Group 9 - Data-Driven Optimisation of Supply Chain Management

Data mining problem 1 - Analyse the factors that will result in an order delay

```
# Load necessary libraries
# Load necessary libraries
library(readxl)
library(dplyr)
library(lubridate)
library(ggplot2)
library(reshape2)
library(corrplot)
library(reshape2)
library(rpart)
library(caret)
library(lattice)
library(randomForest)
library(forecast)
library(pROC)
library(tidyr)

# Data mining problem 1 - Analyse the factors that will result in an order de
lay

# Load the dataset
data <- read_excel("C:/Users/Xi/Desktop/incom2024_delay_example_dataset.xlsx
")

# Data Preprocessing
data<- na.omit(data)
data <- unique(data)
data<- data[data$customer_state != '91732', ]

# Label Transformation
data <- data %>%
  mutate(label = ifelse(label %in% c(-1, 0), 0, label)) %>%
  mutate(label = case_when(
    label == 0 ~ "Not Delayed",
    label == 1 ~ "Delayed",
    TRUE ~ as.character(label)
  ))
data$label <- as.factor(data$label)

# Convert character columns to factors
data <- data %>%mutate(across(where(is.character), as.factor))

set.seed(42)
```

```r
# Create training set (40% of the data)
train_index <- createDataPartition(data$label, p = 0.4, list = FALSE)
train_data <- data[train_index, ]

# Remaining data split into validation and testing sets (each 50% of the rema
ining 60%)
temp_data <- data[-train_index, ]
valid_index <- createDataPartition(temp_data$label, p = 0.5, list = FALSE)
valid_data <- temp_data[valid_index, ]
test_data <- temp_data[-valid_index, ]

# Output the size of each dataset
output <- paste("train:", nrow(train_data),
                ", valid:", nrow(valid_data),
                ", test:", nrow(test_data))
cat(output, "\n")


# Train a Random Forest Model (Partially explainable feature)
rf_model <- randomForest(label ~ shipping_mode + order_region + category_name
                                   + order_item_total_amount+ customer_state
                                   + customer_segment + department_name
                                   + payment_type,
                                   data = train_data, importance = TRUE)

# Print the Random Forest model summary
print(rf_model)

## Confusion matrix:
##             Delayed Not Delayed class.error
## Delayed        2089        1501   0.4181058
## Not Delayed     584        2046   0.2220532

# Make predictions on the validation dataset
rf_valid_predictions <- predict(rf_model, valid_data)

# Evaluate model performance on the validation dataset
confusion_matrix_valid <- confusionMatrix(as.factor(rf_valid_predictions), as
.factor(valid_data$label))

# Display the confusion matrix and evaluation metrics
print(confusion_matrix_valid)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Delayed Not Delayed
##   Delayed        1531         403
##   Not Delayed    1162        1569
##
```

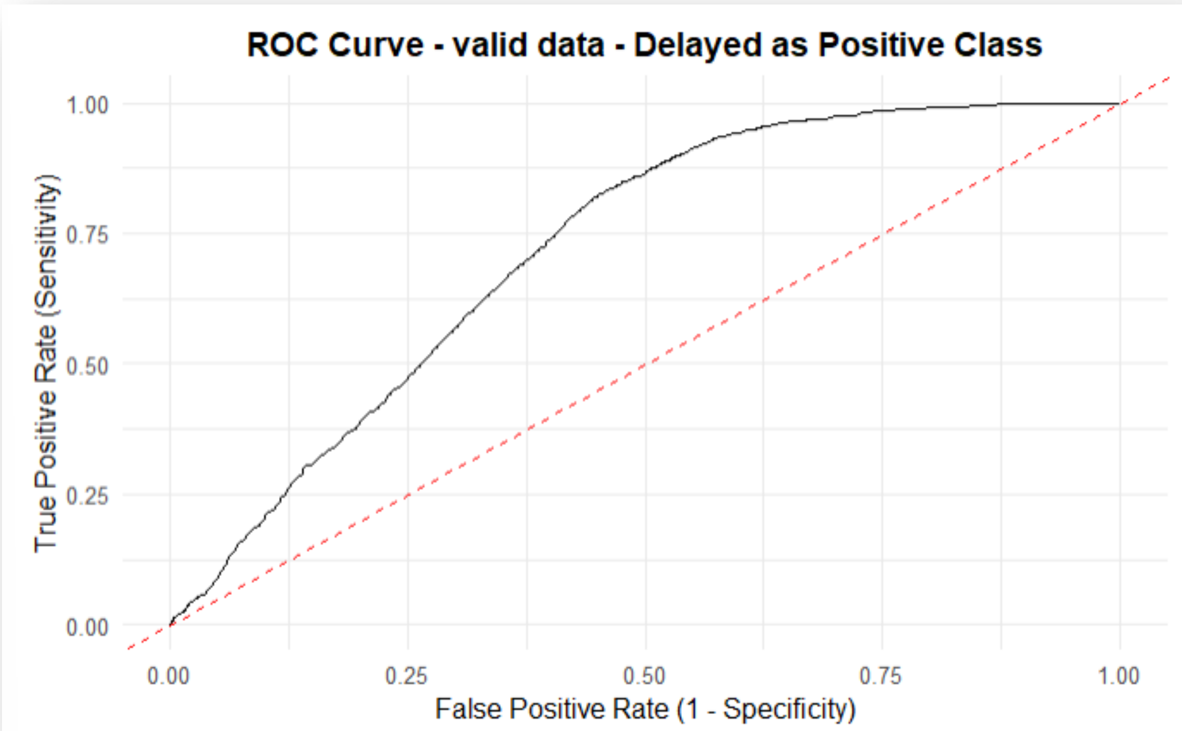
```
##                Accuracy : 0.6645
##                  95% CI : (0.6508, 0.6781)
##     No Information Rate : 0.5773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3463
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5685
##             Specificity : 0.7956
##          Pos Pred Value : 0.7916
##          Neg Pred Value : 0.5745
##              Prevalence : 0.5773
##          Detection Rate : 0.3282
##    Detection Prevalence : 0.4146
##       Balanced Accuracy : 0.6821
##
##        'Positive' Class : Delayed
##
```

```
# Calculate precision, recall, and F1 score for the validation data
valid_precisions_rf <- posPredValue(rf_valid_predictions, valid_data$label, p
ositive = "Delayed")
valid_recall_rf <- sensitivity(rf_valid_predictions, valid_data$label, positi
ve = "Delayed")
valid_f1_score_rf <- (2 * valid_precisions_rf * valid_recall_rf) / (valid_pre
cisions_rf + valid_recall_rf)


# Print precision, recall, and F1 score
print(c("valid Precision" = valid_precisions_rf, "valid Recall" = valid_recal
l_rf, "valid F1 Score" = valid_f1_score_rf))
```

```
## valid Precision    valid Recall  valid F1 Score
##       0.7916236       0.5685110       0.6617679
```

```
# Calculate predicted probabilities for the validation dataset
rf_valid_probabilities <- predict(rf_model, valid_data, type = "prob")
positive_class_probabilities_v <- rf_valid_probabilities[, "Delayed"]

# Generate the ROC curve for validation data
roc_curve_v <- roc(valid_data$label, positive_class_probabilities_v)

# Plot the ROC curve for validation data
ggplot(data.frame(FPR = 1 - roc_curve_v$specificities, TPR = roc_curve_v$sens
itivities), aes(x = FPR, y = TPR)) +
  geom_line() +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "ROC Curve - valid data - Delayed as Positive Class",
       x = "False Positive Rate (1 - Specificity)",
```

```
      y = "True Positive Rate (Sensitivity)") +
  xlim(0, 1) +
  ylim(0, 1) +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```




```
# Calculate AUC for the validation data
auc_value_v <- auc(roc_curve_v)
print(paste("AUC_v:", auc_value_v))

## [1] "AUC_v: 0.719667058085383"

# Make predictions on the test dataset
rf_test_predictions <- predict(rf_model, test_data)

# Evaluate model performance on the test dataset using a confusion matrix
confusion_matrix_test <- confusionMatrix(as.factor(rf_test_predictions), as.f
actor(test_data$label))

# Print the confusion matrix and related statistics
print(confusion_matrix_test)

## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Delayed Not Delayed
```

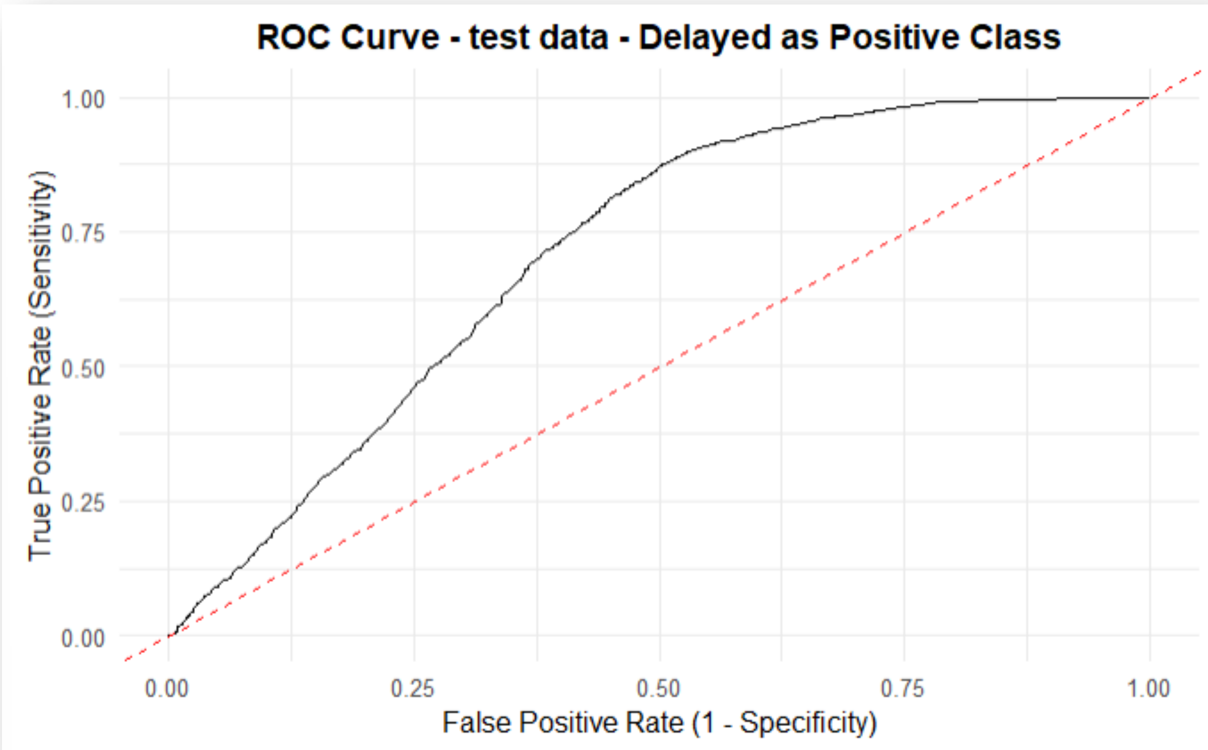
```
##   Delayed        1529         439
##   Not Delayed    1163        1532
##
##                Accuracy : 0.6564
##                  95% CI : (0.6426, 0.6701)
##     No Information Rate : 0.5773
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3291
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5680
##             Specificity : 0.7773
##          Pos Pred Value : 0.7769
##          Neg Pred Value : 0.5685
##              Prevalence : 0.5773
##          Detection Rate : 0.3279
##    Detection Prevalence : 0.4220
##       Balanced Accuracy : 0.6726
##
##        'Positive' Class : Delayed
##
```

```
# Calculate precision, recall, and F1 score for the test data
test_precisions_rf <- posPredValue(rf_test_predictions, test_data$label, posi
tive = "Delayed")
test_recall_rf <- sensitivity(rf_test_predictions, test_data$label, positive
= "Delayed")
test_f1_score_rf <- (2 * test_precisions_rf * test_recall_rf) / (test_precisi
ons_rf + test_recall_rf)


# Print precision, recall, and F1 score for the test data
print(c("test Precision" = test_precisions_rf, "test Recall" = test_recall_rf
, "test F1 Score" = test_f1_score_rf))
```

```
## test Precision    test Recall  test F1 Score
##      0.7769309      0.5679792      0.6562232
```

```
# Calculate predicted probabilities for the test dataset
rf_test_probabilities <- predict(rf_model, test_data, type = "prob")
positive_class_probabilities_t <- rf_test_probabilities[, "Delayed"]

# Generate the ROC curve for the test dataset
roc_curve_t <- roc(test_data$label, positive_class_probabilities_t)

# Plot the ROC curve for the test dataset
ggplot(data.frame(FPR = 1 - roc_curve_t$specificities, TPR = roc_curve_t$sens
itivities), aes(x = FPR, y = TPR)) +
  geom_line() +
```

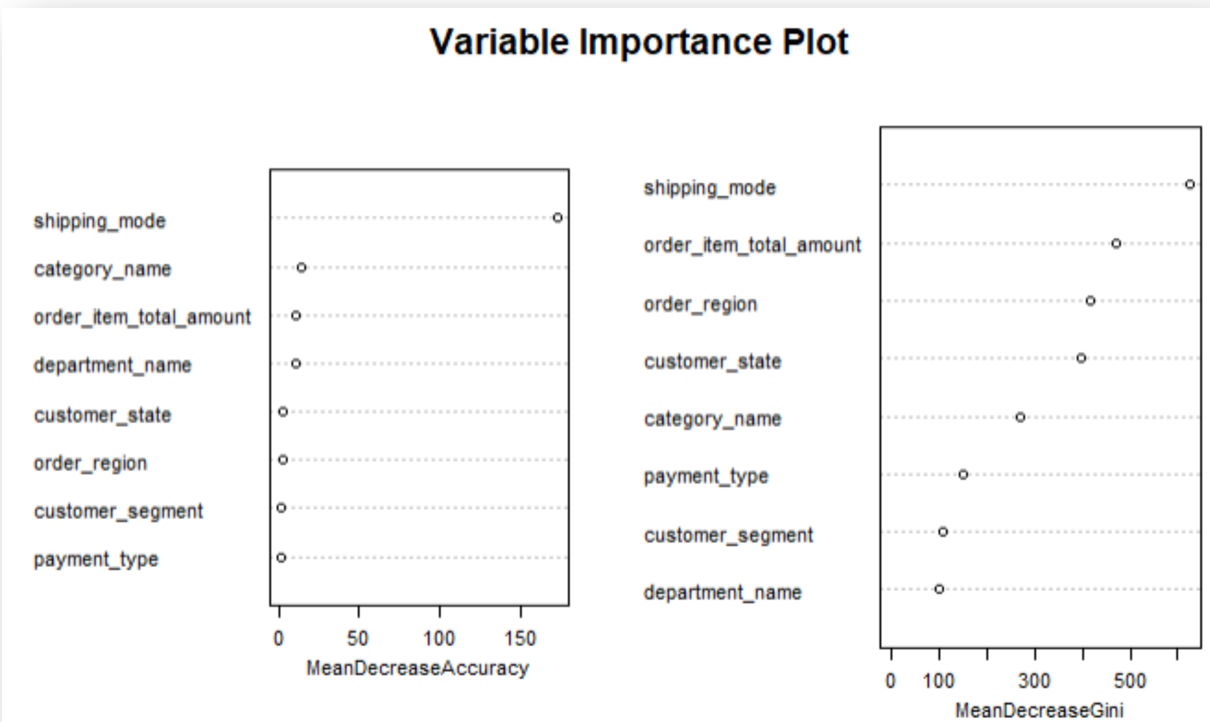
```r
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "ROC Curve - test data - Delayed as Positive Class",
       x = "False Positive Rate (1 - Specificity)",
       y = "True Positive Rate (Sensitivity)") +
  xlim(0, 1) +
  ylim(0, 1) +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```




```r
# Calculate AUC (Area Under the Curve) for the test dataset
auc_value_t <- auc(roc_curve_t)
print(paste("AUC_t:", auc_value_t))

## [1] "AUC_t: 0.710338918779962"

# Print feature importance
importance(rf_model)

##                            Delayed  Not Delayed MeanDecreaseAccuracy
## shipping_mode           131.484018 157.80435648          173.6836934
## order_region              1.259141   1.32107606            1.8662639
## category_name            40.050329 -36.07786975           14.0042911
## order_item_total_amount  23.122924 -16.01573522           10.1118339
## customer_state            1.738902   1.27729678            2.1381485
## customer_segment          1.407772   0.06297998            1.0605772
```

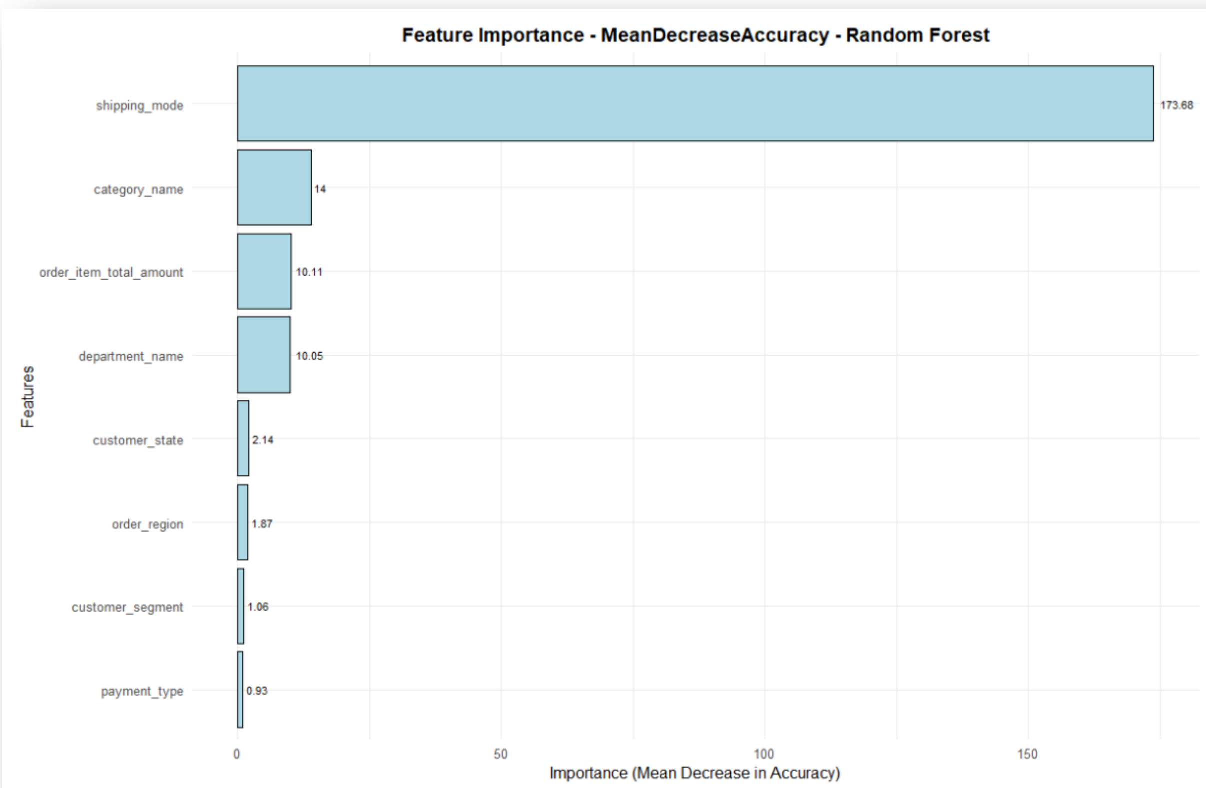
```
## department_name          32.020268 -31.71273204          10.0526398
## payment_type              1.175420   0.14907514           0.9320742
##                         MeanDecreaseGini
## shipping_mode                  623.83810
## order_region                   416.53759
## category_name                  269.93401
## order_item_total_amount        470.17352
## customer_state                 398.97601
## customer_segment               107.18833
## department_name                 99.23072
## payment_type                   148.63597
```

```
# Plot the importance of features
varImpPlot(rf_model,
           main = "",
           col = "black",
           cex = 0.7)
title(main = "Variable Importance Plot", line = 2, cex.main = 1.2, font.main
= 2)
```



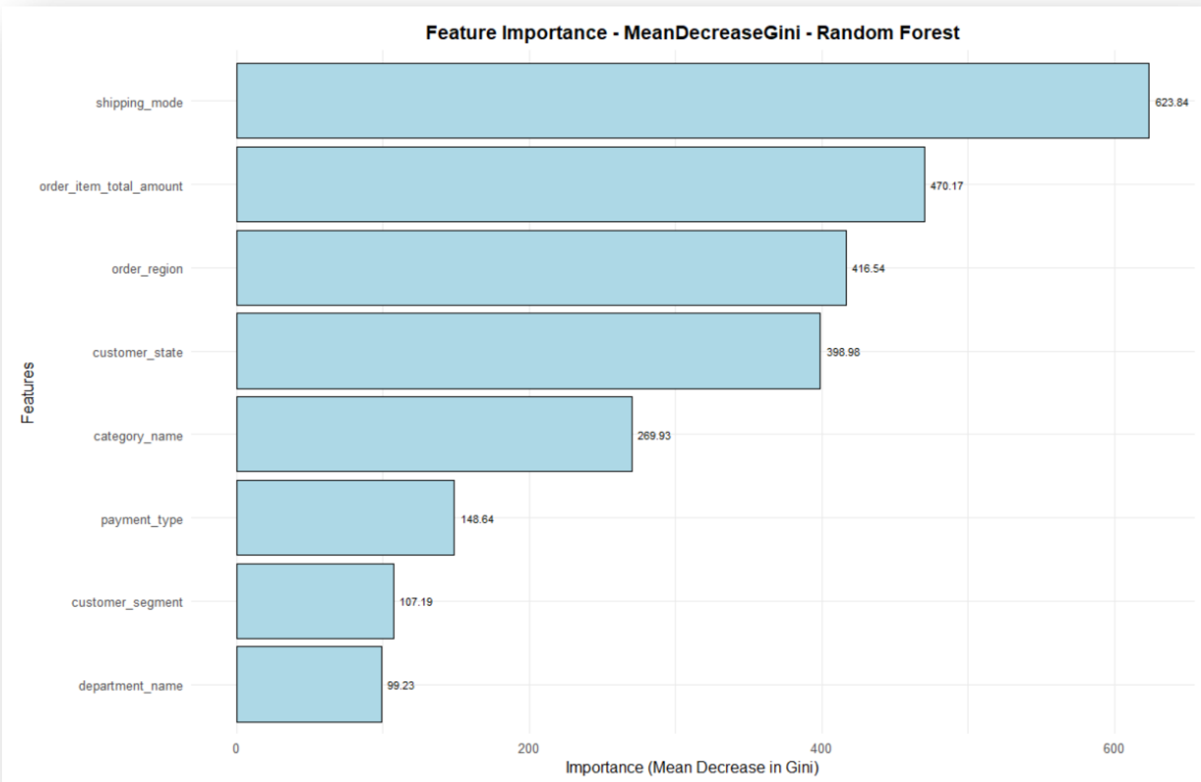

```
# Extract important features
importance_values <- importance(rf_model)
importance_df <- data.frame(
  Feature = rownames(importance_values),
  MeanDecreaseAccuracy = importance_values[, 'MeanDecreaseAccuracy'],
  MeanDecreaseGini = importance_values[, 'MeanDecreaseGini'])
```

```r
# Important Features Chart
ggplot(importance_df, aes(x = reorder(Feature, MeanDecreaseAccuracy), y = Mea
nDecreaseAccuracy)) +
  geom_bar(stat = 'identity', fill = 'lightblue', color = "black") +
  coord_flip() +
  geom_text(aes(label = round(MeanDecreaseAccuracy, 2)),
            hjust = -0.2, size = 3, color = 'black') +
  labs(title = 'Feature Importance - MeanDecreaseAccuracy - Random Forest',
       x = 'Features',
       y = 'Importance (Mean Decrease in Accuracy)') +
  theme_minimal(base_size = 12) +  # Adjust base text size for better readabi
lity
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),  # Center and
bold the title
        axis.text.y = element_text(size = 10),  # Adjust text size for featur
e names
        axis.text.x = element_text(size = 10))  # Adjust text size for the im
portance values
```




```r
# Important Features Chart - Mean Decrease in Gini
ggplot(importance_df, aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDec
reaseGini)) +
```

```
  geom_bar(stat = 'identity', fill = 'lightblue', color = "black") +
  coord_flip() +
  geom_text(aes(label = round(MeanDecreaseGini, 2)),
            hjust = -0.2, size = 3, color = 'black') +
  labs(title = 'Feature Importance - MeanDecreaseGini - Random Forest',
       x = 'Features',
       y = 'Importance (Mean Decrease in Gini)') +
  theme_minimal(base_size = 12) +  # Adjust base text size for better readabi
lity
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),  # Center and
bold the title
        axis.text.y = element_text(size = 10),  # Adjust text size for featur
e names
        axis.text.x = element_text(size = 10))  # Adjust text size for the im
portance values
```

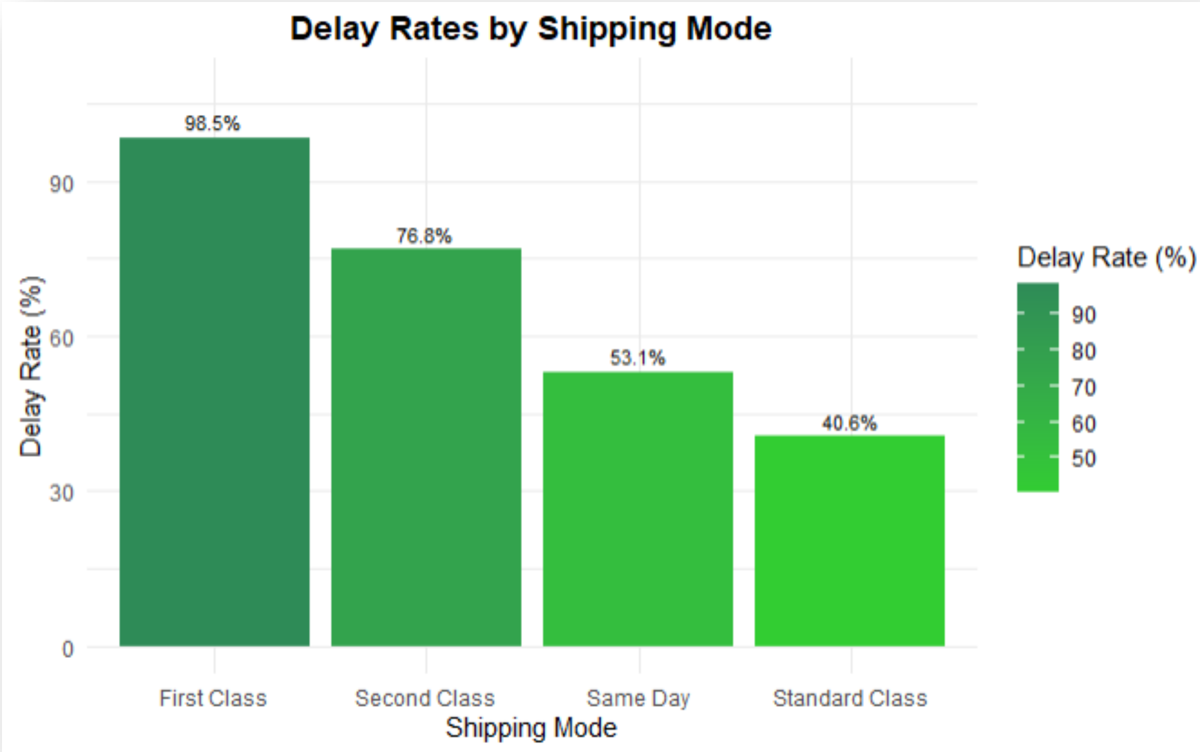



```
# Calculate delay rates per shipping_mode
delay_rate_shipping <- data %>%
  group_by(shipping_mode) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
```

```r
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)

# View the calculated delay rates
print(delay_rate_shipping)
```

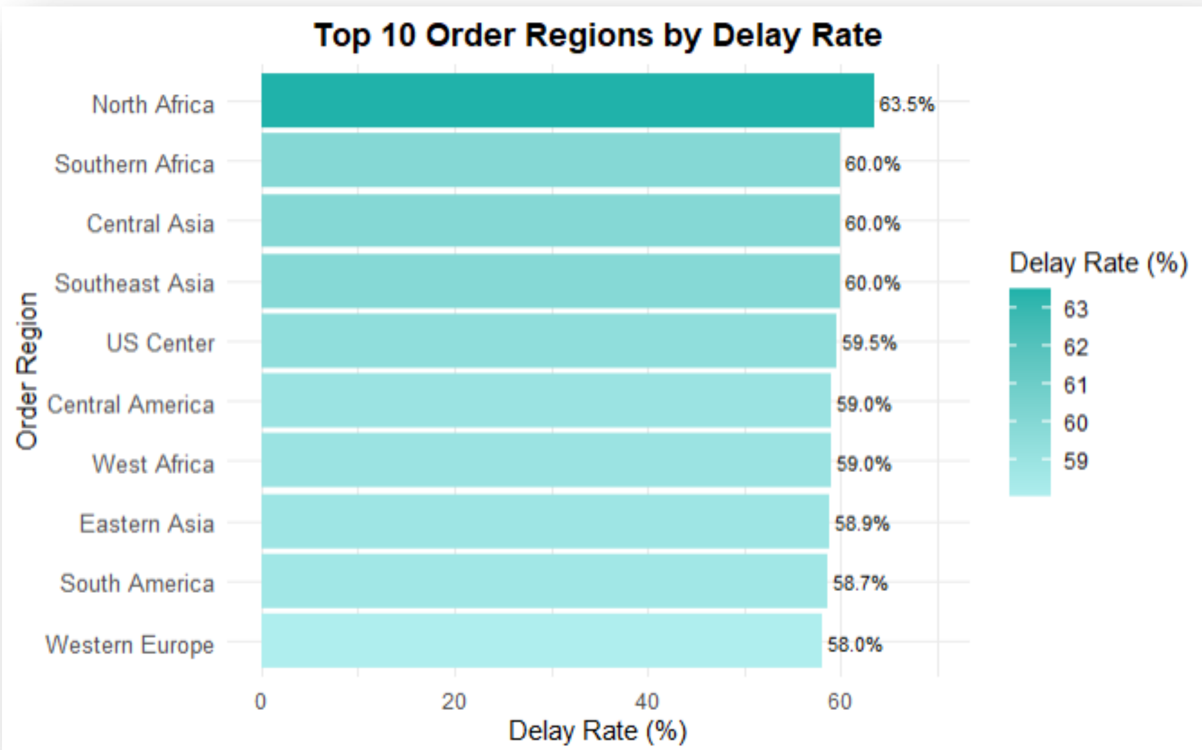
```
## # A tibble: 4 × 4
##   shipping_mode  Total_Orders Delayed_Orders Delay_Rate
##   <fct>                 <int>          <int>      <dbl>
## 1 First Class            2390           2353       98.5
## 2 Same Day                759            403       53.1
## 3 Second Class           3283           2520       76.8
## 4 Standard Class         9116           3699       40.6
```

```r
# Plot the delay rates using ggplot2 with orange gradient and no X-axis rotat
ion
ggplot(delay_rate_shipping, aes(x = reorder(shipping_mode, -Delay_Rate), y =
Delay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)), vjust = -0.5, size =
3) +
  scale_fill_gradient(low = "limegreen", high = "seagreen") +  # Define gradi
ent colors
  labs(
    title = "Delay Rates by Shipping Mode",
    x = "Shipping Mode",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"  # Add legend title for fill
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.x = element_text(angle = 0, hjust = 0.5)  # Remove rotation
  ) +
  ylim(0, max(delay_rate_shipping$Delay_Rate) * 1.1)  # Adds space for labels
```



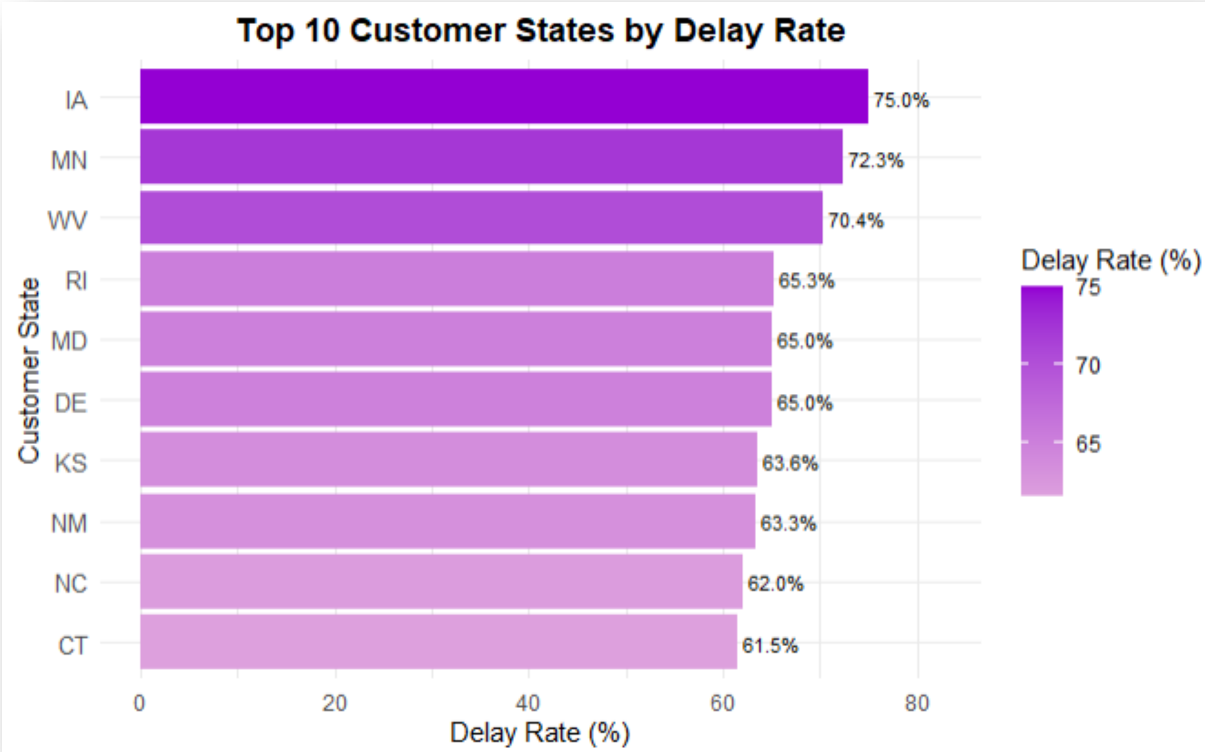

```
# Calculate delay rates for each order_region
delay_rate_region <- data %>%
  group_by(order_region) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)

# Select the top 10 regions with the highest delay rates
top10_delay_region <- delay_rate_region %>%
  arrange(desc(Delay_Rate)) %>%  # Sort by Delay_Rate in descending order
  slice(1:10)                    # Select the top 10 rows

# Plot the top 10 delay rates by order_region using ggplot2 with horizontal b
ars and color gradient
ggplot(top10_delay_region, aes(x = reorder(order_region, Delay_Rate), y = Del
ay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)),
            hjust = -0.1, size = 3) +  # Adjust horizontal justification for
better label placement
  scale_fill_gradient(low = "paleturquoise", high = "lightseagreen") +  # Lig
ht to dark gradient
```

```
  labs(
    title = "Top 10 Order Regions by Delay Rate",
    x = "Order Region",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"  # Legend title for the fill
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.y = element_text(size = 10)  # Adjust Y-axis (formerly X-axis)
text size if needed
  ) +
  ylim(0, max(top10_delay_region$Delay_Rate) * 1.1) +  # Adds space for label
s
  coord_flip()  # Flip the coordinates for horizontal bars
```



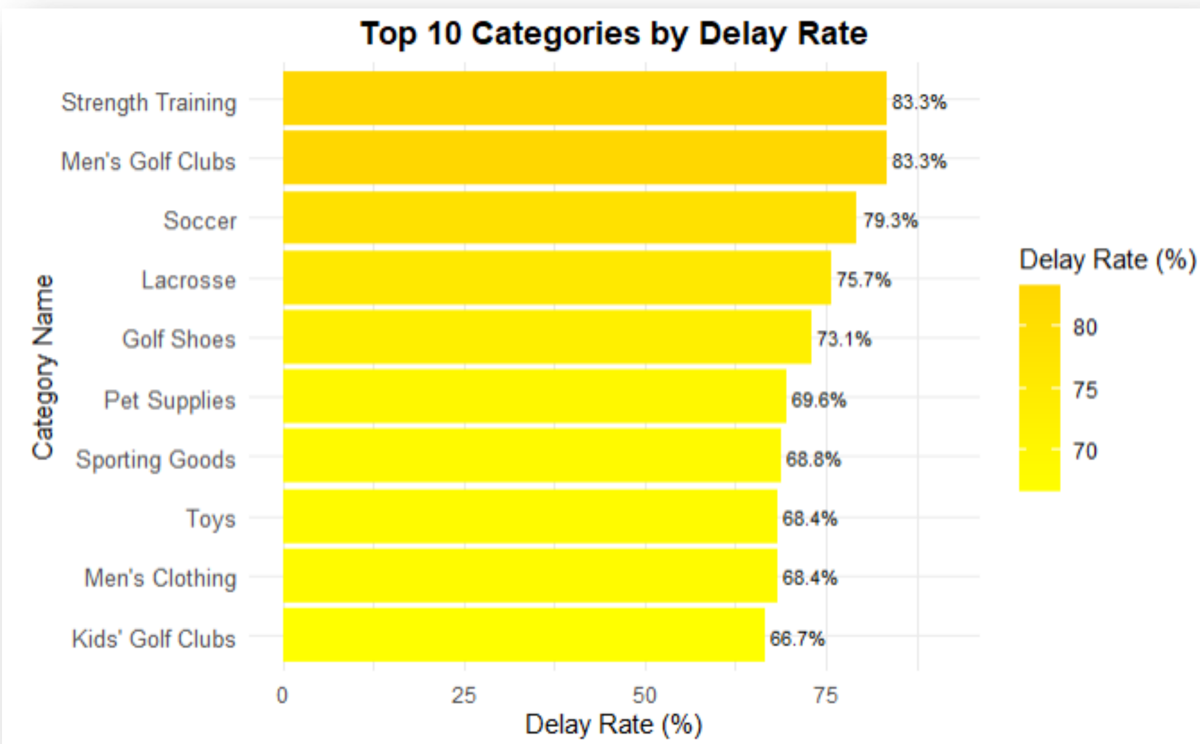

```
# Calculate delay rates per customer_state
delay_rate_state <- data %>%
  group_by(customer_state) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)
```

```r
# Select the top 10 states with the highest delay rates
top10_delay_state <- delay_rate_state %>%
  arrange(desc(Delay_Rate)) %>%  # Sort by Delay_Rate in descending order
  slice(1:10)                     # Select the top 10 rows

# Plot the top 10 delay rates by customer_state using ggplot2 with horizontal
 bars and color gradient
ggplot(top10_delay_state, aes(x = reorder(customer_state, Delay_Rate), y = De
lay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)),
            hjust = -0.1, size = 3) +  # Adjust horizontal justification for
better label placement
  scale_fill_gradient(low = "plum", high = "darkviolet") +  # Light to dark v
iolet gradient
  labs(
    title = "Top 10 Customer States by Delay Rate",
    x = "Customer State",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"  # Legend title for the fill
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.y = element_text(size = 10)  # Adjust Y-axis (formerly X-axis)
text size if needed
  ) +
  ylim(0, max(top10_delay_state$Delay_Rate) * 1.1) +  # Adds space for labels
  coord_flip()  # Flip the coordinates for horizontal bars
```



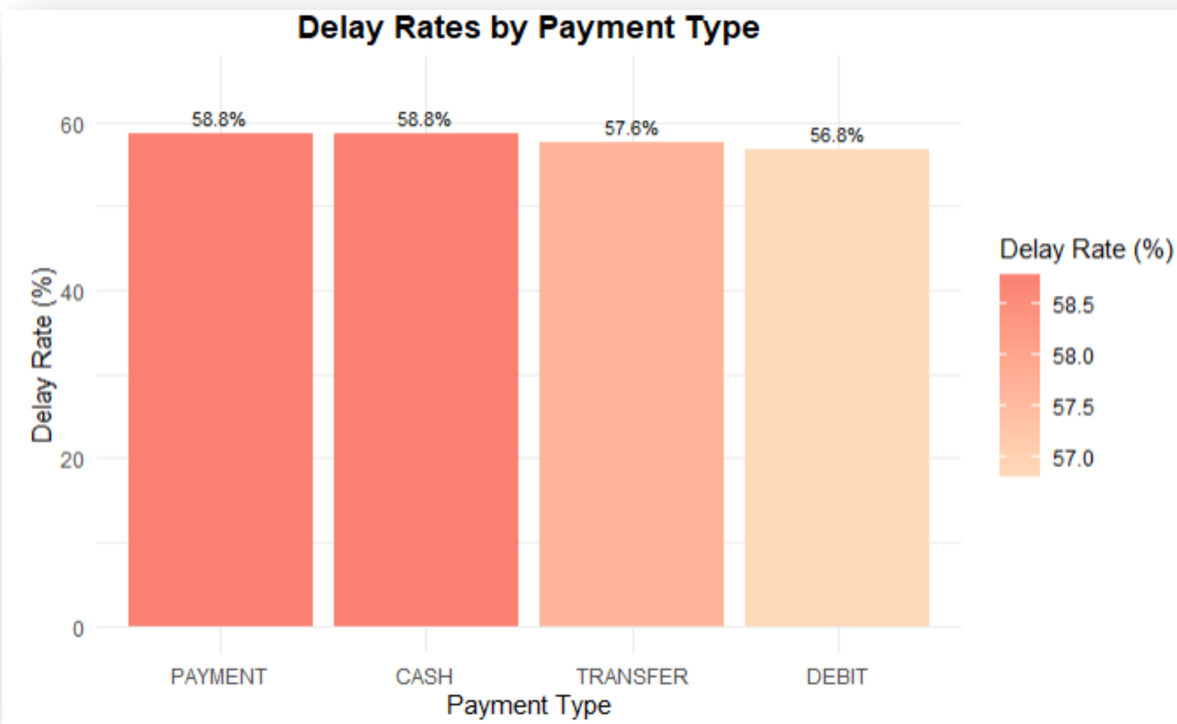

```r
# Calculate delay rates per category_name
delay_rate_category <- data %>%
  group_by(category_name) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)

# Select the top 10 categories with the highest delay rates
top10_delay_category <- delay_rate_category %>%
  arrange(desc(Delay_Rate)) %>%  # Sort by Delay_Rate in descending order
  slice(1:10)                     # Select the top 10 rows

# Plot the top 10 delay rates by category_name using ggplot2 with horizontal
bars and color gradient
ggplot(top10_delay_category, aes(x = reorder(category_name, Delay_Rate), y =
Delay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)),
            hjust = -0.1, size = 3) +  # Adjust horizontal justification for
better label placement
  scale_fill_gradient(low = "yellow", high = "gold") +  # Light to dark gradi
ent
```

```r
  labs(
    title = "Top 10 Categories by Delay Rate",
    x = "Category Name",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"  # Legend title for the fill
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.y = element_text(size = 10)  # Adjust Y-axis (formerly X-axis) 
text size if needed
  ) +
  ylim(0, max(top10_delay_category$Delay_Rate) * 1.1) +  # Adds space for lab
els
  coord_flip()  # Flip the coordinates for horizontal bars
```

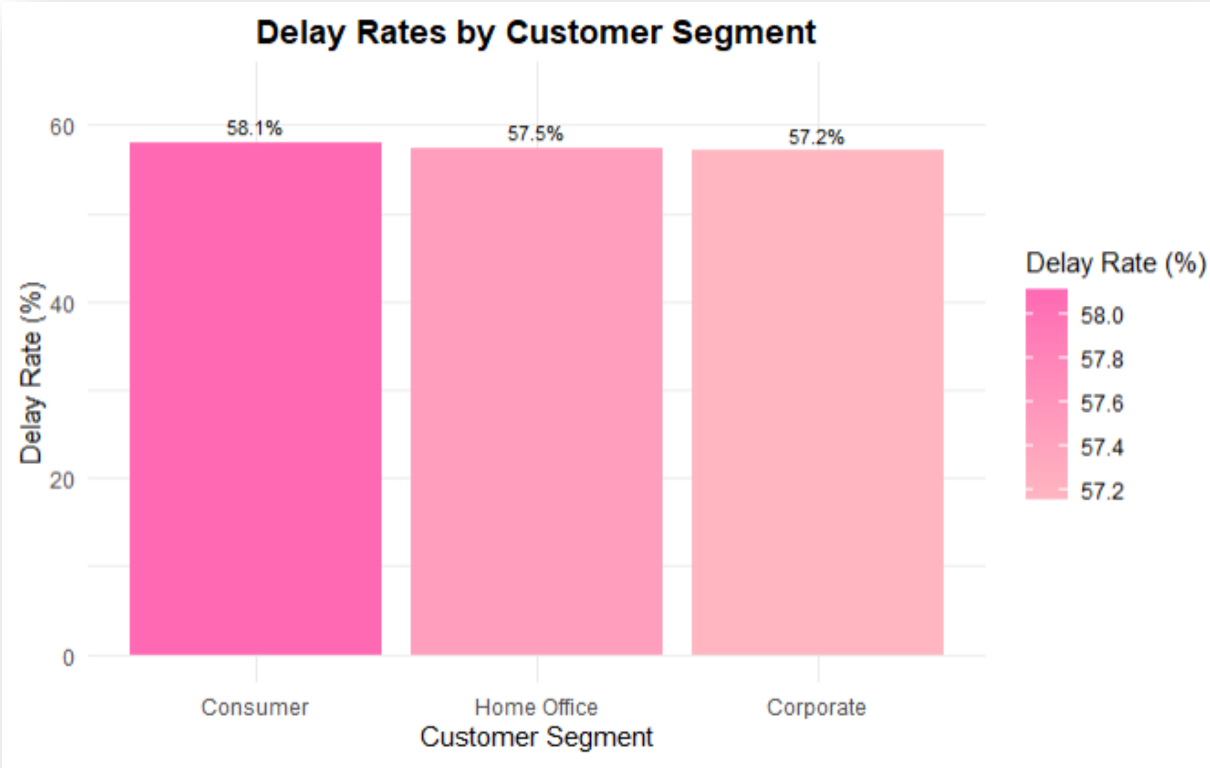



```r
# Calculate delay rates per payment_type
delay_rate_payment <- data %>%
  group_by(payment_type) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)
```

```
delay_rate_payment <- delay_rate_payment %>%
  filter(!is.na(payment_type))  # Remove any NA payment types if present
                      # Select the top 5 rows

# Plot the delay rates by payment_type using ggplot2
ggplot(delay_rate_payment, aes(x = reorder(payment_type, -Delay_Rate), y = De
lay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)), vjust = -0.5, size =
3) +
  scale_fill_gradient(low = "peachpuff", high = "salmon") +  # Light to dark
gradient
  labs(
    title = "Delay Rates by Payment Type",
    x = "Payment Type",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"  # Legend title for the fill
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.x = element_text(angle = 0, hjust = 0.5)  # Keep X-axis labels
horizontal
  ) +
  ylim(0, max(delay_rate_payment$Delay_Rate) * 1.1)  # Adds space for labels
```

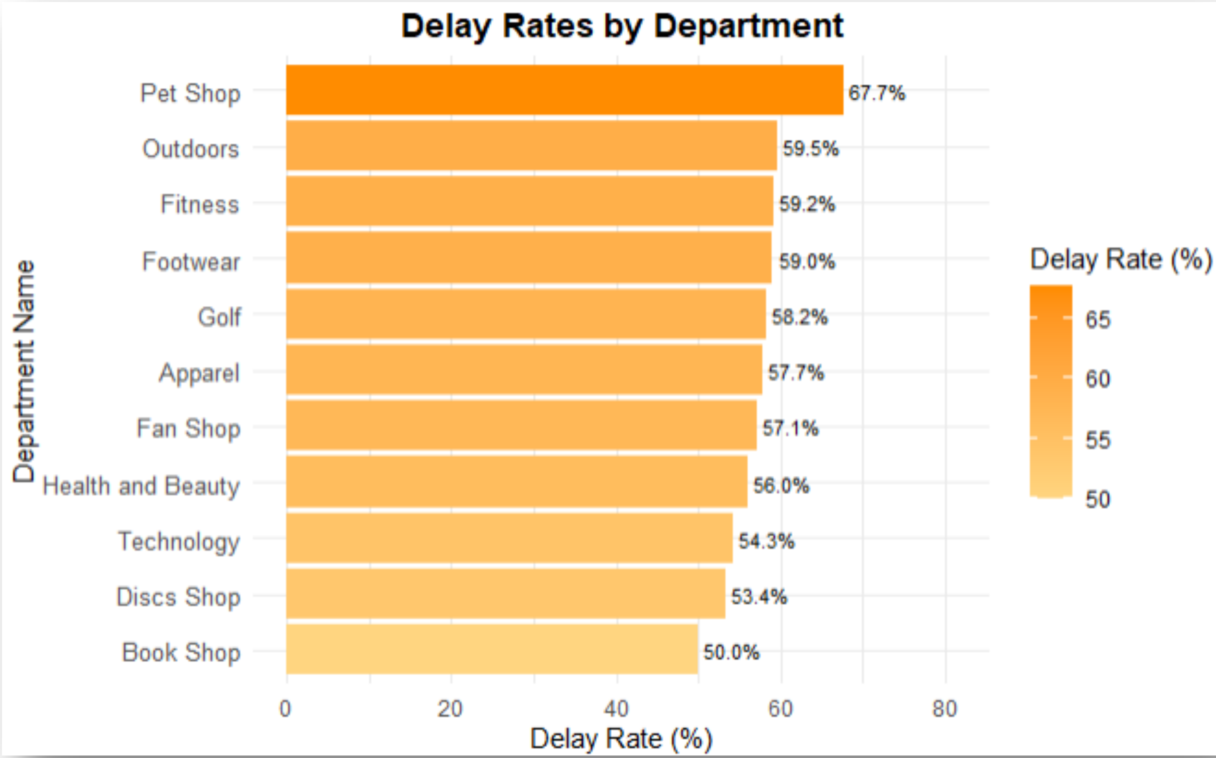


```
# Calculate delay rates per customer_segment
delay_rate_customer_segment <- data %>%
  group_by(customer_segment) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)

ggplot(delay_rate_customer_segment, aes(x = reorder(customer_segment, -Delay_
Rate), y = Delay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)), vjust = -0.5, size =
3) +
  scale_fill_gradient(low = "lightpink", high = "hotpink") +
   labs(
    title = "Delay Rates by Customer Segment",
    x = "Customer Segment",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.x = element_text(angle = 0, hjust = 0.5)   # Keep X-axis labels
 horizontal
  ) +
  ylim(0, max(delay_rate_customer_segment$Delay_Rate) * 1.1)
```



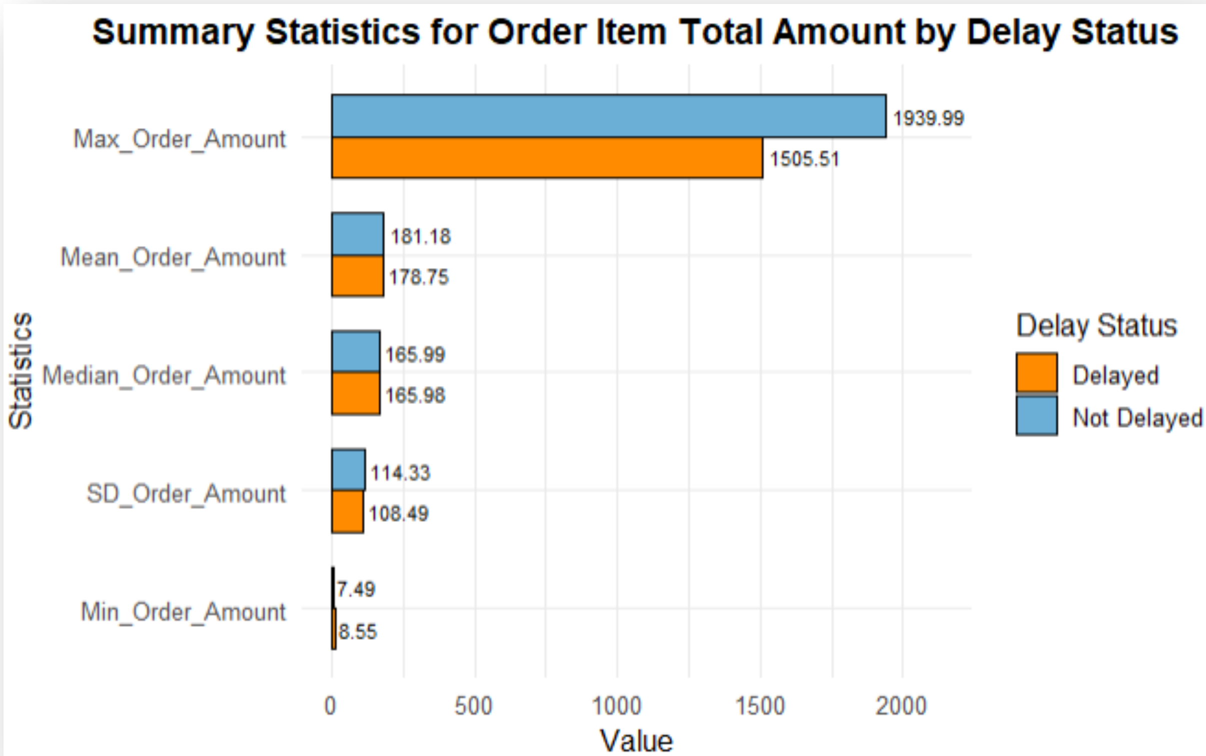

```
# Calculate delay rates for each department_name
delay_rate_department <- data %>%
  group_by(department_name) %>%
  summarize(
    Total_Orders = n(),
    Delayed_Orders = sum(label == "Delayed")
  ) %>%
  mutate(Delay_Rate = (Delayed_Orders / Total_Orders) * 100)

# View the calculated delay rates
print(delay_rate_department)

## # A tibble: 11 × 4
##    department_name   Total_Orders Delayed_Orders Delay_Rate
##    <fct>                    <int>          <int>      <dbl>
##  1 Apparel                   4273           2466       57.7
##  2 Book Shop                   26             13       50
##  3 Discs Shop                 163             87       53.4
##  4 Fan Shop                  5707           3259       57.1
##  5 Fitness                    179            106       59.2
##  6 Footwear                  1208            713       59.0
##  7 Golf                      3064           1782       58.2
##  8 Health and Beauty           25             14       56
##  9 Outdoors                   778            463       59.5
```

```
## 10 Pet Shop                          31              21        67.7
## 11 Technology                        94              51        54.3

# Plot the delay rates by department_name using ggplot2 with horizontal bars
and color gradient
ggplot(delay_rate_department, aes(x = reorder(department_name, Delay_Rate), y
 = Delay_Rate, fill = Delay_Rate)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", Delay_Rate)),
            hjust = -0.1, size = 3) +  # Adjust horizontal justification for
better label placement
  scale_fill_gradient(low = "#FFD580", high = "#FF8C00") +  # Gradient from l
ight orange to dark orange
  labs(
    title = "Delay Rates by Department",
    x = "Department Name",
    y = "Delay Rate (%)",
    fill = "Delay Rate (%)"  # Legend title for fill
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.y = element_text(size = 10)  # Adjust Y-axis (formerly X-axis)
text size if needed
  ) +
  ylim(0, max(delay_rate_department$Delay_Rate) * 1.2) +  # Increase space to
 accommodate labels
  coord_flip()  # Flip the coordinates for horizontal bars
```




```
# Calculate summary statistics for order_item_total_amount by delay status
order_amount_summary <- data %>%
  group_by(label) %>%
  summarize(
    Mean_Order_Amount = mean(order_item_total_amount, na.rm = TRUE),
    Median_Order_Amount = median(order_item_total_amount, na.rm = TRUE),
    Min_Order_Amount = min(order_item_total_amount, na.rm = TRUE),
    Max_Order_Amount = max(order_item_total_amount, na.rm = TRUE),
    SD_Order_Amount = sd(order_item_total_amount, na.rm = TRUE)
  )
print(order_amount_summary)

## # A tibble: 2 × 6
##   label   Mean_Order_Amount Median_Order_Amount Min_Order_Amount Max_Order_
Amount
##   <fct>               <dbl>               <dbl>            <dbl>
 <dbl>
## 1 Delay…               179.                166.             8.55
 1506.
## 2 Not D…               181.                166.             7.49
 1940.
## # ℹ 1 more variable: SD_Order_Amount <dbl>

# Reshape the data to long format for plotting
order_amount_long <- order_amount_summary %>%
```

```r
  pivot_longer(cols = Mean_Order_Amount:SD_Order_Amount,
               names_to = "Statistic",
               values_to = "Value")

# Create a horizontal bar chart for order item total amount statistics
ggplot(order_amount_long, aes(x = reorder(Statistic, Value), y = Value, fill
= label)) +
  geom_bar(stat = "identity", color = "black", position = "dodge", width = 0.
7) +
  coord_flip() +  # Flip for horizontal bars
  geom_text(aes(label = round(Value, 2)),
            hjust = -0.1, size = 3, color = 'black',
            position = position_dodge(width = 0.7)) +  # Bring labels closer
to the bars
  labs(
    title = "Summary Statistics for Order Item Total Amount by Delay Status",
    x = "Statistics",
    y = "Value",
    fill = "Delay Status"
  ) +
  theme_minimal(base_size = 12) +  # Adjust base text size for readability
  scale_fill_manual(values = c("#FF8C00", "#6BAED6")) +  # Customize colors
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),  # Center and bold
 the title
    axis.text.y = element_text(size = 10),  # Adjust size of y-axis labels
    axis.text.x = element_text(size = 10)   # Adjust size of x-axis values
  ) +
  expand_limits(y = max(order_amount_long$Value) * 1.1)  # Extend y-axis to c
reate space for labels
```


Summary Statistics for Order Item Total Amount by Delay Status
Statistics
Max_Order_Amount
Mean_Order_Amount
Median_Order_Amount
SD_Order_Amount
Min_Order_Amount
1939.99
1505.51
181.18
178.75
165.99
165.98
114.33
108.49
7.49
8.55
0
500
1000
1500
2000
Value
Delay Status
Delayed
Not Delayed