique","Unique_percentage","missing","missing_Percentage",
"mean","min","Q1","median","Q3","max","sd"),
  TestNumericTableSummary)
#glimpse(TestNumericTableSummary)
```

#Train Numeric Data Report

```
TrainNumericSummaryFinal <- TrainNumericTableSummary %>%
  pivot_longer("sessionId":"revenue", names_to = "variable", values_to =
"value") %>%
  pivot_wider(names_from = stat, values_from = value)%>%
  #mutate(missing_pct = 100*missing/n,
  #unique_pct = 100*unique/n) %>%
  select(variable, n, missing,  unique, everything())

#glimpse(TrainNumericSummaryFinal)
```

#Test Numeric Data Report

```
TestNumericSummaryFinal <- TestNumericTableSummary %>%
  pivot_longer("sessionId":"newVisits", names_to = "variable", values_to =
"value") %>%
  pivot_wider(names_from = stat, values_from = value)%>%
```

```
  #mutate(missing_pct = 100*missing/n,
  #unique_pct = 100*unique/n) %>%
  select(variable, n, missing,  unique, everything())

glimpse(TestNumericSummaryFinal)

## Rows: 11
## Columns: 13
## $ variable           <chr> "sessionId", "custId", "visitStartTime",
"visitNumb…
## $ n                  <dbl> 69672, 69672, 69672, 69672, 69672, 69672,
69672, 69…
## $ missing            <dbl> 0, 0, 0, 0, 0, 0, 0, 67667, 11, 40357, 23485
## $ unique             <dbl> 69672, 47247, 69576, 284, 20653, 2, 2, 5, 155,
2, 2
## $ Unique_percentage  <dbl> 1.000000e+02, 6.781347e+01, 9.986221e+01,
4.076243e…
## $ missing_Percentage <dbl> 0.00000000, 0.00000000, 0.00000000, 0.00000000,
0.0…
## $ mean               <dbl> 4.744871e+12, 4.924171e+04, 1.485066e+09,
3.795915e…
## $ min                <dbl> 100000110, 1794, 1470035429, 1, 0, 0, 0, 1, 1,
1, 1
## $ Q1                 <dbl> 2.370975e+12, 2.550275e+04, 1.477563e+09,
1.000000e…
## $ median             <dbl> 4.757850e+12, 4.937150e+04, 1.484069e+09,
1.000000e…
## $ Q3                 <dbl> 7.107800e+12, 7.287100e+04, 1.492896e+09,
2.000000e…
## $ max                <dbl> 9.449600e+12, 9.628900e+04, 1.501657e+09,
2.840000e…
## $ sd                 <dbl> 2.721195e+12, 2.721195e+04, 9.051804e+06,
1.424728e…

library(knitr)
options(digits=3)
options(scipen=99)

#Train Numeric Data report
TrainNumericSummaryFinal %>% kable()
```

variable	n	missing	unique	Unique_percentage	missing_Percentage	mean	min	Q1	median	Q3	max	sd
sessionId	700	0	700	100.000	0.000	4708198750205.9	200000120	2328800000017	4688000000014	7079450000017	9449700000019	2732187943023.7

variable	n	missing	unique	Unique_percentage	missing_Percentage	mean	min	Q1	median	Q3	max	sd
	71		71			51		3	6	0	4	50
custId	70071	0	47249	67.430	0.000	48874.987	1795	25081	48673	72588	96290	27321.879
visitStartTime	70071	0	69951	99.829	0.000	1485110879.818	1470035066	1477552704	1484102061	1493099922	1501655863	9106581.658
visitNumber	70071	0	155	0.221	0.000	3.146	1	1	1	2	155	8.660
timeSinceLastVisit	70071	0	20970	29.927	0.000	256450.236	0	0	0	10375	30074517	1164717.354
isMobile	70071	0	2	0.003	0.000	0.229	0	0	0	0	1	0.420
isTrueDirect	70071	0	2	0.003	0.000	0.400	0	0	0	1	1	0.490
adwordsClickInfo.page	70071	68260	6	0.009	97.415	1.008	1	1	1	1	7	0.179
pagevi	7	8	1	0.221	0.011	6.304	1	1	2	6	469	11.69

variable	n	missing	unique	Unique_percentage	missing_Percentage	mean	min	Q1	median	Q3	max	sd
ews	0071		55									3
bounces	70071	40719	2	0.003	58.111	1.000	1	1	1	1	1	0.000
newVisits	70071	23944	2	0.003	34.171	1.000	1	1	1	1	1	0.000
revenue	70071	0	5850	8.349	0.000	10.165	0	0	0	0	15981	99.534

#Data Report for Non-Numeric Table #This function will work for Every column

```
getmodes <- function(v,type=1) {
  if(sum(is.na(v))==length(v)){
    return(NA)
  }
  tbl <- table(v)
  m1<-which.max(tbl)
  if (type==1) {
    return (names(m1)) #1st mode
  }
  else if (type==2) {
    return (names(which.max(tbl[-m1]))) #2nd mode
  }
  else if (type==-1) {
    return (names(which.min(tbl))) #least common mode
  }
  else {
    stop("Invalid type selected")
  }
}
```

```r
getmodesCnt <- function(v,type=1) {
  tbl <- table(v)
  m1<-which.max(tbl)
  if (type==1) {
    return (max(tbl)) #1st mode freq
  }
  else if (type==2) {
    return (max(tbl[-m1])) #2nd mode freq
  }
  else if (type==-1) {
    return (min(tbl)) #least common freq
  }
  else {
    stop("Invalid type selected")
  }
}
```

This function will run the get modes individually for every column and display

```r
getmodes_df <- function(df, type = 1) {
  modes_list <- list()  # Create an empty list to store modes for each column

  for (col in colnames(df)) {
    modes_list[[col]] <- getmodes(df[[col]], type)  # Apply getmodes to each
column
  }

  return(modes_list)
}
```

#getmodes_df(TrainDiscrete,type=1) #getmodes_df(TrainDiscrete,type=2) #getmodes_df(TrainDiscrete,type=-1)

```r
TrainDiscreteSummary <- function(x){
  c(length(x), n_distinct(x), sum(is.na(x)),  getmodes(x, type=1),
getmodesCnt(x, type =1),
    getmodes(x, type=2), getmodesCnt(x, type =2), getmodes(x, type= -1),
getmodesCnt(x, type = -1))
}

result1 <- lapply(TrainDiscrete, TrainDiscreteSummary)

## Warning in max(tbl[-m1]): no non-missing arguments to max; returning -Inf

## Warning in max(tbl[-m1]): no non-missing arguments to max; returning -Inf

result_matrix <- do.call(cbind, result1)
```

```
## Warning in base::cbind(...): number of rows of result is not a multiple of
## vector length (arg 21)

# Convert the matrix into a dataframe
TrainDiscreteTableSummary <- as.data.frame(result_matrix)

#test
result2 <- lapply(TestDiscrete, TrainDiscreteSummary)

## Warning in max(tbl[-m1]): no non-missing arguments to max; returning -Inf

## Warning in max(tbl[-m1]): no non-missing arguments to max; returning -Inf

result_matrix <- do.call(cbind, result2)

## Warning in base::cbind(...): number of rows of result is not a multiple of
## vector length (arg 21)

# Convert the matrix into a dataframe
TestDiscreteTableSummary <- as.data.frame(result_matrix)

# Assign the first vector as column names
#colnames(result_df) <- result_df[1, ]
#result_df <- as.data.frame(do.call(cbind, result1))

#TrainDiscreteTableSummary <- TrainDiscrete %>%
#summarize(across(everything(), TrainDiscreteSummary))
```

#Train Discrete Summary Report

```
TrainDiscreteTableSummary <-cbind(
  stat=c("n","unique","missing","1st mode", "first_mode_freq", "2nd mode",
"second_mode_freq", "least common", "least common freq"),
  TrainDiscreteTableSummary)


DiscreteFactorSummaryFinal <- TrainDiscreteTableSummary %>%
  pivot_longer("channelGrouping":"adwordsClickInfo.isVideoAd", names_to =
"variable", values_to = "value") %>%
  pivot_wider(names_from = stat, values_from = value) %>%
  mutate(across(c(2,3,4,6,8,10), as.double), missing_pct = 100*missing/n,
         unique_pct = 100*unique/n, freq_ratio = as.numeric(first_mode_freq)
/ as.numeric(second_mode_freq))%>%
  select(variable, n, missing, missing_pct, unique, unique_pct, freq_ratio,
everything())

## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `across(c(2, 3, 4, 6, 8, 10), as.double)`.
## Caused by warning:
## ! NAs introduced by coercion

#glimpse(DiscreteFactorSummaryFinal)
```

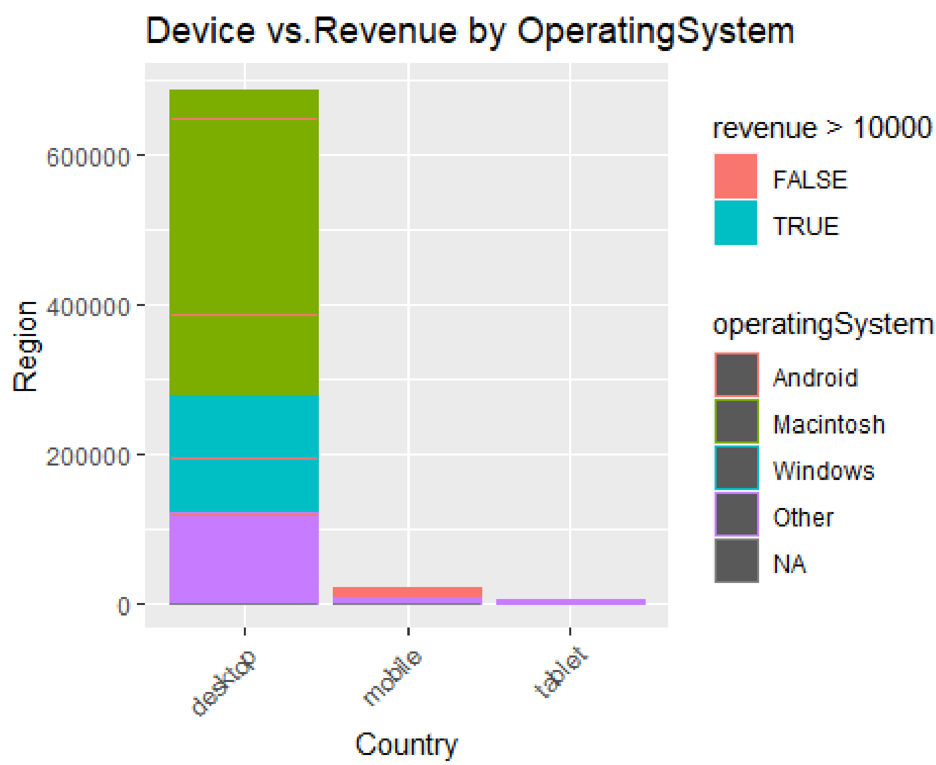
#Test Discrete Table summary report

```
#test
TestDiscreteTableSummary <-cbind(
  stat=c("n","unique","missing","1st mode", "first_mode_freq", "2nd mode",
"second_mode_freq", "least common", "least common freq"),
  TrainDiscreteTableSummary)
TestDiscreteFactorSummaryFinal <- TestDiscreteTableSummary %>%
  pivot_longer("channelGrouping":"adwordsClickInfo.isVideoAd", names_to =
"variable", values_to = "value") %>%
  pivot_wider(names_from = stat, values_from = value) %>%
  mutate(across(c(2,3,4,6,8,10), as.double), missing_pct = 100*missing/n,
         unique_pct = 100*unique/n, freq_ratio = as.numeric(first_mode_freq)
/ as.numeric(second_mode_freq))%>%
  select(variable, n, missing, missing_pct, unique, unique_pct, freq_ratio,
everything())

## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `across(c(2, 3, 4, 6, 8, 10), as.double)`.
## Caused by warning:
## ! NAs introduced by coercion

glimpse(TestDiscreteFactorSummaryFinal)

## Rows: 22
## Columns: 13
## $ variable          <chr> "channelGrouping", "browser",
"operatingSystem", "…
## $ n                 <dbl> 70071, 70071, 70071, 70071, 70071, 70071,
70071, 7…
## $ missing           <dbl> 0, 1, 307, 0, 85, 85, 85, 38485, 49183, 39028,
334…
## $ missing_pct       <dbl> 0.00000, 0.00143, 0.43813, 0.00000, 0.12131,
0.121…
## $ unique            <dbl> 8, 28, 16, 3, 6, 23, 177, 310, 73, 478, 5015,
184,…
## $ unique_pct        <dbl> 0.01142, 0.03996, 0.02283, 0.00428, 0.00856,
0.032…
## $ freq_ratio        <dbl> 2.03, 4.30, 1.01, 3.89, 3.10, 8.06, 12.14,
3.25, 2…
## $ `1st mode`        <chr> "Organic Search", "Chrome", "Macintosh",
"desktop"…
## $ first_mode_freq   <dbl> 27503, 51584, 23970, 53986, 42508, 38860,
36941, 1…
## $ `2nd mode`        <chr> "Social", "Safari", "Windows", "mobile",
"Asia", "…
## $ second_mode_freq  <dbl> 13528, 12007, 23707, 13868, 13697, 4823, 3044,
346…
## $ `least common`    <chr> "(Other)", "Apple-iPhone7C2", "Nintendo 3DS",
"tab…
```

```
## $ `least common freq` <dbl> 3, 1, 1, 2217, 888, 1, 1, 1, 1, 1, 1, 1, 4, 1,
735…
```

#Visualizations #We are performing two visualizations on train data.

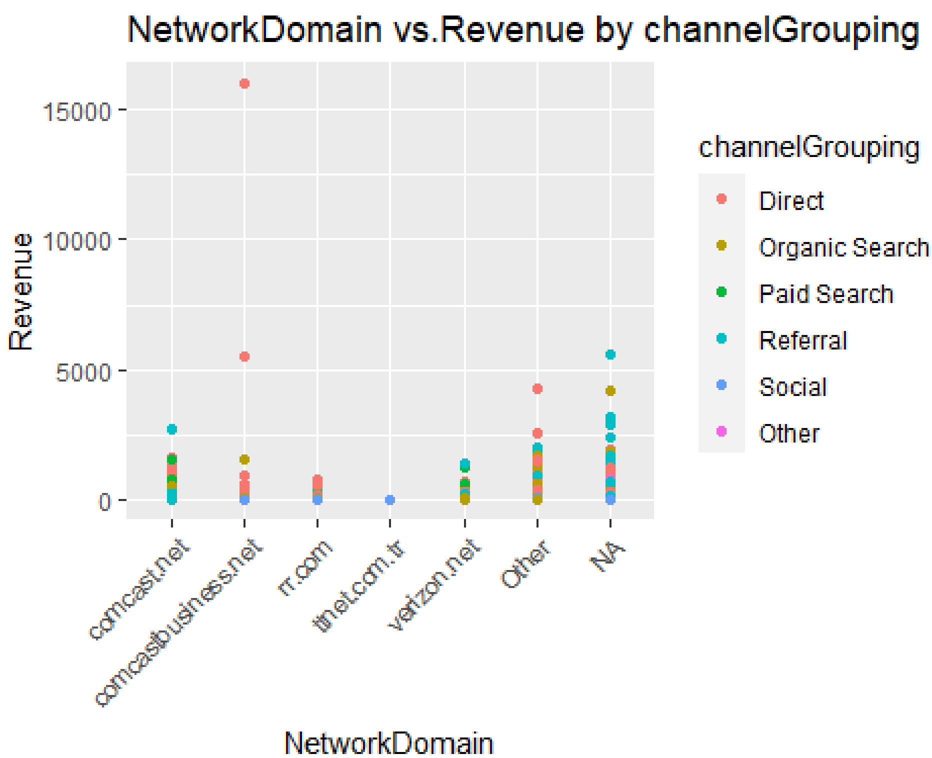
```
#Train Data
FilterTrain <- Train %>%
mutate(operatingSystem = fct_lump_n(operatingSystem, n = 3))

ggplot(FilterTrain, aes(x = deviceCategory, y = revenue, color =
operatingSystem, fill=revenue>10000 )) +
  geom_bar(stat="identity") +
  labs(title = "Device vs.Revenue by OperatingSystem",
       x = "Country", y = "Region") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



#Plot is generated for Device vs Revenue for different operating Systems. From this graph we can understand that the revenue is more for Desktop devices especially with Macintosh operating system and next windows operating system.The least revenue is for tablet devices. It is evident that revenue is highest for users using Desktop with Macintosh operating system whereas least for tablet and average for mobile devices with Android operating system. This gives us a brief understanding of revenue for different devices.

```
#Train Data
filtered_Train1 <- Train %>%
  mutate(networkDomain = fct_lump_n(networkDomain, n =
5),channelGrouping=fct_lump_n(channelGrouping, n=5))
```

```
  ggplot(filtered_Train1,aes(x = networkDomain, y = revenue, color =
channelGrouping )) +
  geom_point() +
  labs(title = "NetworkDomain vs.Revenue by channelGrouping",
       x = "NetworkDomain", y = "Revenue") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



#Plot is generated for Network Domain vs Revenue for different channels. From this graph we can understand how revenue is changing based on network domain the users are using and which channel is widely used. We can observe in comcast.net network domain most of the users are under Referral channel grouping and if also we can see for almost all the network domains are falling under Direct or Referral channel grouping. So we can conclude revenue is highest for Direct or Referral channel grouping.

```
library(knitr)
options(digits=3)
options(scipen=99)
```

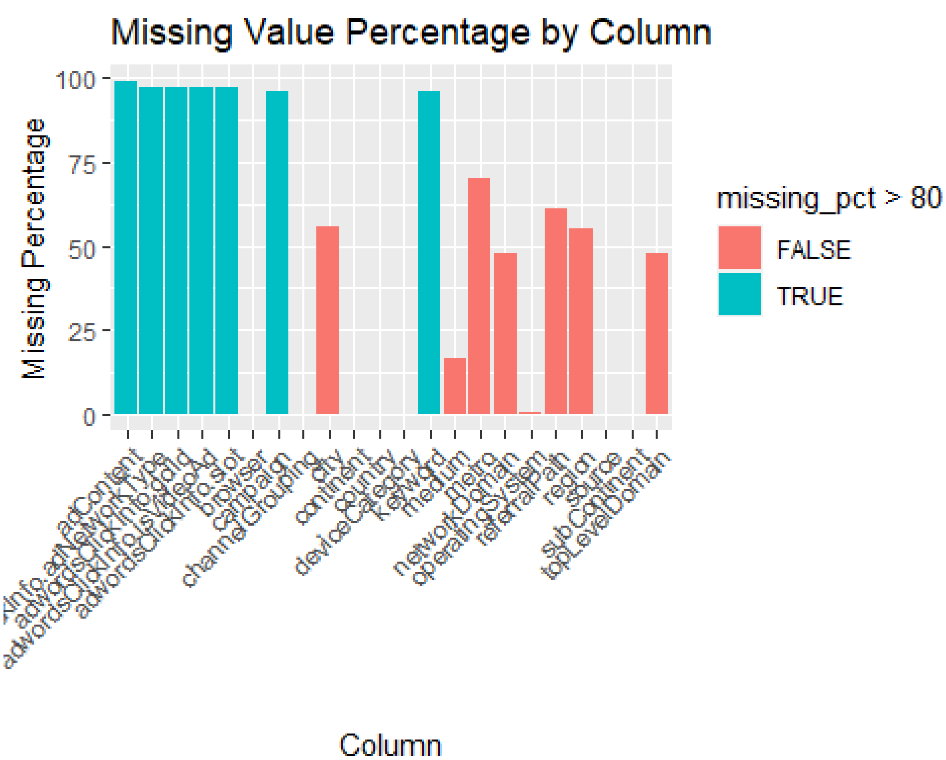
#DiscreteFactorSummaryFinal %>% kable()

1.(a).(ii). Data preparation. Choose two of the most critical data preparation actions you took and explain the reasoning for these actions.

#beginning of Data Preparation and Preprocessing

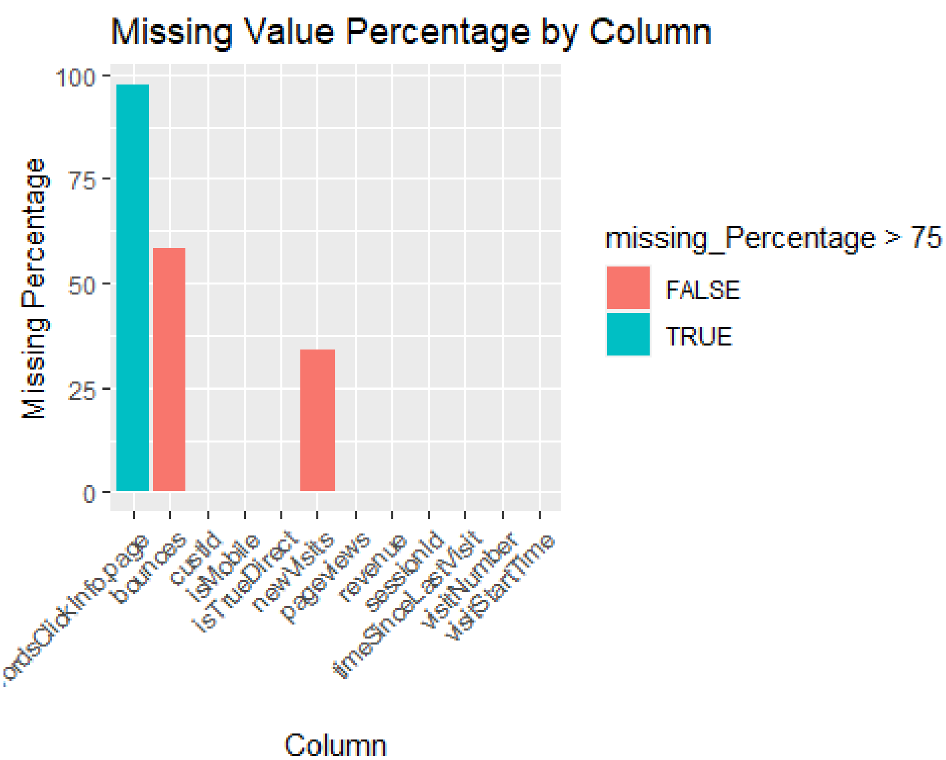
#Since we have to do preprocessing for both Test Data and Training Data we'll create functions

```
#Train Discrete variables
ggplot(data = DiscreteFactorSummaryFinal,mapping = ( aes(x = variable, y
=missing_pct, fill=missing_pct>80 ))) +
  geom_bar(stat="identity") +
  labs(
    title = "Missing Value Percentage by Column",
    x = "Column",
    y = "Missing Percentage"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis
labels for better readability
```




```
#Train Numeric variables
ggplot(data = TrainNumericSummaryFinal,mapping = ( aes(x = variable, y
=missing_Percentage, fill=missing_Percentage>75 ))) +
  geom_bar(stat="identity") +
  labs(
    title = "Missing Value Percentage by Column",
    x = "Column",
    y = "Missing Percentage"
  ) +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis
labels for better readability
```



#removing columns with more than 80% missing values

```
columns_to_remove <- c("adContent", "adwordsClickInfo.adNetworkType",
"adwordsClickInfo.gclId","adwordsClickInfo.isVideoAd","campaign","adwordsClic
kInfo.slot","adwordsClickInfo.page","keyword")
```

#Removing selected columns for train

```
Train_preprocess <- Train %>%
  select(-one_of(columns_to_remove))
#Train_preprocess
```

#Removing selected columns for test

```
Test_preprocess <- Test %>%
  select(-one_of(columns_to_remove))
#Test_preprocess
```

#Train Data

```
Train_preprocess.imp<-Train_preprocess
columns_to_impute <- c("pageviews")

for (col_name in columns_to_impute) {
  missing <- is.na(Train_preprocess[[col_name]])
```

```
  if (sum(missing) > 0) {
    Train_preprocess.imp[missing, col_name] <- mice.impute.pmm(
      Train_preprocess.imp[[col_name]],
      !missing,
      Train_preprocess.imp$custId
    )
  }
}
```

#test

```
Test_preprocess.imp<-Test_preprocess
columns_to_impute <- c( "pageviews")

for (col_name in columns_to_impute) {
  missing <- is.na(Test_preprocess[[col_name]])
  if (sum(missing) > 0) {
    Test_preprocess.imp[missing, col_name] <- mice.impute.pmm(
      Test_preprocess.imp[[col_name]],
      !missing,
      Test_preprocess.imp$custId
    )
  }
}

#Replacing Na's with zero's to create two factor variables for Train
Train_preprocess.imp$bounces <- ifelse(is.na(Train_preprocess.imp$bounces),
0, Train_preprocess.imp$bounces)
Train_preprocess.imp$newVisits <-
ifelse(is.na(Train_preprocess.imp$newVisits), 0,
Train_preprocess.imp$newVisits)

#Replacing Na's with zero's to create two factor variables
Test_preprocess.imp$bounces <- ifelse(is.na(Test_preprocess.imp$bounces), 0,
Test_preprocess.imp$bounces)
Test_preprocess.imp$newVisits <- ifelse(is.na(Test_preprocess.imp$newVisits),
0, Test_preprocess.imp$newVisits)
```

#install.packages("party")

```
library(party)
```

#so We'll first impute columns with missing values percent of less than 10% by replacing them with mode of the column. But,if missing values is greater than 7000, that is roughly 10% of the Total column then we'll replace NA values with 'Other' and create a seperate level of category for the misiing values

```
#Train
Train_preprocess.imp1<- Train_preprocess.imp
col_names<- names(Train_preprocess.imp1)
for (col in col_names)
```

```
{
  #print(Train_preprocess.imp1[[col]])
  missing <- is.na(Train_preprocess.imp1[[col]])
  if(is.character(Train_preprocess.imp1[[col]])){
    if (sum(missing) < 7000 & sum(missing) > 0) {
      Train_preprocess.imp1[[col]][is.na(Train_preprocess.imp1[[col]])] <-
getmodes(Train_preprocess.imp1[[col]])
    }
    else{
      Train_preprocess.imp1[[col]][is.na(Train_preprocess.imp1[[col]])] <-
"Other"

    }
  }
}
```

#printing the number of missing values in the entire table

```
total_missing_values<-sum(is.na(Train_preprocess.imp1))
total_missing_values

## [1] 0

#Test
Test_preprocess.imp1<- Test_preprocess.imp
col_names<- names(Test_preprocess.imp1)
for (col in col_names)
{
  #print(Train_preprocess.imp1[[col]])
  missing <- is.na(Test_preprocess.imp1[[col]])
  if(is.character(Test_preprocess.imp1[[col]])){
    if (sum(missing) < 7000 & sum(missing) > 0) {
      Test_preprocess.imp1[[col]][is.na(Test_preprocess.imp1[[col]])] <-
getmodes(Test_preprocess.imp1[[col]])
    }
    else{
      Test_preprocess.imp1[[col]][is.na(Test_preprocess.imp1[[col]])] <-
"Other"

    }
  }
}
```

#printing the number of missing values in the entire table

```
total_missing_values<-sum(is.na(Test_preprocess.imp1))
total_missing_values

## [1] 0
```

#Preprocessing step 3:Converting all characters to Factor Variables

```
#Train
char_vars <- sapply(Train_preprocess.imp1, is.character)

Train_preprocess.imp1[char_vars] <- lapply(Train_preprocess.imp1[char_vars],
as.factor)
```

#test

```
char_vars <- sapply(Test_preprocess.imp1, is.character)

Test_preprocess.imp1[char_vars] <- lapply(Test_preprocess.imp1[char_vars],
as.factor)

#performing PCA
#Hpca <-prcomp(TrainNumeric,scale. = TRUE)

#Hpca

#Plottinh LDA
#lda_result <- lda(revenue ~ ., data = Train_preprocess.imp1)
# Print the LDA results
#lda_result

#<-Glass[, sapply(Glass, is.numeric)]
```

#preprocessing step 4:Removing Outliers

```
corMat <- cor(TrainNumeric)
corMat

##                        sessionId   custId visitStartTime visitNumber
## sessionId                1.00000  1.00000        0.00462     -0.0112
## custId                   1.00000  1.00000        0.00462     -0.0112
## visitStartTime           0.00462  0.00462        1.00000      0.0486
## visitNumber             -0.01117 -0.01117        0.04856      1.0000
## timeSinceLastVisit      -0.00325 -0.00325        0.07667      0.0618
## isMobile                 0.00349  0.00349        0.12130     -0.0491
## isTrueDirect            -0.00595 -0.00595        0.06498      0.2636
## adwordsClickInfo.page         NA       NA             NA          NA
## pageviews                     NA       NA             NA          NA
## bounces                       NA       NA             NA          NA
## newVisits                     NA       NA             NA          NA
## revenue                  0.00451  0.00451        0.00315      0.0199
##                        timeSinceLastVisit  isMobile isTrueDirect
## sessionId                        -0.00325  0.003491    -0.005955
## custId                           -0.00325  0.003491    -0.005955
## visitStartTime                    0.07667  0.121298     0.064977
## visitNumber                       0.06178 -0.049106     0.263565
## timeSinceLastVisit                1.00000 -0.040659     0.167647
## isMobile                         -0.04066  1.000000     0.000692
## isTrueDirect                      0.16765  0.000692     1.000000
```

```
## adwordsClickInfo.page                  NA        NA           NA
## pageviews                              NA        NA           NA
## bounces                                NA        NA           NA
## newVisits                              NA        NA           NA
## revenue                           0.02167 -0.046842     0.070025
##                       adwordsClickInfo.page pageviews bounces newVisits
## sessionId                                NA        NA      NA        NA
## custId                                   NA        NA      NA        NA
## visitStartTime                           NA        NA      NA        NA
## visitNumber                              NA        NA      NA        NA
## timeSinceLastVisit                       NA        NA      NA        NA
## isMobile                                 NA        NA      NA        NA
## isTrueDirect                             NA        NA      NA        NA
## adwordsClickInfo.page                     1        NA      NA        NA
## pageviews                                NA         1      NA        NA
## bounces                                  NA        NA       1        NA
## newVisits                                NA        NA      NA         1
## revenue                                  NA        NA      NA        NA
##                        revenue
## sessionId              0.00451
## custId                 0.00451
## visitStartTime         0.00315
## visitNumber            0.01988
## timeSinceLastVisit     0.02167
## isMobile              -0.04684
## isTrueDirect           0.07003
## adwordsClickInfo.page       NA
## pageviews                   NA
## bounces                     NA
## newVisits                   NA
## revenue                1.00000
```

#SessionID and cust Id have high correlation so we cannot use them together to avoid multicollinearity.

#Grouping the entire table on the basis of Cust ID

```
TrainGroupedData<-Train_preprocess.imp1 %>%
  group_by(custId)%>%
  summarize(sessionId=
max(sessionId),visitNumber=max(visitNumber),timeSinceLastVisit=mean(timeSince
LastVisit),continent=getmodes(continent),

subContinent=getmodes(subContinent),country=getmodes(country),region=getmodes
(region),city=getmodes(city),

metro=getmodes(metro),channelGrouping=getmodes(channelGrouping),visitStartTim
e=min(visitStartTime),
            browser=getmodes(browser),
operatingSystem=getmodes(operatingSystem),isMobile=getmodes(isMobile),
```

```
deviceCategory=getmodes(deviceCategory),networkDomain=getmodes(networkDomain)
,topLevelDomain=getmodes(topLevelDomain),

source=getmodes(source),medium=getmodes(medium),isTrueDirect=getmodes(isTrueD
irect),

referralPath=getmodes(referralPath),pageviews=sum(pageviews),bounces=getmodes
(bounces),newVisits=getmodes(newVisits),
            revenue=log(sum(revenue+1)))%>%
  as_tibble()
char_vars <- sapply(TrainGroupedData, is.character)

TrainGroupedData[char_vars] <- lapply(TrainGroupedData[char_vars], as.factor)

glimpse(TrainGroupedData)

## Rows: 47,249
## Columns: 26

## Warning in grepl(",", levels(x), fixed = TRUE): input string 1 is invalid
in
## this locale

## Warning in grepl(",", levels(x), fixed = TRUE): input string 2 is invalid
in
## this locale

## Warning in grepl(",", levels(x), fixed = TRUE): input string 4 is invalid
in
## this locale

## Warning in grepl(",", levels(x), fixed = TRUE): input string 5 is invalid
in
## this locale

## Warning in grepl(",", levels(x), fixed = TRUE): input string 1 is invalid
in
## this locale

## $ custId             <int> 1795, 1797, 1799, 1800, 1801, 1803, 1804, 1807,
181…
## $ sessionId          <dbl> 200000120, 400000140, 600000160, 700000170,
8000001…
## $ visitNumber        <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
2, …
## $ timeSinceLastVisit <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 8412, 0, 0, 0, 0, 0, 0,
0, …
## $ continent          <fct> Asia, Americas, Asia, Africa, Americas, Europe,
Asi…
## $ subContinent       <fct> Southern Asia, Northern America, Southern Asia,
Eas…
```

```
## $ country            <fct> India, United States, India, Zambia, United
States,…
## $ region             <fct> Tamil Nadu, Other, Other, Other, California,
Other,…
## $ city               <fct> Chennai, Other, Other, Other, San Francisco,
Other,…
## $ metro              <fct> Other, Other, Other, Other, San Francisco-
Oakland-S…
## $ channelGrouping    <fct> Social, Social, Organic Search, Social, Direct,
Org…
## $ visitStartTime     <int> 1493117200, 1473037945, 1483011213, 1471890172,
149…
## $ browser            <fct> Chrome, Safari, Chrome, Safari, Chrome, Chrome,
Saf…
## $ operatingSystem    <fct> Windows, Macintosh, Windows, Macintosh,
Android, Ma…
## $ isMobile           <fct> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
1, …
## $ deviceCategory     <fct> desktop, desktop, desktop, desktop, mobile,
desktop…
## $ networkDomain      <fct> airtel.in, comcast.net, Other, ipb.na, Other,
wanad…
## $ topLevelDomain     <fct> in, net, Other, na, Other, fr, Other, com, th,
Othe…
## $ source             <fct> quora.com, youtube.com, google, youtube.com,
(direc…
## $ medium             <fct> referral, referral, organic, referral, Other,
organ…
## $ isTrueDirect       <fct> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
0, …
## $ referralPath       <fct> "/How-can-one-get-a-Google-T-shirt-in-India",
"/yt/…
## $ pageviews          <int> 1, 1, 1, 1, 6, 6, 1, 1, 2, 5, 1, 1, 8, 24, 14,
2, 3…
## $ bounces            <fct> 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
0, …
## $ newVisits          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, …
## $ revenue            <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 0.000,
0.000, 0.…
```

#test

```
TestGroupedData<-Test_preprocess.imp1 %>%
  group_by(custId)%>%

summarize(sessionId=max(sessionId),visitNumber=max(visitNumber),timeSinceLast
Visit=mean(timeSinceLastVisit),continent=getmodes(continent),

subContinent=getmodes(subContinent),country=getmodes(country),region=getmodes
```

```
(region),city=getmodes(city),

metro=getmodes(metro),channelGrouping=getmodes(channelGrouping),visitStartTim
e=min(visitStartTime),
            browser=getmodes(browser),
operatingSystem=getmodes(operatingSystem),isMobile=getmodes(isMobile),

deviceCategory=getmodes(deviceCategory),networkDomain=getmodes(networkDomain)
,topLevelDomain=getmodes(topLevelDomain),

source=getmodes(source),medium=getmodes(medium),isTrueDirect=getmodes(isTrueD
irect),

referralPath=getmodes(referralPath),pageviews=sum(pageviews),bounces=getmodes
(bounces),newVisits=getmodes(newVisits))%>%
  as_tibble()
char_vars <- sapply(TestGroupedData, is.character)

TestGroupedData[char_vars] <- lapply(TestGroupedData[char_vars], as.factor)

#Doing Final Conversions
#columns_to_convert <- c("bounces", "newVisits")
#TrainGroupedData[columns_to_convert] <-
lapply(TrainGroupedData[columns_to_convert], as.factor)

# Calculate the mean of the non-zero values
#mean_non_zero <-
mean(TrainGroupedData$timeSinceLastVisit[TrainGroupedData$timeSinceLastVisit
!= 0])
```

Preprocessing Part 3: Transforminmg the predictor values to log(x+1) form

```
#library(MASS)

#boxcox
TrainGroupedData$timeSinceLastVisit <-
log(TrainGroupedData$timeSinceLastVisit+1)
TestGroupedData$timeSinceLastVisit <-
log(TestGroupedData$timeSinceLastVisit+1)

TrainGroupedData$pageviews <- log(TrainGroupedData$pageviews+1)
TestGroupedData$pageviews <- log(TestGroupedData$pageviews+1)

TrainGroupedData$visitNumber <- log(TrainGroupedData$visitNumber+1)
TestGroupedData$visitNumber <- log(TestGroupedData$visitNumber+1)

#TestGroupedData[char_vars] <- lapply(TestGroupedData[char_vars],
as.character)
```

1.(a).(iii). Modeling. Build an OLS model and 3 or more regression variant models (these may include robust regression, PLS, PCR, ridge regression, LASSO, elasticnet, MARS, or SVR) and summarize their performance in a table (as shown in Table 1). Clearly state your resampling approach. Note: You may combine models, techniques, etc

```
#OLS Model
ctrl <- trainControl(
  method = "repeatedcv",        # Cross-validation method ("cv" for k-fold cross-validation)
  number = 10,          # Number of folds (5 for 5-fold cross-validation)
  #verboseIter = TRUE,  # Display progress
  summaryFunction = defaultSummary  # Use default summary function
)

# Fit your model with 5-fold cross-validation
OlsCVmodel <- train(
  revenue ~ custId+(sessionId*visitNumber*timeSinceLastVisit*pageviews)+operatingSystem+channelGrouping+continent+deviceCategory+isMobile+medium+bounces+newVisits,
  #revenue ~ .,# Specify the formula (dependent variable ~ predictor variables)
  data = TrainGroupedData,   # Specify your dataset
  method = "lm",      # Specify the modeling method (e.g., linear regression)
  trControl = ctrl,
  metric="RMSE"# Use the training control settings created earlier
)
cvsummary<-summary(OlsCVmodel)
cvsummary

##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.099 -0.158  0.003  0.166  6.041
##
## Coefficients: (2 not defined because of singularities)
##                                                              Estimate
## (Intercept)                                       0.310960470765812735
## custId                                           -0.000001356879579814
## sessionId                                                           NA
## visitNumber                                      -1.265042198346758573
## timeSinceLastVisit                               -0.109094026365473745
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.708 on 47199 degrees of freedom
## Multiple R-squared:  0.716,  Adjusted R-squared:  0.716
## F-statistic: 2.43e+03 on 49 and 47199 DF,  p-value: <0.0000000000000002
```

```
#olspredictions <- predict(OlsCVmodel, newdata = TestGroupedData)
#summary(olspredictions)





#TestGroupedData1<- TestGroupedData %>%
  #mutate(predRevenue=olspredictions)

#FinalPredictions1<- TestGroupedData1 %>%
  #select(custId, predRevenue)

#write.csv(FinalPredictions, file = "filepredictionOLS.csv", row.names =
FALSE)

CV_RMSE <- OlsCVmodel$results$RMSE
CV_R2<- cvsummary$r.squared
cat("CVRMSE:", CV_RMSE , "\n")
```

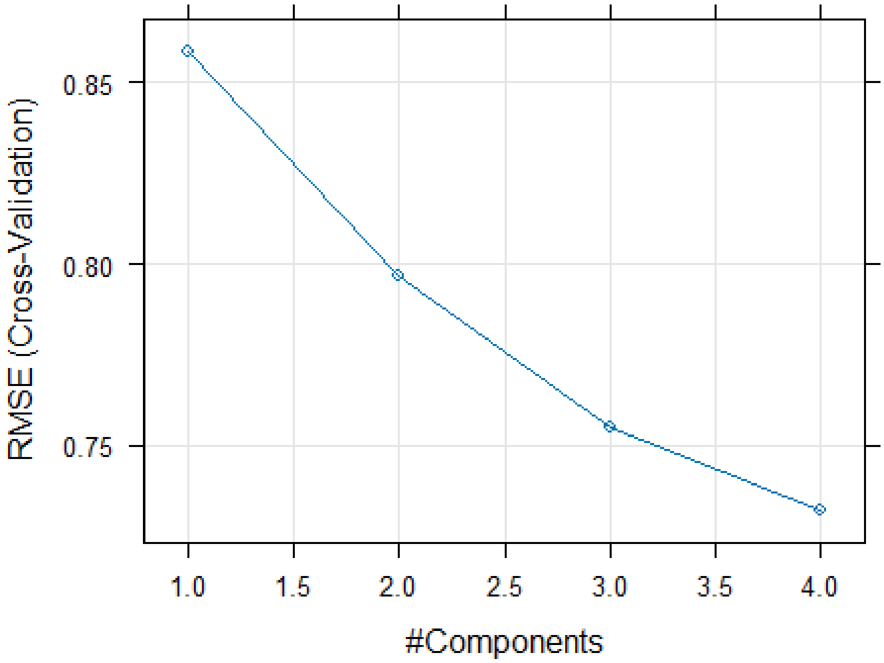
```
## CVRMSE: 0.708
```

```
cat("CV R-squared:",CV_R2  , "\n")
```

```
## CV R-squared: 0.716
```

```
#PLS
ctrl <- trainControl(
  method = "cv",         # Cross-validation method (e.g., k-fold)
  number = 5,            # Number of folds
  savePredictions = TRUE, # Save predictions for final model

)

# Perform hyperparameter tuning with cross-validation
set.seed(123)  # For reproducibility
ncompnum<-(seq(1,4,1))
pls_model <- train(
  revenue ~
custId+(sessionId*visitNumber*timeSinceLastVisit*pageviews)+operatingSystem+c
hannelGrouping+continent+deviceCategory+isMobile+medium+bounces+newVisits,
  data = TrainGroupedData,
  method = "pls",
```

```r
  trControl = ctrl,
  #ncomp=5,
  tuneGrid = expand.grid(ncomp = ncompnum),
  metric="RMSE",
  preProc=c("center","scale"))

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19,
uniqueCut =
## 10, : These variables have zero variances: operatingSystemNintendo 3DS

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19,
uniqueCut =
## 10, : These variables have zero variances: operatingSystemNintendo WiiU

pls_model$results

##   ncomp  RMSE Rsquared   MAE  RMSESD RsquaredSD   MAESD
## 1     1 0.859    0.582 0.437 0.01230    0.01058 0.00491
## 2     2 0.797    0.640 0.479 0.00936    0.00799 0.00506
## 3     3 0.755    0.676 0.463 0.01080    0.00847 0.00667
## 4     4 0.732    0.696 0.416 0.01183    0.00919 0.00618

plsresults<-pls_model$results %>%
  filter(ncomp == pls_model$bestTune$ncomp)


plsRMSE<- plsresults$RMSE
cat("RMSE:", plsRMSE, "\n")

## RMSE: 0.732

plsR2<- plsresults$Rsquared
cat("R-squared:", plsR2, "\n")

## R-squared: 0.696

plot(pls_model)
```




```
#Creating Lasso model
lambda_val=seq(0,1,0.1)
ctrl <- trainControl(
  method = "cv",          # Cross-validation method (e.g., k-fold)
  number = 10,            # Number of folds
  savePredictions = TRUE, # Save predictions for final model

)
# Perform hyperparameter tuning with cross-validation
set.seed(123)  # For reproducibility
Lasso_model <- train(
  revenue ~
custId+(sessionId*visitNumber*timeSinceLastVisit*pageviews)+operatingSystem+c
hannelGrouping+continent+deviceCategory+isMobile+medium+bounces+newVisits,
  data = TrainGroupedData,                          # Your response variable
  method = "glmnet",             # Machine learning method (Ridge
Regression)
  trControl = ctrl,              # Cross-validation control
  tuneGrid = expand.grid(lambda=lambda_val, alpha=1),
  metric="RMSE",
  preProc=c("center","scale")
  )

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19,
uniqueCut =
## 10, : These variables have zero variances: operatingSystemNintendo WiiU
```

```
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19,
uniqueCut =
## 10, : These variables have zero variances: operatingSystemNintendo 3DS
```

Retrieve and display the results of Lasso hyperparameter tuning

```
Lasso_model$results

##    alpha lambda  RMSE Rsquared   MAE RMSESD RsquaredSD   MAESD
## 1      1    0.0 0.709    0.714 0.394 0.0155     0.0126 0.00865
## 2      1    0.1 0.761    0.678 0.431 0.0121     0.0126 0.00686
## 3      1    0.2 0.786    0.675 0.420 0.0112     0.0127 0.00626
## 4      1    0.3 0.821    0.675 0.438 0.0111     0.0126 0.00554
## 5      1    0.4 0.869    0.674 0.487 0.0118     0.0125 0.00537
## 6      1    0.5 0.926    0.673 0.538 0.0131     0.0124 0.00585
## 7      1    0.6 0.992    0.670 0.592 0.0148     0.0122 0.00669
## 8      1    0.7 1.064    0.663 0.649 0.0166     0.0119 0.00758
## 9      1    0.8 1.140    0.651 0.709 0.0183     0.0115 0.00833
## 10     1    0.9 1.218    0.643 0.767 0.0200     0.0111 0.00945
## 11     1    1.0 1.297    0.627 0.827 0.0216     0.0147 0.01061
```

Filter the best result based on lambda

```
Lassoresults <- Lasso_model$results %>%
  filter(lambda == Lasso_model$bestTune$lambda)

Lassoresults

##   alpha lambda  RMSE Rsquared   MAE RMSESD RsquaredSD   MAESD
## 1     1      0 0.709    0.714 0.394 0.0155     0.0126 0.00865
```

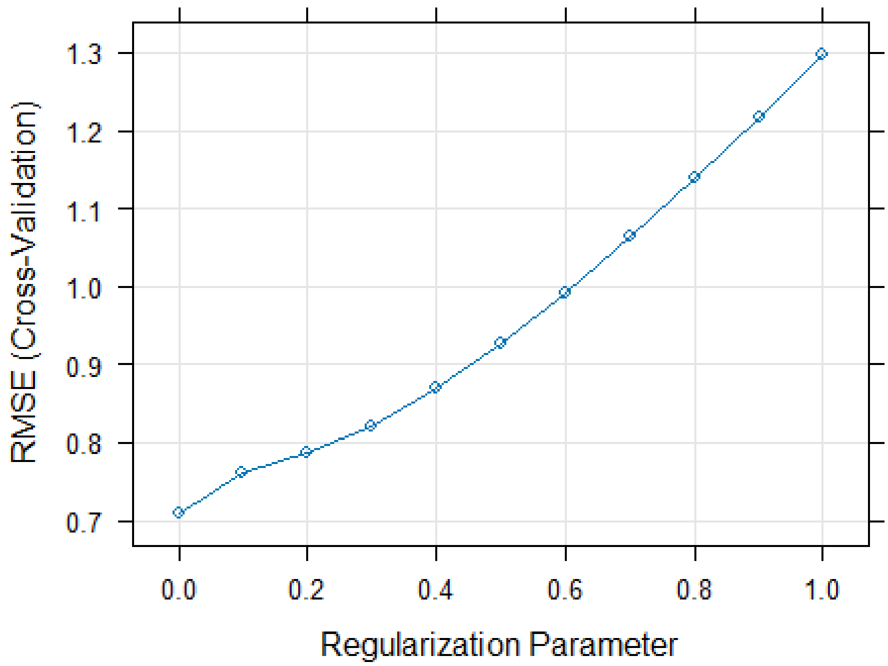
Calculate and display RMSE (Root Mean Squared Error) and display R-squared

```
rmse <- Lassoresults$RMSE
cat("RMSE:", rmse, "\n")

## RMSE: 0.709

r_squared <- Lassoresults$Rsquared
cat("R-squared:", r_squared, "\n")

## R-squared: 0.714
```

We got these values for Alpha=1 and Lambda=0

I have plotted a coefficient path plot to visualize how the coefficients of predictor variables change as the regularization parameter lambda varies in a Lasso model

```
plot(Lasso_model, xvar = "lambda", label = TRUE)
```



#MarsModel

#Since Mars Model takes a long time to execute, we will be posting screenshots from the R file

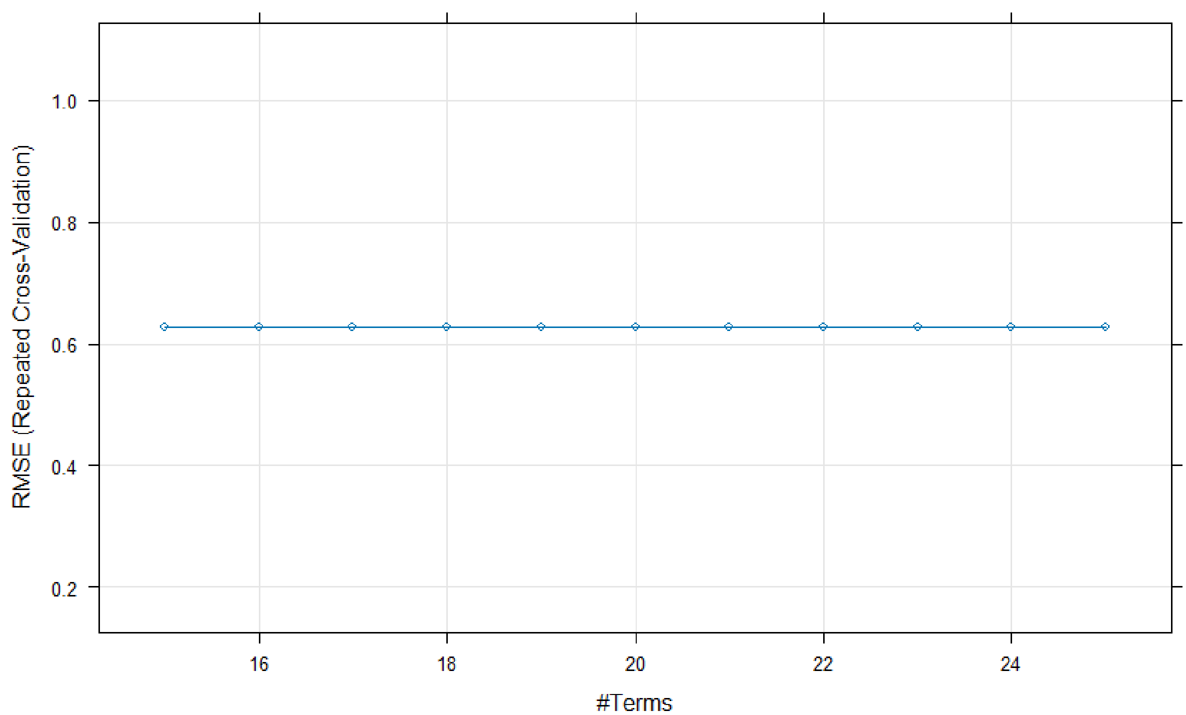
ctrl <- trainControl(

```
  method = "repeatedcv",           # Cross-validation method (e.g., k-fold)
  number = 10,

)

set.seed(123)  # For reproducibility
mars_model <- train(

revenue~(visitNumber*timeSinceLastVisit*pageviews)+operatingSystem+browser+chann
elGrouping+continent+
   deviceCategory+isMobile+medium+bounces+newVisits+referralPath,
  data = TrainGroupedData,                 # Your response variable
  method = "earth",         # Machine learning method (MARS)
  trControl = ctrl,         # Cross-validation control
  metric="RMSE",
  preProc=c("center","scale"),
  tuneGrid = expand.grid(degree=2,nprune=15:25)
)


mars_model
summary(mars_model)
```

```
> mars_model
Multivariate Adaptive Regression Spline

47249 samples
   12 predictor

Pre-processing: centered (42), scaled (42)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 42524, 42524, 42524, 42524, 42524, 42524, ...
Resampling results across tuning parameters:

  nprune  RMSE   Rsquared  MAE
  15      0.626  0.777     0.23
  16      0.626  0.777     0.23
  17      0.626  0.777     0.23
  18      0.626  0.777     0.23
  19      0.626  0.777     0.23
  20      0.626  0.777     0.23
  21      0.626  0.777     0.23
  22      0.626  0.777     0.23
  23      0.626  0.777     0.23
  24      0.626  0.777     0.23
  25      0.626  0.777     0.23

Tuning parameter 'degree' was held constant at a value of 2
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 15 and degree = 2.
```

```
mmresult <- mars_model$results %>%
+       filter(nprune == mars_model$bestTune$nprune, degree ==
mars_model$bestTune$degree)
> mmresult
```

```
> mmresult <- mars_model$results %>%
+     filter(nprune == mars_model$bestTune$nprune, degree == mars_model$bestTune$degree)
> mmresult
  degree nprune  RMSE Rsquared  MAE RMSESD RsquaredSD  MAESD
1      2     15 0.626    0.777 0.23 0.0254     0.0173 0.0106
> rmse <- mmresult$RMSE
> cat("RMSE:", rmse, "\n")
RMSE: 0.626
>
>
> r_squared <- mmresult$Rsquared
> cat("R-Squared:", r_squared, "\n")
R-Squared: 0.777
>
> plot(mars_model)
>
```



Displaying the Tabular Data for comparison of Different Models

Model	Notes	Hyperparameters	Cv RMSE	CV R2
OLS	lm	N/A	0.708	0.716
PLS	pls	Ncomp=4	0.732	0.696
Lasso	Lasso Regression	Alpha=1 and Lambda=0	0.709	0.714
MARS		Degree=2 and Nprune=15:25	0.626	0.777

By Observing the above data, we can infer that Mars Model has the best RMSE and R squared, which means it is the best performing model and we have choose that to be our final model

Q1.iv. (10 points) Debrief. For your best predictions, describe your approach, e.g., did you examine interactions? did you use any type of model stacking? what was your secret sauce? Did you have any problems during the modeling process? If so, how did you overcome those?

We removed all the missing values and for catergoical values we removed stratigically and we thought to replace column with less than 10% missing values with the mode and for columns with more than 10% missing values we converted the missing values into another category. We also did log transformations of the predictor variables we think this was our secret source which improved the model performance by a huge margin. And we used the combination of polinomial terms and categorical variables which reduced the rmse even further.