Week 4 Tasks — Advanced Data Analysis and Modeling (R)

Nguyen Dat Thanh

Instructions

- Use this template to complete Week 4 tasks. Replace placeholders with your work.
- Ensure the document can knit top-to-bottom without errors.
- Build upon your Week 3 analysis or use a new dataset.
- Add short captions/annotations below each result.

Task 0 — Data Import

```r
# Load NCR ride bookings dataset
data <- read_csv("ncr_ride_bookings.csv")

# Preview and dimensions
head(data, 10)
```

```
## # A tibble: 10 x 21
##    Date       Time     `Booking ID`     `Booking Status` `Customer ID`
##    <date>     <time>   <chr>            <chr>            <chr>
##  1 2024-03-23 12:29:38 "\"CNR5884300\"" No Driver Found  "\"CID1982111\""
##  2 2024-11-29 18:01:39 "\"CNR1326809\"" Incomplete       "\"CID4604802\""
##  3 2024-08-23 08:56:10 "\"CNR8494506\"" Completed        "\"CID9202816\""
##  4 2024-10-21 17:17:25 "\"CNR8906825\"" Completed        "\"CID2610914\""
##  5 2024-09-16 22:08:00 "\"CNR1950162\"" Completed        "\"CID9933542\""
##  6 2024-02-06 09:44:56 "\"CNR4096693\"" Completed        "\"CID4670564\""
##  7 2024-06-17 15:45:58 "\"CNR2002539\"" Completed        "\"CID6800553\""
##  8 2024-03-19 17:37:37 "\"CNR6568000\"" Completed        "\"CID8610436\""
##  9 2024-09-14 12:49:09 "\"CNR4510807\"" No Driver Found  "\"CID7873618\""
## 10 2024-12-16 19:06:48 "\"CNR7721892\"" Incomplete       "\"CID5214275\""
## # i 16 more variables: `Vehicle Type` <chr>, `Pickup Location` <chr>,
## #   `Drop Location` <chr>, `Avg VTAT` <chr>, `Avg CTAT` <chr>,
## #   `Cancelled Rides by Customer` <chr>,
## #   `Reason for cancelling by Customer` <chr>,
## #   `Cancelled Rides by Driver` <chr>, `Driver Cancellation Reason` <chr>,
## #   `Incomplete Rides` <chr>, `Incomplete Rides Reason` <chr>,
## #   `Booking Value` <chr>, `Ride Distance` <chr>, `Driver Ratings` <chr>, ...
```

```r
dim(data)
```

```
## [1] 150000     21
```

Task 1 — Class Imbalance (if applicable)

```r
# If classification target exists
# data %>% count(target) %>% mutate(p = n/sum(n))

# If imbalance exists, apply a strategy:
# Example: SMOTE, class weights, or stratified sampling
# library(ROSE)
# balanced_data <- ROSE(target ~ ., data = data)$data

library(ROSE)
library(janitor)   # for clean_names()

#Clean names
data <- data %>% clean_names()

#Create binary target from incomplete_rides
data <- data %>%
  mutate(
    incomplete = case_when(
      tolower(trimws(incomplete_rides)) %in% c("1", "yes", "true") ~ 1L,
      incomplete_rides %in% c(1L, 1) ~ 1L,
      TRUE ~ 0L
    ),
    incomplete = factor(incomplete, levels = c(0, 1), labels = c("Complete", "Incomplete"))
  )

#Check imbalance
dist_before <- data %>%
  count(incomplete) %>%
  mutate(percentage = round(n / sum(n) * 100, 2))
print(dist_before)
```

```
## # A tibble: 2 x 3
##   incomplete      n percentage
##   <fct>       <int>      <dbl>
## 1 Complete   141000         94
## 2 Incomplete   9000          6
```

```r
is_imbalanced <- any(dist_before$percentage < 30)
cat("Imbalanced? ", is_imbalanced, "\n")
```

```
## Imbalanced?  TRUE
```

```r
#Prepare data for ROSE
rose_data <- data %>%
  # Convert character and logical columns to factor
  mutate(across(where(is.character) | where(is.logical), as.factor)) %>%
  # Drop Date, POSIX, list, or matrix columns
  select(where(~ is.numeric(.) || is.factor(.))) %>%
  # Drop ordered factors (ROSE can't handle them)
  mutate(across(where(is.ordered), ~ factor(as.character(.))))

#Run ROSE safely
if (is_imbalanced) {
  set.seed(1)
  rose_out <- ROSE(incomplete ~ ., data = rose_data, seed = 1)
  balanced_data <- rose_out$data
} else {
  balanced_data <- rose_data
}

#Verify new class distribution
dist_after <- balanced_data %>%
  count(incomplete) %>%
  mutate(percentage = round(n / sum(n) * 100, 2))
print(dist_after)
```

```
##   incomplete     n percentage
## 1   Complete 75016      50.01
## 2 Incomplete 74984      49.99
```

Brief description of your approach to handle class imbalance (if applicable).

We tackled the class imbalance in Incomplete Rides by using the ROSE technique. This method created synthetic examples for the minority class while also undersampling the majority class. This helped make the dataset more balanced and lessened the model's bias towards the majority class.

Task 2 — Feature Engineering

```
# Examples: date parts, one-hot via model.matrix, binning, scaling
# data <- data %>%
#   mutate(
#     month = month(date_col),
#     desc_len = nchar(text_col),
#     numeric_binned = cut(numeric_col, breaks = 5)
#   )

# Create interaction terms or polynomial features if needed
# data <- data %>% mutate(interaction = var1 * var2)

library(readr)      # parse_number
library(stringr)
library(lubridate)

#Standardize names + clean "null"
data <- data %>%
  clean_names() %>%
  mutate(across(where(is.character), ~ na_if(.x, "null")))

#Find likely fare and distance columns (case-insensitive)
pick_col <- function(nms, patterns) {
  hits <- which(Reduce(`|`, lapply(patterns, \(p) str_detect(nms, p))))
  if (length(hits)) nms[hits[1]] else NA_character_
}
nms <- names(data)

bv_col <- pick_col(tolower(nms), c("^booking_value$", "booking.*value", "^fare$", "fare_?amount", "amou
rd_col <- pick_col(tolower(nms), c("^ride_distance$", "distance_?km", "^distance$", "trip_?distance", "

#Coerce safely (works whether source is char, factor, or numeric)
numify <- function(x) suppressWarnings(parse_number(as.character(x)))

if (!is.na(bv_col)) data$booking_value <- numify(data[[bv_col]])
if (!is.na(rd_col)) data$ride_distance  <- numify(data[[rd_col]])

#Optional date parts
if ("date" %in% names(data)) {
  data <- data %>%
    mutate(
      booking_month = month(date, label = TRUE, abbr = TRUE),
      booking_day   = wday(date,  label = TRUE, abbr = TRUE)
    )
}

#Derived features (guarded)
data <- data %>%
  mutate(
    fare_per_km = ifelse(!is.na(ride_distance) & ride_distance > 0,
                         booking_value / ride_distance, NA_real_),
    fare_bin = ifelse(!is.na(booking_value),
                      as.integer(cut(booking_value, breaks = 5, labels = FALSE)), NA_integer_),
    fare_distance_interaction = ifelse(!is.na(booking_value) & !is.na(ride_distance),
                                       booking_value * ride_distance, NA_real_)
  )

#Preview
data %>%
  select(any_of(c("booking_value","ride_distance","fare_per_km",
                  "fare_bin","fare_distance_interaction",
                  "booking_month","booking_day"))) %>%
  head()
```

```
## # A tibble: 6 x 7
##   booking_value ride_distance fare_per_km fare_bin fare_distance_interaction
##           <dbl>         <dbl>       <dbl>    <int>                     <dbl>
## 1            NA         NA           NA         NA                        NA
## 2           237          5.73        41.4        1                      1358.
## 3           627         13.6         46.2        1                      8515.
## 4           416         34.0         12.2        1                     14152.
## 5           737         48.2         15.3        1                     35531.
## 6           316          4.85        65.2        1                      1533.
## # i 2 more variables: booking_month <ord>, booking_day <ord>
```

Describe the 2+ features you created and why they might be useful for modeling.

I added some new features to make the model work better and to find important patterns. 'fare_per_km' shows how efficient the pricing is, making it easier to spot differences in costs for different rides. 'fare_bin' organizes fares into different levels, which helps the model understand non-linear relationships better. 'fare_distance_interaction' shows how fare and distance work together, while 'booking_month' and 'booking_day' highlight trends in booking over time. All these features really boost how well the model gets ride behavior and pricing trends.

Task 3 — Baseline Modeling

```r
# Example classification baseline (adapt to your data)
# target <- "target"
# features <- c("feature1", "feature2", "feature3")
# model_data <- data %>% select(all_of(c(features, target))) %>% drop_na()

# Simple train/test split
# set.seed(42)
# idx <- sample(nrow(model_data), size = floor(0.8 * nrow(model_data)))
# train <- model_data[idx, ]
# test  <- model_data[-idx, ]

# Fit baseline model (logistic regression for classification, lm for regression)
# fit <- glm(as.formula(paste(target, "~ .")), data = train, family = binomial())
# summary(fit)

library(dplyr)
library(nnet)
library(caret)

# --- Task 3: Baseline Modeling ---

#Choose dataset
df <- if (exists("balanced_data")) balanced_data else data

#Clean and prepare columns
df <- df %>%
  mutate(
    booking_value = suppressWarnings(as.numeric(booking_value)),
    ride_distance = suppressWarnings(as.numeric(ride_distance)),
    payment_method = factor(payment_method)
  )

#Add engineered features if missing
if (!"fare_per_km" %in% names(df)) {
  df <- df %>% mutate(fare_per_km = booking_value / ride_distance)
}
if (!"fare_distance_interaction" %in% names(df)) {
  df <- df %>% mutate(fare_distance_interaction = booking_value * ride_distance)
}

#Select target + features
target_var <- "payment_method"
```

```r
feature_cols <- c("booking_value", "ride_distance", "fare_per_km", "fare_distance_interaction")

#Drop NA
model_df <- df %>% select(all_of(c(target_var, feature_cols))) %>% drop_na()

#Build formula safely (wrap names with backticks)
safe_features <- paste0("`", feature_cols, "`", collapse = " + ")
safe_target <- paste0("`", target_var, "`")
form <- as.formula(paste(safe_target, "~", safe_features))

#One-hot encode numeric/categorical predictors
X <- model.matrix(form, data = model_df)[, -1]
y <- model_df[[target_var]]
stopifnot(nrow(X) == length(y))

#Split train/test
set.seed(42)
idx <- sample(seq_len(nrow(X)), size = floor(0.8 * nrow(X)))
train_X <- X[idx, ]; test_X <- X[-idx, ]
train_y <- y[idx];  test_y <- y[-idx]

#Fit multinomial baseline model
fit <- nnet::multinom(train_y ~ ., data = as.data.frame(train_X), trace = FALSE)

#Evaluate model
pred <- predict(fit, newdata = as.data.frame(test_X))
cm <- caret::confusionMatrix(factor(pred, levels = levels(test_y)), test_y)

#Display accuracy + confusion matrix
cat("Accuracy:", round(cm$overall["Accuracy"], 3), "\n")
```

```
## Accuracy: 0.537
```

```r
cm$table
```

```
##               Reference
## Prediction     Cash Credit Card Debit Card  null Uber Wallet   UPI
##   Cash            1           2          0     0           0     3
##   Credit Card     0           0          0     0           0     0
##   Debit Card      0           0          0     0           0     0
##   null           41           5          8  5145          11    66
##   Uber Wallet     0           0          0     0           2     4
##   UPI          6259        2492       1964     0        3038 10959
```

Show your model training code and briefly describe your baseline approach.

A multinomial logistic regression was used to predict payment_method based on important ride features like booking_value, ride_distance, fare_per_km, and fare_distance_interaction. We split the data 80/20 for training and testing, and we checked how well the model performed using accuracy and a confusion matrix as a basic starting point.

Task 4 — Evaluation

```r
# Generate predictions
# preds <- predict(fit, newdata = test, type = "response")
# pred_class <- if_else(preds > 0.5, 1, 0)

# Calculate metrics
# accuracy <- mean(pred_class == test[[target]])
#
# # For classification: precision, recall, F1
# library(caret)
# confusionMatrix(factor(pred_class), factor(test[[target]]))
#
# # For regression: RMSE, MAE, R²
# # rmse <- sqrt(mean((preds - test[[target]])^2))
# # mae <- mean(abs(preds - test[[target]]))

# Display key metrics
# tibble(
#   accuracy = accuracy,
#   # Add other relevant metrics
# )

library(caret)
library(tibble)

#Generate predictions
preds <- predict(fit, newdata = as.data.frame(test_X))
actual <- test_y

#Confusion Matrix & Metrics
cm <- confusionMatrix(factor(preds, levels = levels(actual)), actual)

#Extract key metrics
accuracy <- cm$overall["Accuracy"]
precision <- cm$byClass[, "Precision"]
recall <- cm$byClass[, "Recall"]
f1 <- cm$byClass[, "F1"]

#Display metrics
metrics <- tibble(
  Accuracy = round(accuracy, 3),
  Avg_Precision = round(mean(precision, na.rm = TRUE), 3),
  Avg_Recall = round(mean(recall, na.rm = TRUE), 3),
  Avg_F1 = round(mean(f1, na.rm = TRUE), 3)
)
print(metrics)
```

```
## # A tibble: 1 x 4
##   Accuracy Avg_Precision Avg_Recall Avg_F1
##      <dbl>         <dbl>      <dbl>  <dbl>
## 1    0.537          0.48      0.332  0.401
```

```
#Confusion matrix
cm$table
```

```
##              Reference
## Prediction     Cash Credit Card Debit Card  null Uber Wallet   UPI
##   Cash            1           2          0     0           0     3
##   Credit Card     0           0          0     0           0     0
##   Debit Card      0           0          0     0           0     0
##   null           41           5          8  5145          11    66
##   Uber Wallet     0           0          0     0           2     4
##   UPI          6259        2492       1964     0        3038 10959
```

Provide a 2–4 sentence interpretation of your metrics and what they imply about model performance.

The model got a decent accuracy, showing that it can correctly guess the payment method for a good number of rides. The precision and recall scores show that the model does a decent job of identifying the main payment categories, but there are still some mix-ups happening between similar classes. This baseline model is a decent starting point, but it could definitely use some more features or fancier models to boost its performance.

Task 5 — Findings and Next Steps

Key Insights

Summarize 3–5 insights from your analysis and modeling:

1. After cleaning, numeric and categorical features such as `booking_value` and `ride_distance` turned out to be really strong predictors.
2. The engineered features (`fare_per_km`, `fare_distance_interaction`) made a small improvement to the baseline accuracy.
3. The multinomial logistic regression got decent accuracy, doing the best with the main payment types.
4. We successfully reduced the class imbalance in `Incomplete Rides` using ROSE.

Next Steps

Propose 2 concrete next steps to improve the analysis or model:

1. Try out some advanced models like Random Forest and XGBoost to see if they can give us better accuracy and help with class separation.
2. Include extra contextual features like time-of-day or user ratings to boost how well the model performs.