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

Cao Pham Minh Dang

Pre-task: Downloading data

Data link: https://www.kaggle.com/datasets/prathamtripathi/drug-classification

```
if (!file.exists("./drug200.csv")) stop("Missing file: ./drug200.csv")
data <- read.csv("./drug200.csv", na = c("", "NA", "null"), stringsAsFactors = FALSE)

tibble(col = names(data), n_missing = colSums(is.na(data))) %>%
  mutate(p_missing = n_missing / nrow(data)) %>%
  arrange(desc(p_missing))
```

```
## # A tibble: 6 x 3
##   col         n_missing p_missing
##   <chr>           <dbl>     <dbl>
## 1 Age                 0         0
## 2 Sex                 0         0
## 3 BP                  0         0
## 4 Cholesterol         0         0
## 5 Na_to_K             0         0
## 6 Drug                0         0
```

```
summary(data)
```

```
##       Age            Sex                 BP            Cholesterol
##  Min.   :15.00   Length:200         Length:200         Length:200
##  1st Qu.:31.00   Class :character   Class :character   Class :character
##  Median :45.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :44.31
##  3rd Qu.:58.00
##  Max.   :74.00
##     Na_to_K           Drug
##  Min.   : 6.269   Length:200
##  1st Qu.:10.445   Class :character
##  Median :13.937   Mode  :character
##  Mean   :16.084
##  3rd Qu.:19.380
##  Max.   :38.247
```

```
colnames(data)
```

```
## [1] "Age"         "Sex"         "BP"          "Cholesterol" "Na_to_K"
## [6] "Drug"
```

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 1 — Class Imbalance (if applicable)

```
# quick class distribution
data %>% count(Drug) %>% mutate(p = n / sum(n))
```

```
##    Drug  n     p
## 1 DrugY 91 0.455
## 2 drugA 23 0.115
## 3 drugB 16 0.080
## 4 drugC 16 0.080
## 5 drugX 54 0.270
```

```
# Split before resampling to avoid leakage
set.seed(42)
train_index <- createDataPartition(data$Drug, p = 0.8, list = FALSE)
train_raw <- data[train_index, , drop = FALSE]
test_raw  <- data[-train_index, , drop = FALSE]

# Check training imbalance
train_raw %>% count(Drug) %>% mutate(p = n / sum(n))
```

```
##    Drug  n          p
## 1 DrugY 73 0.45061728
## 2 drugA 19 0.11728395
## 3 drugB 13 0.08024691
## 4 drugC 13 0.08024691
## 5 drugX 44 0.27160494
```

```
# If imbalance exists, apply oversampling only on training set
max_count <- max(table(train_raw$Drug))

train_balanced <- train_raw %>%
  group_by(Drug) %>%
  slice_sample(n = max_count, replace = TRUE) %>%
  ungroup()

train_balanced %>% count(Drug) %>% mutate(p = n / sum(n))
```

```
## # A tibble: 5 x 3
##   Drug      n     p
##   <chr> <int> <dbl>
## 1 DrugY    73   0.2
```

```
## 2 drugA    73   0.2
## 3 drugB    73   0.2
## 4 drugC    73   0.2
## 5 drugX    73   0.2
```

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

- I first split the dataset into training and testing subsets to prevent data leakage.
- I applied random oversampling on the training set only, where each minority class was resampled (with replacement) until all classes reached the size of the majority class.

=> This ensured a balanced training distribution, allowing the model to learn all drug categories equally rather than being biased toward the dominant class. The test set remained untouched to preserve an unbiased evaluation of model performance.

Task 2 — Feature Engineering

```r
# Keep target aside
train_y <- train_balanced$Drug
test_y  <- test_raw$Drug

# Feature engineering on predictors (training)
train_x <- train_balanced %>%
  select(-Drug) %>%
  mutate(
    BP_Chol = paste(BP, Cholesterol, sep = "_"),
    NaK_BP_Interaction = Na_to_K *
      case_when(BP == "HIGH" ~ 3, BP == "NORMAL" ~ 2, TRUE ~ 1)
  ) %>%
  mutate(across(c(Sex, BP, Cholesterol, BP_Chol), as.factor))

# Apply same transforms to test (important: use same factor levels where possible)
test_x <- test_raw %>%
  select(-Drug) %>%
  mutate(
    BP_Chol = paste(BP, Cholesterol, sep = "_"),
    NaK_BP_Interaction = Na_to_K *
      case_when(BP == "HIGH" ~ 3, BP == "NORMAL" ~ 2, TRUE ~ 1)
  ) %>%
  mutate(across(c(Sex, BP, Cholesterol, BP_Chol), as.factor))

# Align factor levels between train and test for categorical columns
for (f in intersect(names(train_x)[sapply(train_x, is.factor)], names(test_x))) {
  levels(test_x[[f]]) <- union(levels(train_x[[f]]), levels(test_x[[f]]))
  levels(train_x[[f]]) <- union(levels(train_x[[f]]), levels(test_x[[f]]))
}

# One-hot encode (model.matrix) - produce same columns using formula with train then subset for test
mm_train <- model.matrix(~ . - 1, data = train_x) %>% as.data.frame()
mm_test  <- model.matrix(~ . - 1, data = test_x) %>% as.data.frame()
```

```r
# Ensure test has same columns as train
missing_cols <- setdiff(names(mm_train), names(mm_test))
if (length(missing_cols) > 0) {
  mm_test[missing_cols] <- 0
}
mm_test <- mm_test[names(mm_train)]

# Add target back
mm_train$Drug <- train_y
mm_test$Drug  <- test_y

# Remove near-zero variance predictors to help multinom
nzv <- nearZeroVar(mm_train, saveMetrics = TRUE)
keep_vars <- rownames(nzv)[!nzv$nzv]
mm_train <- mm_train[, keep_vars]
mm_test  <- mm_test[, intersect(names(mm_test), keep_vars)]
```

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

- `BP_Chol`: a combined categorical feature representing each patient's Blood Pressure (BP) and Cholesterol level together (e.g. HIGH_HIGH, LOW_NORMAL). This is because drug prescriptions often depend on the joint condition of BP and cholesterol rather than each factor independently.
- `NaK_BP_Interaction`: a numeric interaction feature that multiplies the sodium-to-potassium ratio (Na_to_K) by a numeric encoding of BP severity (e.g. 3 for HIGH, 2 for NORMAL, 1 for LOW). This will reflect how the impact of sodium–potassium balance may vary by blood pressure level, therefore increase the chance of choosing the correct drug type.

Task 3 — Baseline Modeling

```r
# Preprocess: center/scale numeric predictors (exclude Drug)
predictor_names <- setdiff(names(mm_train), "Drug")

preproc <- preProcess(mm_train[, predictor_names], method = c("center", "scale"))
train_pp <- predict(preproc, mm_train[, predictor_names])
test_pp  <- predict(preproc, mm_test[, predictor_names])

# Reattach target as factor with identical levels
train_pp$Drug <- factor(mm_train$Drug)
test_pp$Drug  <- factor(mm_test$Drug, levels = levels(train_pp$Drug))

# Fit multinomial logistic regression (baseline)
model_baseline <- multinom(Drug ~ ., data = train_pp, trace = FALSE, MaxNWts = 10000)
```

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

For the baseline model, I used a multinomial logistic regression trained on centered and scaled numeric predictors. This model was chosen because it's simple, interpretable, and provides a solid performance benchmark for multi-class classification tasks. Before fitting, I applied preprocessing to standardize all numeric features (ensuring equal contribution to the model) and ensured consistent factor levels across training and testing sets.

Task 4 — Evaluation

```r
pred_raw <- predict(model_baseline, newdata = test_pp, type = "class")
pred_factor <- factor(pred_raw, levels = levels(train_pp$Drug))
ref_factor  <- factor(test_pp$Drug, levels = levels(train_pp$Drug))

conf_mat <- confusionMatrix(data = pred_factor, reference = ref_factor)
conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction drugA drugB drugC drugX DrugY
##      drugA     4     0     0     0     1
##      drugB     0     2     0     0     0
##      drugC     0     1     3     0     0
##      drugX     0     0     0     8     0
##      DrugY     0     0     0     2    17
##
## Overall Statistics
##
##                Accuracy : 0.8947
##                  95% CI : (0.752, 0.9706)
##     No Information Rate : 0.4737
##     P-Value [Acc > NIR] : 5.827e-08
##
##                   Kappa : 0.8455
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: drugA Class: drugB Class: drugC Class: drugX
## Sensitivity                1.0000      0.66667      1.00000       0.8000
## Specificity                0.9706      1.00000      0.97143       1.0000
## Pos Pred Value             0.8000      1.00000      0.75000       1.0000
## Neg Pred Value             1.0000      0.97222      1.00000       0.9333
## Prevalence                 0.1053      0.07895      0.07895       0.2632
## Detection Rate             0.1053      0.05263      0.07895       0.2105
## Detection Prevalence       0.1316      0.05263      0.10526       0.2105
## Balanced Accuracy          0.9853      0.83333      0.98571       0.9000
##                      Class: DrugY
## Sensitivity                0.9444
## Specificity                0.9000
## Pos Pred Value             0.8947
## Neg Pred Value             0.9474
## Prevalence                 0.4737
## Detection Rate             0.4474
## Detection Prevalence       0.5000
## Balanced Accuracy          0.9222
```

```r
tibble(
  Accuracy = unname(conf_mat$overall["Accuracy"]),
  Kappa = unname(conf_mat$overall["Kappa"])
)
```

```
## # A tibble: 1 x 2
##   Accuracy Kappa
##      <dbl> <dbl>
## 1    0.895 0.846
```

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

- Accuracy measures the overall proportion of correct predictions but can be misleading if the data are imbalanced.
- The Kappa statistic adjusts for the accuracy that might occur by random chance, providing a more reliable measure of agreement between predictions and true labels.
- The baseline multinomial logistic regression achieved an accuracy of about 0.895 and a Kappa of 0.846, indicating strong agreement between predicted and true drug classes beyond random chance. Most classes show high sensitivity and specificity (>0.9), meaning the model correctly identifies most drugs with few false positives or negatives.

Task 5 — Findings and Next Steps

Key Insights

Summarize 3–5 insights from your analysis and modeling:

1. Initial data had clear class imbalance in `Drug` — oversampling the training set achieved a balanced training distribution.
2. Combined feature `BP_Chol` and `NaK_BP_Interaction` improved separability for some drug classes (check variable importance).
3. Multinomial logistic regression works as a baseline but may need regularization or fewer predictors when many dummies exist.
4. Near-zero variance predictors and perfectly collinear dummy columns were removed to stabilize training.

Next Steps

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

1. Try a regularized multinomial model (e.g., `glmnet` with `family = "multinomial"`) to handle high-dimensional dummies and prevent overfitting.
2. Evaluate using stratified cross-validation on the *training* set (after oversampling) and hold out the test set strictly for final evaluation. Also try class-weighted algorithms (xgboost with class weights) as an alternative to oversampling.