Assignment 6 Kaggle Competition

Nicholas Jacob, Yechang Qi and Zayne McLaughlin

2024-09-05

Preparation and Modeling

(a) i. Data understanding

```
# Display the Descriptive Summary of Numeric Variables
numericSummary %>%
  kable(digits = 2, format = "latex", booktabs = TRUE,
        caption = "Descriptive Summary of Numeric Variables") %>%
  kable_styling(font_size = 12,latex_options = c("H", "scale_down")) %>%
  row_spec(0, bold = TRUE) %>% # Make header bold
  row_spec(1:nrow(numericSummary), extra_latex_after = "\\addlinespace[0.5em]")
```

Table 1: Descriptive Summary of Numeric Variables

variable	n	missing	missing_pct	unique	unique_pct	mean	min	Q1	median	Q3	max	sd
sessionId	70071	0	0.00	70071	100.00	4.7e+12	2.0e+08	2.3e+12	4.7e+12	7.1e+12	9.4e+12	2.7e+12
custId	70071	0	0.00	47249	67.43	4.9e+04	1.8e+03	2.5e+04	4.9e+04	7.3e+04	9.6e+04	2.7e+04
visitStartTime	70071	0	0.00	69951	99.83	1.5e+09	1.5e+09	1.5e+09	1.5e+09	1.5e+09	1.5e+09	9.1e+06
visitNumber	70071	0	0.00	155	0.22	3.1e+00	1.0e+00	1.0e+00	1.0e+00	2.0e+00	1.6e+02	8.7e+00
timeSinceLastVisit	70071	0	0.00	20970	29.93	2.6e+05	0.0e+00	0.0e+00	0.0e+00	1.0e+04	3.0e+07	1.2e+06
isMobile	70071	0	0.00	2	0.00	2.3e-01	0.0e+00	0.0e+00	0.0e+00	0.0e+00	1.0e+00	4.2e-01
isTrueDirect	70071	0	0.00	2	0.00	4.0e-01	0.0e+00	0.0e+00	0.0e+00	1.0e+00	1.0e+00	4.9e-01
adwordsClickInfo.page	70071	68260	97.42	6	0.01	1.0e+00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	7.0e+00	1.8e-01
pageviews	70071	8	0.01	155	0.22	6.3e+00	1.0e+00	1.0e+00	2.0e+00	6.0e+00	4.7e+02	1.2e+01
bounces	70071	40719	58.11	2	0.00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	0.0e+00
newVisits	70071	23944	34.17	2	0.00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	1.0e+00	0.0e+00
revenue	70071	0	0.00	5850	8.35	1.0e+01	0.0e+00	0.0e+00	0.0e+00	0.0e+00	1.6e+04	1.0e+02
targetRevenue	70071	0	0.00	5085	7.26	1.1e+00	0.0e+00	0.0e+00	0.0e+00	2.4e+00	9.7e+00	2.0e+00

Table 1 provides a Descriptive Summary of Numeric Variables in the dataset. Some variables, like adwordsClickInfo.page, have a high percentage of missing data (97.42%). The median revenue is 0, indicating that many customers may not generate revenue.

```r
# Display the final summary table with kable
factorSummaryFinal %>%
  kable(digits = 2, format = "latex", booktabs = TRUE,
        caption = "Descriptive Summary of Categorical Variables") %>%
  kable_styling(font_size = 12,latex_options = c("H", "scale_down")) %>%
  row_spec(0, bold = TRUE) %>%  # Make header bold
  row_spec(1:nrow(factorSummaryFinal), extra_latex_after = "\\addlinespace[0.25em]")
```

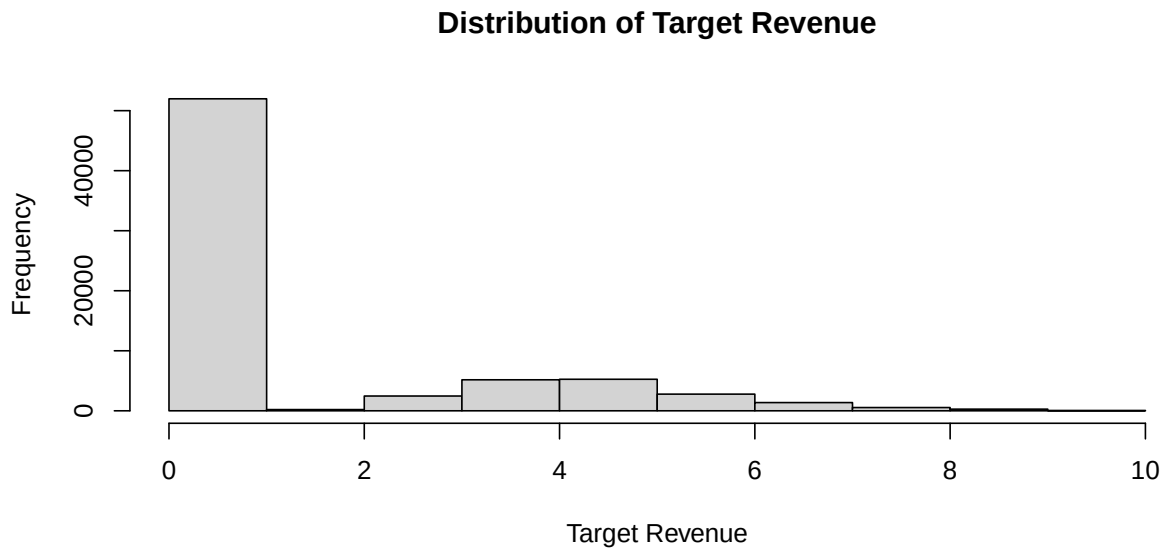
Table 2: Descriptive Summary of Categorical Variables

variable	n	missing	missing_pct	unique	unique_pct	freqRatio	1st mode	first_mode_freq	2nd mode	second_mode_freq
date	70071	0	0.00	366	0.52	1.03	2016-12-05	362	2016-11-28	352
channelGrouping	70071	0	0.00	8	0.01	2.03	Organic Search	27503	Social	13528
browser	70071	1	0.00	28	0.04	4.30	Chrome	51584	Safari	12007
operatingSystem	70071	307	0.44	16	0.02	1.01	Macintosh	23970	Windows	23707
deviceCategory	70071	0	0.00	3	0.00	3.89	desktop	53986	mobile	13868
continent	70071	85	0.12	6	0.01	3.10	Americas	42508	Asia	13697
subContinent	70071	85	0.12	23	0.03	8.06	Northern America	38860	Southeast Asia	4823
country	70071	85	0.12	177	0.25	12.14	United States	36941	India	3044
region	70071	38485	54.92	310	0.44	3.25	California	11254	New York	3468
metro	70071	49183	70.19	73	0.10	2.86	San Francisco-Oakland-San Jose CA	10072	New York NY	3526
city	70071	39028	55.70	478	0.68	1.32	Mountain View	4569	New York	3465
networkDomain	70071	33448	47.73	5015	7.16	2.11	comcast.net	2890	verizon.net	1372
topLevelDomain	70071	33448	47.73	184	0.26	2.39	net	15027	com	6297
campaign	70071	67310	96.06	7	0.01	1.35	AW - Dynamic Search Ads Whole Site	1229	Data Share Promo	911
source	70071	2	0.00	132	0.19	2.30	google	29233	youtube.com	12708
medium	70071	11827	16.88	6	0.01	1.02	organic	27503	referral	27010
keyword	70071	67412	96.21	416	0.59	4.68	6qEhsCssdK0z36ri	997	1hZbAqLCbjwfgOH7	213

Table 2 provides a Descriptive Summary of Categorical Variables in the dataset. We excluded variables such as referralPath, adContent, adwordsClickInfo.page, adwordsClickInfo.slot, adwordsClickInfo.gclId, and adwordsClickInfo.adNetworkType due to the high proportion of missing values and their limited relevance to the analysis.

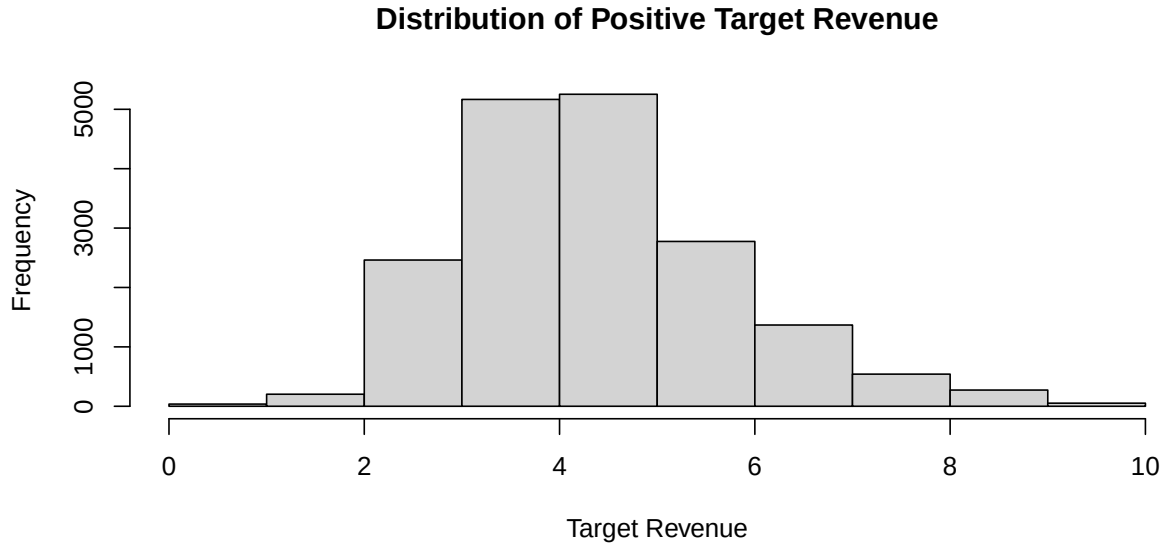
Some variables, like metro, have a high percentage of missing data (70.19%). For variables like "browser", Chrome is the most dominant category, followed by Safari. Similarly, for "deviceCategory", most users are using desktop.

```r
hist(train$targetRevenue, breaks = seq(0, 10, 1),
     main = "Distribution of Target Revenue", xlab = "Target Revenue", ylab = "Frequency")
```



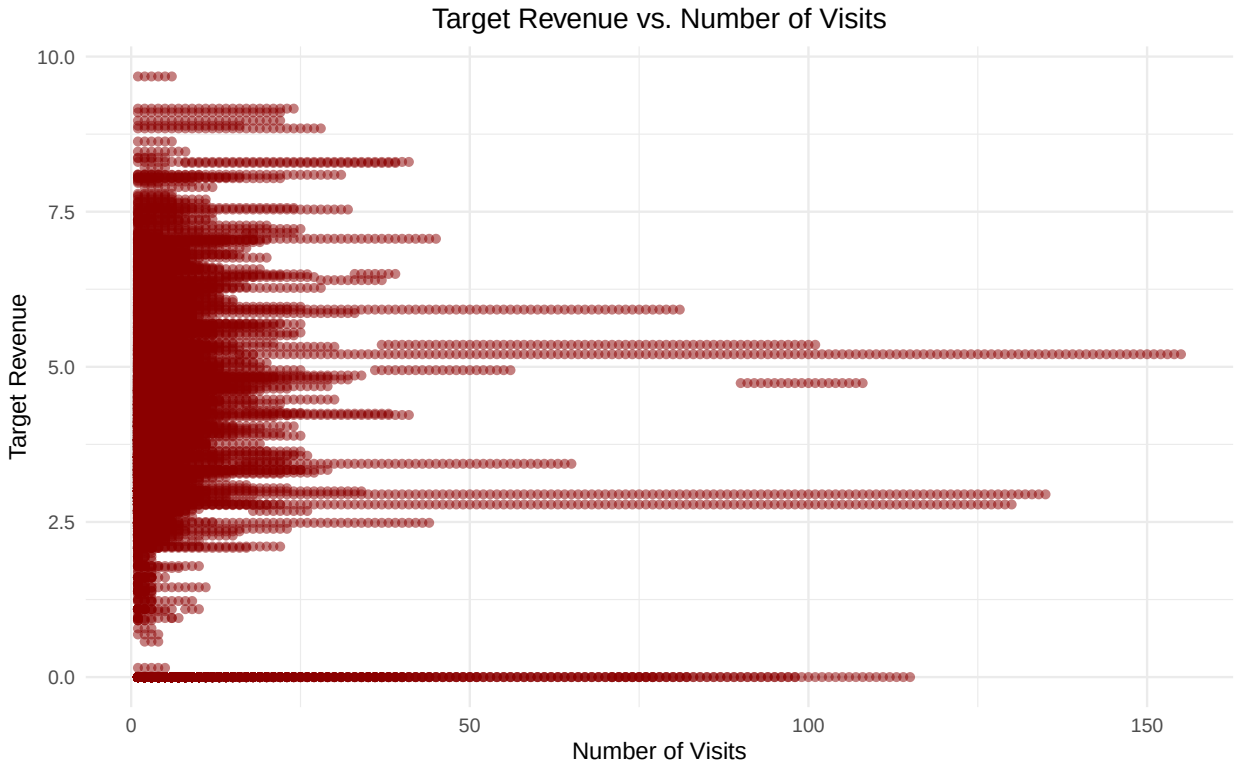
The graph shows that most observations have a target revenue of 0, which causes a significant peak at 0 on the x-axis.

```r
hist(train[train$targetRevenue>0,]$targetRevenue, breaks = seq(0,10,1),
     main = "Distribution of Positive Target Revenue", xlab = "Target Revenue", ylab = "Frequency")
```



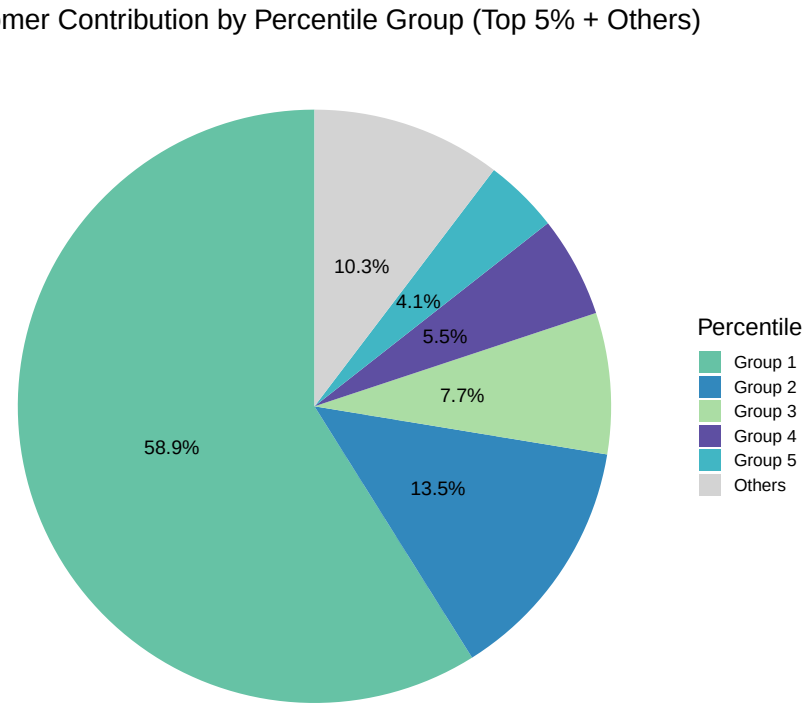
After excluding the zero value, the graph shows that the distribution of positive income values is more even, with a peak near the middle of the revenue range.

```r
# Scatter plot of number of visits vs. target revenue
ggplot(train, aes(x = visitNumber, y = targetRevenue)) +
  geom_point(alpha = 0.5, color = "darkred") +
  labs(title = "Target Revenue vs. Number of Visits",
       x = "Number of Visits", y = "Target Revenue") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



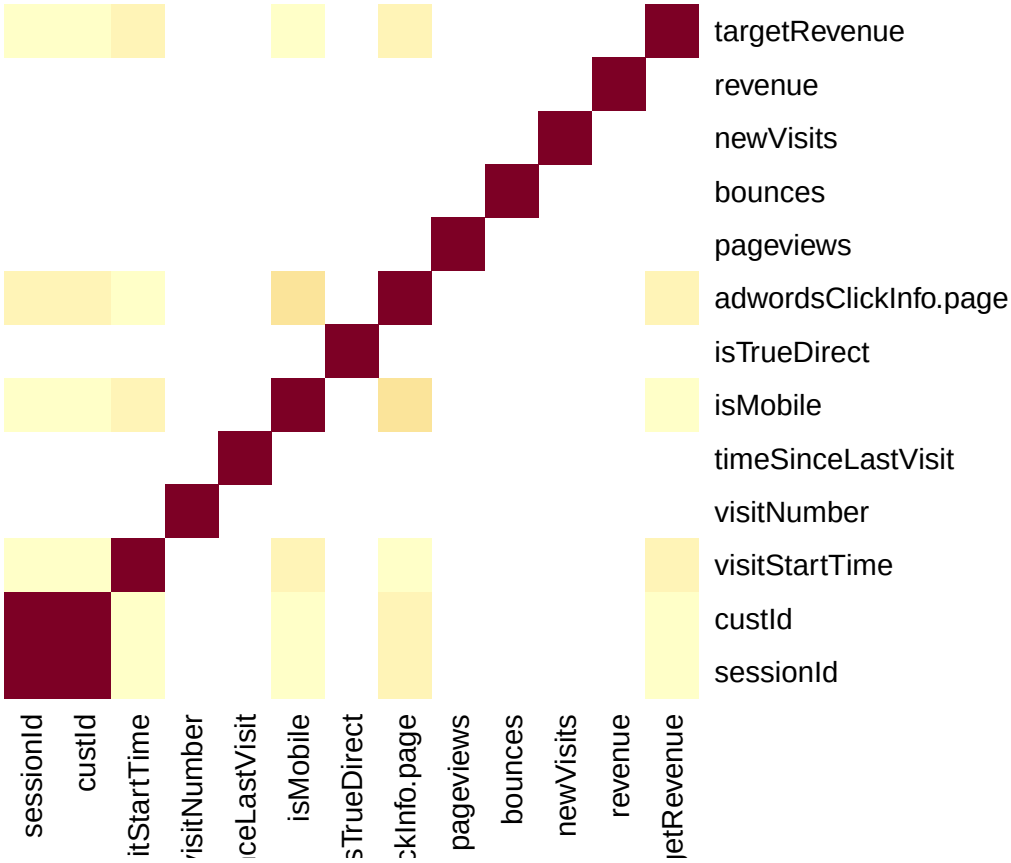
This scatter plot illustrates the relationship between the number of visits and the total revenue for customers. Most customers have a small number of visits and generate relatively low total revenue. The outliers, those who visit more frequently or generate higher revenue, could indeed be considered high net value customers. These customers contribute significantly more to the business in terms of revenue despite being a smaller group.

```r
# Create the pie chart, showing labels only for the top 5% and "Others"
ggplot(group_contribution, aes(x = "", y = group_percentage, fill = group)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  geom_text(aes(label = paste0(round(group_percentage, 1), "%")),
            position = position_stack(vjust = 0.5), size = 3) +
  scale_fill_manual(values = custom_colors) +
  labs(title = "Customer Contribution by Percentile Group (Top 5% + Others)",
       fill = "Percentile") +
  theme_void() +  # Clean look
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.key.size = unit(0.4, "cm"),
    legend.text = element_text(size = 8)
  )
```



This pie plot provides a clear understanding of how heavily the revenue relies on a few top customer groups, emphasizing the importance of focusing efforts on retaining and nurturing these high-value customers.

```r
# Remove rows with NA
train_numeric_NNA <- na.omit(trainNumeric)
cor_matrix <- cor(train_numeric_NNA)
par(mar = c(5, 5, 5, 5))
# Create the heatmap
heatmap(cor_matrix, Colv = NA, Rowv = NA,
        scale = "none")
```



The heatmap suggests that there are few strong correlations between the variables in the train dataset, meaning that most variables act fairly independently of each other.

(a) ii. Data preparation

Cleaning Datas

```
trainFactor <- train %>%
  dplyr::select(where(is.character))%>% ##|where(is.logical) )%>%
  mutate_all(na_if,"")%>%
  mutate_all(factor)%>%
  mutate_all(fct_na_value_to_level)
trainNumeric <- train %>%
  dplyr::select(where(is.numeric))%>%
  replace(is.na(.),0)
train <- cbind(trainNumeric,trainFactor)
```

We replace missing values with zeros, ensuring that all features have valid values, allowing the model to learn from all the data without removing rows or variables.

```
# Remove outliers beyond 3 standard deviations
train_ro <- train %>%
  filter(abs(scale(targetRevenue)) < 3)
```

We decided to remove outliers beyond 3 standard deviations in our model because these extreme values may disproportionately skew the results and negatively impact the model's performance.

```
trainNumeric$nonZero <- as.factor(train$revenue>0)
train$nonZero <- as.factor(train$revenue>0)
```

Identifying customers with non-zero revenue is essential because it helps separate the customers who made purchases from those who did not. This separation allows us to handle the two groups differently in the modeling process. We will focus regression models only on non-zero revenue cases first, then we will focus on all revenue cases.

```
trainFactor <- trainFactor %>%
  mutate(month = as.factor(month(ymd(date))))
```

We want to add the month as a factor rather than just the date. we find this is a better indicator of sales. Note we made it a factor, NOT a quantitative variable.

(a) iii. Modeling

Identifying The Zeros

```
trainControl <- trainControl(method="cv", number=5)
metric <- "Accuracy"
grid <- expand.grid(.k=seq(1,20,by=1))

fit.knn <- train(nonZero~.-targetRevenue - revenue,#
                 data=trainNumeric, method="lda",
                 metric=metric ,
                 trControl=trainControl#,
                 #tuneGrid = grid
                 )
knn.k1 <- fit.knn$bestTune # keep this Initial k for testing with knn() function in next section
print(fit.knn)
```

```
## Linear Discriminant Analysis
```

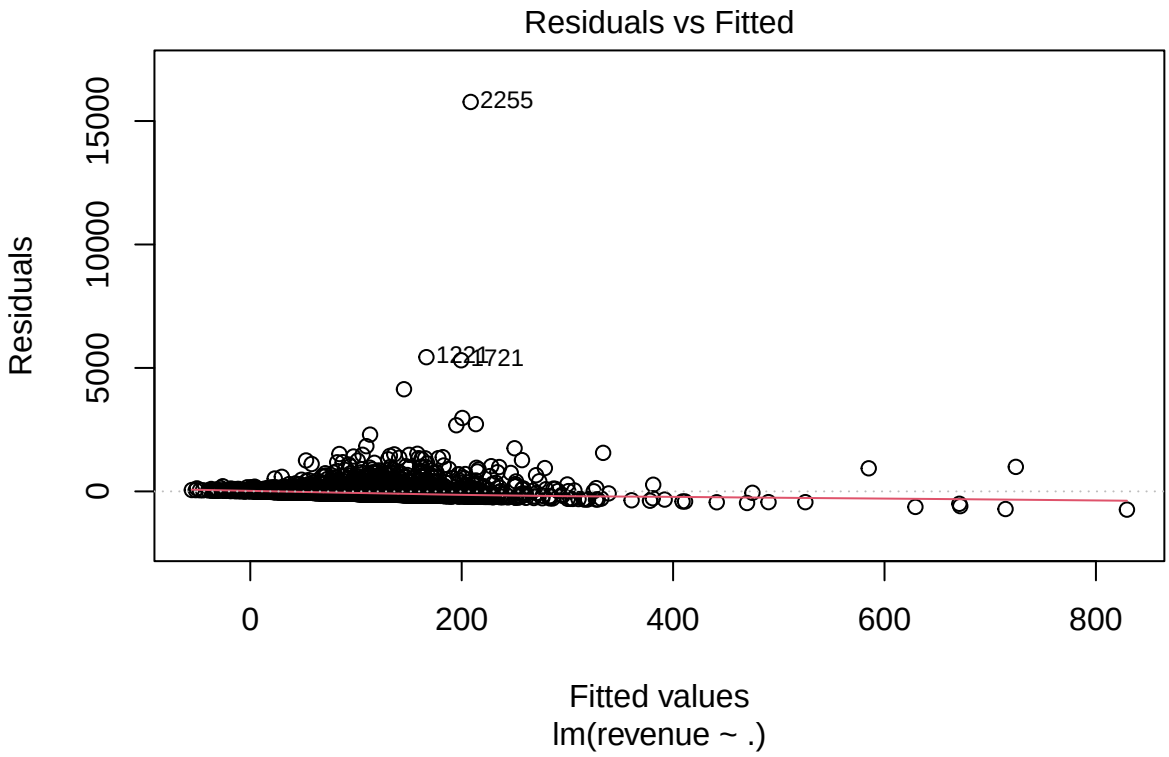
```
##
## 70071 samples
##    13 predictor
##     2 classes: 'FALSE', 'TRUE'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 56056, 56056, 56057, 56058, 56057
## Resampling results:
##
##   Accuracy  Kappa
##   0.935     0.501
```

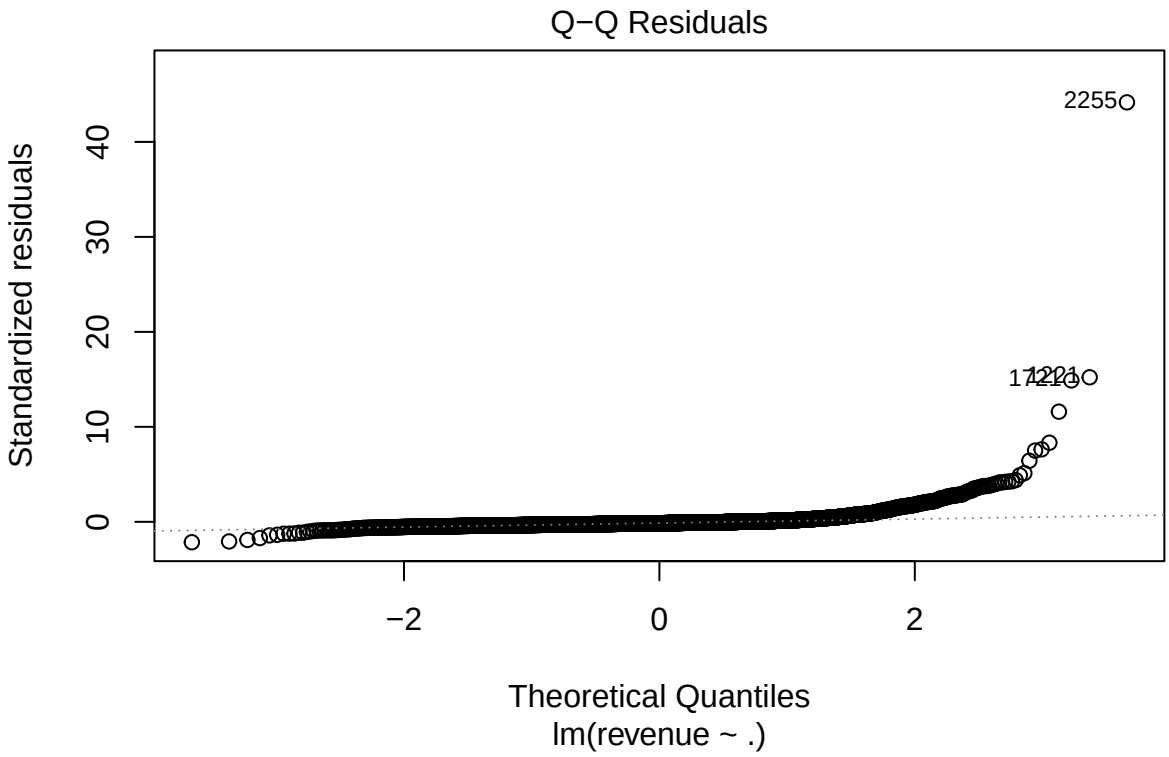
Once the zero's have been identified, not need to predict a value for those. Can build the regression on just the values that are non-zero.

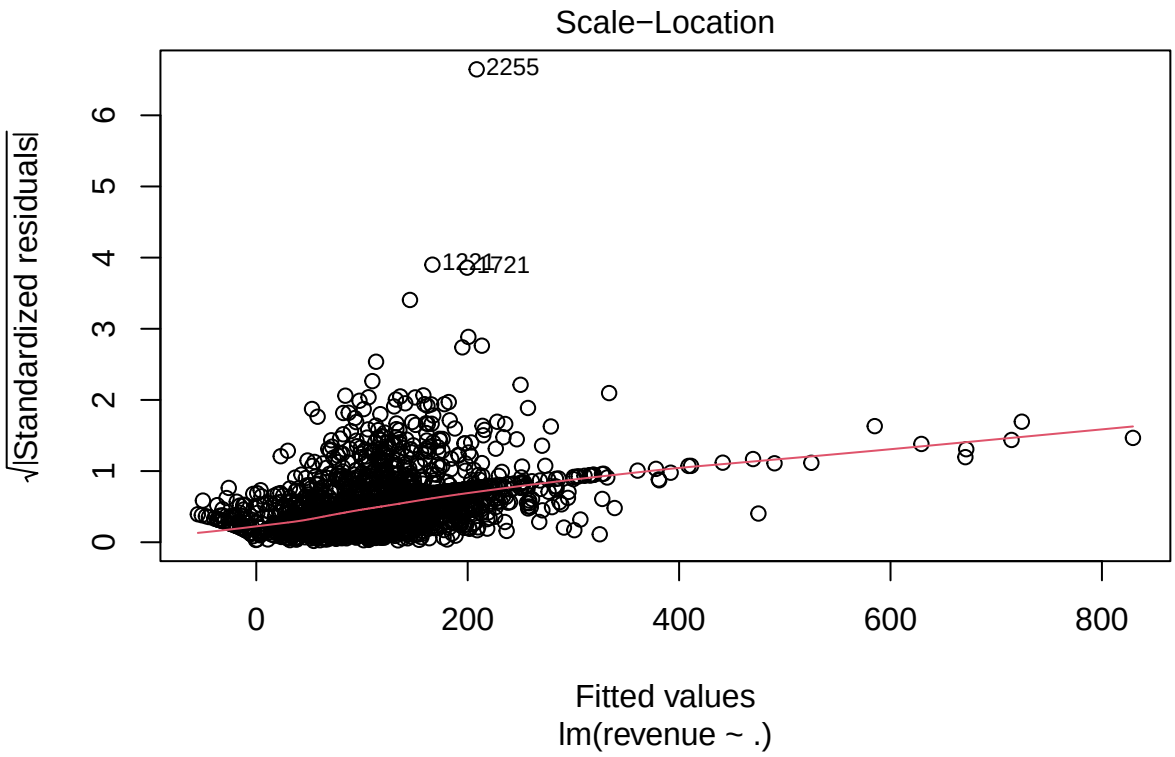
```
ldafit <- lda(nonZero~.,#
                data=trainNumeric %>%
                     select(-c("targetRevenue","revenue"))
              )

predNonZero <- predict(ldafit,trainNumeric)
```

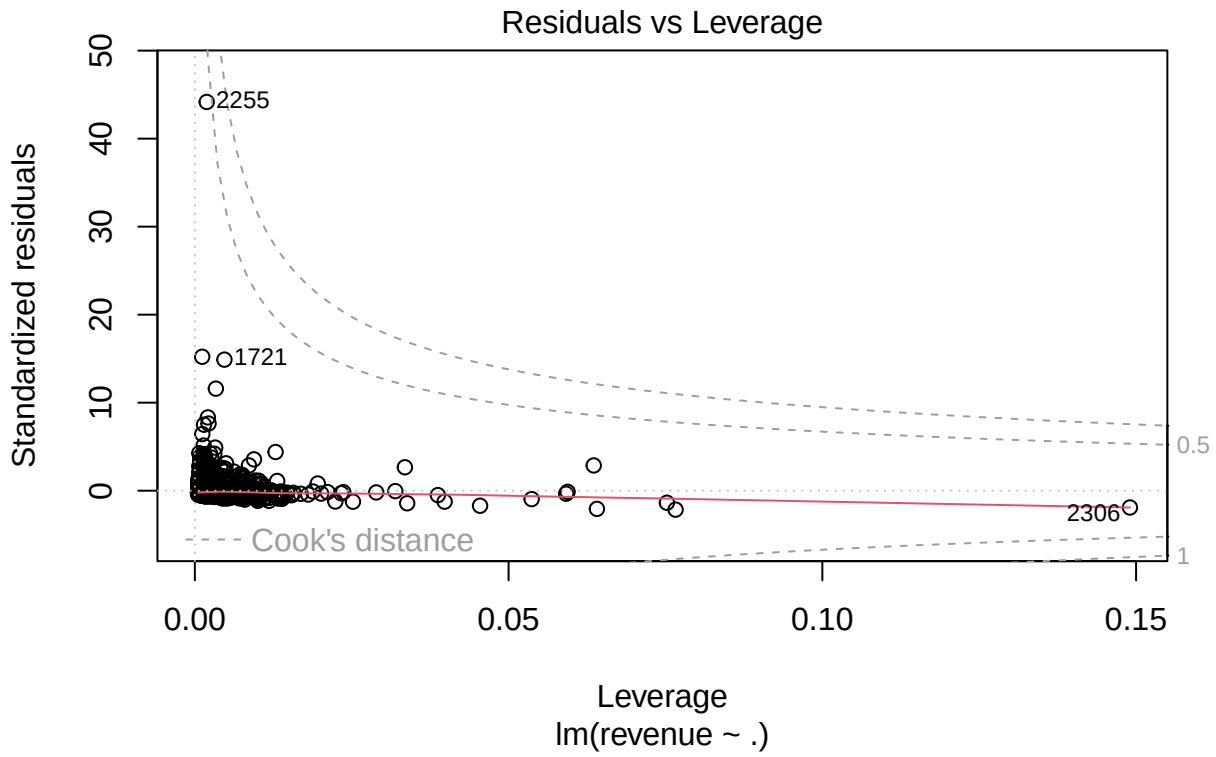
Here is the OLS regression on the revenue that are non-zero:

```
lmod <- lm(revenue ~ .,
           data =trainNumeric%>%
             select(-c("targetRevenue","nonZero"))%>%
             filter(as.logical(predNonZero$class))
          )

plot(lmod)
```


Residuals vs Fitted
2255
1221
1721
Residuals
15000
10000
5000
0
0
200
400
600
800
Fitted values
lm(revenue ~ .)


Q−Q Residuals
Standardized residuals
0
10
20
30
40
2255
1721
1221
−2
0
2
Theoretical Quantiles
lm(revenue ~ .)


Scale−Location
2255
1221
1721
√|Standardized residuals|
0
1
2
3
4
5
6
0
200
400
600
800
Fitted values
lm(revenue ~ .)



Here is the logistic regression on the revenue that are non-zero:

```
glmfit <- glm(nonZero~.,
              data = train %>%
                select(-c("targetRevenue","revenue","adContent","adwordsClickInfo.page",
                        "adwordsClickInfo.slot","adwordsClickInfo.gclId",
                        "adwordsClickInfo.adNetworkType","adwordsClickInfo.slot",
                        "country", "region","metro","city","networkDomain","topLevelDomain",
                        "medium","campaign","keyword","referralPath","sessionId","custId",
                        "date","source","channelGrouping","browser","operatingSystem"
                        )
                       ),#I need less variables to be computed this is maxing my working memory
              family="binomial"
              )
```

```
predVal = predict(lmod,trainNumeric)
rmse_value <- sqrt(mean((trainNumeric$revenue - predVal)^2))
cat("RMSE: ", rmse_value, "\n")
```

```
## RMSE:  112
```

```
sst <- sum((trainNumeric$revenue - mean(trainNumeric$revenue))^2)
sse <- sum((trainNumeric$revenue - predVal)^2)
r2_value <- 1 - (sse / sst)
cat("R-squared: ", r2_value, "\n")
```

```
## R-squared:  -0.275
```

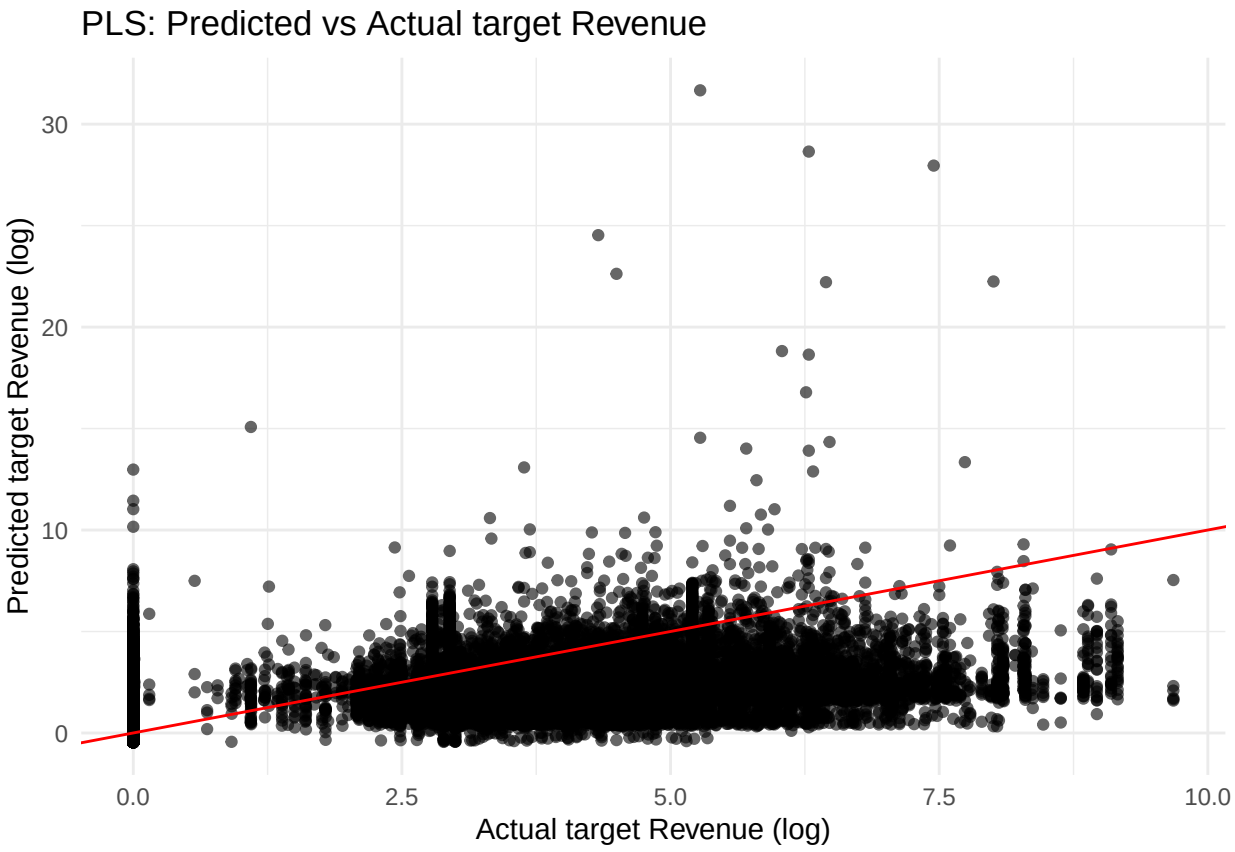
```
trainNumeric$predVal <- replace(predVal,!as.logical(predNonZero$class)|predVal <0,0)
```

Now, we turn to look at the revenue including 0. Here is the PLS model:

```
trainControl <- trainControl(method = "cv", number = 5)
plsFit <- train(targetRevenue ~ ., data = trainNumeric,
                method = "pls",
                preProcess = c("center", "scale"),
                trControl = trainControl,
                tuneLength = 10)
print(plsFit)
```

```
## Partial Least Squares
##
## 70071 samples
##     7 predictor
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 56056, 56057, 56058, 56056, 56057
## Resampling results across tuning parameters:
##
##   ncomp  RMSE  Rsquared  MAE
##   1      1.65  0.342     1.19
##   2      1.65  0.348     1.19
##   3      1.65  0.348     1.19
##   4      1.65  0.348     1.19
##   5      1.65  0.348     1.19
##   6      1.65  0.348     1.19
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 6.
```

```
predictedValues <- predict(plsFit, newdata = trainNumeric)
trainNumeric$predictedRevenue <- predictedValues

ggplot(trainNumeric, aes(x = targetRevenue, y = predictedRevenue)) +
  geom_point(alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(title = "PLS: Predicted vs Actual target Revenue", x = "Actual target Revenue (log)", y = "Predi
  theme_minimal()
```




```
# Calculate RMSE and R² for training data predictions
rmse <- sqrt(mean((trainNumeric$targetRevenue - trainNumeric$predictedRevenue)^2))
r_squared <- cor(trainNumeric$targetRevenue, trainNumeric$predictedRevenue)^2

cat("Training RMSE:", rmse, "\n")
```

```
## Training RMSE: 1.64
```

```
cat("Training R²:", r_squared, "\n")
```

```
## Training R²: 0.348
```

Then, we transformed some categorical variables into dummy variables. We used lasso regression model to predict the target revenue.

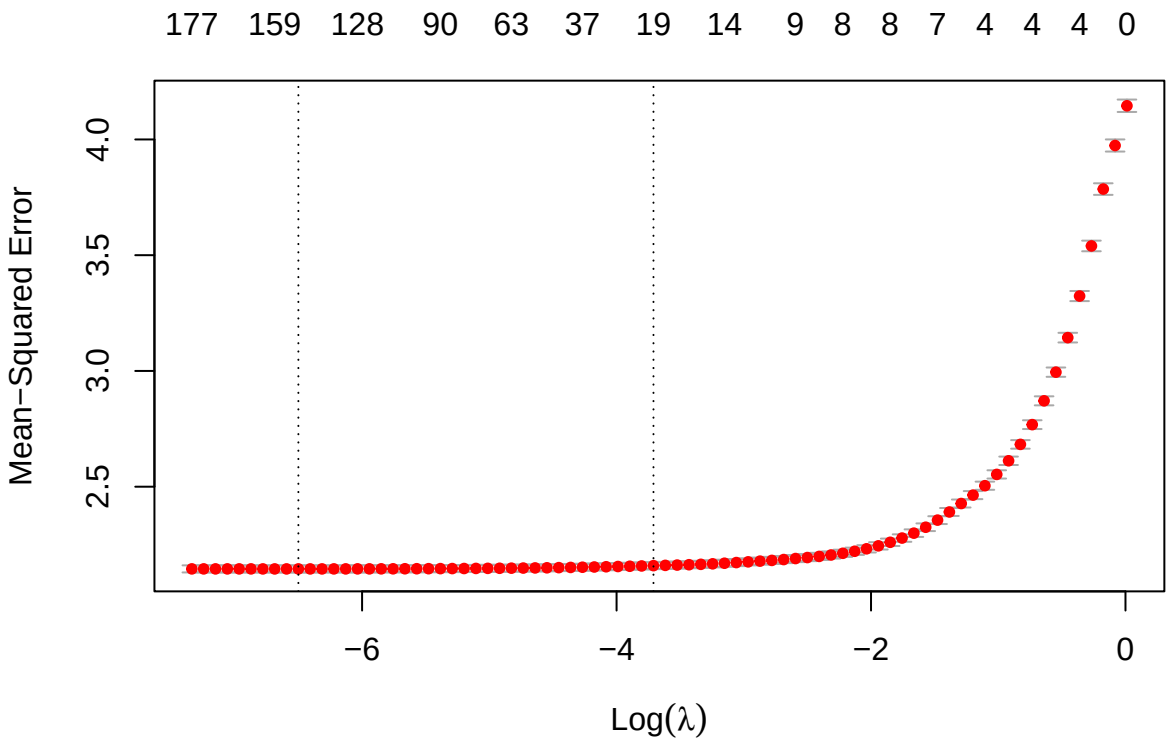
```
trainFactor <- train %>%
  select(channelGrouping, browser, deviceCategory, subContinent, source)
trainFactor_dummies <- model.matrix(~ . -1, data=trainFactor) %>%
  as.data.frame()
# Keep only selected categorical columns
trainNumeric <- train %>%
  dplyr::select(where(is.numeric))
train_full <- cbind(trainNumeric, trainFactor_dummies)%>%
  replace(is.na(.),0)
```

```
X <- as.matrix(train_full %>% select(-targetRevenue,-revenue))  # Exclude the target variable
y <- train_full$targetRevenue
# Fit the LASSO model with cross-validation
```

```r
set.seed(2024)
lasso_glm_model <- cv.glmnet(X, y, alpha = 1, family = "gaussian", standardize = TRUE)

print(lasso_glm_model$lambda.min)
```

```
## [1] 0.0015
```

```r
plot(lasso_glm_model)
```




```r
predictions <- predict(lasso_glm_model, newx = X, s = "lambda.min")
predictions <- as.numeric(predictions)
rmse_value <- rmse(y, predictions)
cat("RMSE: ", rmse_value, "\n")
```

```
## RMSE:  1.46
```

```r
SST <- sum((y - mean(y))^2)  # Total sum of squares
SSE <- sum((y - predictions)^2)  # Sum of squared errors
r2_value <- 1 - (SSE / SST)
cat("R-squared: ", r2_value, "\n")
```

```
## R-squared:  0.484
```

Summary of Model Performance

```r
df <- data.frame(
  Model = c("OLS", "Logistic", "lasso", "PLS"),
```

```
  Method = c("lm", "glm", "glmnet", "plsr"),
  Package = c("stats", "stats", "glmnet", "pls"),
  Hyperparameter = c("N/A", "N/A", "\\lambda", "ncomp"),
  Value = c("N/A", "N/A", "0.0015", "6"),
  RMSE = c("112", "N/A", "1.46", "1.18"),
  R2 = c("-0.275", "N/A", "0.484", "0.664")
)
```

```
# Generate the table using kable
kable(df, "latex", booktabs = TRUE, linesep = "",caption = "Summary of Model Performance") %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 3: Summary of Model Performance

Model	Method	Package	Hyperparameter	Value	RMSE	R2
OLS	lm	stats	N/A	N/A	112	-0.275
Logistic	glm	stats	N/A	N/A	N/A	N/A
lasso	glmnet	glmnet	\lambda	0.0015	1.46	0.484
PLS	plsr	pls	ncomp	6	1.18	0.664

(a) iv. Debrief

No interaction terms or model stacking were used; just a clean LASSO regression model with properly prepared data.

Data Preparation: Categorical data (like channelGrouping, browser, etc.) was converted into dummy variables so the model could work with them. Any missing data was replaced with 0 to avoid issues during modeling.

LASSO Regression: We used LASSO regression, which is good for selecting important features and avoiding overfitting. Cross-validation helped find the best value of lambda, a key parameter that controls how much the model is regularized preventing it from being too complex or too simple.

Model Performance: The model's predictions were evaluated using two metrics:

RMSE : This was 1.46, showing how far the predictions were from the true values on average.

R-squared: The model explained 48.4% of the variance in the revenue data.

Challenges: Converting categorical data and replacing missing values were the main challenges, but both were handled smoothly.

(b) Creating an Output For Competition

https://www.kaggle.com/competitions/2024-ise-dsa-5103-ida-hw-6/leaderboard