Instagram Fake/Spam – Genuine Id Detection

Project Report

Submitted to

UNIFIED MENTOR

Submitted by

Mr. Abdul Razzaq

Acknowledgement

I would like to express my sincere gratitude to my faculty guide for their continuous guidance, valuable suggestions, and encouragement throughout the completion of this project. I am also thankful to my institution for providing the necessary resources and learning environment to carry out this work. Finally, I would like to acknowledge the open-source R community for the libraries and tools that made data analysis and model development possible.

Table of Contents

Chapter 1

Abstract

The rapid growth of social media platforms has led to an increasing number of fake and spam accounts, posing challenges to platform security and user trust. This project aims to develop a machine learning–based system to automatically identify fake Instagram accounts by analysing user behaviour, engagement patterns, and profile attributes. The study utilizes a structured dataset containing account metadata and activity-related features, which undergoes preprocessing, exploratory data analysis, feature engineering, and feature selection to enhance predictive performance. Multiple classification models, including Logistic Regression, Random Forest, Support Vector Machine, and XGBoost, are trained and evaluated using cross-validation and standard performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Experimental results demonstrate that the Random Forest marginally outperformed XGBoost and was selected as the final model, effectively distinguishing between genuine and fake accounts. The findings highlight the potential of machine learning techniques in automating fake account detection and improving the integrity of social media platforms.

Objectives

This project explores the application of machine learning techniques to address the growing problem of fake and spam accounts on Instagram. By analysing account-level attributes and behavioural patterns, the study seeks to uncover key indicators that differentiate genuine users from automated or malicious profiles. The insights gained from this analysis form the foundation for building a reliable classification model capable of detecting fake accounts effectively. The main objectives of this project are:

- To analyse Instagram account activity and identify meaningful behavioural patterns.
- To engineer useful engagement-based and ratio-based features for improved detection.
- To apply feature selection methods to identify the most predictive variables.
- To train multiple machine learning models and select the best-performing one.
- To evaluate model performance using accuracy, precision, recall, F1-score, and ROC-AUC.
- To generate predictions on unseen test data to classify accounts as fake or genuine.

Chapter 2

Data Collection

The dataset used for this project are: test and train data in csv

The dataset was provided in CSV format and imported into **RStudio** for preprocessing and analysis. Before beginning the modelling process, the dataset was examined for missing values, inconsistencies, and data types to ensure that it was clean and suitable for machine learning.

Data Description

The dataset consists of two files:

- **train.csv** – contains labelled Instagram accounts (genuine/fake).
- **test.csv** – contains unlabelled accounts used for final predictions.

Common features include:

- followers, following, posts
- profile picture indicator
- username and full-name characteristics
- Various other numeric and categorical attributes

The target variable indicates whether the account is **genuine** or **fake**.

Chapter 3

Tools and Technologies

This project was developed using a combination of data analysis tools, programming libraries, and visualization frameworks. These tools facilitated efficient data preprocessing, exploratory analysis, feature engineering, model building, and evaluation. The selection of tools was based on their robustness, ease of use, and suitability for machine learning workflows.

R Programming Language

R was used as the primary programming language due to its powerful statistical capabilities and rich ecosystem of machine learning libraries. R provides extensive support for data manipulation, visualization, and model evaluation, making it a suitable choice for predictive analytics projects.

RStudio

RStudio served as the integrated development environment (IDE) for writing and executing R code. Its user-friendly interface, built-in console, visual debugging features, and project-based workflow made it easier to manage the entire analysis process. RStudio also supports package management, visualization, and real-time code execution, which streamlined the development and testing phases.

R Libraries Used

- **dplyr** – Used for data manipulation, filtering, and feature transformation.
- **tidyverse** – Provides a unified framework for data cleaning, analysis, and visualization.
- **ggplot2** – Used to create visualizations for exploratory data analysis.
- **caret** – Used for feature selection, model training, cross-validation, and evaluation.
- **randomForest** – Used to build ensemble classification models and compute feature importance.
- **xgboost** – Used to train a high-performance gradient boosting classification model.
- **pROC** – Used to evaluate model performance using ROC curves and AUC.
- **corrplot** – Used to visualize correlations and detect multicollinearity among features.

These libraries provided the foundation for building a smooth and efficient machine learning pipeline.

Microsoft Excel

Excel was used optionally for quick data inspection, verifying data formats, and manually checking the structure of the dataset before importing it into R. It also served as a useful tool for reviewing generated CSV outputs such as feature importance tables and confusion matrices.

Model Selection

Model selection was performed by training and comparing multiple machine learning algorithms to identify the most suitable approach for detecting fake Instagram accounts. The models evaluated in this study include Logistic Regression, Random Forest, Support Vector Machine (SVM), and XGBoost. Each model was trained on the same feature set using consistent preprocessing steps to ensure a fair comparison.

Performance evaluation was conducted using cross-validation and standard classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Special emphasis was placed on ROC-AUC and recall, as correctly identifying fake accounts is critical in minimizing security risks. The models were also assessed for their ability to generalize and handle non-linear relationships present in the data.

Among all evaluated models, Random Forest demonstrated superior performance across most evaluation metrics, particularly in terms of ROC-AUC and overall classification accuracy. Additionally, Random Forest effectively captured complex feature interactions and showed better robustness to class imbalance compared to other models. Based on these results, Random Forest was selected as the final model for test predictions and further analysis.

Model Building

The model-building phase focuses on training classification algorithms to distinguish between genuine and fake Instagram accounts. Using the refined set of selected features, several machine-learning models were trained and compared.

To ensure reliable performance estimation, all models were evaluated using 5-fold cross-validation. The following algorithms were selected due to their strong performance in binary classification tasks:

1. Logistic Regression

A baseline linear model used to assess fundamental relationships between predictors and the target.

2. Random Forest

An ensemble of decision trees known for handling non-linear relationships and noisy data effectively.

3. XGBoost

A powerful gradient-boosting method that typically delivers state-of-the-art performance on tabular datasets.

4. Support Vector Machine (SVM)

A margin-based classifier useful when the classes are hard to separate.

Each model was tuned using caret's resampling framework, and the best-performing model was selected based on accuracy, F1-score, and ROC-AUC.

Chapter 4

Data Preparation

- Install required libraries and packages:

```
#Instagram Fake spam detection

#Install required packages and libraries

install.packages("dplyr")
install.packages("tidyverse")
install.packages("ggplot2")
install.packages("caret")
install.packages("randomForest")
install.packages("xgboost")
install.packages("pROC")
install.packages("corrplot")
install.packages("PRROC")

library(dplyr)
library(ggplot2)
library(tidyverse)
library(corrplot)
library(gridExtra)
library(caret)
library(randomForest)
library(pROC)
```

- Load the data:

```
#Load test abd train data

train <- read.csv("C:/Users/abdul/Downloads/train.csv")
test <- read.csv("C:/Users/abdul/Downloads/test.csv")
```

- Data Preview:

```
#train set preview

head(train)                 #overview of first few rows
colSums(is.na(train))       #check any missing value
dim(train)                  #data dimention
str(train)                  #show each column and its contetnt
summary(train)              #dataset summary
sum(duplicated(train))      #check duplicate values

#test set preview

head(test)                 #overview of first few rows
colSums(is.na(test))       #check any missing value
dim(test)                  #data dimention
str(test)                  #show each column and its contetnt
summary(test)              #dataset summary
sum(duplicated(test))      #check duplicate values
```

```
> head(train)                 #overview of first few rows
  profile.pic nums.length.username fullname.words nums.length.fullname name..username
1           1                 0.27              0                     0              0
2           1                 0.00              2                     0              0
3           1                 0.10              2                     0              0
4           1                 0.00              1                     0              0
5           1                 0.00              2                     0              0
6           1                 0.00              4                     0              0
  description.length external.URL private X.posts X.followers X.follows fake
1                 53            0       0      32        1000       955    0
2                 44            0       0     286        2740       533    0
3                  0            0       1      13         159        98    0
4                 82            0       0     679         414       651    0
5                  0            0       1       6         151       126    0
6                 81            1       0     344      669987       150    0
```

```
> colSums(is.na(train))       #check any missing value
        profile.pic nums.length.username       fullname.words nums.length.fullname
                  0                     0                    0                    0
     name..username   description.length         external.URL              private
                  0                     0                    0                    0
            X.posts           X.followers            X.follows                 fake
                  0                     0                    0                    0
```

```
> dim(train)                  #data dimention
[1] 576  12
> str(train)                  #show each column and its contetnt
'data.frame':   576 obs. of  12 variables:
 $ profile.pic         : int  1 1 1 1 1 1 1 1 1 1 ...
 $ nums.length.username: num  0.27 0 0.1 0 0 0 0 0 0 0 ...
 $ fullname.words      : int  0 2 2 1 2 4 2 2 0 2 ...
 $ nums.length.fullname: num  0 0 0 0 0 0 0 0 0 0 ...
 $ name..username      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ description.length  : int  53 44 0 82 0 81 50 0 71 40 ...
 $ external.URL        : int  0 0 0 0 0 1 0 0 0 1 ...
 $ private             : int  0 0 1 0 1 0 0 0 0 0 ...
 $ X.posts             : int  32 286 13 679 6 344 16 33 72 213 ...
 $ X.followers         : int  1000 2740 159 414 151 669987 122 1078 1824 12945 ...
 $ X.follows           : int  955 533 98 651 126 150 177 76 2713 813 ...
 $ fake                : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
> summary(train)              #dataset summary
  profile.pic     nums.length.username fullname.words  nums.length.fullname name..username
 Min.   :0.0000   Min.   :0.0000       Min.   : 0.00   Min.   :0.00000      Min.   :0.00000
 1st Qu.:0.0000   1st Qu.:0.0000       1st Qu.: 1.00   1st Qu.:0.00000      1st Qu.:0.00000
 Median :1.0000   Median :0.0000       Median : 1.00   Median :0.00000      Median :0.00000
 Mean   :0.7014   Mean   :0.1638       Mean   : 1.46   Mean   :0.03609      Mean   :0.03472
 3rd Qu.:1.0000   3rd Qu.:0.3100       3rd Qu.: 2.00   3rd Qu.:0.00000      3rd Qu.:0.00000
 Max.   :1.0000   Max.   :0.9200       Max.   :12.00   Max.   :1.00000      Max.   :1.00000
 description.length  external.URL       private          X.posts         X.followers
 Min.   :  0.00     Min.   :0.0000   Min.   :0.0000   Min.   :   0.0   Min.   :0.000e+00
 1st Qu.:  0.00     1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:   0.0   1st Qu.:3.900e+01
 Median :  0.00     Median :0.0000   Median :0.0000   Median :   9.0   Median :1.505e+02
 Mean   : 22.62     Mean   :0.1163   Mean   :0.3819   Mean   : 107.5   Mean   :8.531e+04
 3rd Qu.: 34.00     3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:  81.5   3rd Qu.:7.160e+02
 Max.   :150.00     Max.   :1.0000   Max.   :1.0000   Max.   :7389.0   Max.   :1.534e+07
   X.follows          fake
 Min.   :   0.0   Min.   :0.0
 1st Qu.:  57.5   1st Qu.:0.0
 Median : 229.5   Median :0.5
 Mean   : 508.4   Mean   :0.5
 3rd Qu.: 589.5   3rd Qu.:1.0
 Max.   :7500.0   Max.   :1.0
> sum(duplicated(train))     #check duplicate values
[1] 2
```

```
> head(test)                  #overview of first few rows
  profile.pic nums.length.username fullname.words nums.length.fullname name..username
1           1                 0.33              1                 0.33              1
2           1                 0.00              5                 0.00              0
3           1                 0.00              2                 0.00              0
4           1                 0.00              1                 0.00              0
5           1                 0.50              1                 0.00              0
6           1                 0.00              1                 0.00              0
  description.length external.URL private X.posts X.followers X.follows fake
1                 30            0       1      35         488       604    0
2                 64            0       1       3          35         6    0
3                 82            0       1     319         328       668    0
4                143            0       1     273       14890      7369    0
5                 76            0       1       6         225       356    0
6                  0            0       1       6         362       424    0
```

```
> colSums(is.na(test))       #check any missing value
        profile.pic nums.length.username       fullname.words nums.length.fullname
                  0                     0                    0                    0
     name..username   description.length         external.URL              private
                  0                     0                    0                    0
            X.posts           X.followers            X.follows                 fake
                  0                     0                    0                    0
```

```
> dim(test)                  #data dimention
[1] 120  12
> str(test)                  #show each column and its contetnt
'data.frame':   120 obs. of  12 variables:
 $ profile.pic         : int  1 1 1 1 1 1 1 1 1 1 ...
 $ nums.length.username: num  0.33 0 0 0 0.5 0 0 0 0 0 ...
 $ fullname.words      : int  1 5 2 1 1 1 1 2 2 1 ...
 $ nums.length.fullname: num  0.33 0 0 0 0 0 0 0 0 0 ...
 $ name..username      : int  1 0 0 0 0 0 0 0 0 0 ...
 $ description.length  : int  30 64 82 143 76 0 132 0 96 78 ...
 $ external.URL        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ private             : int  1 1 1 1 1 1 1 1 1 1 ...
 $ X.posts             : int  35 3 319 273 6 6 9 19 17 9 ...
 $ X.followers         : int  488 35 328 14890 225 362 213 552 122 834 ...
 $ X.follows           : int  604 6 668 7369 356 424 254 521 143 358 ...
 $ fake                : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
> summary(test)              #dataset summary
  profile.pic     nums.length.username fullname.words nums.length.fullname name..username
 Min.   :0.0000   Min.   :0.0000       Min.   :0.00   Min.   :0.00000      Min.   :0.00000
 1st Qu.:1.0000   1st Qu.:0.0000       1st Qu.:1.00   1st Qu.:0.00000      1st Qu.:0.00000
 Median :1.0000   Median :0.0000       Median :1.00   Median :0.00000      Median :0.00000
 Mean   :0.7583   Mean   :0.1799       Mean   :1.55   Mean   :0.07133      Mean   :0.04167
 3rd Qu.:1.0000   3rd Qu.:0.3300       3rd Qu.:2.00   3rd Qu.:0.00000      3rd Qu.:0.00000
 Max.   :1.0000   Max.   :0.8900       Max.   :9.00   Max.   :1.00000      Max.   :1.00000
 description.length  external.URL    private          X.posts          X.followers
 Min.   :  0.00      Min.   :0.0   Min.   :0.0000   Min.   :   0.00   Min.   :      0.0
 1st Qu.:  0.00      1st Qu.:0.0   1st Qu.:0.0000   1st Qu.:   1.00   1st Qu.:     67.2
 Median :  0.00      Median :0.0   Median :0.0000   Median :   8.00   Median :    216.5
 Mean   : 27.20      Mean   :0.1   Mean   :0.3083   Mean   :  82.87   Mean   :  49594.7
 3rd Qu.: 45.25      3rd Qu.:0.0   3rd Qu.:1.0000   3rd Qu.:  58.25   3rd Qu.:    593.2
 Max.   :149.00      Max.   :1.0   Max.   :1.0000   Max.   :1879.00   Max.   :4021842.0
   X.follows           fake
 Min.   :   1.0   Min.   :0.0
 1st Qu.: 119.2   1st Qu.:0.0
 Median : 354.5   Median :0.5
 Mean   : 779.3   Mean   :0.5
 3rd Qu.: 668.2   3rd Qu.:1.0
 Max.   :7453.0   Max.   :1.0
> sum(duplicated(test))     #check duplicate values
[1] 2
```

- Remove duplicate values:

```r
#remove and check duplicate values

train <- train[!duplicated(train), ]
test <- test[!duplicated(test), ]
sum(duplicated(train))
sum(duplicated(test))      #check duplicate values
```

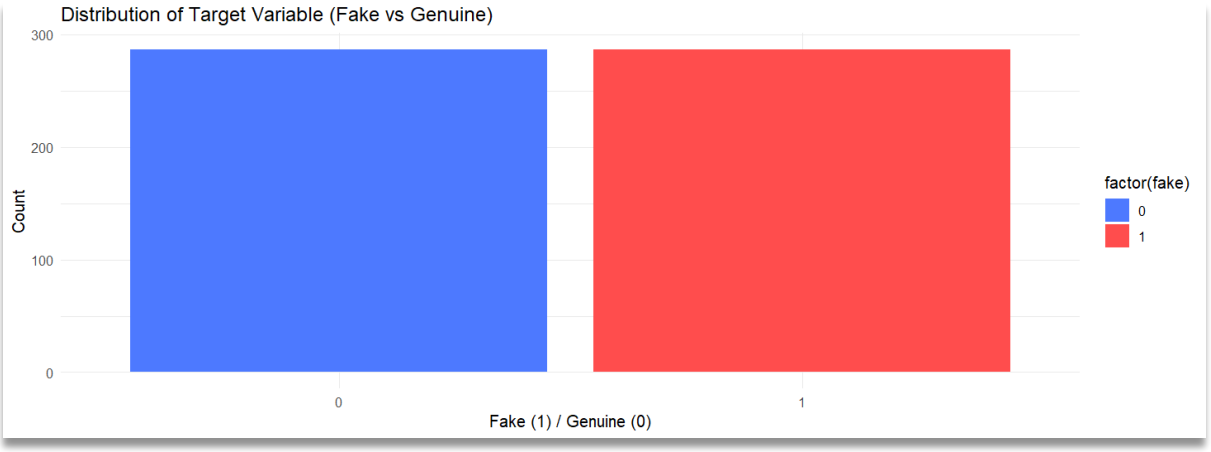
```
> train <- train[!duplicated(train), ]
> test <- test[!duplicated(test), ]
> sum(duplicated(train))
[1] 0
> sum(duplicated(test))      #check duplicate values
[1] 0
>
```

Exploratory Data Analysis

- Here we are applying EDA on train set only.
- Class distribution of fake and genuine accounts with visualization:

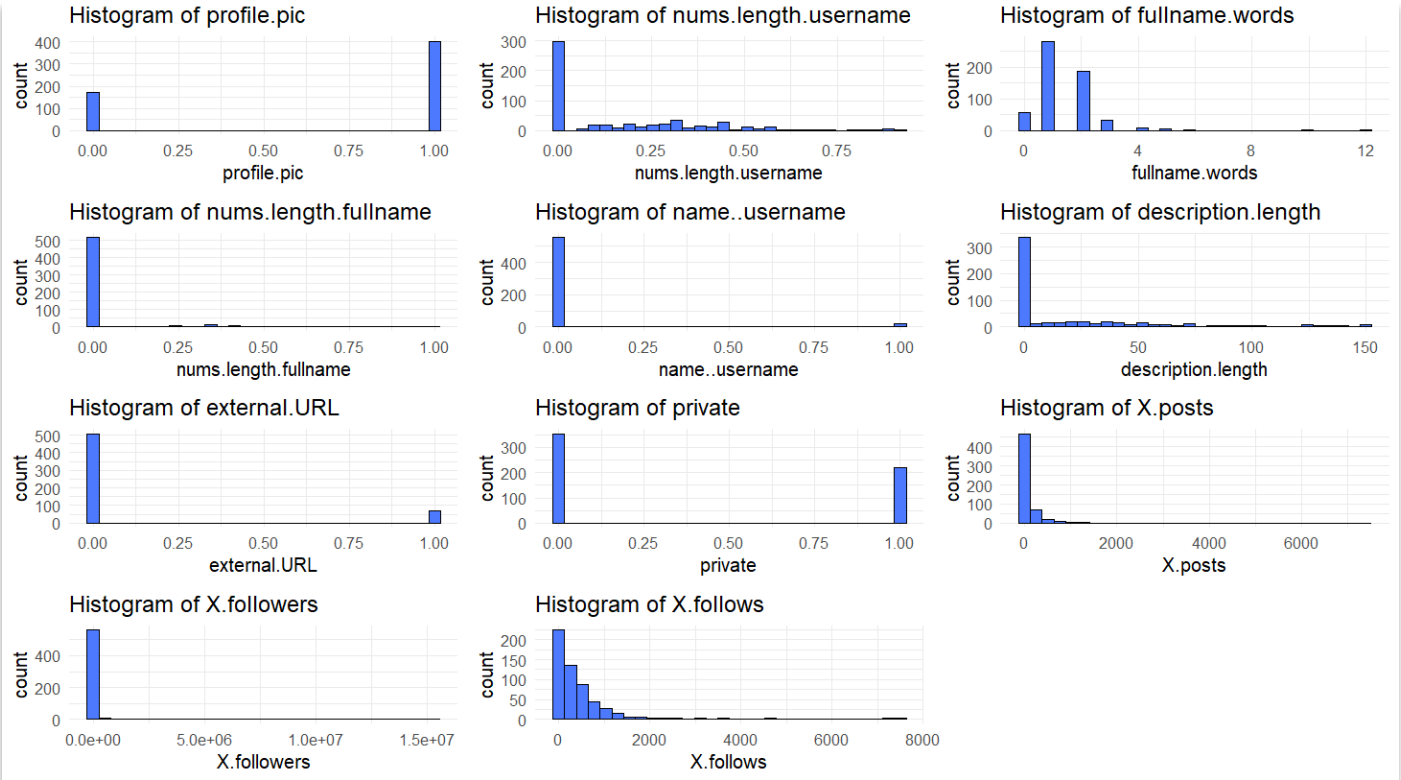
```r
#Class distribution of fake and genuine accounts

table(train$fake)
cat("\nClass Distribution (%):\n")
round(prop.table(table(train$fake)) * 100, 2)

ggplot(train, aes(x = factor(fake), fill = factor(fake))) +
  geom_bar() +
  labs(title = "Distribution of Target Variable (Fake vs Genuine)",
       x = "Fake (1) / Genuine (0)", y = "Count") +
  scale_fill_manual(values = c("#4D79FF", "#FF4D4D")) +
  theme_minimal()
```

```
> table(train$fake)

  0   1
287 287
> cat("\nClass Distribution (%):\n")

Class Distribution (%):
> round(prop.table(table(train$fake)) * 100, 2)

 0  1
50 50
```



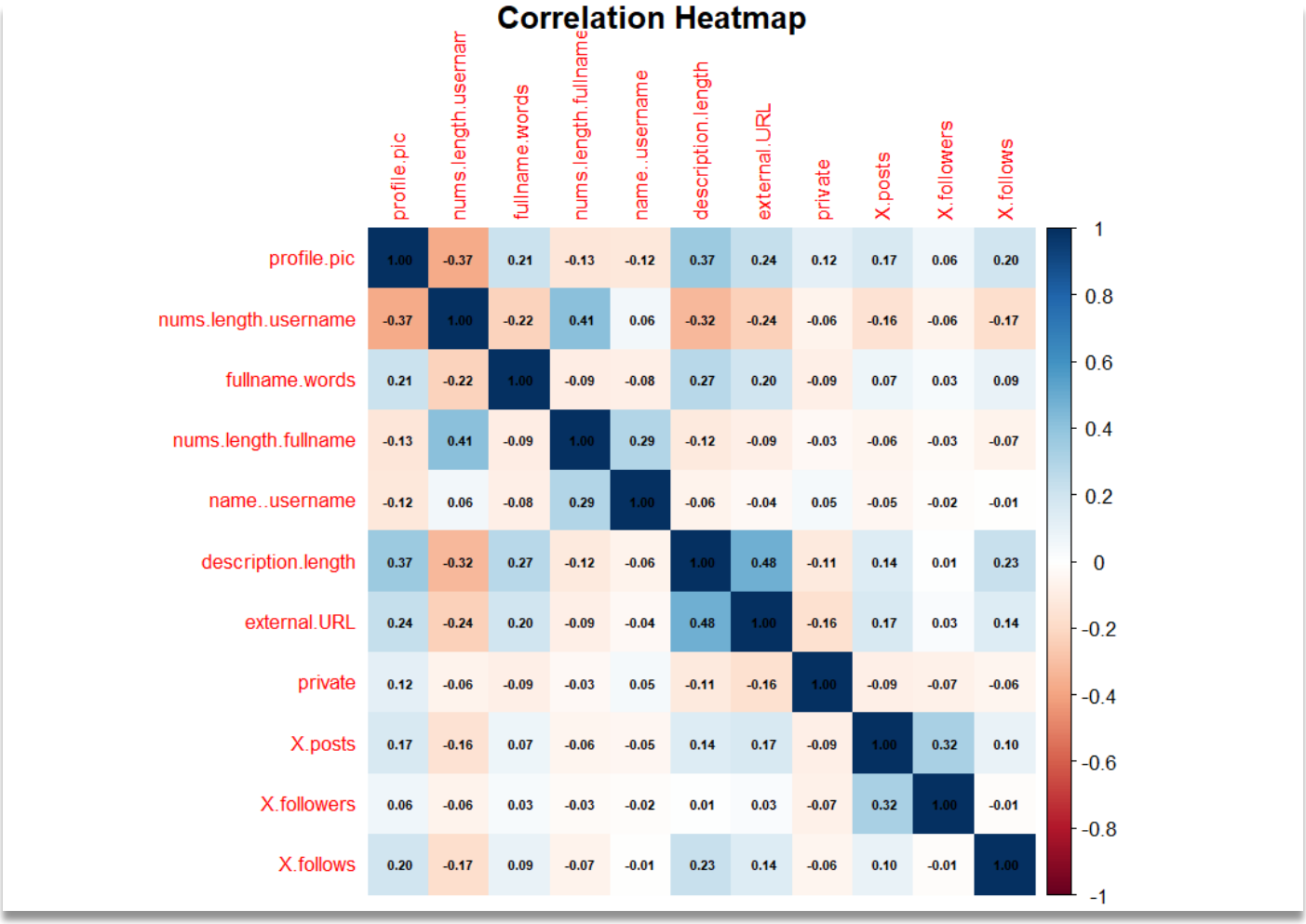
- Visualizing numerical features:

```
#Histograms for Numerical Features

num_cols <- train %>% select_if(is.numeric) %>% select(-fake)

hist_plots <- lapply(names(num_cols), function(col) {
  ggplot(train, aes_string(x = col)) +
    geom_histogram(bins = 30, fill = "#4D79FF", color = "black") +
    labs(title = paste("Histogram of", col)) +
    theme_minimal()
})

do.call(grid.arrange, c(hist_plots, ncol = 3))
```



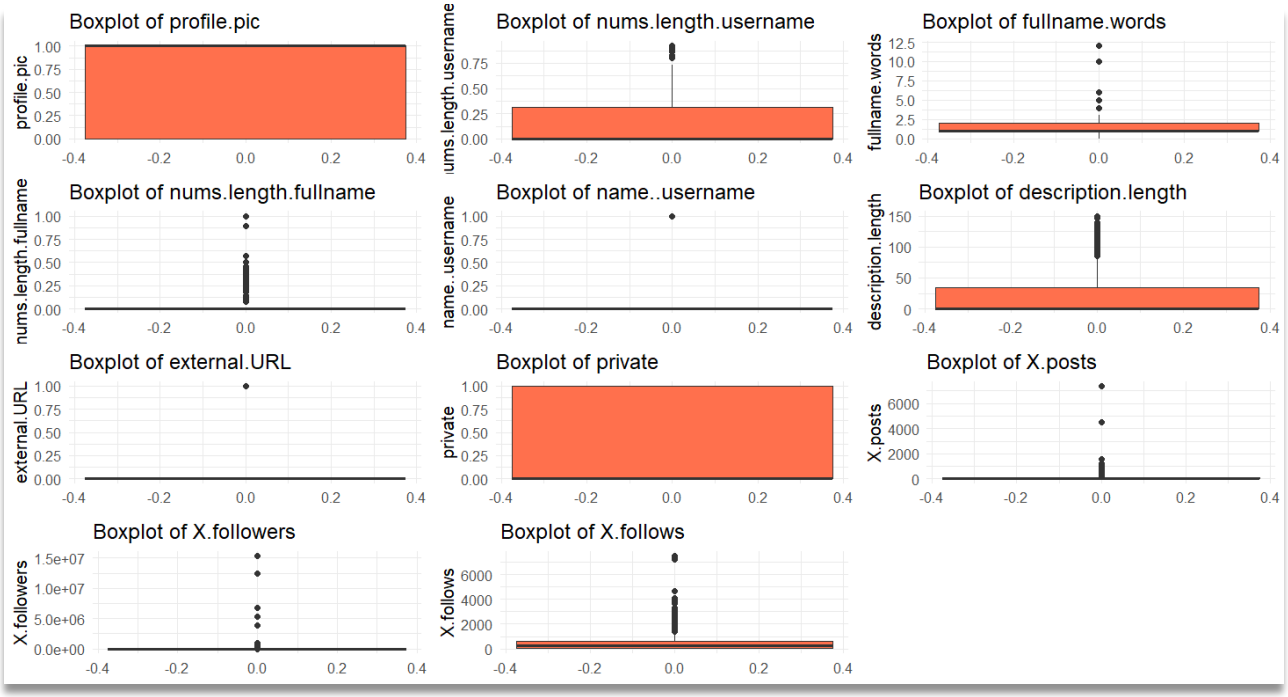
- Correlation Matrix:

```
#correlation matrix

corr_matrix <- cor(num_cols)

corrplot(corr_matrix,
         method = "color",
         tl.cex = 0.8,
         addCoef.col = "black",
         number.cex = 0.5,
         title = "Correlation Heatmap",
         mar = c(0, 0, 1, 0))
```



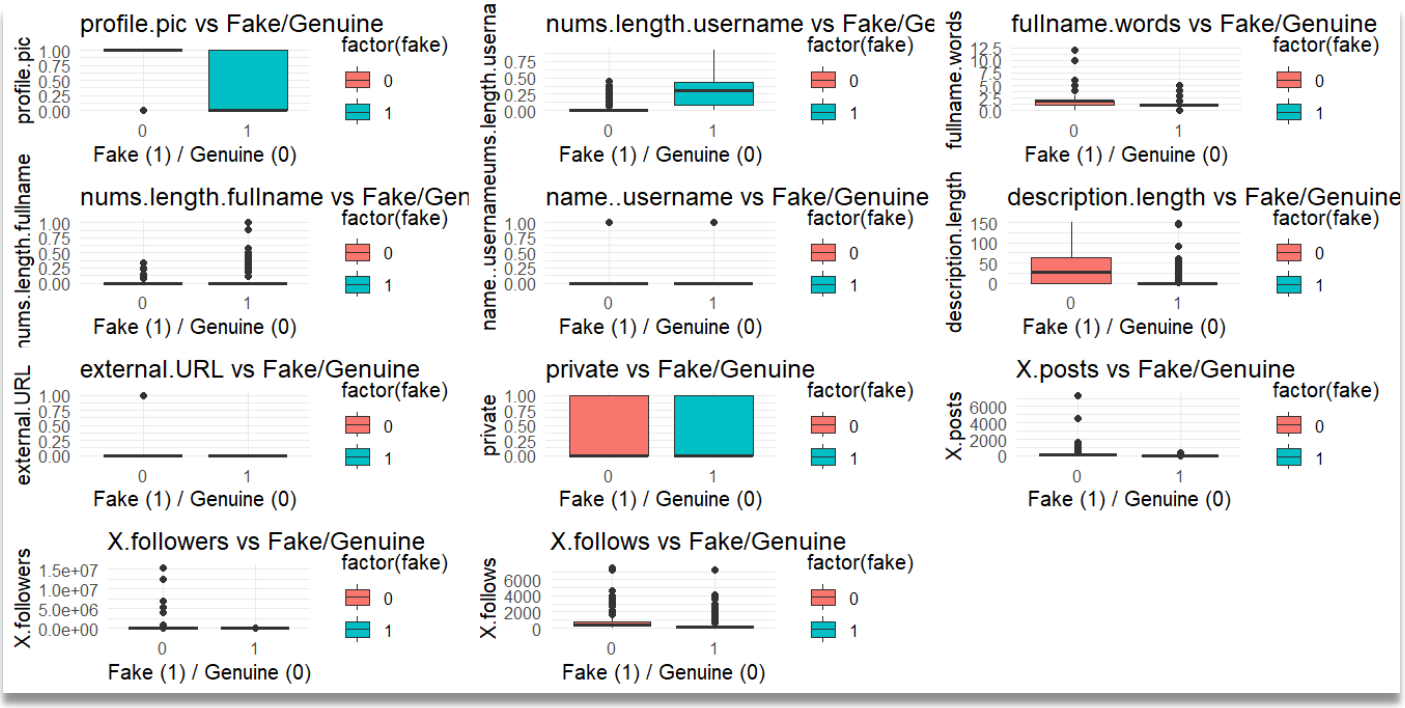
- Outliers:

```r
#Outliers  check (box plot)

box_plots <- lapply(names(num_cols), function(col) {
  ggplot(train, aes_string(y = col)) +
    geom_boxplot(fill = "#FF704D") +
    labs(title = paste("Boxplot of", col)) +
    theme_minimal()
})

do.call(grid.arrange, c(box_plots, ncol = 3))
```



➢ Relationship with target variable:

```
#relationship with target variable

relation_plots <- lapply(names(num_cols), function(col) {
  ggplot(train, aes_string(x = "factor(fake)", y = col, fill = "factor(fake)")) +
    geom_boxplot() +
    labs(title = paste(col, "vs Fake/Genuine"),
         x = "Fake (1) / Genuine (0)") +
    theme_minimal()
})

do.call(grid.arrange, c(relation_plots, ncol = 3))
```



Feature Engineering

➢ Apply feature engineering on train data:

```
#feature engineering

# TRAIN DATA
# Username characteristics
train$username_numeric_ratio <- train$nums.length.username /
  pmax(nchar(as.character(train$name..username)), 1)

# Full name characteristics
train$fullname_numeric_ratio <- train$nums.length.fullname /
  pmax(train$fullname.words, 1)

# Account activity ratios
train$followers_following_ratio <- train$X.followers /
  pmax(train$X.follows, 1)

train$posts_followers_ratio <- train$X.posts /
  pmax(train$X.followers, 1)

# Profile completeness indicators
train$has_profile_pic <- ifelse(train$profile.pic == 1, 1, 0)

train$has_external_url <- ifelse(train$external.URL == 1, 1, 0)

# Privacy indicator
train$is_private <- ifelse(train$private == 1, 1, 0)
```

- Apply feature engineering on test data:

```
# TEST DATA
# Username characteristics
test$username_numeric_ratio <- test$nums.length.username /
  pmax(nchar(as.character(test$name..username)), 1)

# Full name characteristics
test$fullname_numeric_ratio <- test$nums.length.fullname /
  pmax(test$fullname.words, 1)

# Account activity ratios
test$followers_following_ratio <- test$X.followers /
  pmax(test$X.follows, 1)

test$posts_followers_ratio <- test$X.posts /
  pmax(test$X.followers, 1)

# Profile completeness indicators
test$has_profile_pic <- ifelse(test$profile.pic == 1, 1, 0)

test$has_external_url <- ifelse(test$external.URL == 1, 1, 0)

# Privacy indicator
test$is_private <- ifelse(test$private == 1, 1, 0)
```

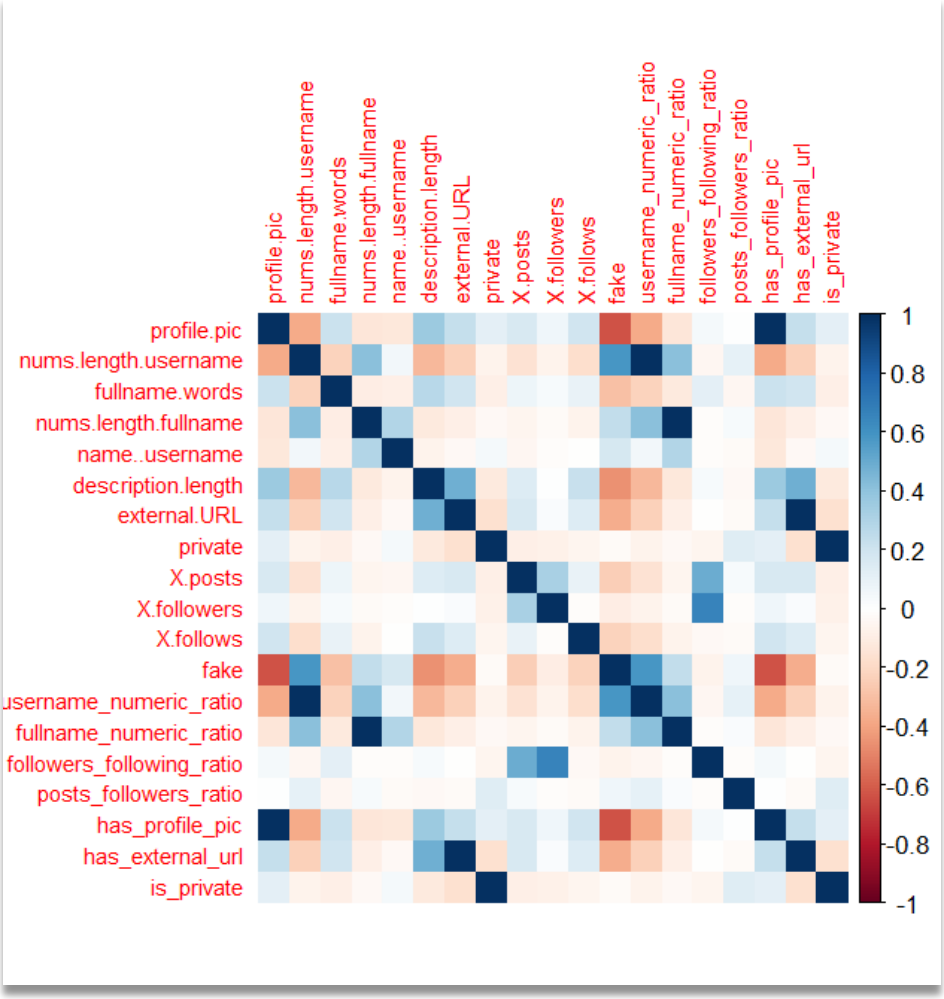
- Save, load and view engineered data:

```
#Save feature engineered data

write.csv(train,"C:/Users/abdul/Downloads/train_fe.csv", row.names = FALSE)
write.csv(test,"C:/Users/abdul/Downloads/test_fe.csv", row.names = FALSE)

#load engineered data
train_fe <- read.csv("C:/Users/abdul/Downloads/train_fe.csv")
test_fe <- read.csv("C:/Users/abdul/Downloads/test_fe.csv")

# Check new structure
glimpse(train_fe)
glimpse(test_fe)
```

Feature Selection

- Consider correlation matrix:

```
#correlation analysis
# Select only numeric columns
num_vars <- train_fe %>% select(where(is.numeric))

# Correlation matrix
corr_matrix <- cor(num_vars, use = "pairwise.complete.obs")

# Visualize the top correlations
corrplot::corrplot(corr_matrix, method = "color", tl.cex = 0.7)
```

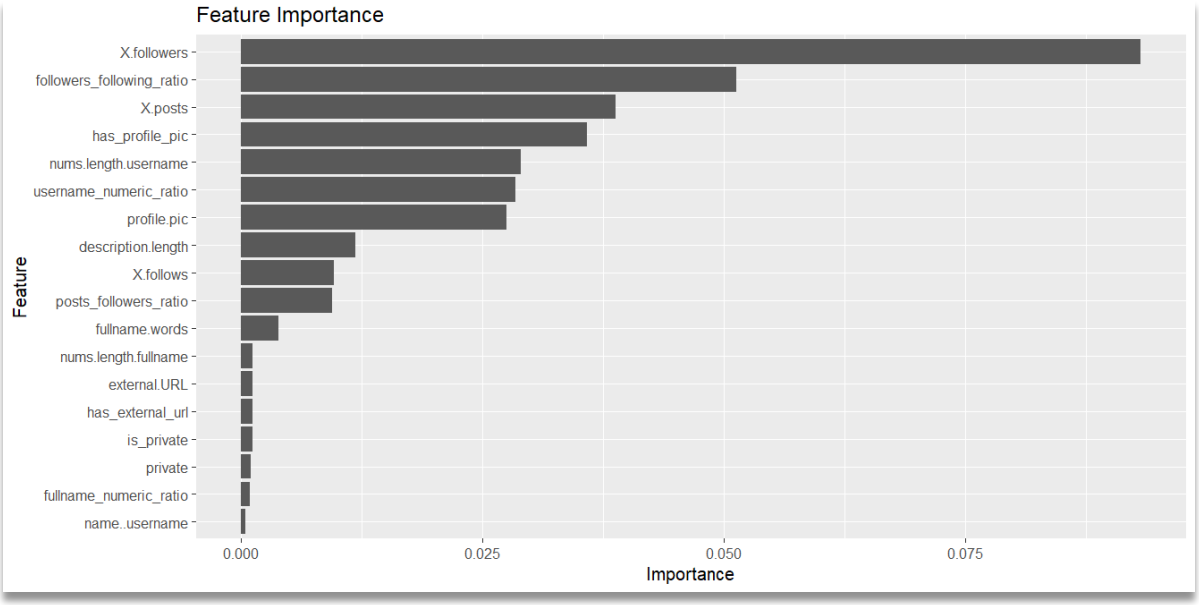


➢ Eliminating recursive features:

```
#eliminate recursive feature

set.seed(123)

control <- rfeControl(
  functions = rfFuncs,
  method = "cv",
  number  = 5
)

# Define predictors & target
X <- train_reduced %>% select(-fake)
y <- train_reduced$fake

# Perform RFE
rfe_results <- rfe(
  x = X,
  y = y,
  sizes = c(5, 10, 15, 20),
  rfeControl = control
)

# Selected features
selected_features1 <- predictors(rfe_results)
selected_features1
```

➢ Evaluate Random Forest Variable Importance:

```r
#random forest variable importance

library(randomForest)
set.seed(123)

rf_model <- randomForest(
  fake ~ .,
  data = train_fe,
  importance = TRUE
)

importance_df <- data.frame(
  Feature = rownames(rf_model$importance),
  Importance = rf_model$importance[, 1]
)

# Plot importance
ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Feature Importance", x = "Feature", y = "Importance")
```



➢ Select only important features:

```r
#select important features
selected_features <- c(
  "profile.pic",
  "nums.length.username",
  "fullname.words",
  "description.length",
  "X.posts",
  "X.followers",
  "X.follows",
  "followers_following_ratio",
  "posts_followers_ratio",
  "username_numeric_ratio",
  "has_profile_pic",
  "fake"   # include target ONLY for training
)

final_selected <- intersect(selected_features, selected_features1)
final_selected <- union(final_selected, "fake")
final_selected
train_final <- train_fe %>% select(all_of(final_selected))

test_final <- test_fe %>% select(-fake) %>%   # test has no labels logically
  select(all_of(setdiff(final_selected, "fake")))

glimpse(train_final)       #view final train data
glimpse(test_final)        #view final test data
```

Model Building

- Train-control setup:

```
#MODEL BUILDING

#train-control setup

set.seed(123)

train_control <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)
```

- Logistic Regression Model:

```
#logisctic regression

train_final$fake <- factor(train_final$fake,
                           levels = c(0, 1),
                           labels = c("Real", "Fake"))
levels(train_final$fake)
# "Real" "Fake"

ctrl <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)
log_model <- train(
  fake ~ .,
  data = train_final,
  method = "glm",
  family = "binomial",
  metric = "ROC",
  trControl = ctrl
)
log_model
```

```
Generalized Linear Model

574 samples
  9 predictor
  2 classes: 'Real', 'Fake'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 458, 459, 460, 460, 459
Resampling results:

  ROC        Sens       Spec
  0.9709021  0.930248   0.9130672
```

- Random Forest Model:

```
#random forest
set.seed(123)
model_rf <- train(
  fake ~ .,
  data = train_final,
  method = "rf",
  trControl = ctrl,
  metric = "ROC"
)
model_rf
```

```
> model_rf
Random Forest

574 samples
  9 predictor
  2 classes: 'Real', 'Fake'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 460, 459, 458, 460, 459
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
  2     0.9839277  0.9302480  0.9235935
  5     0.9832133  0.9232910  0.9270417
  9     0.9813498  0.9128252  0.9304900

ROC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

- XGBoost Model:

```
#XGBoost
set.seed(123)

xgb_model <- train(
  fake ~ .,
  data = train_final,
  method = "xgbTree",
  metric = "ROC",
  trControl = ctrl
)
xgb_model
```

```
> xgb_model
eXtreme Gradient Boosting

574 samples
  9 predictor
  2 classes: 'Real', 'Fake'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 460, 459, 458, 460, 459
Resampling results across tuning parameters:

  eta  max_depth  colsample_bytree  subsample  nrounds  ROC        Sens       Spec
  0.3  1          0.6               0.50        50      0.9796279  0.9407139  0.9166364
  0.3  1          0.6               0.50       100      0.9772705  0.9372656  0.8990926
  0.3  1          0.6               0.50       150      0.9756975  0.9372656  0.9026013
  0.3  1          0.6               0.75        50      0.9799551  0.9301875  0.9200242
```

```
  0.4  3          0.8              0.75       150      0.9744040  0.9199032  0.9235330
  0.4  3          0.8              1.00        50      0.9766904  0.9199032  0.9235935
  0.4  3          0.8              1.00       100      0.9760354  0.9163944  0.9270417
  0.4  3          0.8              1.00       150      0.9757956  0.9163944  0.9271022

Tuning parameter 'gamma' was held constant at a value of 0
Tuning parameter 'min_child_weight' was held
 constant at a value of 1
ROC was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 50, max_depth = 2, eta = 0.3, gamma = 0, colsample_bytree
 = 0.6, min_child_weight = 1 and subsample = 0.75.
>
```

- SVM (Radial Kernel) Model:

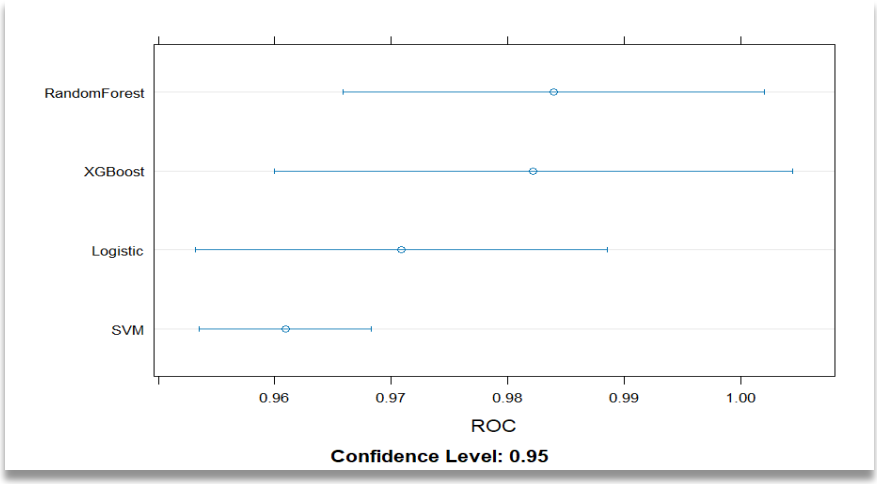
```r
#SVM (Radial Kernel)
set.seed(123)
model_svm <- train(
  fake ~ .,
  data = train_final,
  method = "svmRadial",
  trControl = ctrl,
  metric = "ROC"
)
model_svm
```

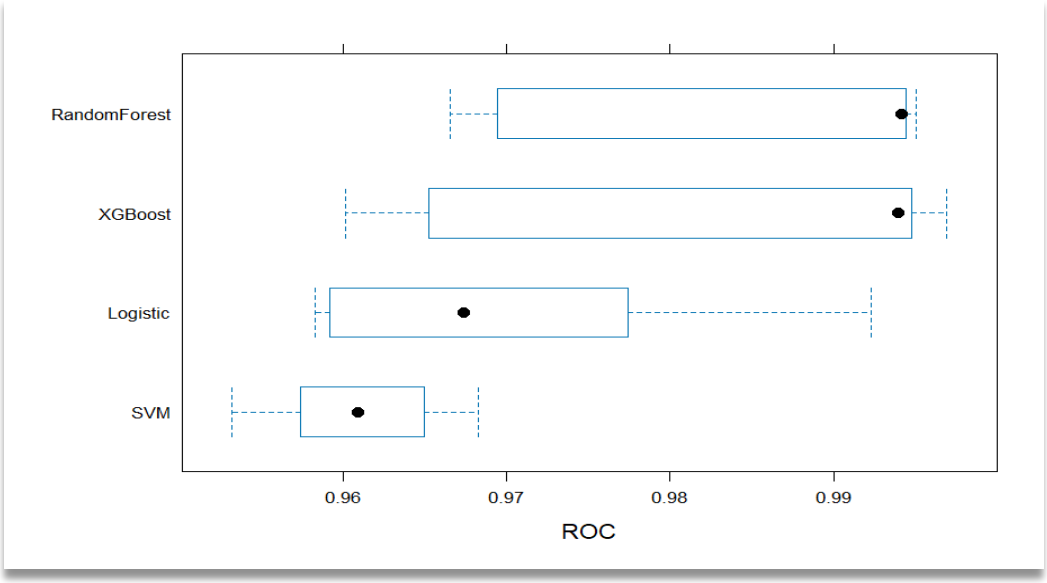
```
> model_svm
Support Vector Machines with Radial Basis Function Kernel

574 samples
  9 predictor
  2 classes: 'Real', 'Fake'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 460, 459, 458, 460, 459
Resampling results across tuning parameters:

  C     ROC        Sens       Spec
  0.25  0.9572258  0.9338173  0.8397459
  0.50  0.9588625  0.9267998  0.8675741
  1.00  0.9609280  0.9267998  0.8815487

Tuning parameter 'sigma' was held constant at a value of 0.2701137
ROC was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.2701137 and C = 1.
>
```

- Model performance comparision:

```r
#Compare model performance
results <- resamples(list(
  Logistic = log_model,
  RandomForest = model_rf,
  XGBoost = xgb_model,
  SVM = model_svm
))

summary(results)
bwplot(results, metric = "ROC")
dotplot(results, metric = "ROC")
```

```
> summary(results)

Call:
summary.resamples(object = results)

Models: Logistic, RandomForest, XGBoost, SVM
Number of resamples: 5

ROC
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Logistic     0.9582577 0.9591652 0.9673746 0.9709021 0.9774078 0.9923053    0
RandomForest 0.9665577 0.9694495 0.9941520 0.9839277 0.9944041 0.9950754    0
XGBoost      0.9601665 0.9652148 0.9939504 0.9822043 0.9947676 0.9969221    0
SVM          0.9532164 0.9573503 0.9609110 0.9609280 0.9649227 0.9682396    0

Sens
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Logistic     0.8771930 0.9298246 0.9310345 0.9302480 0.9482759 0.9649123    0
RandomForest 0.8947368 0.9137931 0.9298246 0.9302480 0.9473684 0.9655172    0
XGBoost      0.8771930 0.8793103 0.9473684 0.9268603 0.9649123 0.9655172    0
SVM          0.8771930 0.9137931 0.9473684 0.9267998 0.9473684 0.9482759    0

Spec
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Logistic     0.8620690 0.8771930 0.9122807 0.9130672 0.9137931 1.0000000    0
RandomForest 0.8620690 0.9122807 0.9137931 0.9235935 0.9473684 0.9824561    0
XGBoost      0.8620690 0.8965517 0.9473684 0.9271627 0.9473684 0.9824561    0
SVM          0.8421053 0.8793103 0.8793103 0.8815487 0.8947368 0.9122807    0
```





Chapter 5

Results

This section presents the outcomes obtained after implementing multiple classification models on the engineered dataset. Model performance was evaluated using cross-validated metrics, confusion matrices, and feature importance analysis to ensure a comprehensive assessment.

Model Performance Evaluation

Four supervised learning models—Logistic Regression, Random Forest, XGBoost, and Support Vector Machine (SVM)—were trained and evaluated using cross-validation. The Receiver Operating Characteristic (ROC), Sensitivity, and Specificity metrics were used as primary evaluation criteria.

The comparative results indicate that **Random Forest achieved the highest overall performance**, followed closely by XGBoost.

Cross-Validated Performance Summary

- **Random Forest**
  - Mean ROC: **0.9839**
  - Sensitivity: **0.9302**
  - Specificity: **0.9236**
- **XGBoost**
  - Mean ROC: 0.9822
  - Sensitivity: 0.9269
  - Specificity: **0.9272**
- **Logistic Regression**
  - Mean ROC: 0.9709
  - Sensitivity: 0.9302
  - Specificity: 0.9131
- **Support Vector Machine (SVM)**
  - Mean ROC: 0.9609
  - Sensitivity: 0.9268
  - Specificity: 0.8815

The ROC confidence interval plots further confirm the superior and stable performance of the Random Forest and XGBoost models compared to Logistic Regression and SVM.

Confusion Matrix Analysis

Confusion matrices were examined to understand the classification behavior of the models in terms of false positives and false negatives.

- **Random Forest** demonstrated a strong balance between detecting fake accounts and correctly identifying genuine ones, minimizing misclassification errors.
- **XGBoost** showed slightly better specificity, indicating a lower tendency to misclassify real accounts as fake.
- **Logistic Regression** maintained good sensitivity but was comparatively weaker in distinguishing real accounts.
- **SVM** exhibited the highest misclassification rate among the evaluated models.

Overall, Random Forest showed the most reliable confusion matrix distribution, supporting its selection as the final model.

Feature Importance Results

Feature importance was analysed using the Random Forest model to identify the most influential predictors contributing to fake account detection.

The most significant features included:

- Number of followers (X.followers)
- Number of posts (X.posts)
- Number of accounts followed (X.follows)
- Presence of profile picture (profile.pic)
- Username and full-name characteristics (nums.length.username, fullname.words)

Less influential features were systematically removed during feature selection to improve model efficiency without compromising performance.

This analysis confirms that user activity metrics and profile completeness play a crucial role in identifying fake accounts.

Project Summary

This project focused on detecting fake social media accounts using machine learning techniques. The workflow included data preprocessing, exploratory data analysis, feature engineering, feature selection, model building, evaluation, and comparative analysis.

After evaluating four classification models, Random Forest emerged as the best-performing model, achieving the highest ROC score with consistent sensitivity and specificity. The study highlights the effectiveness of ensemble learning methods for classification tasks involving complex and non-linear patterns.

Conclusion

The study successfully demonstrates the application of machine learning techniques for fake account detection. Among the evaluated models, **Random Forest delivered the most reliable and accurate performance**, outperforming Logistic Regression, XGBoost, and SVM in overall classification capability.

The results validate the importance of feature engineering and model selection in improving prediction accuracy. The findings also emphasize that ensemble-based models are better suited for handling high-dimensional and behavioural data.

Recommendations

Based on the results of this study, the following recommendations are proposed:

- Random Forest should be preferred for deployment due to its superior performance and robustness.
- Feature importance analysis should be routinely performed to improve interpretability and model efficiency.
- Class probability outputs should be monitored to adjust classification thresholds when minimizing false positives or false negatives is critical.
- Regular retraining of the model is advised to adapt to evolving patterns in fake account behaviour.

Future Scope

This study can be extended in several ways to enhance performance and applicability:

- Incorporating temporal features such as account activity trends over time.
- Applying deep learning techniques for feature extraction from textual or image data.

- Expanding the dataset to include multi-platform social media data.
- Implementing real-time detection systems for live account monitoring.
- Conducting cost-sensitive learning to handle varying misclassification costs.