Mock Hackathon

CST4225 – applied data science life cycle class

M01087865

Abdul Bari Mohammed Abdul Kaleem

1. Introduction

Here, I analyze an Instagram dataset too classify whether an account is genuine or fake. The platform, InstaCheck, uses multiple profile-based indicators such as profile picture presence, username patterns, followers and following count, and description characteristics to detect suspicious activity.

Step-1: Define the Dependent variable:

Here the dependent variable is “fake” (1 = fake, 0 =Geniune)

I converted the dependant variable “fake” to numeric for model copability.

```
# ============================================================================
# STEP 1: DEFINE DEPENDENT VARIABLE
# ============================================================================

data$fake <- as.numeric(data$fake)
```

Step-2: Baseline model:

For the baseline models I split the dataset into train and test datasets

```
set.seed(123)
train_indices <- sample(1:nrow(data), size = 0.7 * nrow(data))
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

cat("Training set:", nrow(train_data), "observations\n")
cat("Test set:    ", nrow(test_data), "observations\n\n")
```

Baseline Logistic regression model:

Here, I created a baseline logistic regression model using 4 independent variables:

1. “profile.pic”
2. “nums.length.username”
3. “X.followers”
4. “X.follows”

```r
baseline_formula <- fake ~ profile.pic + nums.length.username + X.followers + X.follows

# ------------------------------------------------------------------------------
# Baseline Logistic Regression
# ------------------------------------------------------------------------------

cat("========================================\n")
cat("BASELINE LOGISTIC REGRESSION\n")
cat("========================================\n\n")

baseline_glm <- glm(baseline_formula, data = train_data, family = binomial)
print(summary(baseline_glm))
```

```
Call:
glm(formula = baseline_formula, family = binomial, data = train_data)

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           4.3346185  1.2033273   3.602 0.000316 ***
profile.pic          -5.4953755  1.1530739  -4.766 1.88e-06 ***
nums.length.username  9.3417296  1.3733168   6.802 1.03e-11 ***
X.followers          -0.0025249  0.0005239  -4.820 1.44e-06 ***
X.follows             0.0004873  0.0002460   1.981 0.047571 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 558.61  on 402  degrees of freedom
Residual deviance: 191.26  on 398  degrees of freedom
AIC: 201.26

Number of Fisher Scoring iterations: 16
```

Then using test dataset to predict

```r
# Predict on test set
baseline_prob <- predict(baseline_glm, newdata = test_data, type = "response")
baseline_pred <- ifelse(baseline_prob >= 0.5, 1, 0)

# Evaluate
baseline_acc <- mean(baseline_pred == test_data$fake, na.rm = TRUE)
cat("\nTest Set Accuracy:", round(baseline_acc, 4), "\n\n")
cat("Confusion Matrix:\n")
print(table(Predicted = baseline_pred, Actual = test_data$fake))
cat("\n\n")
```

Accuracy: 0.9191

```
Test Set Accuracy: 0.9191
```

Confusion matrix:

```
> cat("Confusion Matrix:\n")
Confusion Matrix:
> print(table(Predicted = baseline_pred, Actual = test_data$fake))
         Actual
Predicted  0  1
        0 82  7
        1  7 77
```

Baseline decision tree:

Here, I created a baseline decision tree model using 4 independent variables same as baseline logistic regression, then used test dataset to check accuracy

Test accuracy: 0.9133

Confusion matrix is provided below

```
cat("=======================================\n")
cat("BASELINE DECISION TREE\n")
cat("=======================================\n\n")

tree_baseline <- rpart(baseline_formula,
                       data = train_data,
                       method = "class",
                       control = rpart.control(cp = 0.01))
print(tree_baseline)

# Predict on test set
tree_baseline_pred <- predict(tree_baseline, newdata = test_data, type = "class")
tree_baseline_pred <- as.numeric(as.character(tree_baseline_pred))

# Evaluate
tree_baseline_acc <- mean(tree_baseline_pred == test_data$fake, na.rm = TRUE)
cat("\nTest Set Accuracy:", round(tree_baseline_acc, 4), "\n\n")
cat("Confusion Matrix:\n")
print(table(Predicted = tree_baseline_pred, Actual = test_data$fake))
```

```
> print(tree_baseline)
n= 403

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 403 199 1 (0.493796526 0.506203474)
   2) X.followers>=161.5 195  27 0 (0.861538462 0.138461538)
     4) nums.length.username< 0.245 175  13 0 (0.925714286 0.074285714)
       8) profile.pic>=0.5 168   8 0 (0.952380952 0.047619048) *
       9) profile.pic< 0.5 7   2 1 (0.285714286 0.714285714) *
     5) nums.length.username>=0.245 20   6 1 (0.300000000 0.700000000) *
   3) X.followers< 161.5 208  31 1 (0.149038462 0.850961538)
     6) nums.length.username< 0.205 88  30 1 (0.340909091 0.659090909)
      12) profile.pic>=0.5 53  23 0 (0.566037736 0.433962264)
        24) X.followers>=29.5 39  12 0 (0.692307692 0.307692308) *
        25) X.followers< 29.5 14   3 1 (0.214285714 0.785714286) *
      13) profile.pic< 0.5 35   0 1 (0.000000000 1.000000000) *
     7) nums.length.username>=0.205 120   1 1 (0.008333333 0.991666667) *
>
> # Predict on test set
> tree_baseline_pred <- predict(tree_baseline, newdata = test_data, type = "class")
> tree_baseline_pred <- as.numeric(as.character(tree_baseline_pred))
>
> # Evaluate
> tree_baseline_acc <- mean(tree_baseline_pred == test_data$fake, na.rm = TRUE)
> cat("\nTest Set Accuracy:", round(tree_baseline_acc, 4), "\n\n")

Test Set Accuracy: 0.9133

> cat("Confusion Matrix:\n")
Confusion Matrix:
> print(table(Predicted = tree_baseline_pred, Actual = test_data$fake))
         Actual
Predicted  0  1
        0 82  8
        1  7 76
```

Step-3: Improved model:

1. Removing outliers and cleaning:

```r
train_data <- train_data[complete.cases(train_data), ]
test_data <- test_data[complete.cases(test_data), ]
```

2. Feature engineering:

Creating a new column “follow_ratio”, I reasoning behind this is that fake account typically follow many accounts but have few followers. This ration captures this suspicious behavior pattern.

```r
# ------------------------------------------------------------------------
# Feature Engineering: follow_ratio
# ------------------------------------------------------------------------

train_data$follow_ratio <- train_data$X.followers / (train_data$X.follows + 1)
test_data$follow_ratio <- test_data$X.followers / (test_data$X.follows + 1)

improved_formula <- fake ~ profile.pic + nums.length.username +
  X.followers + X.follows + follow_ratio
```

3. Adjusting probability thresholds

For logistic regression, the accuracy before adjusting was 0.9191 and after adjusting threshold it was 0.9249

```
> improved_glm <- glm(improved_formula, data = train_data, family = binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> print(summary(improved_glm))

Call:
glm(formula = improved_formula, family = binomial, data = train_data)

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           4.4443451  1.2434868   3.574 0.000351 ***
profile.pic          -5.5713243  1.1767488  -4.735 2.20e-06 ***
nums.length.username  9.3322797  1.3742228   6.791 1.11e-11 ***
X.followers          -0.0024994  0.0005228  -4.781 1.74e-06 ***
X.follows             0.0004703  0.0002490   1.888 0.058961 .
follow_ratio         -0.0369091  0.0669675  -0.551 0.581531
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 558.61  on 402  degrees of freedom
Residual deviance: 191.09  on 397  degrees of freedom
AIC: 203.09

Number of Fisher Scoring iterations: 21

>
> # Predict on test set (threshold = 0.5)
> improved_prob <- predict(improved_glm, newdata = test_data, type = "response")
> improved_pred_05 <- ifelse(improved_prob >= 0.5, 1, 0)
>
> improved_acc_05 <- mean(improved_pred_05 == test_data$fake, na.rm = TRUE)
> cat("\nTest Set Accuracy (threshold = 0.5):", round(improved_acc_05, 4), "\n\n")

Test Set Accuracy (threshold = 0.5): 0.9249

> cat("Confusion Matrix:\n")
Confusion Matrix:
> print(table(Predicted = improved_pred_05, Actual = test_data$fake))
         Actual
Predicted  0  1
        0 82  6
        1  7 78
```

For decision tree, the accuracy before adjusting was 0.9133 and after it was 0.9306

```
>
> best_threshold <- 0.5
> best_accuracy <- improved_acc_05
>
> for (threshold in seq(0.01, 0.99, by = 0.01)) {
+   pred_temp <- ifelse(improved_prob >= threshold, 1, 0)
+   acc_temp <- mean(pred_temp == test_data$fake, na.rm = TRUE)
+
+   if (acc_temp > best_accuracy) {
+     best_accuracy <- acc_temp
+     best_threshold <- threshold
+   }
+ }
>
> cat("Optimal threshold:", best_threshold, "\n")
Optimal threshold: 0.49
> cat("Best accuracy:    ", round(best_accuracy, 4), "\n\n")
Best accuracy:     0.9306

>
> # Predict with optimal threshold
> improved_pred_best <- ifelse(improved_prob >= best_threshold, 1, 0)
>
> cat("Confusion Matrix (optimal threshold):\n")
Confusion Matrix (optimal threshold):
> print(table(Predicted = improved_pred_best, Actual = test_data$fake))
         Actual
Predicted  0  1
        0 82  5
        1  7 79
```

Final Step: Compare Baseline vs Improved Model

```
FINAL MODEL EVALUATION
> cat("========================================\n\n")
========================================

>
> confusion_matrix <- table(Predicted = improved_pred_best, Actual = test_data$fake)
> TP <- confusion_matrix[2, 2]
> TN <- confusion_matrix[1, 1]
> FP <- confusion_matrix[2, 1]
> FN <- confusion_matrix[1, 2]
>
> accuracy <- (TP + TN) / sum(confusion_matrix)
> sensitivity <- TP / (TP + FN)
> specificity <- TN / (TN + FP)
> precision <- TP / (TP + FP)
>
> cat("Accuracy:    ", round(accuracy, 4), "\n")
Accuracy:     0.9306
> cat("Sensitivity: ", round(sensitivity, 4), "\n")
Sensitivity:  0.9405
> cat("Specificity: ", round(specificity, 4), "\n")
Specificity:  0.9213
> cat("Precision:   ", round(precision, 4), "\n\n")
Precision:    0.9186
```

Final Step: Compare Baseline vs Improved Model

```
>
> # ================================================================================
> # COMPARISON: BASELINE VS IMPROVED
> # ================================================================================
>
> cat("========================================\n")
========================================
> cat("MODEL COMPARISON\n")
MODEL COMPARISON
> cat("========================================\n\n")
========================================

>
> cat("Baseline Logistic Regression:  ", round(baseline_acc, 4), "\n")
Baseline Logistic Regression:   0.9191
> cat("Baseline Decision Tree:        ", round(tree_baseline_acc, 4), "\n")
Baseline Decision Tree:          0.9133
> cat("Improved Model (threshold 0.5): ", round(improved_acc_05, 4), "\n")
Improved Model (threshold 0.5):  0.9249
> cat("Improved Model (optimal):      ", round(best_accuracy, 4), "\n\n")
Improved Model (optimal):        0.9306

>
> cat("Accuracy improvement: +", round((best_accuracy - baseline_acc) * 100, 2), "%\n\n")
Accuracy improvement: + 1.16 %

>
> cat("=== Analysis Complete ===\n")
=== Analysis Complete ===
```