FinalProjectShark

Ricardo Lu

2024-11-30

# Load necessary libraries  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(car)

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

library(caret)

## Loading required package: lattice

library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(cluster)  
library(corrplot)

## corrplot 0.95 loaded

library(randomForest)

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

library(tree)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(tidyr)  
library(tidytext)  
library(stringr)  
library(ggfortify)  
set.seed(1) # Set seed for reproducibility

#EDA on Dataset  
# Load and explore the dataset  
df <- read.csv("Dataset 3 — Shark tank pitches.csv") # Load the dataset  
attach(df) # Attach the dataset for easy reference  
str(df) # Check the structure of the dataset

## 'data.frame': 495 obs. of 19 variables:  
## $ deal : logi FALSE TRUE TRUE FALSE FALSE TRUE ...  
## $ description : chr "Bluetooth device implant for your ear." "Retail and wholesale pie factory with two retail locations in New Jersey." "Ava the Elephant is a godsend for frazzled parents of young children everywhere. This talking medicine dispense"| \_\_truncated\_\_ "Organizing, packing, and moving services delivered by college women." ...  
## $ episode : int 1 1 1 1 1 2 2 2 2 2 ...  
## $ category : chr "Novelties" "Specialty Food" "Baby and Child Care" "Consumer Services" ...  
## $ entrepreneurs : chr "Darrin Johnson" "Tod Wilson" "Tiffany Krumins" "Nick Friedman, Omar Soliman" ...  
## $ location : chr "St. Paul, MN" "Somerset, NJ" "Atlanta, GA" "Tampa, FL" ...  
## $ website : chr "" "http://whybake.com/" "http://www.avatheelephant.com/" "http://collegehunkshaulingjunk.com/" ...  
## $ askedFor : int 1000000 460000 50000 250000 1200000 500000 200000 100000 500000 250000 ...  
## $ exchangeForStake : int 15 10 15 25 10 15 20 20 10 10 ...  
## $ valuation : int 6666667 4600000 333333 1000000 12000000 3333333 1000000 500000 5000000 2500000 ...  
## $ season : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ shark1 : chr "Barbara Corcoran" "Barbara Corcoran" "Barbara Corcoran" "Barbara Corcoran" ...  
## $ shark2 : chr "Robert Herjavec" "Robert Herjavec" "Robert Herjavec" "Robert Herjavec" ...  
## $ shark3 : chr "Kevin O'Leary" "Kevin O'Leary" "Kevin O'Leary" "Kevin O'Leary" ...  
## $ shark4 : chr "Daymond John" "Daymond John" "Daymond John" "Daymond John" ...  
## $ shark5 : chr "Kevin Harrington" "Kevin Harrington" "Kevin Harrington" "Kevin Harrington" ...  
## $ title : chr "Ionic Ear" "Mr. Tod's Pie Factory" "Ava the Elephant" "College Foxes Packing Boxes" ...  
## $ episode.season : chr "1-1" "1-1" "1-1" "1-1" ...  
## $ Multiple.Entreprenuers: logi FALSE FALSE FALSE FALSE FALSE FALSE ...

summary(df) # Summarize each column

## deal description episode category   
## Mode :logical Length:495 Min. : 1.00 Length:495   
## FALSE:244 Class :character 1st Qu.: 5.00 Class :character   
## TRUE :251 Mode :character Median :11.00 Mode :character   
## Mean :12.13   
## 3rd Qu.:18.00   
## Max. :29.00   
## entrepreneurs location website askedFor   
## Length:495 Length:495 Length:495 Min. : 10000   
## Class :character Class :character Class :character 1st Qu.: 75000   
## Mode :character Mode :character Mode :character Median : 150000   
## Mean : 258491   
## 3rd Qu.: 250000   
## Max. :5000000   
## exchangeForStake valuation season shark1   
## Min. : 3.00 Min. : 40000 Min. :1.000 Length:495   
## 1st Qu.: 10.00 1st Qu.: 440000 1st Qu.:3.000 Class :character   
## Median : 15.00 Median : 1000000 Median :4.000 Mode :character   
## Mean : 17.54 Mean : 2165615 Mean :4.048   
## 3rd Qu.: 20.00 3rd Qu.: 2000000 3rd Qu.:5.000   
## Max. :100.00 Max. :30000000 Max. :6.000   
## shark2 shark3 shark4 shark5   
## Length:495 Length:495 Length:495 Length:495   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## title episode.season Multiple.Entreprenuers  
## Length:495 Length:495 Mode :logical   
## Class :character Class :character FALSE:334   
## Mode :character Mode :character TRUE :161   
##   
##   
##

head(df) # Preview the first few rows of the data

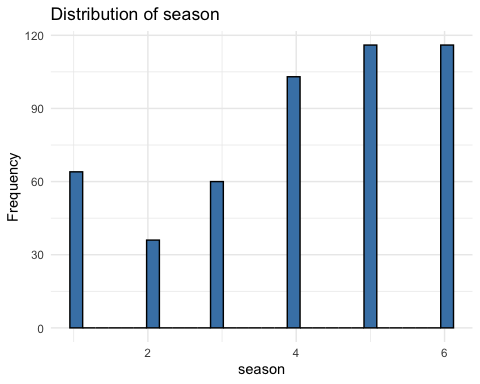
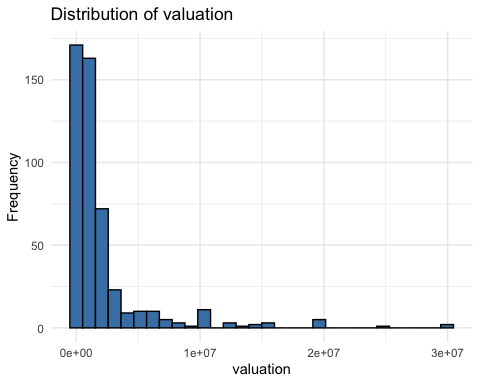
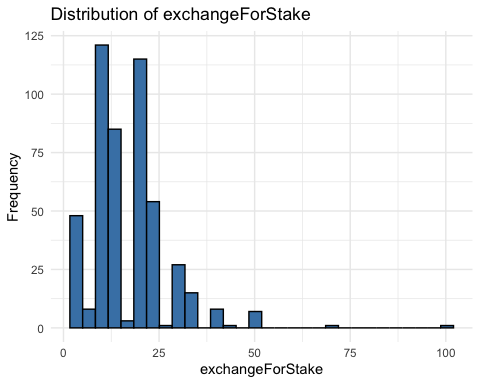
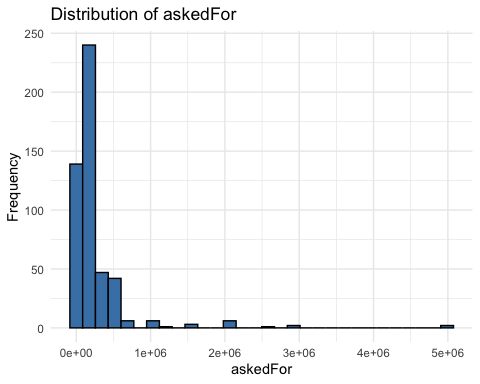
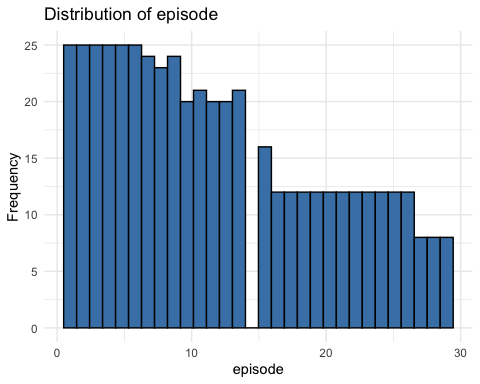
## deal  
## 1 FALSE  
## 2 TRUE  
## 3 TRUE  
## 4 FALSE  
## 5 FALSE  
## 6 TRUE  
## description  
## 1 Bluetooth device implant for your ear.  
## 2 Retail and wholesale pie factory with two retail locations in New Jersey.  
## 3 Ava the Elephant is a godsend for frazzled parents of young children everywhere. This talking medicine dispenser makes it easy to administer medicine to little ones by turning the experience more playful and by providing positive reinforcement.  
## 4 Organizing, packing, and moving services delivered by college women.  
## 5 Interactive media centers for healthcare waiting rooms offering patients web access and educational information.  
## 6 One of the first entrepreneurs to pitch on Shark Tank, Susan Knapp presented A Perfect Pear, her line of pear-focused gourmet food products. Sold across 650 retail stores, the Perfect Pear product portfolio includes jams, jellies, spreads, tapenades, vinegars, marinades, dressings and many others, all designed to showcase the flavors and health benefits of pears.  
## episode category entrepreneurs location  
## 1 1 Novelties Darrin Johnson St. Paul, MN  
## 2 1 Specialty Food Tod Wilson Somerset, NJ  
## 3 1 Baby and Child Care Tiffany Krumins Atlanta, GA  
## 4 1 Consumer Services Nick Friedman, Omar Soliman Tampa, FL  
## 5 1 Consumer Services Kevin Flannery Cary, NC  
## 6 2 Specialty Food Susan Knapp Napa Valley, CA  
## website askedFor exchangeForStake valuation  
## 1 1000000 15 6666667  
## 2 http://whybake.com/ 460000 10 4600000  
## 3 http://www.avatheelephant.com/ 50000 15 333333  
## 4 http://collegehunkshaulingjunk.com/ 250000 25 1000000  
## 5 http://www.wispots.com/ 1200000 10 12000000  
## 6 http://www.aperfectpear.com 500000 15 3333333  
## season shark1 shark2 shark3 shark4  
## 1 1 Barbara Corcoran Robert Herjavec Kevin O'Leary Daymond John  
## 2 1 Barbara Corcoran Robert Herjavec Kevin O'Leary Daymond John  
## 3 1 Barbara Corcoran Robert Herjavec Kevin O'Leary Daymond John  
## 4 1 Barbara Corcoran Robert Herjavec Kevin O'Leary Daymond John  
## 5 1 Barbara Corcoran Robert Herjavec Kevin O'Leary Daymond John  
## 6 1 Barbara Corcoran Robert Herjavec Kevin O'Leary Daymond John  
## shark5 title episode.season  
## 1 Kevin Harrington Ionic Ear 1-1  
## 2 Kevin Harrington Mr. Tod's Pie Factory 1-1  
## 3 Kevin Harrington Ava the Elephant 1-1  
## 4 Kevin Harrington College Foxes Packing Boxes 1-1  
## 5 Kevin Harrington Wispots 1-1  
## 6 Kevin Harrington A Perfect Pear 1-2  
## Multiple.Entreprenuers  
## 1 FALSE  
## 2 FALSE  
## 3 FALSE  
## 4 FALSE  
## 5 FALSE  
## 6 FALSE

# Check for missing values in the dataset  
colSums(is.na(df)) # Count missing values for each column

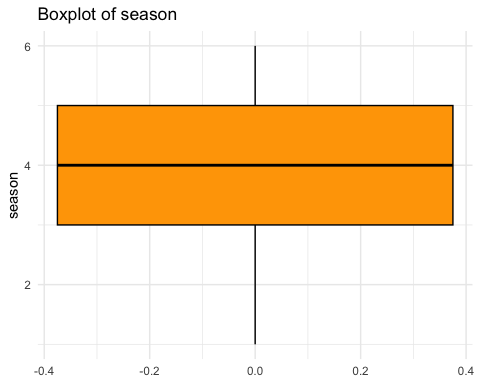
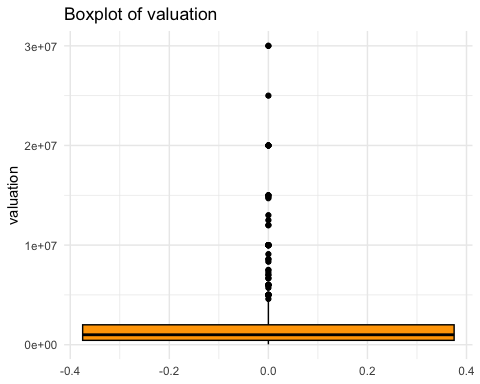
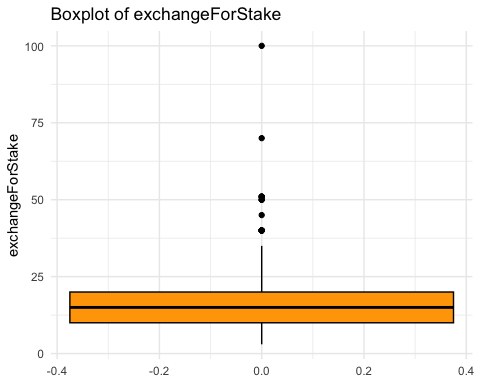
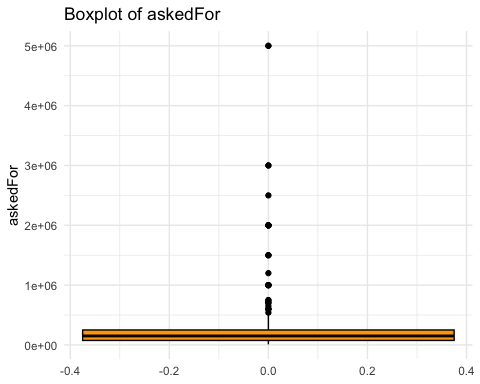
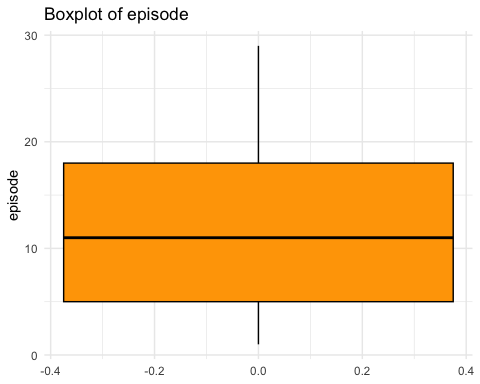
## deal description episode   
## 0 0 0   
## category entrepreneurs location   
## 0 0 0   
## website askedFor exchangeForStake   
## 0 0 0   
## valuation season shark1   
## 0 0 0   
## shark2 shark3 shark4   
## 0 0 0   
## shark5 title episode.season   
## 0 0 0   
## Multiple.Entreprenuers   
## 0

# EDA: Explore numeric features  
numeric\_features <- df %>%  
 select(where(is.numeric)) # Select only numeric columns  
  
# Plot histograms for numeric features  
for (col in colnames(numeric\_features)) {  
 p <- ggplot(df, aes\_string(x = col)) +  
 geom\_histogram(fill = "steelblue", color = "black", bins = 30) +  
 labs(title = paste("Distribution of", col), x = col, y = "Frequency") +  
 theme\_minimal()  
 print(p)  
}

## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation idioms with `aes()`.  
## ℹ See also `vignette("ggplot2-in-packages")` for more information.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



# Plot boxplots for numeric features to identify outliers  
for (col in colnames(numeric\_features)) {  
 p <- ggplot(df, aes\_string(y = col)) +  
 geom\_boxplot(fill = "orange", color = "black") +  
 labs(title = paste("Boxplot of", col), y = col) +  
 theme\_minimal()  
 print(p)  
}



# Analyze categories to see which have more successful deals  
category\_summary <- df %>%  
 group\_by(category) %>%  
 summarise(  
 category\_success = mean(deal == 1, na.rm = TRUE), # Proportion of successful deals  
 count = n() # Total count of occurrences  
 ) %>%  
 arrange(desc(count)) # Sort by count in descending order  
print(category\_summary)

## # A tibble: 54 × 3  
## category category\_success count  
## <chr> <dbl> <int>  
## 1 Specialty Food 0.548 62  
## 2 Novelties 0.457 35  
## 3 Baby and Child Care 0.625 24  
## 4 Online Services 0.455 22  
## 5 Personal Care and Cosmetics 0.4 20  
## 6 Toys and Games 0.526 19  
## 7 Storage and Cleaning Products 0.765 17  
## 8 Outdoor Recreation 0.625 16  
## 9 Electronics 0.429 14  
## 10 Consumer Services 0.231 13  
## # ℹ 44 more rows

# Count the number of deals  
table(deal)

## deal  
## FALSE TRUE   
## 244 251

# Transform skewed numeric columns using log transformation  
df$valuation <- log1p(df$valuation)  
df$askedFor <- log1p(df$askedFor)  
df$exchangeForStake <- log1p(df$exchangeForStake)

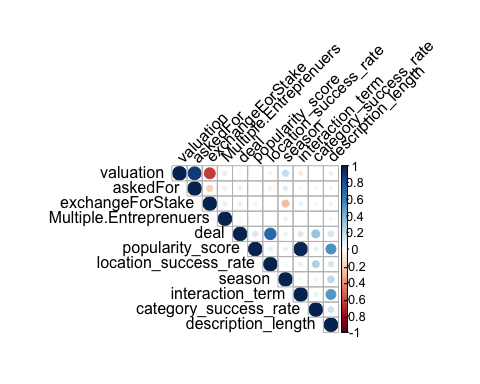
#Feature Engineering  
# Encode deal as numeric for machine learning algorithms  
df$deal <- as.numeric(df$deal)  
  
# Convert the Multiple.Entreprenuers variable to numeric  
df$Multiple.Entreprenuers <- as.numeric(df$Multiple.Entreprenuers)  
  
# Calculate average success rate for each category and add it to the dataset  
category\_success\_rate <- df %>%  
 group\_by(category) %>%  
 summarise(category\_success\_rate = mean(deal == 1, na.rm = TRUE))  
df <- df %>%  
 left\_join(category\_success\_rate, by = "category") # Merge with the original dataset  
  
# Create a new feature: length of the description  
df$description\_length <- nchar(df$description)  
  
# Text Analysis on Descriptions  
  
# Tokenize the description column into words  
df\_tokens <- df %>%  
 select(deal, description) %>% # Include deal column for filtering  
 unnest\_tokens(word, description) # Split descriptions into individual words  
  
# Remove stopwords (common words like "the", "and")  
data("stop\_words") # Load predefined stopwords  
df\_tokens <- df\_tokens %>%  
 anti\_join(stop\_words, by = "word")  
  
# Find the most common words in successful deals  
successful\_tokens <- df\_tokens %>%  
 filter(deal == 1) %>% # Filter for successful deals  
 count(word, sort = TRUE) # Count word frequencies and sort them  
  
# Extract the top 20 words in successful deals  
top\_words <- successful\_tokens %>%  
 slice\_max(n, n = 20) %>%  
 pull(word) # Extract the top words as a vector  
print(top\_words)

## [1] "makes" "company" "free" "designed" "easy"   
## [6] "natural" "line" "products" "water" "kids"   
## [11] "online" "product" "built" "children" "system"   
## [16] "box" "home" "offers" "service" "design"   
## [21] "easier" "fun" "ingredients" "light" "skin"

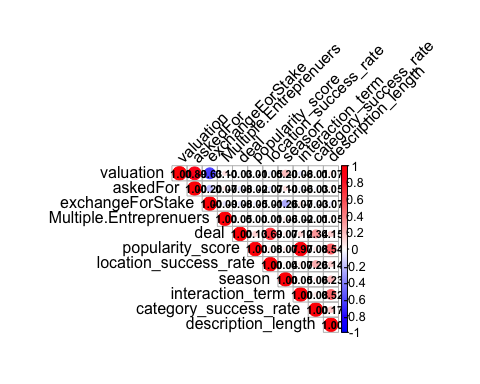
# Calculate a popularity score for each description based on top words  
df$popularity\_score <- sapply(df$description, function(desc) {  
 # Tokenize description into words  
 words <- unlist(strsplit(desc, "\\s+"))  
 # Count how many top words appear in the description  
 sum(words %in% top\_words)  
})  
  
# Interaction Terms  
df$interaction\_term <- df$exchangeForStake \* df$popularity\_score # Interaction between stake and popularity score  
  
# Popular Location Analysis  
df$city <- sapply(strsplit(df$location, ", "), function(x) x[1]) # Extract city names from the location column  
  
# Calculate success rates for each city and join with the dataset  
location\_success\_rate <- df %>%  
 group\_by(city) %>%  
 summarise(location\_success\_rate = mean(deal == 1, na.rm = TRUE))  
df <- df %>%  
 left\_join(location\_success\_rate, by = "city")  
  
# Clustering Analysis  
clustering\_features <- df %>%  
 select(valuation, askedFor, exchangeForStake, Multiple.Entreprenuers, deal, popularity\_score, location\_success\_rate, season, interaction\_term, category\_success\_rate, description\_length)  
str(clustering\_features) # Check the structure of clustering features

## 'data.frame': 495 obs. of 11 variables:  
## $ valuation : num 15.7 15.3 12.7 13.8 16.3 ...  
## $ askedFor : num 13.8 13 10.8 12.4 14 ...  
## $ exchangeForStake : num 2.77 2.4 2.77 3.26 2.4 ...  
## $ Multiple.Entreprenuers: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ deal : num 0 1 1 0 0 1 0 0 0 1 ...  
## $ popularity\_score : int 0 0 3 0 0 3 1 0 0 1 ...  
## $ location\_success\_rate : num 0.5 1 0.636 0.4 0 ...  
## $ season : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ interaction\_term : num 0 0 8.32 0 0 ...  
## $ category\_success\_rate : num 0.457 0.548 0.625 0.231 0.231 ...  
## $ description\_length : int 38 73 244 68 112 365 110 91 111 122 ...

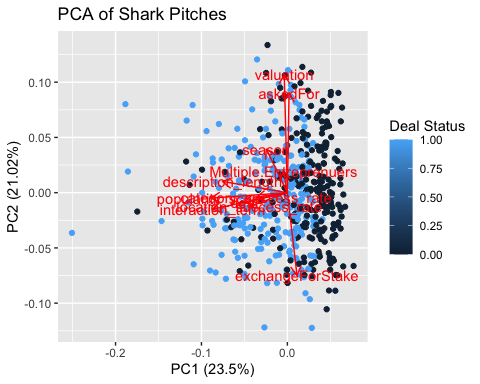
library(corrplot)  
cor\_matrix <- cor(clustering\_features, use = "complete.obs") # Use only complete cases (non-missing data)  
corrplot(cor\_matrix, method = "circle", type = "upper", tl.col = "black", tl.srt = 45)



# Visualize the correlation matrix with numbers  
corrplot(  
 cor\_matrix,  
 method = "circle", # Use circles to represent correlation  
 type = "upper", # Show only the upper triangle of the matrix  
 tl.col = "black", # Set text label color to black  
 tl.srt = 45, # Rotate text labels  
 addCoef.col = "black", # Add correlation coefficients with black color  
 number.cex = 0.7, # Adjust the size of the numbers  
 col = colorRampPalette(c("blue", "white", "red"))(200) # Custom color palette  
)



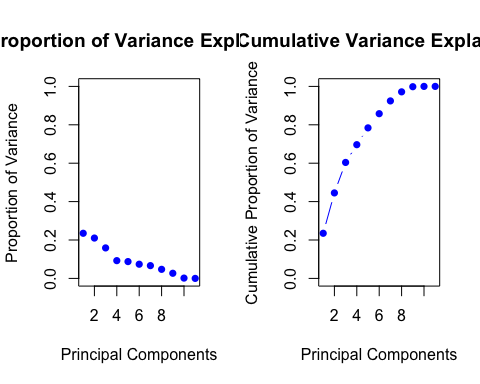
# Perform PCA  
library(ggfortify) # Load library for PCA visualization  
  
# Select features for clustering  
clustering\_features <- df %>%  
 select(valuation, askedFor, exchangeForStake,Multiple.Entreprenuers,deal,popularity\_score,location\_success\_rate,season,interaction\_term,category\_success\_rate,description\_length)  
  
# Scale the features  
pca\_result=prcomp(clustering\_features, scale=TRUE)  
  
# Visualize PCA with loadings and labels, colored by the "deal" status  
autoplot(pca\_result, data = clustering\_features, loadings = TRUE, loadings.label = TRUE, colour = 'deal') +  
 labs(title = "PCA of Shark Pitches", color = "Deal Status") # Add informative title and legend



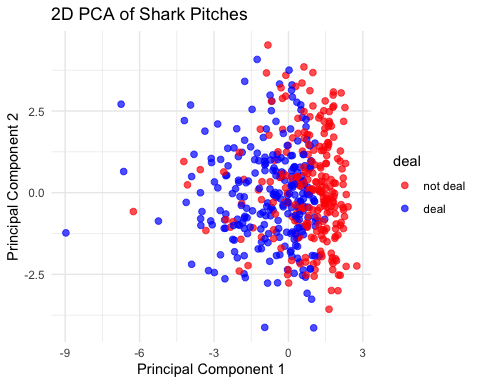
# Check PCA results to explore the variance explained by components  
pca\_result

## Standard deviations (1, .., p=11):  
## [1] 1.6078998 1.5204283 1.3215445 1.0086885 0.9814386 0.9014659 0.8553168  
## [8] 0.7208045 0.5423691 0.1332415 0.0108528  
##   
## Rotation (n x k) = (11 x 11):  
## PC1 PC2 PC3 PC4  
## valuation -0.02201265 0.64167965 -0.01103157 0.20448045  
## askedFor 0.01191285 0.54241126 0.04975023 0.37488879  
## exchangeForStake 0.06742890 -0.44969066 0.10721955 0.19551742  
## Multiple.Entreprenuers -0.02772396 0.11854653 -0.04987021 -0.60771834  
## deal -0.30087972 -0.05143663 -0.55921221 0.08027076  
## popularity\_score -0.53383188 -0.02965214 0.32341240 0.07993741  
## location\_success\_rate -0.26495011 -0.07553906 -0.55389567 0.12975995  
## season -0.14690267 0.23681239 -0.05745329 -0.59183711  
## interaction\_term -0.52319662 -0.08997854 0.33594486 0.09931560  
## category\_success\_rate -0.21756089 -0.02481366 -0.35333405 0.07966545  
## description\_length -0.45354908 0.06114890 0.13528145 -0.11808951  
## PC5 PC6 PC7 PC8  
## valuation 0.06078187 -0.010530914 -0.02004629 -0.01468235  
## askedFor 0.20473600 0.153939316 0.38068040 -0.10902122  
## exchangeForStake 0.21481721 0.281505604 0.68153774 -0.14918331  
## Multiple.Entreprenuers 0.76699036 0.139459205 -0.04240372 -0.04835338  
## deal 0.09719249 -0.223223275 0.02700694 -0.08953514  
## popularity\_score 0.06429227 -0.103804575 -0.14809041 -0.25341340  
## location\_success\_rate 0.10609388 -0.308341408 0.20370105 0.05954711  
## season -0.51203968 -0.001673378 0.43791488 -0.33849211  
## interaction\_term 0.08134670 -0.067923502 -0.06695206 -0.29094863  
## category\_success\_rate -0.14483807 0.829821834 -0.28039420 -0.14582447  
## description\_length -0.07315950 0.169110583 0.22217475 0.81672000  
## PC9 PC10 PC11  
## valuation 0.007633536 0.020479244 7.354783e-01  
## askedFor -0.033163924 -0.010471240 -5.822410e-01  
## exchangeForStake -0.070222885 -0.080087617 3.464420e-01  
## Multiple.Entreprenuers 0.038732603 0.003170951 -4.603827e-05  
## deal -0.720796586 -0.009742675 6.392549e-04  
## popularity\_score 0.059228409 -0.706097698 -4.328976e-03  
## location\_success\_rate 0.670039505 0.011574369 -1.276065e-04  
## season 0.013121447 -0.007838250 2.285309e-04  
## interaction\_term 0.031382703 0.702902359 5.044661e-03  
## category\_success\_rate 0.106706640 -0.002380768 -5.427239e-05  
## description\_length -0.088647625 0.010256459 -6.023373e-04

# Calculate the proportion of variance explained (PVE) by each principal component  
pve = (pca\_result$sdev^2) / sum(pca\_result$sdev^2)  
  
# Visualize the variance explained  
par(mfrow = c(1, 2)) # Arrange plots side-by-side  
plot(pve, ylim = c(0, 1), type = "b", col = "blue", pch = 16,  
 main = "Proportion of Variance Explained",  
 xlab = "Principal Components", ylab = "Proportion of Variance") # PVE for each component  
plot(cumsum(pve), ylim = c(0, 1), type = "b", col = "blue", pch = 16,  
 main = "Cumulative Variance Explained",  
 xlab = "Principal Components", ylab = "Cumulative Proportion of Variance") # Cumulative PVE



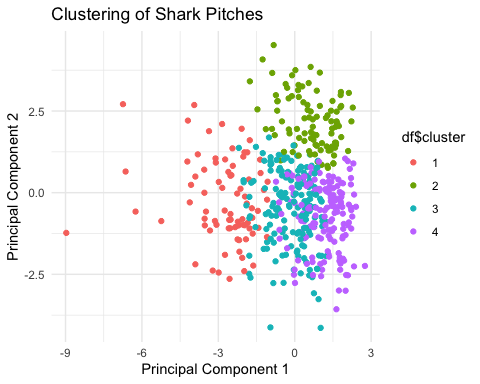
# Prepare PCA Data  
# Create a data frame with the first two principal components and the deal status for visualization  
pca\_data <- data.frame(  
 PC1 = pca\_result$x[, 1], # First principal component  
 PC2 = pca\_result$x[, 2], # Second principal component  
 deal = factor(df$deal, levels = c(0, 1), labels = c("not deal", "deal")) # Convert deal status to factor with labels  
)  
  
# Plot the first two principal components with color representing deal status  
ggplot(pca\_data, aes(x = PC1, y = PC2, color = deal)) +  
 geom\_point(size = 2, alpha = 0.7) + # Points with size 2 and some transparency  
 labs(title = "2D PCA of Shark Pitches", x = "Principal Component 1", y = "Principal Component 2") +  
 theme\_minimal() + # Use a clean, minimal theme  
 scale\_color\_manual(values = c("not deal" = "red", "deal" = "blue")) # Customize colors for deal status



# K-Means Clustering  
set.seed(1) # Set seed for reproducibility  
  
# Scale the features for clustering  
scaled\_clustering\_features <- scale(clustering\_features)  
  
# Perform K-Means clustering with different numbers of clusters  
km.2 <- kmeans(scaled\_clustering\_features, 2) # 2 clusters  
km.3 <- kmeans(scaled\_clustering\_features, 3) # 3 clusters  
km.4 <- kmeans(scaled\_clustering\_features, 4) # 4 clusters  
km.5 <- kmeans(scaled\_clustering\_features, 5) # 5 clusters  
  
# View results for 3 clusters  
km.3

## K-means clustering with 3 clusters of sizes 94, 168, 233  
##   
## Cluster means:  
## valuation askedFor exchangeForStake Multiple.Entreprenuers deal  
## 1 -0.1509613 -0.1850265 0.005676021 -0.08107271 0.53855916  
## 2 0.9825120 0.7956876 -0.745144845 0.20762974 -0.06170422  
## 3 -0.6475178 -0.4990688 0.534981922 -0.11699984 -0.17278220  
## popularity\_score location\_success\_rate season interaction\_term  
## 1 1.4703720 0.45834111 0.2145111 1.4878141  
## 2 -0.3462777 -0.11792447 0.3117189 -0.4082141  
## 3 -0.3435207 -0.09988306 -0.3112997 -0.3058994  
## category\_success\_rate description\_length  
## 1 0.528487111 1.3245705  
## 2 -0.008429057 -0.1265910  
## 3 -0.207131789 -0.4431002  
##   
## Clustering vector:  
## [1] 2 2 1 3 2 1 3 3 2 2 3 3 1 2 2 3 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [38] 3 3 2 3 2 3 3 2 3 3 2 3 3 2 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 2 3 3 3 2 2 3 1  
## [75] 3 1 3 3 3 3 3 2 3 3 3 3 3 3 3 2 3 3 3 3 3 3 1 3 2 3 2 3 2 3 3 3 3 2 1 1 3  
## [112] 1 3 3 3 1 3 3 3 2 1 1 1 3 3 1 1 3 1 3 3 3 3 2 2 3 3 3 2 3 2 2 2 2 2 3 1 3  
## [149] 3 2 3 1 3 2 3 1 3 3 2 3 3 2 2 3 1 3 2 2 3 2 3 2 3 3 3 2 3 3 1 3 2 2 2 2 3  
## [186] 1 3 1 2 3 3 3 2 2 3 3 3 3 3 3 3 2 3 2 3 3 3 3 2 2 2 2 3 3 2 3 3 3 2 2 3 3  
## [223] 1 3 2 2 3 2 3 1 3 3 2 3 3 2 1 1 1 2 3 2 1 3 2 1 3 3 3 3 3 2 3 1 2 3 2 1 3  
## [260] 2 2 2 1 3 3 2 2 1 1 3 3 3 3 1 1 1 2 1 3 2 3 3 3 1 2 3 1 3 3 2 1 3 3 2 2 2  
## [297] 2 2 1 2 1 3 1 2 3 2 2 1 3 1 3 3 3 3 2 1 2 3 1 2 2 1 3 2 2 3 2 2 1 3 3 3 2  
## [334] 3 2 1 3 1 3 1 1 1 1 3 2 1 2 3 1 1 2 2 2 2 1 2 3 3 1 2 1 1 3 2 3 3 2 1 2 2  
## [371] 1 3 1 2 3 1 1 3 1 3 2 3 2 2 3 2 3 2 2 3 3 1 2 3 2 2 2 3 3 2 2 2 2 2 2 2 1  
## [408] 3 3 2 1 1 1 1 2 3 3 2 3 2 3 1 2 1 2 3 1 3 2 3 2 3 1 1 1 3 2 2 2 2 2 2 1 3  
## [445] 2 3 2 1 3 3 2 2 3 3 2 3 2 2 3 2 3 1 2 2 2 2 1 3 2 2 3 3 2 2 2 2 3 2 2 2 3  
## [482] 2 2 1 2 3 2 2 2 2 1 2 1 3 1  
##   
## Within cluster sum of squares by cluster:  
## [1] 955.6481 1339.8906 1676.9536  
## (between\_SS / total\_SS = 26.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

# Add cluster labels from the 4-cluster solution back to the original dataset  
df$cluster <- as.factor(km.4$cluster) # Convert cluster labels to a factor for easier interpretation  
  
# Visualize the clusters in the PCA space  
ggplot(data = data.frame(pca\_result$x), aes(PC1, PC2, color = df$cluster)) +  
 geom\_point() + # Plot PCA components with cluster color  
 labs(title = "Clustering of Shark Pitches", x = "Principal Component 1", y = "Principal Component 2") +  
 theme\_minimal() # Clean visualization style



# Evaluate clustering performance by total within-cluster sum of squares  
km.2$tot.withinss # Total within-cluster sum of squares for 2 clusters

## [1] 4593.703

km.3$tot.withinss # Total within-cluster sum of squares for 3 clusters

## [1] 3972.492

km.4$tot.withinss # Total within-cluster sum of squares for 4 clusters

## [1] 3514.844

km.5$tot.withinss # Total within-cluster sum of squares for 5 clusters

## [1] 3286.088

# Summarize clusters by key features  
df\_summary <- df %>%  
 group\_by(cluster) %>% # Group by cluster  
 summarise(  
 avg\_valuation = mean(valuation, na.rm = TRUE), # Average valuation for each cluster  
 avg\_askedFor = mean(askedFor, na.rm = TRUE), # Average funding requested for each cluster  
 avg\_funding\_ratio = mean(exchangeForStake, na.rm = TRUE), # Average funding ratio (equity offered)  
 popularity\_score\_mean = mean(popularity\_score, na.rm = TRUE), # Average popularity score  
 successful\_rate = mean(deal, na.rm = TRUE), # Proportion of successful deals in each cluster  
 location\_popular\_mean = mean(location\_success\_rate, na.rm = TRUE), # Average location success score  
 multiple\_entreprenuers = mean(Multiple.Entreprenuers, na.rm = TRUE), # Proportion of pitches with multiple entrepreneurs  
 category\_popular\_mean = mean(category\_success\_rate, na.rm = TRUE), # Average category success rate  
 description\_length\_mean = mean(description\_length, na.rm = TRUE), # Average description length  
 count = n() # Number of pitches in each cluster  
 )  
  
# Print the summarized cluster features  
print(df\_summary)

## # A tibble: 4 × 11  
## cluster avg\_valuation avg\_askedFor avg\_funding\_ratio popularity\_score\_mean  
## <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 1 13.7 11.8 2.79 3.04   
## 2 2 15.5 13.1 2.32 0.490  
## 3 3 13.5 11.6 2.81 0.536  
## 4 4 13.2 11.6 3.05 0.474  
## # ℹ 6 more variables: successful\_rate <dbl>, location\_popular\_mean <dbl>,  
## # multiple\_entreprenuers <dbl>, category\_popular\_mean <dbl>,  
## # description\_length\_mean <dbl>, count <int>

# Load required libraries for decision trees and random forests  
library(tree)  
library(rpart)  
library(rpart.plot)  
library(randomForest)  
  
# Convert the target variable 'deal' into a factor for classification models  
df$deal <- as.factor(df$deal)  
  
# ----------------------------------------------  
# Random Forest Model 1: Comprehensive Feature Set  
# ----------------------------------------------  
  
# Train a random forest model using a wide set of features  
rf\_model\_1 <- randomForest(  
 deal ~ valuation + askedFor + exchangeForStake + interaction\_term +   
 category\_success\_rate + description\_length + location\_success\_rate +   
 Multiple.Entreprenuers + popularity\_score + season,  
 data = df, # Dataset to train on  
 importance = TRUE, # Calculate feature importance  
 ntree = 2000, # Number of trees to grow  
 cp = 0.0001, # Complexity parameter for tree splitting  
 do.trace = 100 # Show progress every 100 trees  
)

## ntree OOB 1 2  
## 100: 23.43% 26.64% 20.32%  
## 200: 23.23% 25.00% 21.51%  
## 300: 23.03% 25.00% 21.12%  
## 400: 23.23% 25.41% 21.12%  
## 500: 23.43% 25.41% 21.51%  
## 600: 23.03% 25.41% 20.72%  
## 700: 22.63% 25.00% 20.32%  
## 800: 22.63% 25.00% 20.32%  
## 900: 22.83% 24.59% 21.12%  
## 1000: 22.42% 24.59% 20.32%  
## 1100: 22.42% 24.59% 20.32%  
## 1200: 22.63% 25.00% 20.32%  
## 1300: 22.42% 25.00% 19.92%  
## 1400: 23.03% 25.00% 21.12%  
## 1500: 23.03% 25.00% 21.12%  
## 1600: 23.03% 25.00% 21.12%  
## 1700: 22.42% 25.00% 19.92%  
## 1800: 22.42% 24.59% 20.32%  
## 1900: 22.42% 24.59% 20.32%  
## 2000: 22.63% 25.00% 20.32%

# Print the random forest model results  
rf\_model\_1

##   
## Call:  
## randomForest(formula = deal ~ valuation + askedFor + exchangeForStake + interaction\_term + category\_success\_rate + description\_length + location\_success\_rate + Multiple.Entreprenuers + popularity\_score + season, data = df, importance = TRUE, ntree = 2000, cp = 1e-04, do.trace = 100)   
## Type of random forest: classification  
## Number of trees: 2000  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 22.63%  
## Confusion matrix:  
## 0 1 class.error  
## 0 183 61 0.2500000  
## 1 51 200 0.2031873

# View the importance of features in the model  
importance(rf\_model\_1)

## 0 1 MeanDecreaseAccuracy  
## valuation 7.4621504 3.86938482 8.8103279  
## askedFor 2.6036895 1.22432994 2.9325543  
## exchangeForStake 4.2596875 7.02917255 8.0250484  
## interaction\_term 0.4667398 6.38295412 5.4309445  
## category\_success\_rate 23.7376324 15.74602678 27.5570215  
## description\_length 0.7691998 -2.00681759 -0.9221837  
## location\_success\_rate 145.3797326 137.37914724 170.9890896  
## Multiple.Entreprenuers -2.1041529 2.86674273 0.5857326  
## popularity\_score 2.5007874 5.38206078 6.1722319  
## season -0.8979710 -0.03904669 -0.6993202  
## MeanDecreaseGini  
## valuation 19.383015  
## askedFor 16.595548  
## exchangeForStake 13.194978  
## interaction\_term 10.377800  
## category\_success\_rate 30.618585  
## description\_length 26.705573  
## location\_success\_rate 109.853077  
## Multiple.Entreprenuers 4.150149  
## popularity\_score 5.699279  
## season 9.778696

# ----------------------------------------------  
# Random Forest Model 2: Reduced Feature Set  
# ----------------------------------------------  
  
# Train a random forest model with a smaller set of features  
rf\_model\_2 <- randomForest(  
 deal ~ valuation + exchangeForStake + interaction\_term +   
 category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers,  
 data = df, # Dataset to train on  
 importance = TRUE, # Calculate feature importance  
 na.action = na.omit, # Handle missing values by omitting them  
 ntree = 2000, # Number of trees to grow  
 do.trace = 100, # Show progress every 100 trees  
 cp = 0.0001 # Complexity parameter for tree splitting  
)

## ntree OOB 1 2  
## 100: 23.23% 23.77% 22.71%  
## 200: 21.82% 23.36% 20.32%  
## 300: 23.03% 25.82% 20.32%  
## 400: 22.42% 24.59% 20.32%  
## 500: 22.63% 25.82% 19.52%  
## 600: 22.83% 25.82% 19.92%  
## 700: 23.23% 25.82% 20.72%  
## 800: 22.63% 25.82% 19.52%  
## 900: 22.42% 25.00% 19.92%  
## 1000: 22.22% 24.59% 19.92%  
## 1100: 22.42% 25.00% 19.92%  
## 1200: 22.22% 24.59% 19.92%  
## 1300: 22.42% 24.59% 20.32%  
## 1400: 22.42% 24.59% 20.32%  
## 1500: 22.42% 24.59% 20.32%  
## 1600: 22.22% 24.59% 19.92%  
## 1700: 22.22% 24.18% 20.32%  
## 1800: 22.02% 23.77% 20.32%  
## 1900: 22.02% 23.77% 20.32%  
## 2000: 21.82% 23.77% 19.92%

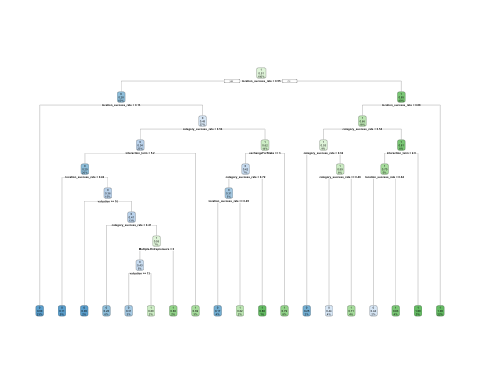
# Print the random forest model results  
rf\_model\_2

##   
## Call:  
## randomForest(formula = deal ~ valuation + exchangeForStake + interaction\_term + category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers, data = df, importance = TRUE, ntree = 2000, do.trace = 100, cp = 1e-04, na.action = na.omit)   
## Type of random forest: classification  
## Number of trees: 2000  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 21.82%  
## Confusion matrix:  
## 0 1 class.error  
## 0 186 58 0.2377049  
## 1 50 201 0.1992032

# View the importance of features in the model  
importance(rf\_model\_2)

## 0 1 MeanDecreaseAccuracy  
## valuation 8.081325 5.179989 9.811654  
## exchangeForStake 9.072864 15.482460 18.042181  
## interaction\_term 4.364082 5.812992 7.127485  
## category\_success\_rate 28.354362 23.286793 34.850582  
## location\_success\_rate 165.541115 150.422085 195.767047  
## Multiple.Entreprenuers 3.352773 7.047063 7.340581  
## MeanDecreaseGini  
## valuation 33.208806  
## exchangeForStake 19.289672  
## interaction\_term 18.728966  
## category\_success\_rate 39.881298  
## location\_success\_rate 123.576766  
## Multiple.Entreprenuers 6.497235

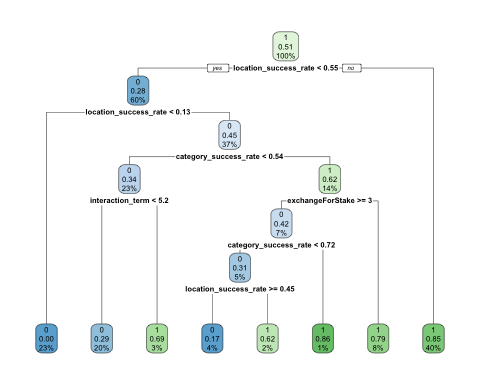
# ----------------------------------------------  
# Decision Tree Model 1: Basic Tree with Optimal Complexity Parameter  
# ----------------------------------------------  
  
# Train a decision tree model with a reduced set of features  
mytree <- rpart(  
 deal ~ valuation + exchangeForStake + interaction\_term +   
 category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers,  
 data = df, # Dataset to train on  
 control = rpart.control(cp = 0.0001) # Complexity parameter for tree splitting  
)  
  
# Plot the decision tree  
rpart.plot(mytree)



# Extract the optimal complexity parameter (cp) based on cross-validation  
opt\_cp <- mytree$cptable[which.min(mytree$cptable[,"xerror"]), "CP"]  
opt\_cp # Print the optimal complexity parameter

## [1] 0.006147541

# ----------------------------------------------  
# Decision Tree Model 2: Pruned Tree with Optimal Complexity Parameter  
# ----------------------------------------------  
  
# Train a pruned decision tree using the optimal complexity parameter  
mytree <- rpart(  
 deal ~ valuation + exchangeForStake + interaction\_term +   
 category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers,  
 data = df, # Dataset to train on  
 control = rpart.control(cp = 0.006147541) # Pruned tree with adjusted cp  
)  
  
# Plot the pruned decision tree  
rpart.plot(mytree)



# ----------------------------------------------  
# Random Forest Model 3: Pruned Random Forest  
# ----------------------------------------------  
  
# Train a random forest model with pruning (using the adjusted cp value)  
rf\_model\_3 <- randomForest(  
 deal ~ valuation + exchangeForStake + interaction\_term +   
 category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers,  
 data = df, # Dataset to train on  
 importance = TRUE, # Calculate feature importance  
 ntree = 2000, # Number of trees to grow  
 do.trace = 100, # Show progress every 100 trees  
 cp = 0.006147541 # Complexity parameter for pruning  
)

## ntree OOB 1 2  
## 100: 21.62% 25.00% 18.33%  
## 200: 22.42% 24.18% 20.72%  
## 300: 22.63% 24.59% 20.72%  
## 400: 22.63% 24.59% 20.72%  
## 500: 22.42% 24.59% 20.32%  
## 600: 22.63% 25.00% 20.32%  
## 700: 22.63% 24.59% 20.72%  
## 800: 22.63% 24.18% 21.12%  
## 900: 22.83% 24.59% 21.12%  
## 1000: 22.22% 23.77% 20.72%  
## 1100: 22.63% 24.59% 20.72%  
## 1200: 22.22% 24.59% 19.92%  
## 1300: 22.02% 23.77% 20.32%  
## 1400: 22.02% 24.18% 19.92%  
## 1500: 22.02% 24.18% 19.92%  
## 1600: 21.82% 24.18% 19.52%  
## 1700: 22.22% 24.18% 20.32%  
## 1800: 22.22% 24.18% 20.32%  
## 1900: 22.42% 24.59% 20.32%  
## 2000: 22.63% 24.59% 20.72%

# Print the random forest model results  
rf\_model\_3

##   
## Call:  
## randomForest(formula = deal ~ valuation + exchangeForStake + interaction\_term + category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers, data = df, importance = TRUE, ntree = 2000, do.trace = 100, cp = 0.006147541)   
## Type of random forest: classification  
## Number of trees: 2000  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 22.63%  
## Confusion matrix:  
## 0 1 class.error  
## 0 184 60 0.2459016  
## 1 52 199 0.2071713

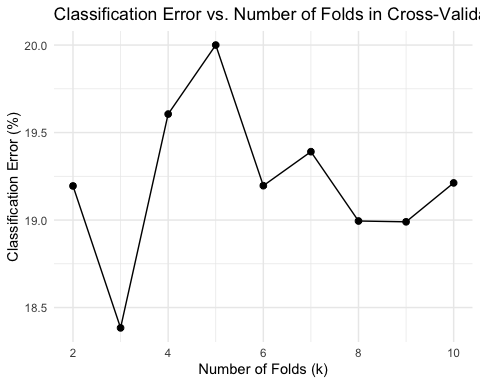
# View the importance of features in the pruned random forest  
importance(rf\_model\_3)

## 0 1 MeanDecreaseAccuracy  
## valuation 7.86407418 5.067652 9.795603  
## exchangeForStake 7.86905671 15.699941 17.412697  
## interaction\_term 3.44255718 4.146319 5.438710  
## category\_success\_rate 29.68033790 21.748035 35.464788  
## location\_success\_rate 160.04942067 154.629254 194.890563  
## Multiple.Entreprenuers -0.02276181 6.686491 5.136683  
## MeanDecreaseGini  
## valuation 32.862648  
## exchangeForStake 19.282325  
## interaction\_term 18.686619  
## category\_success\_rate 40.131672  
## location\_success\_rate 123.735836  
## Multiple.Entreprenuers 6.417495

# Define a range of k values for k-fold cross-validation  
k\_values <- 2:10  
set.seed(1) # Set seed for reproducibility  
  
# Initialize a vector to store classification error results for each k  
classification\_error\_results <- numeric(length(k\_values))  
  
# Loop through each value of k  
for (k in k\_values) {  
 # Create k folds for cross-validation  
 folds <- cut(seq(1, nrow(df)), breaks = k, labels = FALSE)  
   
 # Initialize a vector to store classification errors for each fold  
 fold\_classification\_error <- numeric(k)  
   
 # Perform cross-validation for the current k  
 for (i in 1:k) {  
 # Identify the indices for the test set for the current fold  
 test\_idx <- which(folds == i, arr.ind = TRUE)  
 test\_data <- df[test\_idx, ] # Create the test dataset  
 train\_fold <- df[-test\_idx, ] # Create the training dataset  
   
 # Train a logistic regression model on the training fold  
 mlogit <- glm(  
 deal ~ valuation + exchangeForStake + interaction\_term +   
 category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers,  
 data = train\_fold,  
 family = "binomial" # Logistic regression for binary classification  
 )  
   
 # Predict probabilities on the test set  
 predictions <- predict(mlogit, newdata = test\_data, type = "response")  
   
 # Convert probabilities to binary predictions (threshold = 0.5)  
 predicted\_class <- ifelse(predictions > 0.5, 1, 0)  
   
 # Convert actual deal values to numeric for comparison  
 actual\_class <- as.numeric(test\_data$deal) - 1 # Assuming levels are 0 and 1  
   
 # Calculate classification error for the current fold  
 fold\_classification\_error[i] <- mean(predicted\_class != actual\_class) \* 100 # Percentage error  
 }  
   
 # Store the average classification error for the current k  
 classification\_error\_results[k - 1] <- mean(fold\_classification\_error)  
}  
  
# Combine k values and classification error results into a data frame for visualization  
cv\_results <- data.frame(k = k\_values, ClassificationError = classification\_error\_results)  
  
# View the cross-validation results  
print(cv\_results)

## k ClassificationError  
## 1 2 19.19485  
## 2 3 18.38384  
## 3 4 19.60563  
## 4 5 20.00000  
## 5 6 19.19630  
## 6 7 19.39063  
## 7 8 18.99458  
## 8 9 18.98990  
## 9 10 19.21224

# Visualize the relationship between k and classification error  
library(ggplot2)  
ggplot(cv\_results, aes(x = k, y = ClassificationError)) +  
 geom\_line() + # Line plot  
 geom\_point(size = 2) + # Add points  
 labs(  
 title = "Classification Error vs. Number of Folds in Cross-Validation",  
 x = "Number of Folds (k)",  
 y = "Classification Error (%)"  
 ) +  
 theme\_minimal() # Minimal theme for cleaner visualization



# Calculate Variance Inflation Factor (VIF) to check for multicollinearity  
vif\_values <- vif(mlogit) # Calculate VIF for logistic regression model  
print(vif\_values) # Print VIF values

## valuation exchangeForStake interaction\_term   
## 1.719361 1.774437 1.029432   
## category\_success\_rate location\_success\_rate Multiple.Entreprenuers   
## 1.016014 1.071866 1.022274

# Load the gbm library for Boosting  
library(gbm)

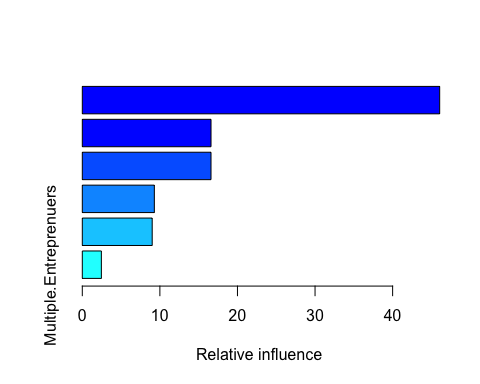
## Loaded gbm 2.2.2

## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3

set.seed(1) # Set seed for reproducibility  
  
# Ensure the target variable is numeric for boosting (convert factor levels to 0 and 1)  
df$deal <- as.numeric(df$deal) - 1  
  
# Define the range of k values for cross-validation  
k\_values <- 2:10  
  
# Initialize a vector to store classification error results for boosting  
classification\_error\_results <- numeric(length(k\_values))  
  
# Loop through each value of k  
for (k in k\_values) {  
 # Create k folds for cross-validation  
 folds <- cut(seq(1, nrow(df)), breaks = k, labels = FALSE)  
   
 # Initialize a vector to store classification errors for each fold  
 fold\_classification\_error <- numeric(k)  
   
 # Perform cross-validation for the current k  
 for (i in 1:k) {  
 # Identify the indices for the test set for the current fold  
 test\_idx <- which(folds == i, arr.ind = TRUE)  
 test\_data <- df[test\_idx, ] # Create the test dataset  
 train\_fold <- df[-test\_idx, ] # Create the training dataset  
   
 # Train the boosting model on the training fold  
 boosted <- gbm(  
 deal ~ valuation + exchangeForStake + interaction\_term +   
 category\_success\_rate + location\_success\_rate + Multiple.Entreprenuers,  
 data = train\_fold, # Training dataset  
 distribution = "bernoulli", # Distribution for binary classification  
 n.trees = 20000, # Number of trees  
 interaction.depth = 4 # Depth of each tree  
 )  
   
 # Predict class probabilities on the test set  
 predicted\_probabilities <- predict(boosted, newdata = test\_data, n.trees = 20000, type = "response")  
   
 # Convert probabilities to binary predictions (threshold = 0.5)  
 predicted\_class <- ifelse(predicted\_probabilities > 0.5, 1, 0)  
   
 # Calculate classification error for the current fold  
 fold\_classification\_error[i] <- mean(predicted\_class != test\_data$deal) \* 100 # Percentage error  
 }  
   
 # Store the average classification error for the current k  
 classification\_error\_results[k - 1] <- mean(fold\_classification\_error)  
}  
  
# Combine k values and classification error results into a data frame  
cv\_results <- data.frame(k = k\_values, ClassificationError = classification\_error\_results)  
  
# View the cross-validation results for Boosting  
print(cv\_results)

## k ClassificationError  
## 1 2 24.44495  
## 2 3 26.46465  
## 3 4 26.66863  
## 4 5 25.85859  
## 5 6 26.47419  
## 6 7 24.83760  
## 7 8 23.82668  
## 8 9 25.25253  
## 9 10 25.44082

# Print summary of the last trained boosting model  
summary(boosted) # Feature importance and partial dependence



## var rel.inf  
## location\_success\_rate location\_success\_rate 46.064917  
## category\_success\_rate category\_success\_rate 16.591597  
## valuation valuation 16.590232  
## interaction\_term interaction\_term 9.286216  
## exchangeForStake exchangeForStake 9.009556  
## Multiple.Entreprenuers Multiple.Entreprenuers 2.457481