Bacay, Gianne P. **BSIT-BTM 3B**
Business Analytics **April 28, 2025**

Machine Problem 3

Employee Attrition Prediction Model Summary

Model & Predictors Recap

A **Random Forest classifier** was trained using the ranger engine with 500 trees, an mtry of 5 candidate predictors at each split and a minimum node size of 5. Prior to fitting the model, all categorical predictors were transformed by one-hot encoding and all numeric predictors were centered and scaled so that variables on different scales contributed equally. **Sixteen features** were used to predict attrition: Age, DistanceFromHome, Department, JobLevel, JobRole, JobSatisfaction, EnvironmentSatisfaction, RelationshipSatisfaction, WorkLifeBalance, YearsSinceLastPromotion, PercentSalaryHike, MonthlyRate, MonthlyIncome, DailyRate, HourlyRate and OverTime. The target variable Attrition was encoded as a factor with levels “No” (stayed) and “Yes” (left).

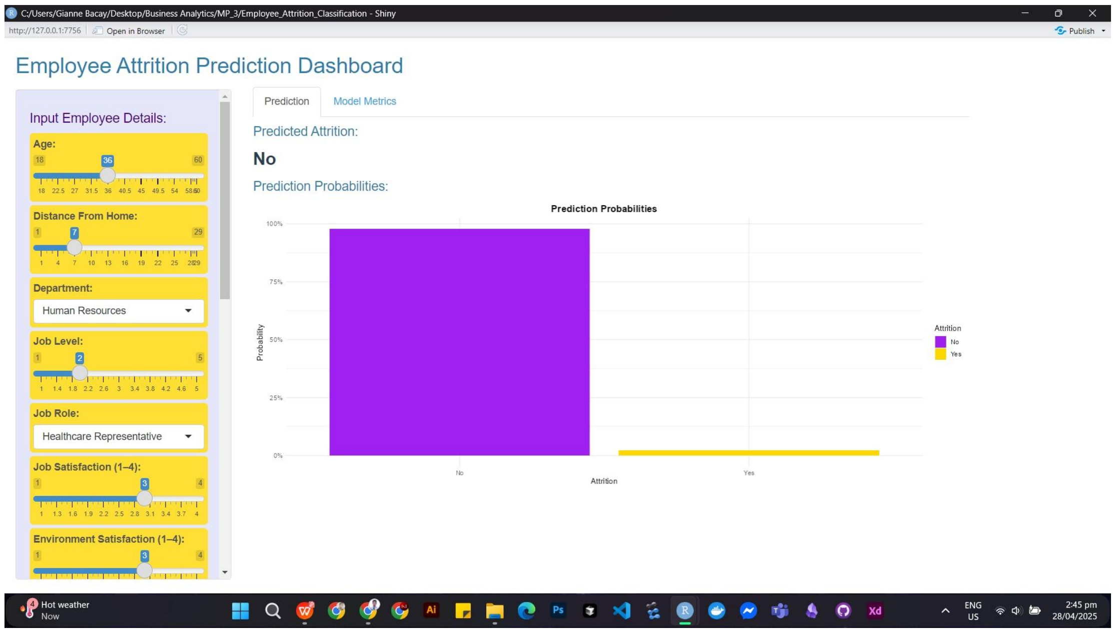
Dashboard Results

When evaluated on a held-out test set of **442 employees**, the model achieved **overall accuracy of 0.8597 or approximately 86%**. The confusion matrix showed **367 true negatives** and **59 false positives** among employees predicted to stay, and **3 false negatives** and **13 true positives** among those predicted to leave. These results correspond to a s**pecificity (true-negative rate) of approximately 86.1 percent** and a **sensitivity (true-positive rate) of approximately 81.3 percent.**

Conclusion

The Random Forest classifier leveraging demographic, satisfaction, tenure and compensation variables delivers strong predictive performance for employee attrition. Its high specificity indicates reliable identification of employees likely to remain, while its sensitivity demonstrates effective detection of those at risk of leaving. **Variables such as OverTime, JobSatisfaction, YearsSinceLastPromotion and MonthlyIncome consistently rank among the most influential predictors.** To further align the model with organizational priorities, one could adjust the classification threshold or apply resampling techniques such as SMOTE or cost-sensitive learning to reduce the remaining false negatives. Overall, this model offers a robust early-warning mechanism for HR teams seeking to prioritize retention efforts.

Dashboard Pages:



C:/Users/Gianne Bacay/Desktop/Business Analytics/MP_3/Employee_Attrition_Classification - Shiny

http://127.0.0.1:7756 | Open in Browser

Employee Attrition Prediction Dashboard

Input Employee Details:

Age: 36

Distance From Home: 7

Department: Human Resources

Job Level: 2

Job Role: Healthcare Representative

Job Satisfaction (1–4): 3

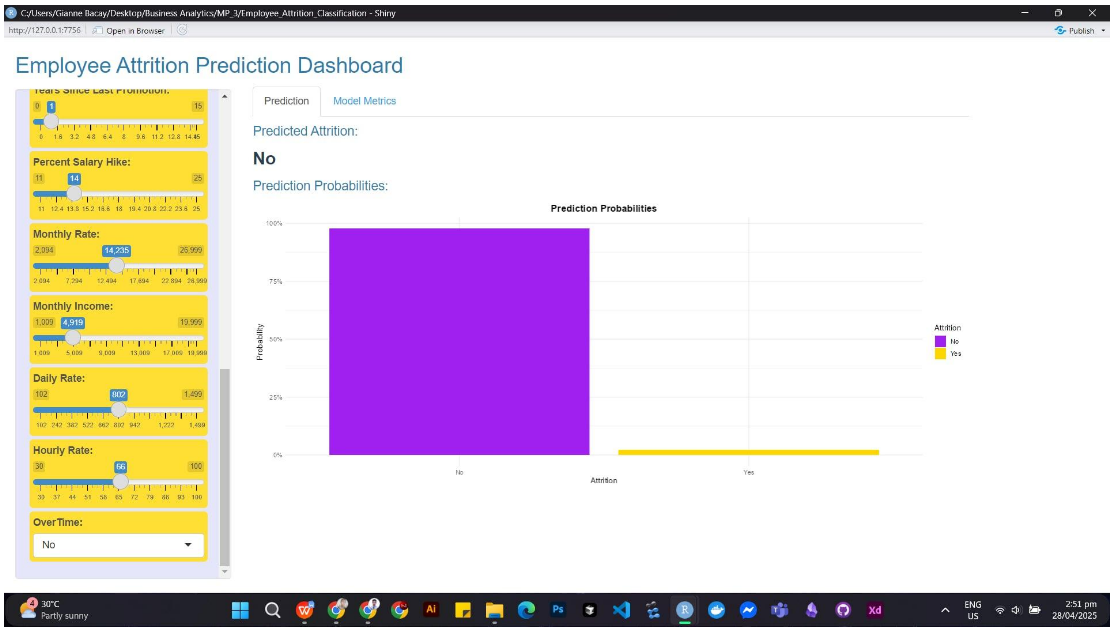
Environment Satisfaction (1–4): 3

Prediction | Model Metrics

Predicted Attrition:

No

Prediction Probabilities:



C:/Users/Gianne Bacay/Desktop/Business Analytics/MP_3/Employee_Attrition_Classification - Shiny

http://127.0.0.1:7756 | Open in Browser

Employee Attrition Prediction Dashboard

Years Since Last Promotion: 1

Percent Salary Hike: 14

Monthly Rate: 14,235

Monthly Income: 4,919

Daily Rate: 802

Hourly Rate: 66

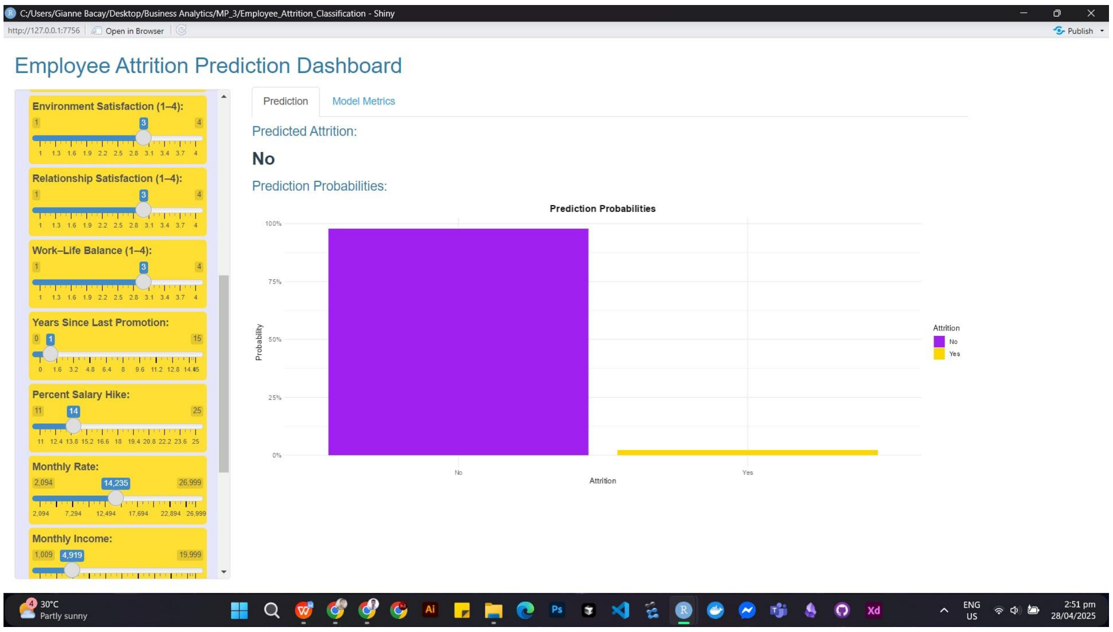
OverTime: No

Prediction | Model Metrics

Predicted Attrition:

No

Prediction Probabilities:



C:/Users/Gianne Bacay/Desktop/Business Analytics/MP_3/Employee_Attrition_Classification - Shiny

http://127.0.0.1:7756 | Open in Browser

Employee Attrition Prediction Dashboard

Environment Satisfaction (1–4): 3

Relationship Satisfaction (1–4): 3

Work–Life Balance (1–4): 3

Years Since Last Promotion: 1

Percent Salary Hike: 14

Monthly Rate: 14,235

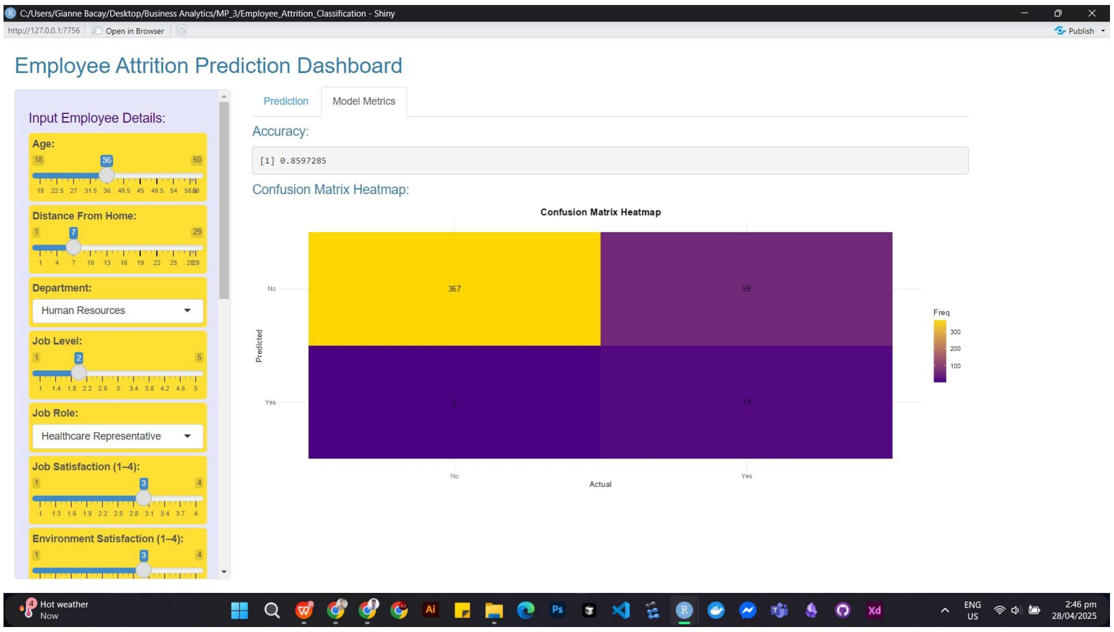
Monthly Income: 4,919

Prediction | Model Metrics

Predicted Attrition:

No

Prediction Probabilities:



Employee Attrition Prediction Dashboard

Input Employee Details:

Age: 36 (18–60)

Distance From Home: 7 (1–29)

Department: Human Resources

Job Level: 2 (1–5)

Job Role: Healthcare Representative

Job Satisfaction (1–4): 3

Environment Satisfaction (1–4): 3

Accuracy:

```
[1] 0.8597285
```

Confusion Matrix Heatmap:

	No	Yes
No	367	59
Yes	3	13

Shiny App Code:

```
library(shiny)
library(shinythemes)
library(tidymodels)
library(tidyverse)

# Read and clean the dataset
raw_data <- read.csv(
  "C:/Users/Gianne Bacay/Desktop/Business
Analytics/MP_3/Employee_Attrition_Classification/dataset.csv",
  stringsAsFactors = FALSE
)
clean_data <- raw_data %>%
  na.omit() %>%      # Remove rows with missing values
  distinct()         # Remove duplicate rows

# Select relevant predictors and prepare types
data <- clean_data %>%
  select(
    Attrition, Age, DistanceFromHome, Department,
    JobLevel, JobRole, JobSatisfaction, EnvironmentSatisfaction,
    RelationshipSatisfaction, WorkLifeBalance, YearsSinceLastPromotion,
    PercentSalaryHike, MonthlyRate, MonthlyIncome, DailyRate,
    HourlyRate, OverTime
  ) %>%
  mutate(
    Attrition = factor(Attrition, levels = c("No", "Yes")),
    Department = factor(Department),
    JobRole    = factor(JobRole),
    OverTime   = factor(OverTime)
  )

# Split for modeling
set.seed(123)
split <- initial_split(data, prop = 0.7, strata = Attrition)
train_data <- training(split)
test_data  <- testing(split)

# Recipe: encode & normalize
attr_recipe <- recipe(Attrition ~ ., data = train_data) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors())

# Model specification
rf_spec <- rand_forest(trees = 500, mtry = 5, min_n = 5) %>%
  set_engine("ranger") %>%
  set_mode("classification")

# Workflow & fit
attr_workflow <- workflow() %>%
  add_recipe(attr_recipe) %>%
  add_model(rf_spec)
rf_fit <- fit(attr_workflow, data = train_data)

# Evaluate on test
test_pred <- predict(rf_fit, test_data, type = "prob") %>%
  bind_cols(predict(rf_fit, test_data)) %>%
  bind_cols(test_data %>% select(Attrition))
accuracy_res <- accuracy(test_pred, truth = Attrition, estimate = .pred_class)
conf_mat_res <- conf_mat(test_pred, truth = Attrition, estimate = .pred_class)

# Define Shiny UI and custom CSS
ui <- fluidPage(
  theme = shinytheme("cerulean"),
  tags$head(
    tags$style(HTML(
      "/* Set white page background */
body { background-color: #FFFFFF; }
/* Sidebar: light purple background */
.sidebar-scroll { background-color: #E6E6FA; }
/* Sidebar headers: darker purple */
.sidebar-scroll h4 { color: #4B0082; }
/* Sidebar input panels: light yellow cards */
.sidebar-scroll .form-group { background-color: #FFDF33; padding: 5px; margin-
bottom: 5px; border-radius: 5px; }
/* Highlight sections: light yellow banners */
.sidebar-scroll .well { background-color: #FFFFE0; }
/* Fix sidebar positioning */
.sidebar-scroll { position: fixed; top: 70px; bottom: 0; overflow-y: auto; width: 300px;
padding-right: 15px; }
.main-static  { margin-left: 330px; }
@media (max-width: 768px) {
  .sidebar-scroll { position: static; width: 100%; height: auto; }
  .main-static  { margin-left: 0; }
}
"    ))
  ),
  titlePanel("Employee Attrition Prediction Dashboard"),
  sidebarLayout(
    sidebarPanel(
      class = "sidebar-scroll",
      h4("Input Employee Details:"),
      sliderInput("Age", "Age:", min = min(data$Age), max = max(data$Age), value =
median(data$Age)),
      sliderInput("DistanceFromHome", "Distance From Home:", min =
min(data$DistanceFromHome), max = max(data$DistanceFromHome), value =
median(data$DistanceFromHome)),
      selectInput("Department", "Department:", choices = levels(data$Department)),
      sliderInput("JobLevel", "Job Level:", min = min(data$JobLevel), max =
max(data$JobLevel), value = median(data$JobLevel)),
      selectInput("JobRole", "Job Role:", choices = levels(data$JobRole)),
      sliderInput("JobSatisfaction", "Job Satisfaction (1–4):", min =
min(data$JobSatisfaction), max = max(data$JobSatisfaction), value =
median(data$JobSatisfaction)),
      sliderInput("EnvironmentSatisfaction", "Environment Satisfaction (1–4):", min =
min(data$EnvironmentSatisfaction), max = max(data$EnvironmentSatisfaction),
value = median(data$EnvironmentSatisfaction)),
      sliderInput("RelationshipSatisfaction", "Relationship Satisfaction (1–4):", min =
min(data$RelationshipSatisfaction), max = max(data$RelationshipSatisfaction), value
= median(data$RelationshipSatisfaction)),
      sliderInput("WorkLifeBalance", "Work–Life Balance (1–4):", min =
min(data$WorkLifeBalance), max = max(data$WorkLifeBalance), value =
median(data$WorkLifeBalance)),
      sliderInput("YearsSinceLastPromotion", "Years Since Last Promotion:", min =
min(data$YearsSinceLastPromotion), max = max(data$YearsSinceLastPromotion),
value = median(data$YearsSinceLastPromotion)),
      sliderInput("PercentSalaryHike", "Percent Salary Hike:", min =
min(data$PercentSalaryHike), max = max(data$PercentSalaryHike), value =
median(data$PercentSalaryHike)),
      sliderInput("MonthlyRate", "Monthly Rate:", min = min(data$MonthlyRate), max
= max(data$MonthlyRate), value = median(data$MonthlyRate)),
      sliderInput("MonthlyIncome", "Monthly Income:", min =
min(data$MonthlyIncome), max = max(data$MonthlyIncome), value =
median(data$MonthlyIncome)),
      sliderInput("DailyRate", "Daily Rate:", min = min(data$DailyRate), max =
max(data$DailyRate), value = median(data$DailyRate)),
      sliderInput("HourlyRate", "Hourly Rate:", min = min(data$HourlyRate), max =
max(data$HourlyRate), value = median(data$HourlyRate)),
      selectInput("OverTime", "OverTime:", choices = levels(data$OverTime))
    ),
    mainPanel(
      class = "main-static",
      tabsetPanel(
        tabPanel("Prediction",
                 h4("Predicted Attrition:"), htmlOutput("predAttr"),
                 h4("Prediction Probabilities:"), plotOutput("probPlot")
        ),
        tabPanel("Model Metrics",
                 h4("Accuracy:"), verbatimTextOutput("accuracy"),
                 h4("Confusion Matrix Heatmap:"), plotOutput("confHeatmap")
        )
      )
    )
  )
)

# Define Shiny Server
server <- function(input, output) {
  new_data <- reactive({
    tibble(
      Age = as.integer(input$Age),
      DistanceFromHome = as.integer(input$DistanceFromHome),
      Department = factor(input$Department, levels = levels(data$Department)),
      JobLevel = as.integer(input$JobLevel),
      JobRole = factor(input$JobRole, levels = levels(data$JobRole)),
      JobSatisfaction = as.integer(input$JobSatisfaction),
      EnvironmentSatisfaction = as.integer(input$EnvironmentSatisfaction),
      RelationshipSatisfaction = as.integer(input$RelationshipSatisfaction),
      WorkLifeBalance = as.integer(input$WorkLifeBalance),
      YearsSinceLastPromotion = as.integer(input$YearsSinceLastPromotion),
      PercentSalaryHike = as.integer(input$PercentSalaryHike),
      MonthlyRate = as.integer(input$MonthlyRate),
      MonthlyIncome = as.integer(input$MonthlyIncome),
      DailyRate = as.integer(input$DailyRate),
      HourlyRate = as.integer(input$HourlyRate),
      OverTime = factor(input$OverTime, levels = levels(data$OverTime))
    )
  })
```

```r
pred_class <- reactive({
  predict(rf_fit, new_data()) %>% pull(.pred_class)
})
pred_prob <- reactive({
  predict(rf_fit, new_data(), type = "prob")
})

output$predAttr <- renderUI({
  div(style = "font-size:24px; font-weight:bold; color:#2C3E50;", pred_class())
})
output$probPlot <- renderPlot({
  prob_df <- pred_prob() %>%
    pivot_longer(
      cols = everything(),
      names_to = "Attrition",
      names_prefix = ".pred_",
      values_to = "Probability"
    )
  gg <- ggplot(prob_df, aes(x = Attrition, y = Probability, fill = Attrition)) +
    geom_col() +
    scale_fill_manual(values = c("No" = "purple", "Yes" = "gold")) +
    scale_y_continuous(labels = scales::percent_format()) +
    labs(title = "Prediction Probabilities", x = "Attrition", y = "Probability") +
    theme_minimal() +
    theme(
      plot.title = element_text(hjust=0.5, face="bold"),
      panel.background = element_rect(fill = "#FFFFFF", color = NA),
      plot.background = element_rect(fill = "#FFFFFF", color = NA)
    )
  print(gg)
})

output$accuracy <- renderPrint({ accuracy_res %>% pull(.estimate) })

output$confHeatmap <- renderPlot({
  gg <- conf_mat_res %>% autoplot(type = "heatmap") +
    scale_fill_gradientn(colors = c("#4B0082", "#FFD700")) +
    labs(x = "Actual", y = "Predicted", title = "Confusion Matrix Heatmap") +
    theme_minimal() +
    theme(
      plot.title = element_text(hjust=0.5, face="bold"),
      panel.background = element_rect(fill = "#FFFFFF", color = NA),
      plot.background = element_rect(fill = "#FFFFFF", color = NA)
    )
  print(gg)
})
```

```
}

# Launch the Shiny App
shinyApp(ui = ui, server = server)
```