Draft Report: Leveraging Artificial Intelligence to Mitigate Delays and Cost Overruns in Public Infrastructure Construction Projects

Souleymane Doumbia

2025-05-22

Contents

Introduction (Analysis and Modeling part of the report)

Large-scale public infrastructure projects in New York City (NYC) are prone to delays and budget overruns, costing the city time, money, and public trust. This capstone project leverages publicly available capital project data to identify, quantify, and model risk factors that contribute to these inefficiencies. The project integrates data from several official sources, including citywide budget and schedule dashboards, milestone tracking datasets, and project-level capital spending records from NYC Open Data.

Initial exploratory data analysis revealed significant variation in project performance. Categories such as **Parks & Recreation** and **Public Buildings** dominate the project portfolio, with a disproportionate share of both delays and cost overruns. Many projects are concentrated in phases like **construction** and **design**, where planning uncertainty and procurement bottlenecks often introduce risks.

To quantify these risks, we performed classification of projects based on:

- Cost performance (`cost_class`): Over, Under, or On Budget (±15% threshold)
- Schedule performance (`delay_class`): Delayed, Early, or On Time (±30-day threshold)
- Combined cost/schedule risk (`high_risk`): Projects that are both delayed and over budget

These classifications form the basis for further **predictive modeling**, where we aim to identify which features — such as project phase, borough, agency, category group, or textual descriptions — are most associated with project failure modes.

---

1. Examining the datasets

1.1 Loading the datasets

```
# Load required libraries
library(readr)
library(dplyr)
library(janitor)

# Load CSV files
budget_schedule <- read_csv("Capital_Projects_Dashboard_-_Citywide_Budget_and_Schedule_20250413.csv") %>%
  clean_names()

project_dollars <- read_csv("Capital_Project_Detail_Data_-_Dollars_20250413.csv") %>%
  clean_names()

project_milestones <- read_csv("Capital_Project_Detail_Data_-_Milestones_20250413.csv") %>%
  clean_names()

# Peek at structure
#glimpse(budget_schedule)
#glimpse(project_dollars)
#glimpse(project_milestones)
```

1.2 Merging all datasets

```
#Basic Join on FMS ID + Project Name

# Merge budget_schedule and project_dollars on fms_id (same as project_id) and project name
merged_projects <- budget_schedule %>%
  inner_join(project_dollars, by = c("fms_id" = "project_id",
                                     "fms_project_name" = "project_descr"))

# View merged results
#glimpse(merged_projects)
```

2. Delay and Cost Overrun Analysis

2.1. Cost Overrun Threshold

```
library(dplyr)
library(lubridate)

# 1. Convert reporting_period to actual date
merged_projects <- merged_projects %>%
  mutate(reporting_date = ymd(paste0(reporting_period, "01")))

# 2. Extract earliest report (initial budget)
budget_earliest <- merged_projects %>%
  group_by(fms_id) %>%
  slice_min(reporting_date, with_ties = FALSE) %>%
  select(fms_id, initial_budget = total_budget)
```

```r
# 3. Extract latest report (final budget/spend)
budget_latest <- merged_projects %>%
  group_by(fms_id) %>%
  slice_max(reporting_date, with_ties = FALSE) %>%
  select(fms_id, latest_budget = total_budget, latest_spend = spend_to_date)

# 4. Merge and classify cost status with 15% threshold
budget_change <- budget_earliest %>%
  left_join(budget_latest, by = "fms_id") %>%
  mutate(
    cost_diff = latest_budget - initial_budget,
    cost_diff_pct = cost_diff / initial_budget,
    cost_class = case_when(
      cost_diff_pct > 0.15 ~ "Over Budget",
      cost_diff_pct < -0.15 ~ "Under Budget",
      TRUE ~ "On Budget"
    )
  )

# 5. View result
#head(budget_change)
```

2.2 Classify Schedule Delay

```r
library(dplyr)
library(lubridate)
library(janitor)

# 1. Clean and prepare milestone data
milestone_clean <- project_milestones %>%
  clean_names() %>%
  mutate(
    orig_start_date = mdy(orig_start_date),
    orig_end_date = mdy(orig_end_date),
    task_end_date = mdy(task_end_date)
  ) %>%
  filter(!is.na(orig_start_date), !is.na(orig_end_date), !is.na(task_end_date))  # Keep valid rows

# 2. Extract final milestone per project
# We'll assume the latest SEQ_NUMBER is the final project phase (according to the data sctructure)
final_milestones <- milestone_clean %>%
  group_by(project_id) %>%
  slice_max(seq_number, with_ties = FALSE) %>%
  mutate(
    planned_duration = as.numeric(orig_end_date - orig_start_date),
    actual_duration = as.numeric(task_end_date - orig_start_date),
    delay_days = actual_duration - planned_duration,
    delay_class = case_when(
      delay_days > 30 ~ "Delayed",
      delay_days < -30 ~ "Early",
      TRUE ~ "On Time"
    )
```

```
  ) %>%
  select(project_id, orig_start_date, orig_end_date, task_end_date, delay_days, delay_class)

# 3. Preview delay classifications
#head(final_milestones)
```

2.3 Merging Cost Overrun and Schedule Status

```
# 1. Merge delay and cost classification into one dataset
project_status <- budget_change %>%
  inner_join(final_milestones, by = c("fms_id" = "project_id"))

# 2. Create combined status label
project_status <- project_status %>%
  mutate(
    status_combined = paste(delay_class, "&", cost_class)
  )

# 3. Preview result
#head(project_status)
```

3. Exploratory Data Analysis

3.1 Adding more features to Project Status Data

```
library(dplyr)
library(janitor)

# 1. Clean column names if not already done
merged_projects <- clean_names(merged_projects)
project_dollars <- clean_names(project_dollars)

# 2. Select and dduplicate relevant columns from merged_projects
merged_info <- merged_projects %>%
  select(
    fms_id,
    borough,
    agency_project_description,
    ten_year_plan_category,
    current_phase,
    spend_to_date_percent
  ) %>%
  distinct()

# 3. Select and deduplicate relevant columns from project_dollars
dollar_info <- project_dollars %>%
  select(
    project_id,
    delay_desc,
    scope_text,
  ) %>%
  distinct()
```

```r
# 4. Join into project_status
project_status <- project_status %>%
  left_join(merged_info, by = "fms_id") %>%
  left_join(dollar_info, by = c("fms_id" = "project_id"))

# 5. Check result
#glimpse(project_status)
#write_csv(project_status, "project_status_enriched.csv")
```

3.2 Cleaning the Enhance Project Data

```r
# Remove empty, NA, or "#NA" descriptions
project_status_clean <- project_status %>%
  filter(
    !is.na(agency_project_description),
    !is.na(delay_desc),
    agency_project_description != "",
    delay_desc != "",
    !agency_project_description %in% c("NA", "#NA"),
    !delay_desc %in% c("NA", "#NA")
  )

project_status_clean <- project_status_clean %>%
  filter(
    !is.na(cost_diff_pct),
    !is.na(ten_year_plan_category),
    !is.na(scope_text)
  )

# View how many rows are left
#nrow(project_status_clean)
```

3.3. Comprehensive Data Cleaning and Classification for Infrastructure Project Insights

```r
library(ggplot2)
library(forcats) # or fct_infreq
library(dplyr)
library(stringr)


# Project profile
project_status_clean <- project_status_clean %>%
  mutate(project_profile = case_when(
    delay_class == "Delayed" & cost_class == "Over Budget"    ~ "High Risk",
    delay_class == "Delayed" & cost_class == "On Budget"      ~ "Schedule Risk",
    delay_class == "Delayed" & cost_class == "Under Budget"   ~ "Time Risk, Cost-Saving",
    delay_class == "On Time" & cost_class == "Over Budget"    ~ "Cost Risk",
    delay_class == "On Time" & cost_class == "On Budget"      ~ "On Track",
    delay_class == "On Time" & cost_class == "Under Budget"   ~ "Lean Delivery",
    delay_class == "Early" & cost_class == "Over Budget"      ~ "Fast but Costly",
    delay_class == "Early" & cost_class == "On Budget"        ~ "Strong Delivery",
```

```
    delay_class == "Early" & cost_class == "Under Budget"     ~ "Exceptional Performance",
    TRUE ~ "Unclassified"
  ))



# Cleaning current_phase
project_status_clean <- project_status_clean %>%
  mutate(
    # Remove leading/trailing spaces and parentheses
    current_phase = str_trim(current_phase),
    current_phase = str_remove_all(current_phase, "[\\(\\)]"),

    # Standardize known variants
    current_phase = case_when(
      is.na(current_phase) | str_to_lower(current_phase) %in% c("n/a", "", "na") ~ "Stopped",

    current_phase %in% c("pre-design", "Pre-Design", "Property Acquisition") ~ "Pre-Design",
    current_phase %in% c("Design", "Design Built", "Design-Build", "Design Build") ~ "Design",
    current_phase %in% c("CONSTRUCTION", "Construction", "construction") ~ "Construction",
    current_phase %in% c("Close-Out") ~ "Close-Out",
    current_phase %in% c("Completed") ~ "Completed",
    current_phase %in% c("Construction Procurement", "Partner-managed", "Consultant Services", "Equipmer
    current_phase %in% c("Pending","Cancelled", "CANCELLED", "Withdrawn", "Terminated", "Inactive", "On

      TRUE ~ current_phase
    )
  )

# Only unique row
project_status_clean <- project_status_clean %>%
  distinct()


# Cleaning and categorizing agency_project_description
project_status_clean <- project_status_clean %>%
  mutate(
    project_theme = case_when(
      # === EXISTING CATEGORIES (KEEP UNCHANGED) ===
      str_detect(agency_project_description, regex("ROOF|ROOFING|PARAPET|FACADE|BULKHEAD|ENVELOPE|CLADDI
      str_detect(agency_project_description, regex("ELEVATOR|LIFT|MODERNIZATION", ignore_case = TRUE)) ~
      str_detect(agency_project_description, regex("ADA |ACCESSIBILITY|ADA COMPLIANCE|ADA REQUIREMENT|in
      str_detect(agency_project_description, regex("RENOVATION|REHABILITATION|BUILDOUT|RESTACKING|UPGRAI
      str_detect(agency_project_description, regex("BATHROOM", ignore_case = TRUE)) ~ "Bathroom Work",
      str_detect(agency_project_description, regex("SAFETY|STREET|SIDEWALK|RAMP", ignore_case = TRUE)) ~
      str_detect(agency_project_description, regex("LIBRARY", ignore_case = TRUE)) ~ "Library Work",
      str_detect(agency_project_description, regex("RELOCATION|RELOCATE|MOVE", ignore_case = TRUE)) ~ "L
      str_detect(agency_project_description, regex("EMERGENCY|REPAIR|REPLACEMENT|INSULATION", ignore_cas
      str_detect(agency_project_description, regex("PRECINCT|FLEET|LOCKER|POLICE|FIRE|VEHICLE| TOW POUNI
      str_detect(agency_project_description, regex("BRIDGE|VIADUCT|OVERPASS| Over ", ignore_case = TRUE)
      str_detect(agency_project_description, regex("DAM|RESERVOIR", ignore_case = TRUE)) ~ "Dam/Reservoi
      str_detect(agency_project_description, regex("SEWER|DRAIN|STORMWATER|SANITARY|WATER MAIN", ignore_
      str_detect(agency_project_description, regex("PARK|PLAYGROUND|RECREATION|FIELD|GREENWAY", ignore_c
```

```
      str_detect(agency_project_description, regex("FERRY|TERMINAL|MARINE|DOCK|BARGE|VESSEL", ignore_cas
      str_detect(agency_project_description, regex("HVAC|MECHANICAL|VENTILATION|BMS|BOILER|CHILLER", ign
      str_detect(agency_project_description, regex("SOLAR|SUSTAINABLE|GREEN INFRASTRUCTURE|STORMWATER MA
      str_detect(agency_project_description, regex("NEW CONSTRUCTION|NEW BUILDING|EXPANSION", ignore_cas

      # === NEW CATEGORIES (FOR UNMATCHED PROJECTS) ===
      str_detect(agency_project_description, regex("MUSEUM|ZOO|AQUARIUM|BOTANICAL GARDEN|CULTURAL CENTER
      str_detect(agency_project_description, regex("SHELTER|TRANSITIONAL HOUSING|FAMILY RESIDENCE|HOMELE
      str_detect(agency_project_description, regex("FLOOD PROTECTION|MITIGATION|BULKHEAD|LEVEE|SHORELINE
      str_detect(agency_project_description, regex("GENERATOR|ELECTRICAL|POWER DISTRIBUTION|TRANSFORMER
      str_detect(agency_project_description, regex("TRIAL COURT|COURTHOUSE|COURTROOM|DA |LAW DEPT|OCA",
      str_detect(agency_project_description, regex("LANDMARK|MEMORIAL|RESTORE|RESTORATION|HISTORIC|ARCH
      str_detect(agency_project_description, regex("PUBLIC ART|PERCENT FOR ART|ART INSTALLATION", ignore
      str_detect(agency_project_description, regex("TUNNEL|SHAFT|CSO|STORAGE|UNDERGROUND|CONNECTION CHA
      str_detect(agency_project_description, regex("TREE|REFORESTATION|PLANTING|HORTICULTURE|GARDEN", ig
      str_detect(agency_project_description, regex("SCHOOL|EDUCATION BUILDING|CLASSROOM|TEACHING|LAB|CU
str_detect(agency_project_description, regex("LOBBY|FLOOR|SPACE|INFRASTRUCTURE|BUILDING SYSTEMS", ignore
str_detect(agency_project_description, regex("PUMP|SLUDGE|SEWAGE|THICKENING", ignore_case = TRUE)) ~ "Wa
str_detect(agency_project_description, regex("FARM|GARDEN|WATER SERVICE|IRRIGATION", ignore_case = TRUE)

      # Default
      TRUE ~ "Unknown"
    )
  )


# Cleaning of delay_desc
project_status_clean <- project_status_clean %>%
  mutate(
    delay_category = case_when(
      str_detect(delay_desc, regex("BUDGETARY CONSTRAINTS|NON-CITY GRANT APPROVAL", ignore_case = TRUE))
      str_detect(delay_desc, regex("CHANGES IN SCOPE/DESIGN", ignore_case = TRUE)) ~ "Scope or Design Ch
      str_detect(delay_desc, regex("SCHEDULING OF UTILITY WORK|UNAVAILABILITY OF PRODUCT|RELEASE OF NEW
      str_detect(delay_desc, regex("UNFORESEEN HAZARDOUS CONDITION|UNFORESEEN SITE/FIELD CONDITION", ign
      str_detect(delay_desc, regex("PENDING APPROVAL OF NECESSARY PERMITS|STATE REQ CONTRACT", ignore_ca
      str_detect(delay_desc, regex("LEGAL ISSUES", ignore_case = TRUE)) ~ "Legal/Contractual Issues",
      str_detect(delay_desc, regex("CONTRACTOR DEFAULT", ignore_case = TRUE)) ~ "Contractor Issues",
      TRUE ~ "Other/Unknown"
    )
  )


# Cleaning ten_year_plan_category
project_status_clean <- project_status_clean %>%
  mutate(
    category_group = case_when(
      str_detect(ten_year_plan_category, regex("PARK|RECREATION|PLAYGROUND|BOARDWALK|ZOOS|FAIR BRIDGES|T
      str_detect(ten_year_plan_category, regex("WATER|TUNNEL|MAIN REPLACEMENT|PLANT|FILTER|CITY TUNNEL|M
      str_detect(ten_year_plan_category, regex("SHELTER|HOUSING|HOMELESS|LOW TO MODERATE INCOME|PUBLIC H
      str_detect(ten_year_plan_category, regex("SIDEWALK|RAMP|HIGHWAY|FERRY|STREET|BRIDGE", ignore_case
      str_detect(ten_year_plan_category, regex("POLICE|COURT|FACILITIES|ADMIN|OFFICE|GARAGE", ignore_cas
      str_detect(ten_year_plan_category, regex("SUSTAINABILITY|GREEN INFRASTRUCTURE|ENVIRONMENT|WATER PL
      TRUE ~ "Miscellaneous / Other"
```

```
    )
  )



# Inspecting the clean data
write_csv(project_status_clean, "project_status_clean.csv")
```

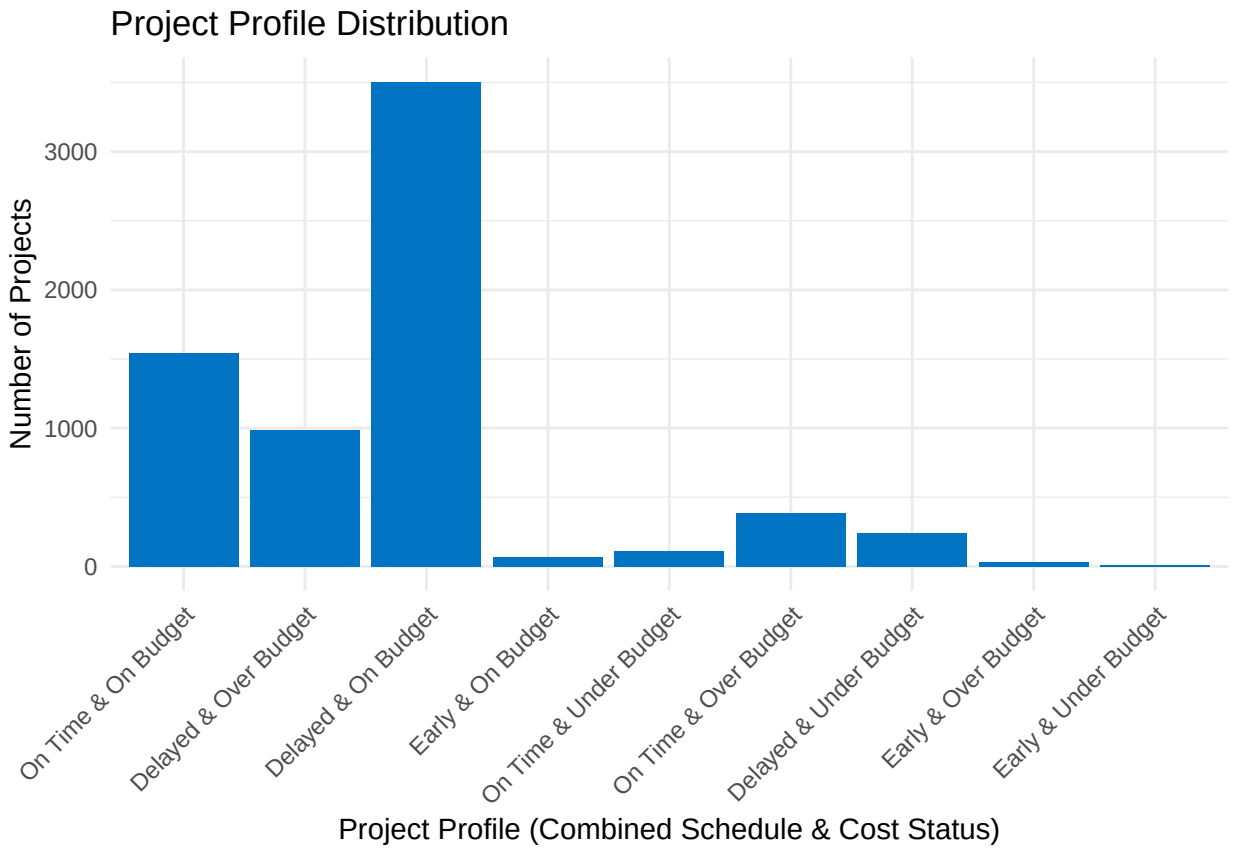
3.4 Summaise table of Project distibution

```
table(project_status_clean$delay_class, project_status_clean$cost_class)
```

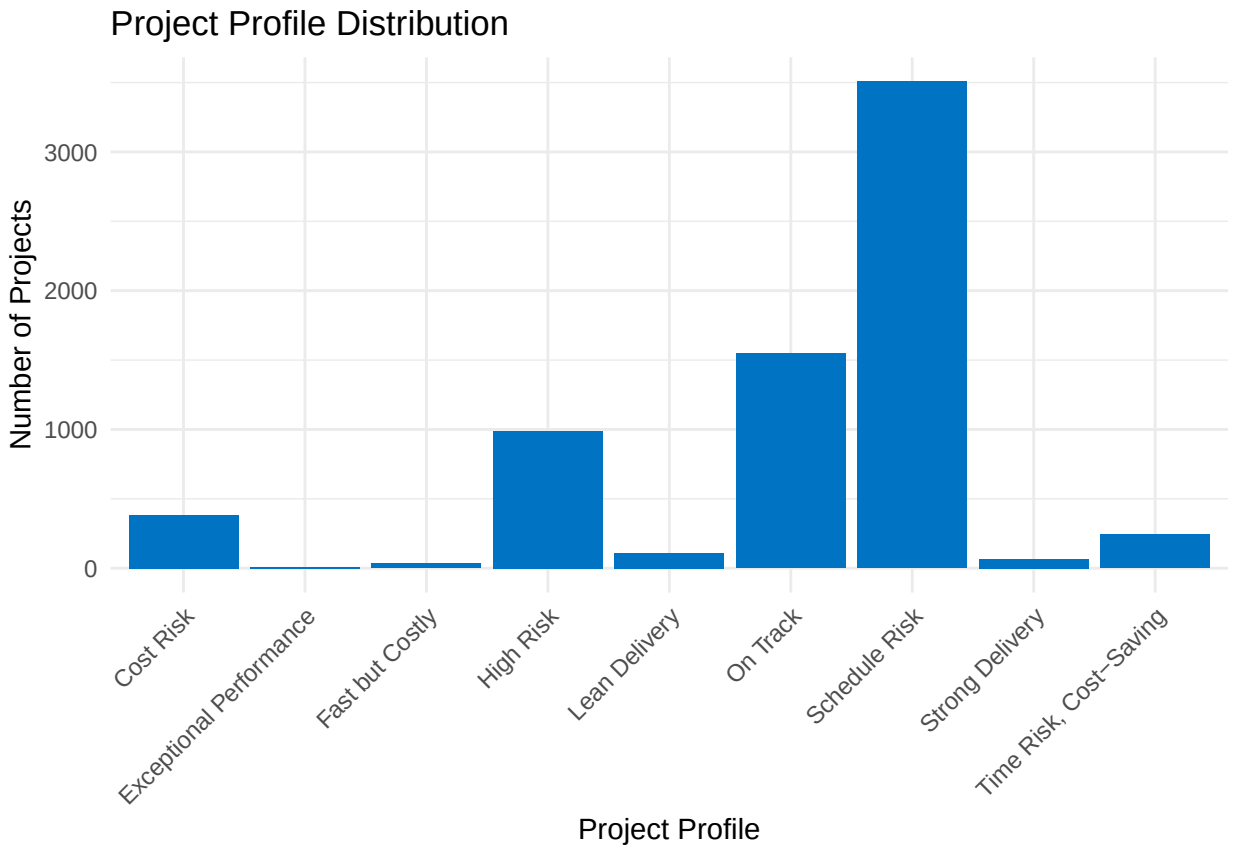
```
##
##           On Budget Over Budget Under Budget
##   Delayed      3506         989          242
##   Early          67          33           10
##   On Time      1546         384          109
```

3.5 EDA by current_phase

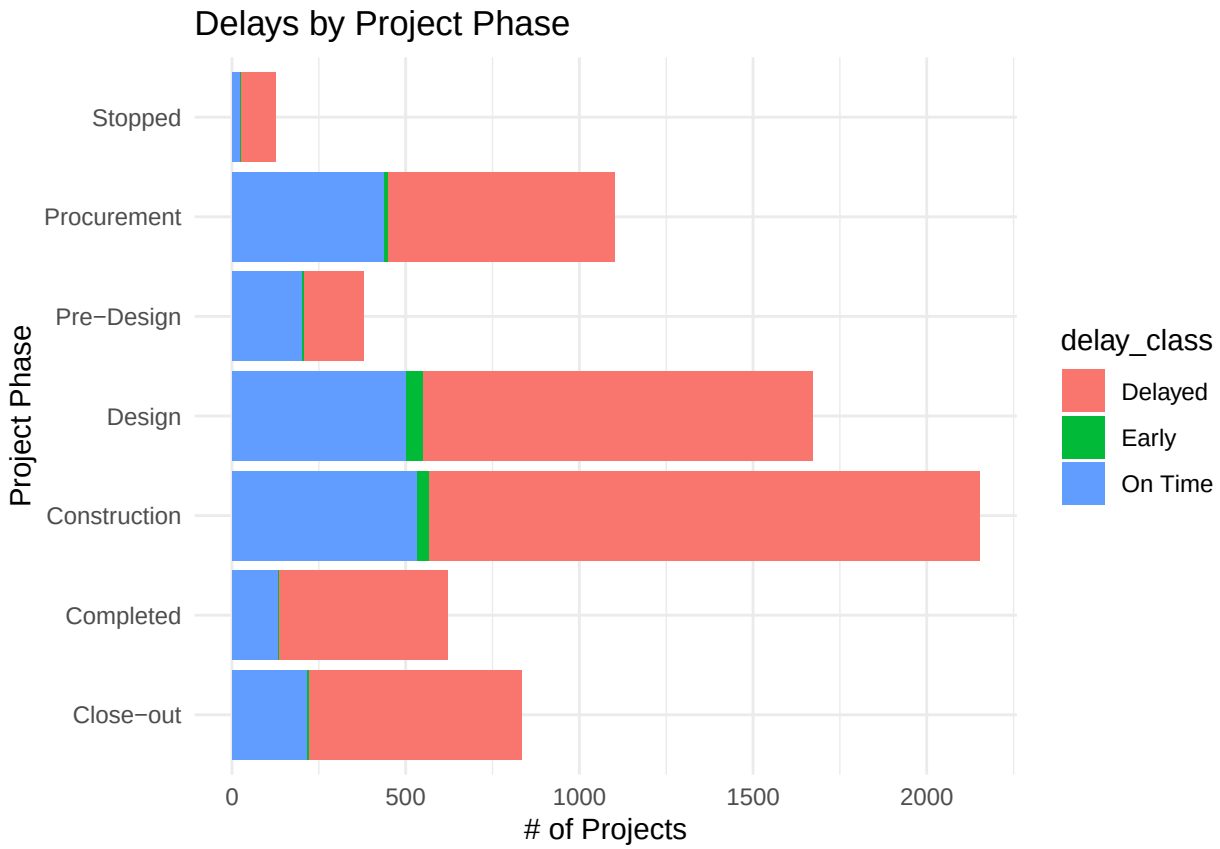
```
# Reorder factor levels by frequency
project_status_clean <- project_status_clean %>%
  mutate(status_combined = fct_infreq(status_combined))

# Bar plot with ordered categories
ggplot(project_status_clean, aes(x = status_combined)) +
  geom_bar(fill = "#0073C2FF") +
  labs(title = "Project Profile Distribution",
       x = "Project Profile (Combined Schedule & Cost Status)",
       y = "Number of Projects") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



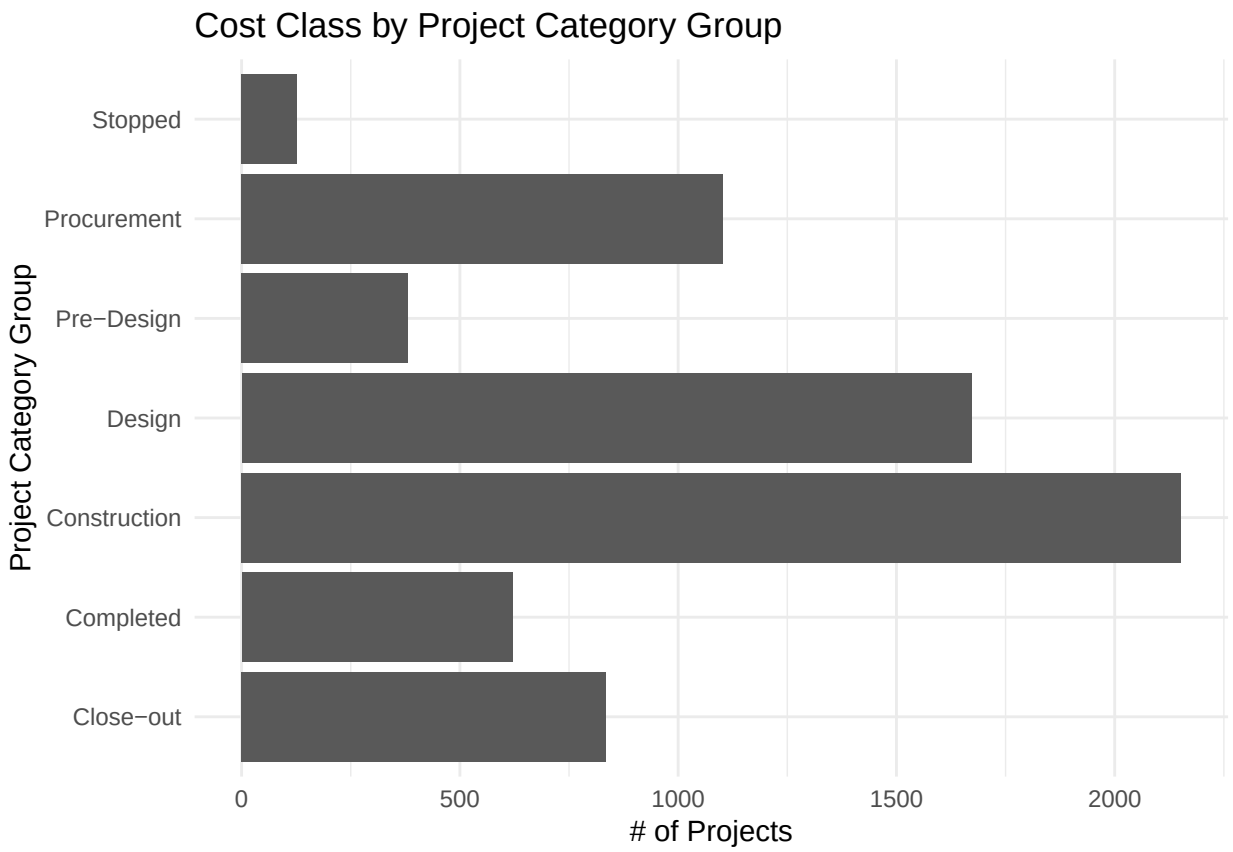

```r
# Project Profile Distribution
ggplot(project_status_clean, aes(x = project_profile)) +
  geom_bar(fill = "#0073C2FF") +
  labs(title = "Project Profile Distribution",
       x = "Project Profile",
       y = "Number of Projects") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# 2. Add High Risk flag
#project_status_clean <- project_status_clean %>%
#  mutate(
#    high_risk = ifelse(delay_class == "Delayed" & cost_class == "Over Budget", "Yes", "No")
#  )

# 3. Summary of high risk projects
#table(project_status_clean$high_risk)


## Goal: Understand how delays and cost overruns relate to project lifecycle stage.
project_status_clean %>%
  count(current_phase, delay_class) %>%
  ggplot(aes(x = current_phase, y = n, fill = delay_class)) +
  geom_col(position = "stack") +
  labs(title = "Delays by Project Phase", x = "Project Phase", y = "# of Projects") +
  #theme(axis.text.x = element_text(angle = 45, hjust = 1))
  coord_flip() +
  theme_minimal()
```

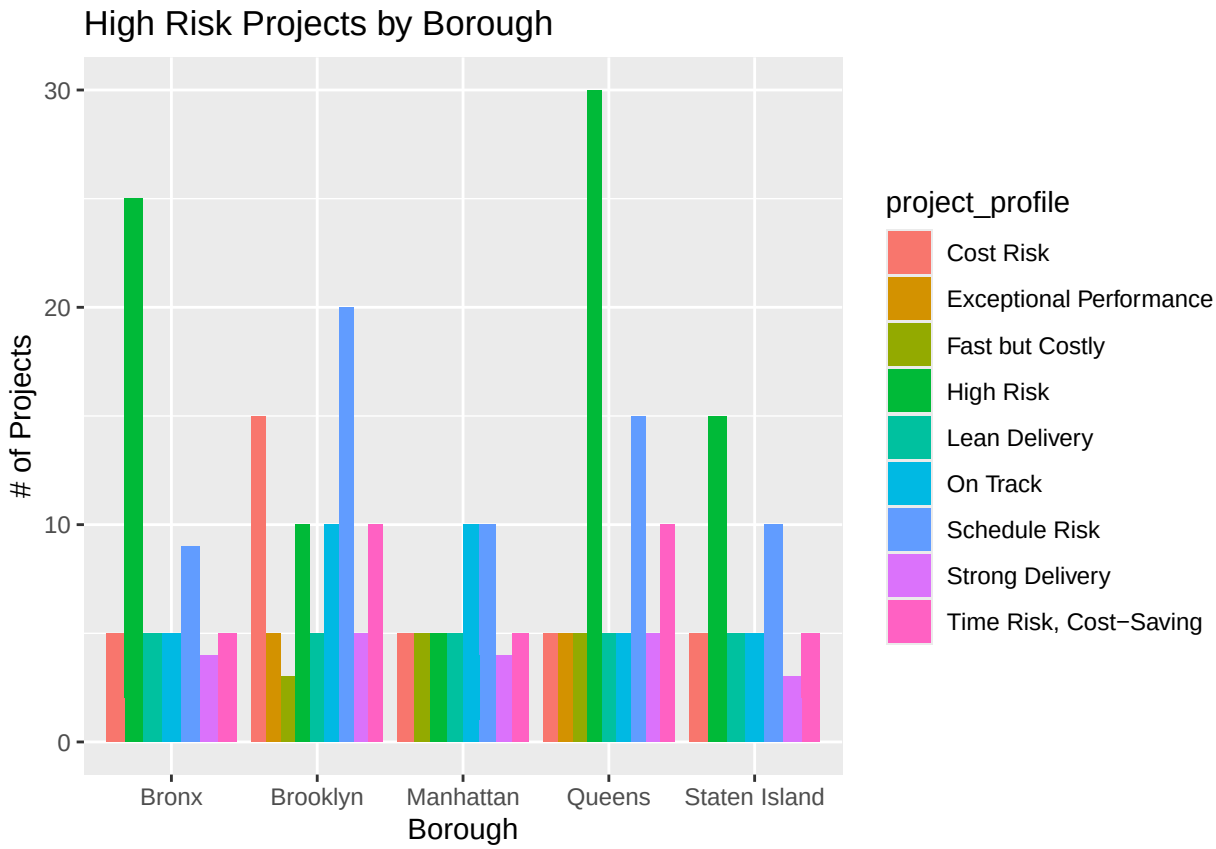



```
project_status_clean %>%
  count(current_phase, cost_class) %>%
  ggplot(aes(x = current_phase, y = n, fill = NULL)) +
  geom_col(position = "stack") +
  labs(title = "Cost Class by Project Category Group",
       x = "Project Category Group",
       y = "# of Projects") +
  coord_flip() +
  theme_minimal()
```



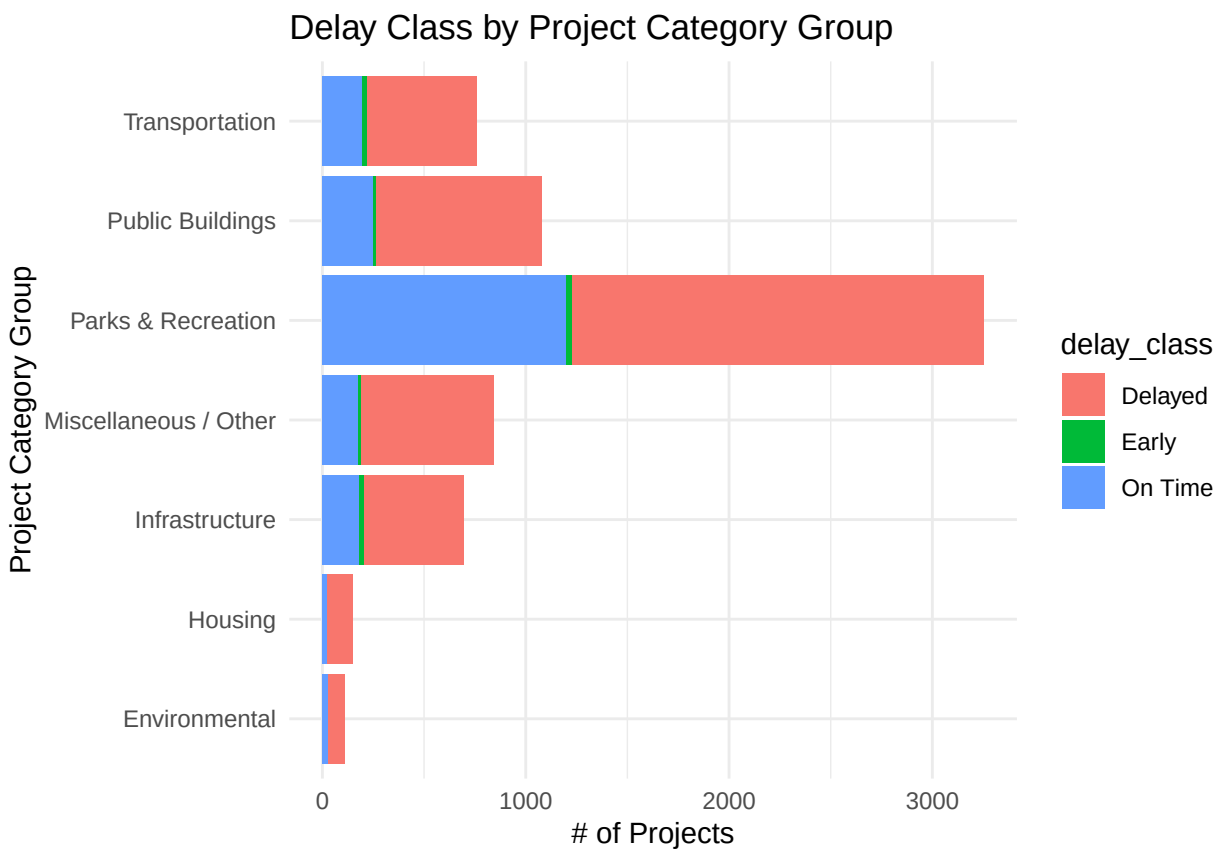
3.6 EDA by borough

```
## Goal: See if spatial patterns exist in delays or costs.
project_status_clean %>%
  filter(borough != "Citywide") %>%
  count(borough, project_profile) %>%
  ggplot(aes(x = borough, y = n, fill = project_profile)) +
  geom_col(position = "dodge") +
  labs(title = "High Risk Projects by Borough", x = "Borough", y = "# of Projects")
```

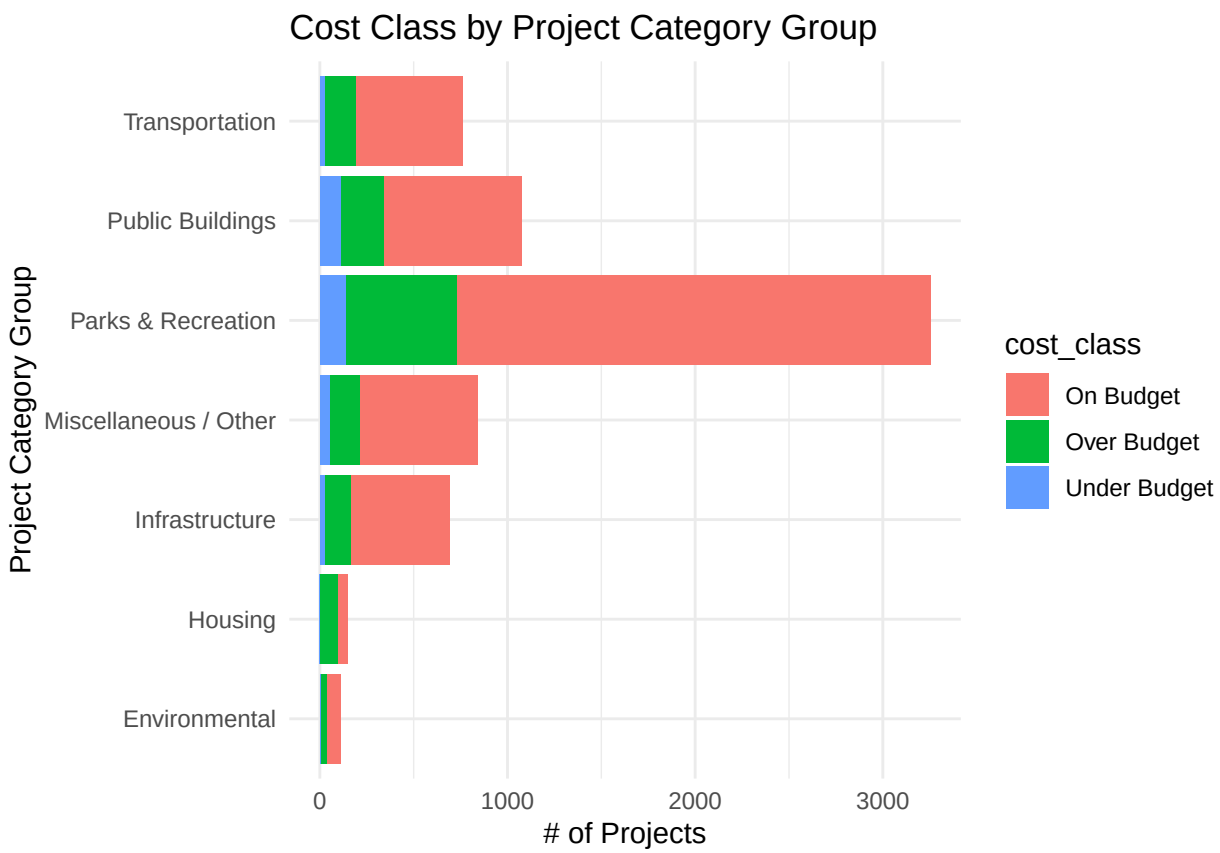


3.7 EDA by "ten_year_plan_category", "Cost Class" and by "high_risk Flag"

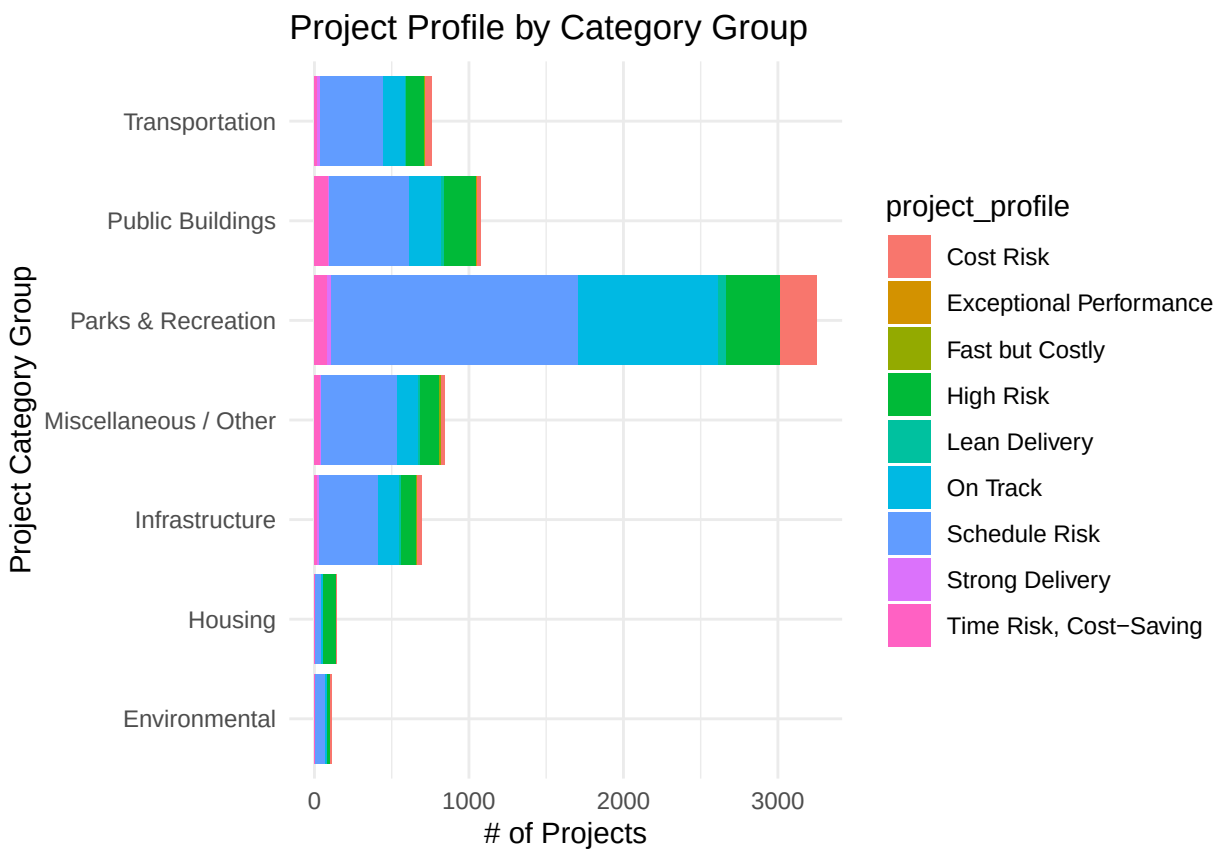
```
library(dplyr)
library(ggplot2)
library(stringr)


# EDA: Delay class by category group
project_status_clean %>%
  count(category_group, delay_class) %>%
  ggplot(aes(x = category_group, y = n, fill = delay_class)) +
  geom_col(position = "stack") +
  labs(title = "Delay Class by Project Category Group",
       x = "Project Category Group",
       y = "# of Projects") +
  coord_flip() +
  theme_minimal()
```



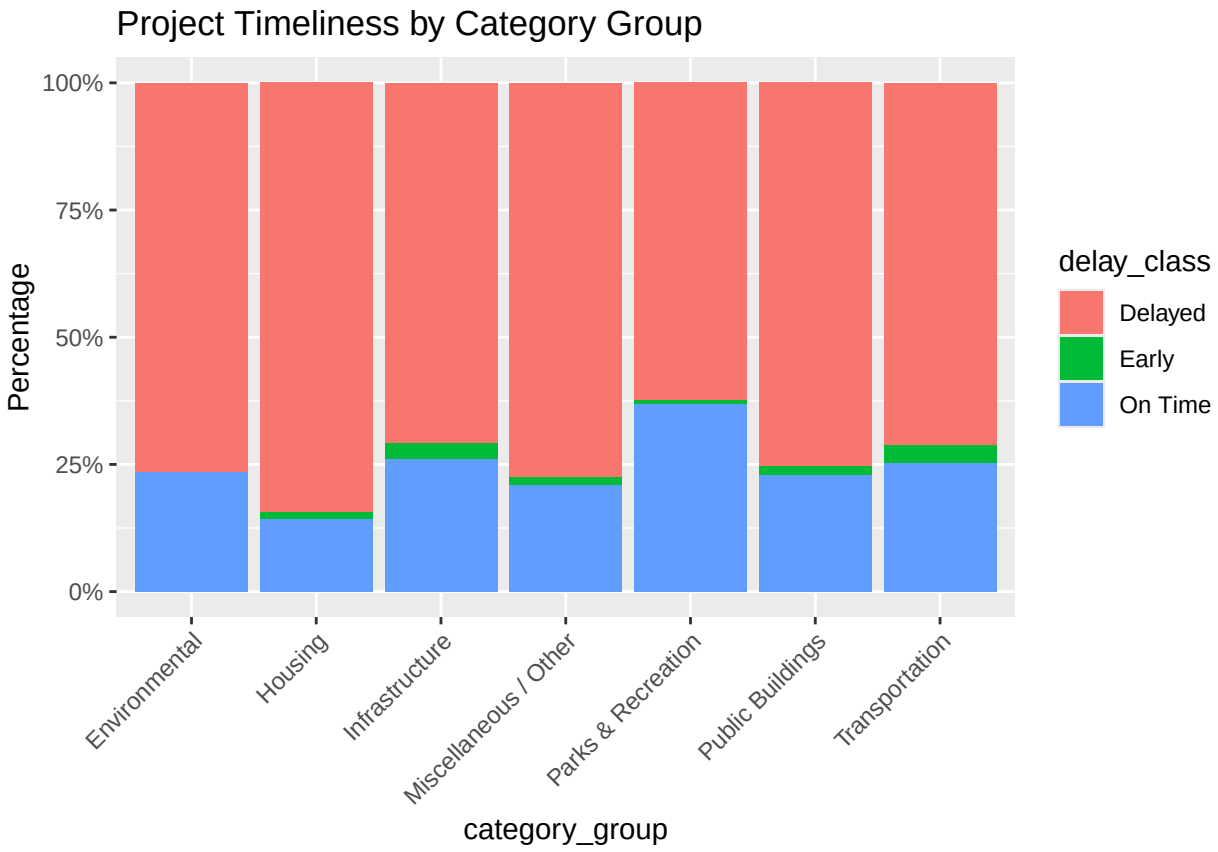

```
# EDA: Cost Class
project_status_clean %>%
  count(category_group, cost_class) %>%
  ggplot(aes(x = category_group, y = n, fill = cost_class)) +
  geom_col(position = "stack") +
  labs(title = "Cost Class by Project Category Group",
       x = "Project Category Group",
       y = "# of Projects") +
  coord_flip() +
  theme_minimal()
```




```r
# EDA: high_risk Flag
project_status_clean %>%
  count(category_group, project_profile) %>%
  ggplot(aes(x = category_group, y = n, fill = project_profile)) +
  geom_col(position = "stack") +
  labs(title = "Project Profile by Category Group",
       x = "Project Category Group",
       y = "# of Projects") +
  coord_flip() +
  theme_minimal()
```




```
# Using percentage delay class in category group
project_status_clean %>%
  group_by(category_group) %>%
  count(delay_class) %>%
  mutate(percentage = n / sum(n) * 100) %>%
  ggplot(aes(x = category_group, y = percentage, fill = delay_class)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(y = "Percentage", title = "Project Timeliness by Category Group") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



3.8 Other EDA 1: Delay class percentage by project category group

```
library(tidyverse)
```

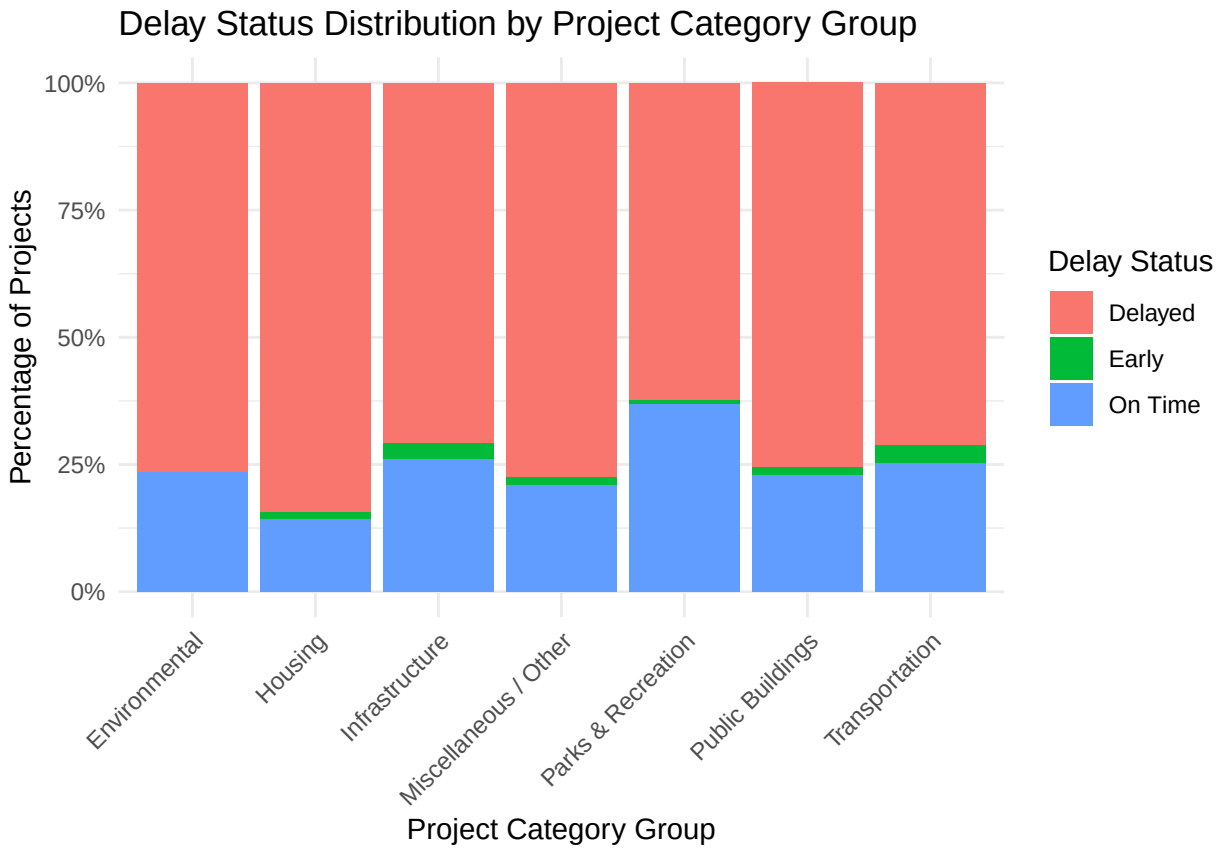
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v purrr  1.0.4     v tidyr  1.3.1
## v tibble 3.2.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

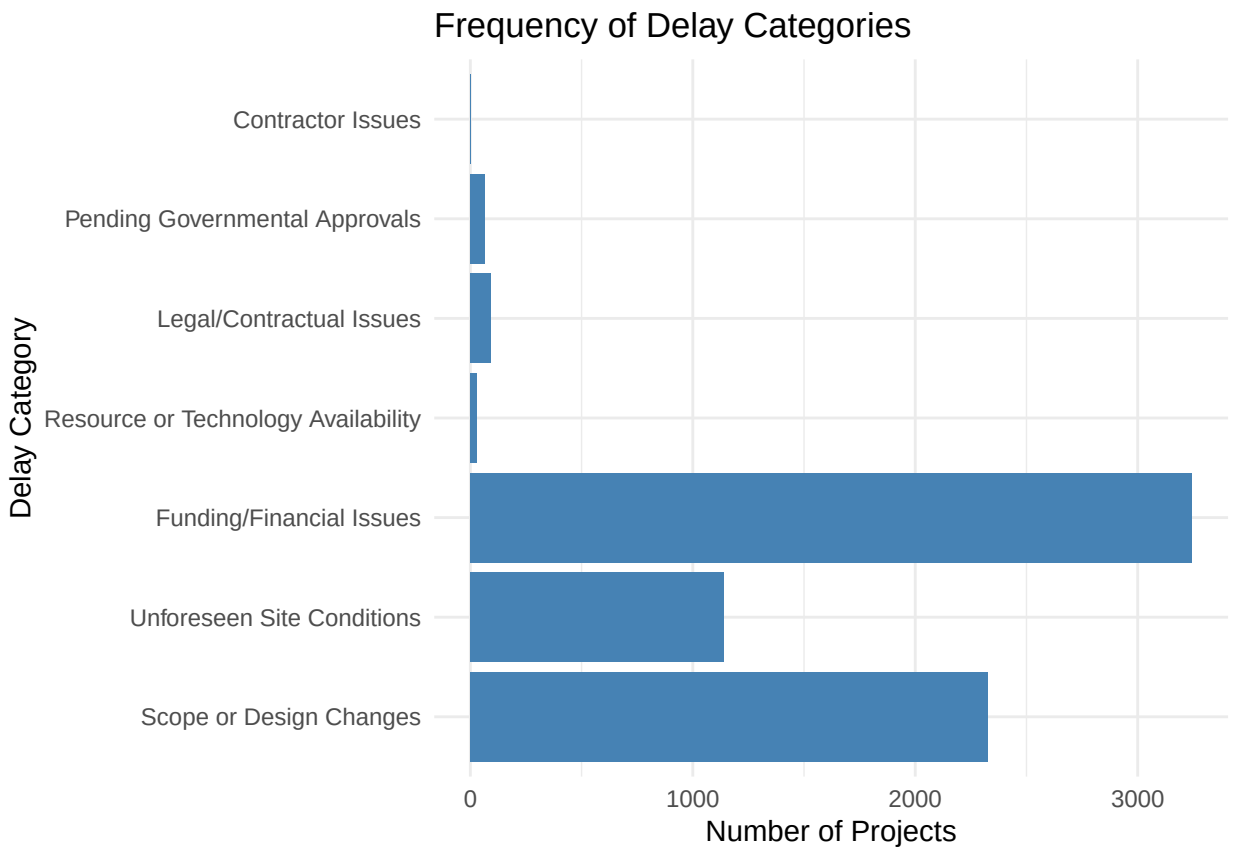
```
library(scales)
```

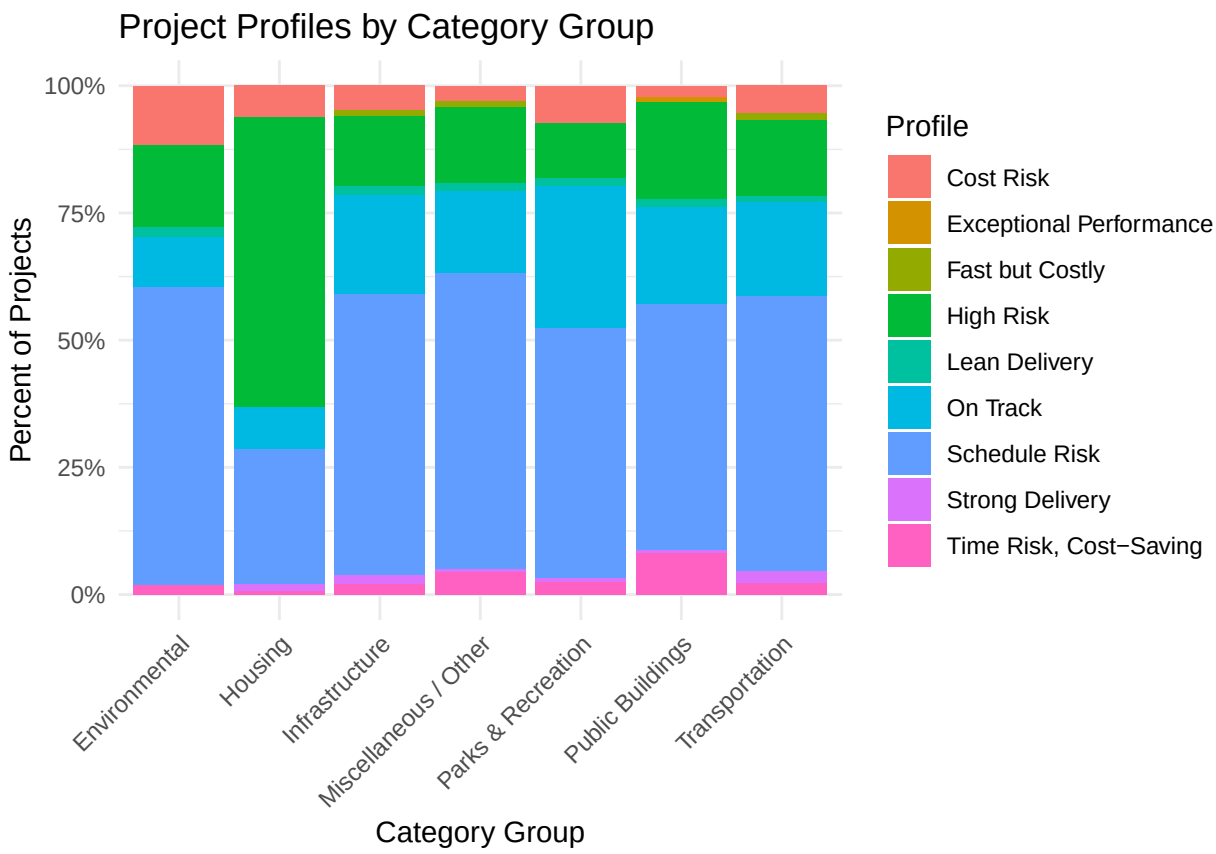
```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
project_status_clean %>%
  group_by(category_group, delay_class) %>%
  summarise(count = n(), .groups = "drop") %>%
```

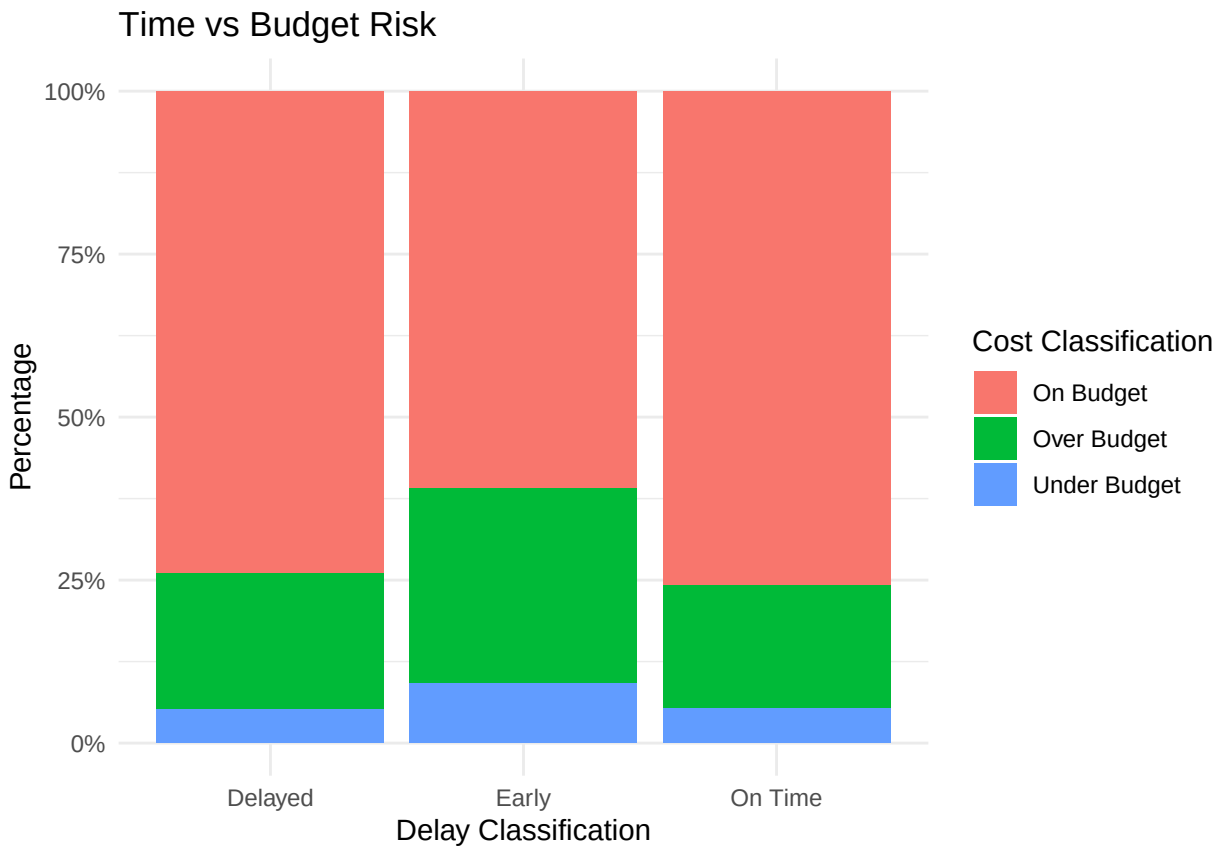
```r
  group_by(category_group) %>%
  mutate(pct = count / sum(count)) %>%
  ggplot(aes(x = category_group, y = pct, fill = delay_class)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = percent_format()) +
  labs(
    title = "Delay Status Distribution by Project Category Group",
    x = "Project Category Group",
    y = "Percentage of Projects",
    fill = "Delay Status"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



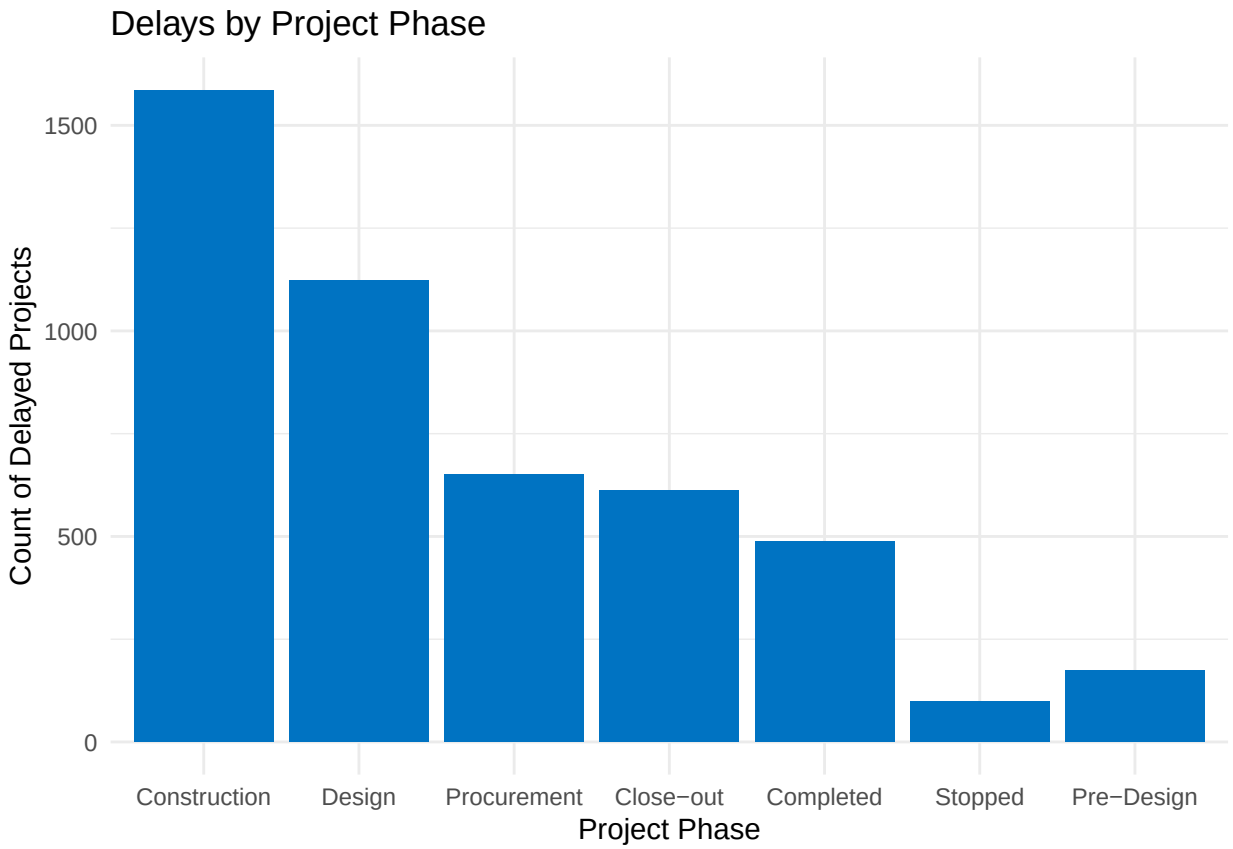

Frequency of Delay Categories
Contractor Issues
Pending Governmental Approvals
Legal/Contractual Issues
Resource or Technology Availability
Funding/Financial Issues
Unforeseen Site Conditions
Scope or Design Changes
Delay Category
0
1000
2000
3000
Number of Projects


Project Profiles by Category Group
100%
75%
50%
25%
0%
Percent of Projects
Environmental
Housing
Infrastructure
Miscellaneous / Other
Parks & Recreation
Public Buildings
Transportation
Category Group
Profile
Cost Risk
Exceptional Performance
Fast but Costly
High Risk
Lean Delivery
On Track
Schedule Risk
Strong Delivery
Time Risk, Cost−Saving

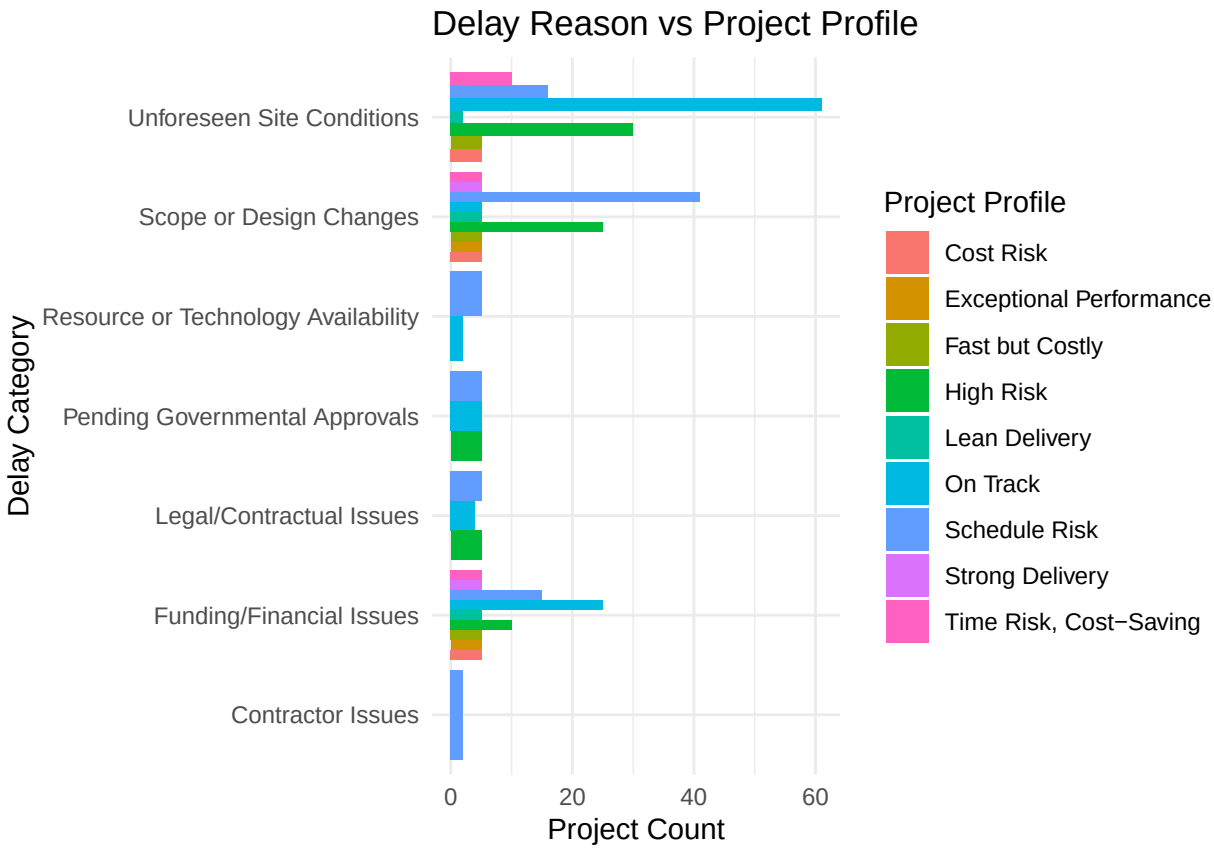
```r
### 1. Time vs Budget Risk: Delay Class vs Cost Class
library(ggplot2)

ggplot(project_status_clean, aes(x = delay_class, fill = cost_class)) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "Time vs Budget Risk",
       x = "Delay Classification",
       y = "Percentage",
       fill = "Cost Classification") +
  theme_minimal()
```




```r
### 2. Delay by Current Phase
project_status_clean %>%
  filter(delay_class == "Delayed") %>%
  count(current_phase) %>%
  ggplot(aes(x = reorder(current_phase, -n), y = n)) +
  geom_col(fill = "#0073C2FF") +
  #coord_flip() +
  labs(title = "Delays by Project Phase",
       x = "Project Phase",
       y = "Count of Delayed Projects") +
  theme_minimal()
```




```
### 3. Delay Category vs Project Profile
library(dplyr)

project_status_clean %>%
  count(delay_category, project_profile) %>%
  ggplot(aes(x = delay_category, y = n, fill = project_profile)) +
  geom_col(position = "dodge") +
  coord_flip() +
  labs(title = "Delay Reason vs Project Profile",
       x = "Delay Category",
       y = "Project Count",
       fill = "Project Profile") +
  theme_minimal()
```



4. Modeling step

4.1 Focus on specific columns for modeling

```
# Exclude specific columns from the dataset "project_status_clean"
# Updated columns to exclude: "agency_project_description", "ten_year_plan_category", "delay_desc", and

project_model_data <- project_status_clean %>%
  select(-c(agency_project_description, ten_year_plan_category, delay_desc, scope_text))

# Inspecting the clean model data
write_csv(project_model_data, "project_model_data.csv")
```

4.2 Adding Weather data

```
# Load temperature anomaly dataset
temp_data <- read.csv("temperature_data.csv")

temp_data <- temp_data %>%
  rename(SEASON = Year) %>%
  mutate(SEASON = as.numeric(SEASON))
#-------------------------------------------------------------------------------

# Storm data
storm_data <- read.csv("ibtracs.ALL.list.v04r01.csv")
```

```r
# Data Cleaning
storm_data <- storm_data[-1, ]
storm_data$SEASON <- as.numeric(storm_data$SEASON)
storm_data <- storm_data %>% filter(SEASON >= 1850 & SEASON <= 2024)

# Calculate yearly storm frequency
storm_frequency <- storm_data %>%
  group_by(SEASON) %>%
  summarise(Number_of_Storms = n(), .groups = "drop")
#-------------------------------------------------------------------------------

# Merge and calculate yearly metrics
storm_correlation <- storm_data %>%
  group_by(SEASON) %>%
  summarise(Cyclone_Frequency = n(), .groups = "drop") %>%
  left_join(temp_data, by = "SEASON")
#-------------------------------------------------------------------------------

# First, rename 'Anomaly' in both datasets before merging
storm_correlation <- storm_correlation %>%
  rename(Temp_Anomaly = Anomaly)

# Merge storm and temperature data
storm_intensity <- storm_data %>%
  filter(SEASON != 'Year') %>%
  group_by(SEASON) %>%
  summarise(Max_Wind_Speed = max(as.numeric(USA_WIND), na.rm = TRUE), .groups = "drop") %>%
  left_join(temp_data, by = "SEASON")


storm_intensity <- storm_intensity %>%
  select(-Anomaly)  # Remove to avoid duplication after join

# Merge using SEASON as the key
weather_data <- left_join(storm_correlation, storm_intensity, by = "SEASON") %>%
  rename(year = SEASON)

write_csv(weather_data, "weather_data.csv")
```

4.3 Addign Labor Data

```r
# Load necessary libraries
library(dplyr)
library(readr)

# Load the datasets
construction_jobs <- read_csv("labor_data_construction_job.csv") %>%
  mutate(construction_job = as.numeric(gsub(",", "", construction_job)))
```

```
## Rows: 24 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (1): year
```

```
## num (1): construction_job
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
unemployment_rate <- read_csv("labor_data_unemployement_rate.csv")
```

```
## Rows: 156 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): borough
## dbl (2): year, unemployment_rate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
population <- read_csv("nyc_borough_population.csv")
```

```
## Rows: 125 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): borough
## dbl (2): year, borough_population
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#-------------------------------------------------------------------------------

# Calculate total NYC population per year
population_totals <- population %>%
  group_by(year) %>%
  summarise(total_population = sum(borough_population, na.rm = TRUE))
#-------------------------------------------------------------------------------

# Merge population and totals to get population share
population_with_share <- population %>%
  left_join(population_totals, by = "year") %>%
  mutate(population_share = borough_population / total_population)
#-------------------------------------------------------------------------------



# Merge with citywide construction job data
labor_data <- population_with_share %>%
  left_join(construction_jobs, by = "year") %>%
  mutate(construction_jobs = round(population_share * construction_job))
#-------------------------------------------------------------------------------

# Merge with unemployment data
labor_data_final <- labor_data %>%
  select(year, borough, construction_jobs) %>%
  left_join(unemployment_rate, by = c("year", "borough")) %>%
  rename(labor_unemp_rate = unemployment_rate)

# View or export the final dataset
#print(labor_data_final)
```

```r
write_csv(labor_data_final, "final_labor_data.csv")
```

4.4 Combining weather and labor data

```r
weather_and_labor_data <- left_join(labor_data_final, weather_data, by = "year")

# Compute Citywide averages for each year
citywide_averages <- weather_and_labor_data %>%
  group_by(year) %>%
  summarise(
    construction_jobs = mean(construction_jobs, na.rm = TRUE),
    labor_unemp_rate = mean(labor_unemp_rate, na.rm = TRUE),
    Cyclone_Frequency = mean(Cyclone_Frequency, na.rm = TRUE),
    Temp_Anomaly = mean(Temp_Anomaly, na.rm = TRUE),
    Max_Wind_Speed = mean(Max_Wind_Speed, na.rm = TRUE)
  ) %>%
  mutate(borough = "Citywide") %>%
  select(year, borough, everything())  # reorder columns to match original

# Bind Citywide rows to original dataset
weather_and_labor_data <- bind_rows(weather_and_labor_data, citywide_averages)

# Checking the weather and labor merged data
write_csv(weather_and_labor_data, "weather_and_labor_data.csv")
```

4.5 Combining project model data with weather+labor data

```r
# Load required libraries
library(dplyr)
library(readr)
library(lubridate)
library(purrr)

## Taking average weather_and_labor_data values per period of the project phase (e.g. 2001-2005)
# Load datasets
project_data <- read_csv("project_model_data.csv")
```

```
## Rows: 6886 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (10): fms_id, cost_class, delay_class, status_combined, borough, curren...
## dbl   (7): initial_budget, latest_budget, latest_spend, cost_diff, cost_diff...
## date  (3): orig_start_date, orig_end_date, task_end_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
weather_labor_data <- read_csv("weather_and_labor_data.csv")
```

```
## Rows: 150 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): borough
```

```
## dbl (6): year, construction_jobs, labor_unemp_rate, Cyclone_Frequency, Temp_...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Extract start and end years
project_data <- project_data %>%
  mutate(
    orig_start_date = ymd(orig_start_date),
    task_end_date = ymd(task_end_date),
    start_year = year(orig_start_date),
    end_year = year(task_end_date)
  )

# Define a function to calculate averages by year range and borough
get_avg_weather_labor <- function(start_year, end_year, borough) {
  subset <- weather_labor_data %>%
    filter(
      tolower(borough) == tolower(!!borough),
      year >= start_year,
      year <= end_year
    )

  if (nrow(subset) == 0) {
    return(tibble(
      avg_construction_jobs = NA_real_,
      avg_labor_unemp_rate = NA_real_,
      avg_cyclone_freq = NA_real_,
      avg_temp_anomaly = NA_real_,
      avg_max_wind_speed = NA_real_
    ))
  }

  return(subset %>%
           summarise(
             avg_construction_jobs = mean(construction_jobs, na.rm = TRUE),
             avg_labor_unemp_rate = mean(labor_unemp_rate, na.rm = TRUE),
             avg_cyclone_freq = mean(Cyclone_Frequency, na.rm = TRUE),
             avg_temp_anomaly = mean(Temp_Anomaly, na.rm = TRUE),
             avg_max_wind_speed = mean(Max_Wind_Speed, na.rm = TRUE)
           ))
}

# Apply the function rowwise to the project data
averaged_weather_labor <- project_data %>%
  mutate(row_id = row_number()) %>%
  group_split(row_id) %>%
  map_dfr(~ bind_cols(.x, get_avg_weather_labor(.x$start_year, .x$end_year, .x$borough)))

# Final enriched dataset
project_model_data_final <- averaged_weather_labor %>%
  select(-row_id)
##------------------------------------------------------------------------------------------------
```

```r
## Mean Imputation for missing averaged_weather_labor values in project_model_data_final
project_model_data_final <- project_model_data_final %>%
  mutate(
    avg_construction_jobs = ifelse(is.na(avg_construction_jobs), mean(avg_construction_jobs, na.rm = TRU
    avg_labor_unemp_rate = ifelse(is.na(avg_labor_unemp_rate), mean(avg_labor_unemp_rate, na.rm = TRUE)
    avg_cyclone_freq = ifelse(is.na(avg_cyclone_freq), mean(avg_cyclone_freq, na.rm = TRUE), avg_cyclon
    avg_temp_anomaly = ifelse(is.na(avg_temp_anomaly), mean(avg_temp_anomaly, na.rm = TRUE), avg_temp_a
    avg_max_wind_speed = ifelse(is.na(avg_max_wind_speed), mean(avg_max_wind_speed, na.rm = TRUE), avg_
  )
##-------------------------------------------------------------------------------------------------

# Replacing ~6% or data being project_theme = Unknown to the most frequent in the borough
# Impute "Unknown" values in project_theme using the most frequent theme in each borough
library(dplyr)

project_model_data_final <- project_model_data_final %>%
  group_by(borough) %>%
  mutate(project_theme = if_else(
    project_theme == "Unknown",
    names(which.max(table(project_theme))),
    project_theme
  )) %>%
  ungroup()


# View(project_model_data_final)
write_csv(project_model_data_final, "project_model_data_final.csv")
```

4.6 Updated Exploratory Data Summary and Visualization

4.6.1 Descriptive Statistics

```r
# LIbraries
library(tidyverse)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```r
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
```

```
##     group_rows
```

```r
project_model_data_final_2 <- project_model_data_final %>%
  select(-c(fms_id, orig_start_date, orig_end_date, task_end_date, end_year, start_year))

summary_table <- project_model_data_final_2 %>%
  select_if(is.numeric) %>%
  summary() %>%
  as.data.frame()
kable(summary_table, caption = "Descriptive Statistics of Numeric Variables") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"), full_width = FALSE)
```

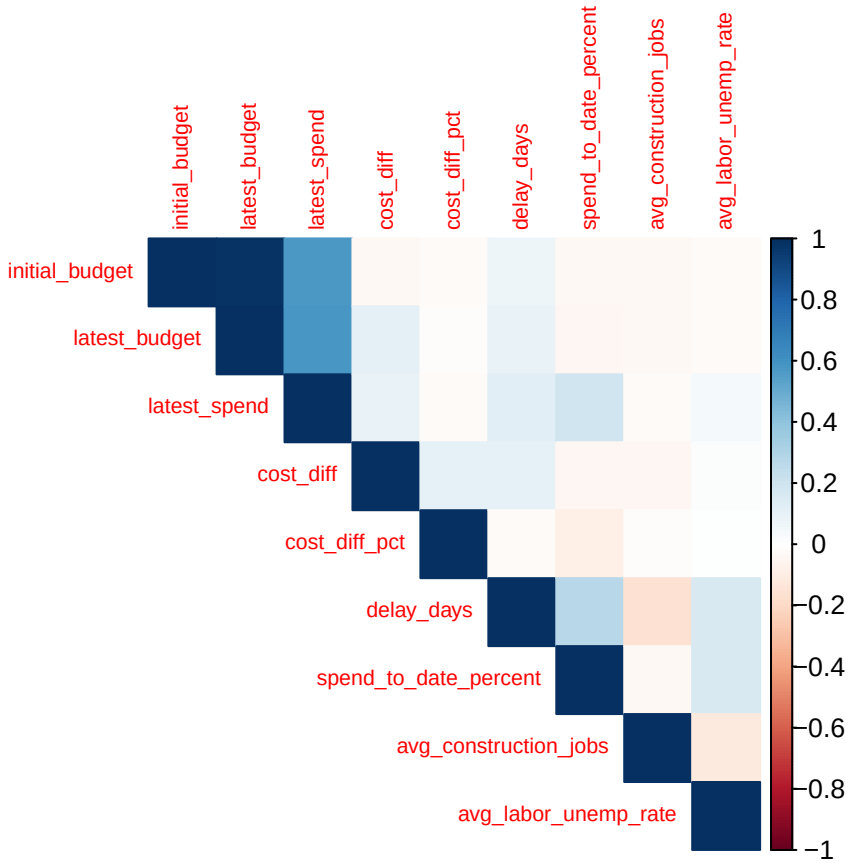
Table 1: Descriptive Statistics of Numeric Variables

Var1	Var2	Freq
	initial_budget	Min. :1.100e+04
	initial_budget	1st Qu.:2.888e+06
	initial_budget	Median :7.210e+06
	initial_budget	Mean :3.913e+07
	initial_budget	3rd Qu.:2.170e+07
	initial_budget	Max. :1.955e+09
	latest_budget	Min. :0.000e+00
	latest_budget	1st Qu.:3.040e+06
	latest_budget	Median :7.691e+06
	latest_budget	Mean :4.057e+07
	latest_budget	3rd Qu.:2.351e+07
	latest_budget	Max. :1.909e+09
	latest_spend	Min. : 0
	latest_spend	1st Qu.: 393800
	latest_spend	Median : 1422821
	latest_spend	Mean : 10879497
	latest_spend	3rd Qu.: 4651588
	latest_spend	Max. :991618354
	cost_diff	Min. :-673968125
	cost_diff	1st Qu.: 0
	cost_diff	Median : 160151
	cost_diff	Mean : 1438560
	cost_diff	3rd Qu.: 873953
	cost_diff	Max. : 418029490
	cost_diff_pct	Min. :-1.00000
	cost_diff_pct	1st Qu.: 0.00000
	cost_diff_pct	Median : 0.02615
	cost_diff_pct	Mean : 0.15641
	cost_diff_pct	3rd Qu.: 0.11165
	cost_diff_pct	Max. :46.70000
	delay_days	Min. :-2557
	delay_days	1st Qu.: 0
	delay_days	Median : 1096
	delay_days	Mean : 1591
	delay_days	3rd Qu.: 2556
	delay_days	Max. : 9223

spend_to_date_percent	Min. :0.0000
spend_to_date_percent	1st Qu.:0.0394
spend_to_date_percent	Median :0.1306
spend_to_date_percent	Mean :0.3058
spend_to_date_percent	3rd Qu.:0.5955
spend_to_date_percent	Max. :1.1288
avg_construction_jobs	Min. : 18297
avg_construction_jobs	1st Qu.: 78617
avg_construction_jobs	Median : 90850
avg_construction_jobs	Mean : 92212
avg_construction_jobs	3rd Qu.:116306
avg_construction_jobs	Max. :138406
avg_labor_unemp_rate	Min. : 3.600
avg_labor_unemp_rate	1st Qu.: 5.450
avg_labor_unemp_rate	Median : 6.340
avg_labor_unemp_rate	Mean : 6.443
avg_labor_unemp_rate	3rd Qu.: 6.960
avg_labor_unemp_rate	Max. :15.100
avg_cyclone_freq	Min. :4750
avg_cyclone_freq	1st Qu.:5965
avg_cyclone_freq	Median :6183
avg_cyclone_freq	Mean :6045
avg_cyclone_freq	3rd Qu.:6324
avg_cyclone_freq	Max. :7222
avg_temp_anomaly	Min. :0.795
avg_temp_anomaly	1st Qu.:0.999
avg_temp_anomaly	Median :1.077
avg_temp_anomaly	Mean :1.077
avg_temp_anomaly	3rd Qu.:1.155
avg_temp_anomaly	Max. :1.430
avg_max_wind_speed	Min. :145.0
avg_max_wind_speed	1st Qu.:157.5
avg_max_wind_speed	Median :158.3
avg_max_wind_speed	Mean :158.2
avg_max_wind_speed	3rd Qu.:160.7
avg_max_wind_speed	Max. :177.5

4.6.2 Correlation Matrix (Numerical Variables)

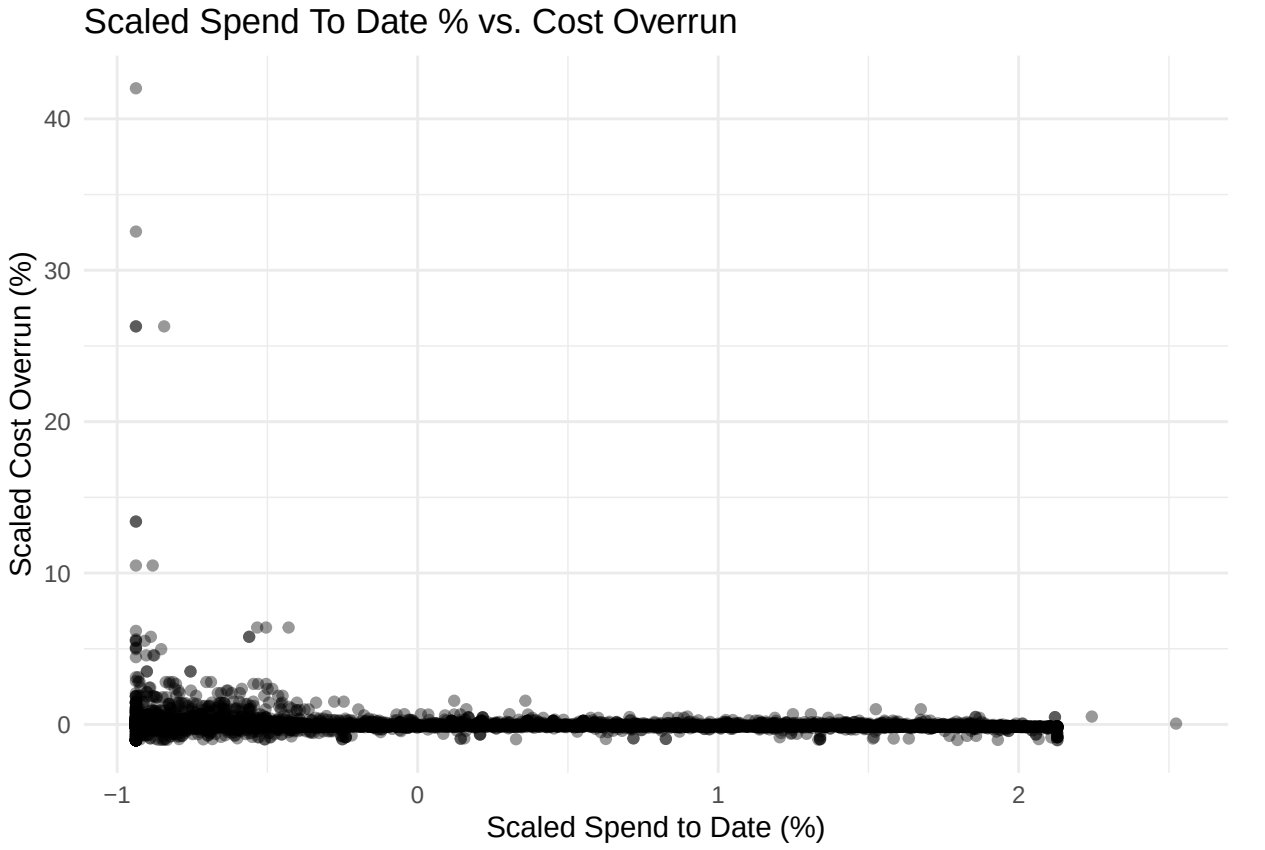
```
numeric_data <- project_model_data_final_2 %>% select_if(is.numeric) %>% select(-c(avg_cyclone_freq, avg
corr_matrix <- cor(numeric_data, use = "complete.obs")
corrplot(corr_matrix, method = "color", type = "upper", tl.cex = 0.7, number.digits = 1, number.cex = 0
```



4.6.3 Pairwise Scatter Plots: Cost/Delay vs Key Features

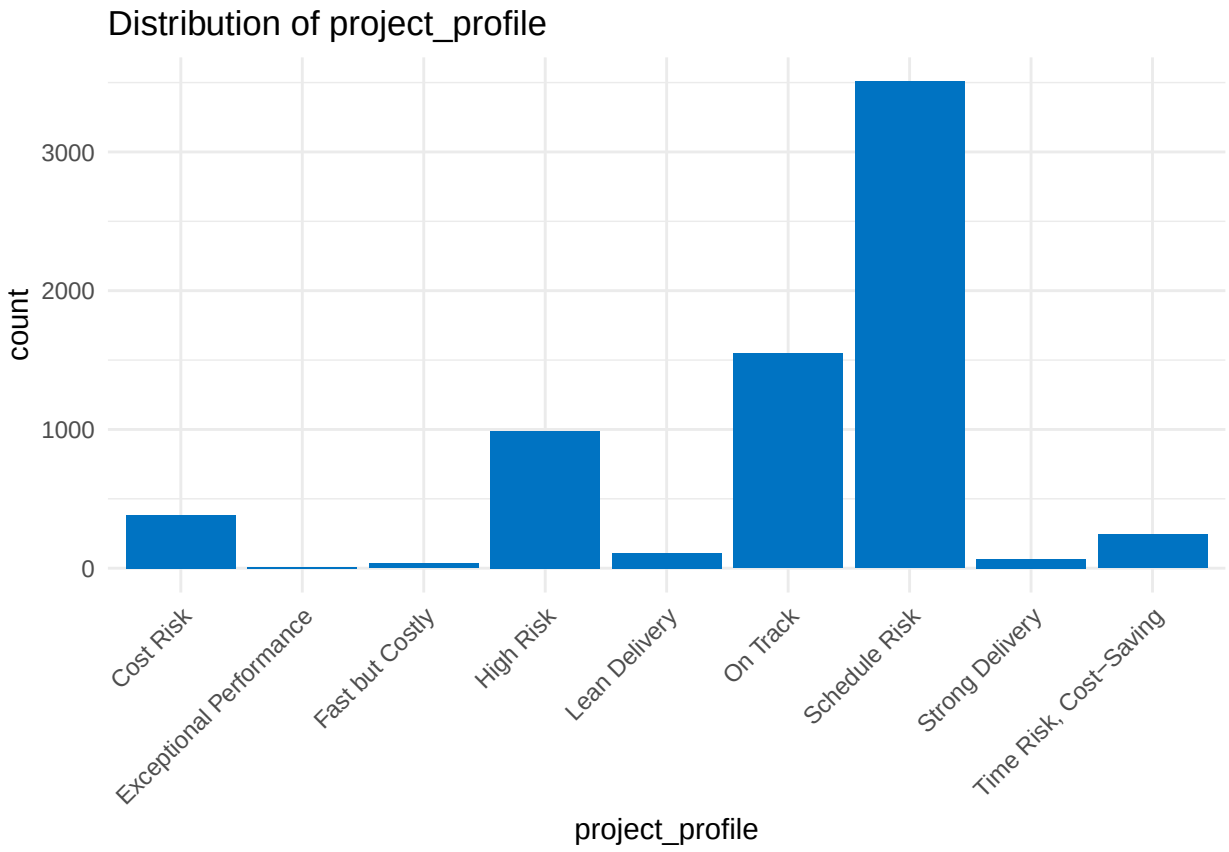
```
plot1 <- ggplot(project_model_data_final_2, aes(x = spend_to_date_percent, y = cost_diff_pct)) +
  geom_point(alpha = 0.4) +
  labs(title = "Spend To Date % vs. Cost Overrun", x = "Initial Budget ($)", y = "Cost Overrun (%)")

plot2 <- ggplot(project_model_data_final_2, aes(x = avg_temp_anomaly, y = delay_days)) +
  geom_point(alpha = 0.4, color = "tomato") +
  labs(title = "Temperature Anomaly vs. Delay Days", x = "Avg Temp Anomaly", y = "Delay (Days)")


# Add scaled variables to your dataset
project_model_data_final_2 <- project_model_data_final_2 %>%
  mutate(
    spend_scaled = scale(spend_to_date_percent),
    cost_diff_scaled = scale(cost_diff_pct)
  )

# Plot with scaled values
plot3 <- ggplot(project_model_data_final_2, aes(x = spend_scaled, y = cost_diff_scaled)) +
  geom_point(alpha = 0.4) +
  labs(
    title = "Scaled Spend To Date % vs. Cost Overrun",
    x = "Scaled Spend to Date (%)",
    y = "Scaled Cost Overrun (%)"
  ) +
```

```
  theme_minimal()

grid.arrange(plot3, ncol = 1)
```



4.6.4 Distribution of Outcome Classes

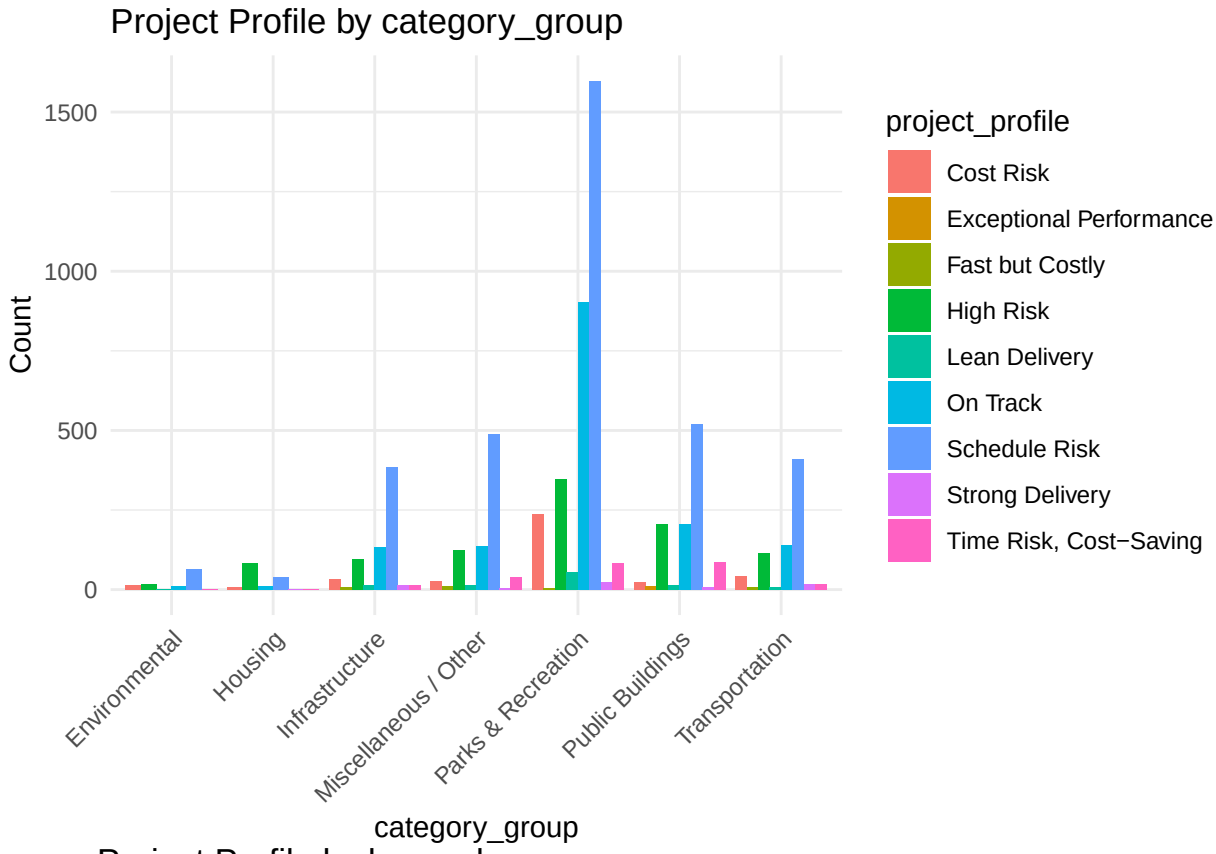
```
plot_cost <- ggplot(project_model_data_final_2, aes(x = cost_class)) +
  geom_bar(fill = "#0073C2FF") +
  labs(title = "Distribution of Cost Class")

plot_delay <- ggplot(project_model_data_final_2, aes(x = delay_class)) +
  geom_bar(fill = "#EFC000FF") +
  labs(title = "Distribution of Delay Class")

plot_profile <- ggplot(project_model_data_final_2, aes(x = project_profile)) +
  geom_bar(fill = "#0073C2FF") +
  labs(title = "Distribution of project_profile") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(plot_profile, ncol = 1)
```

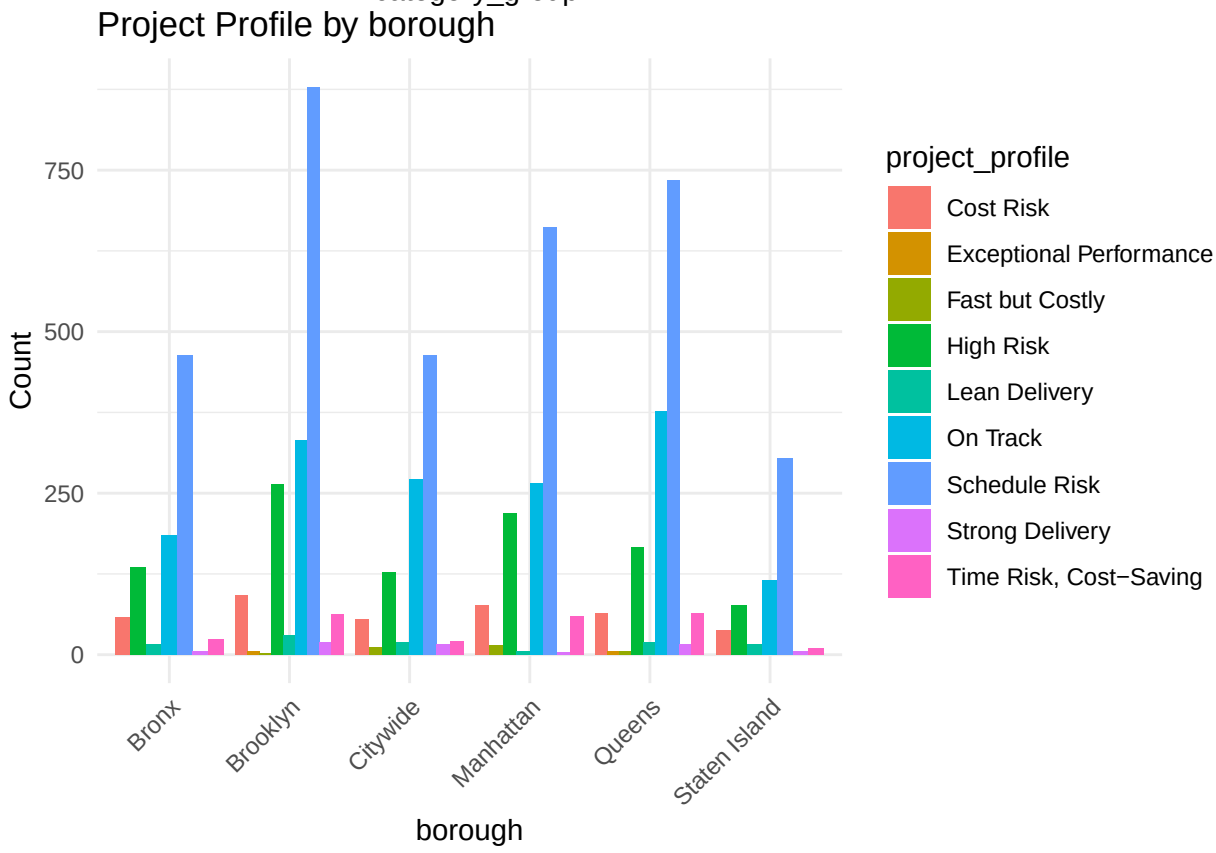


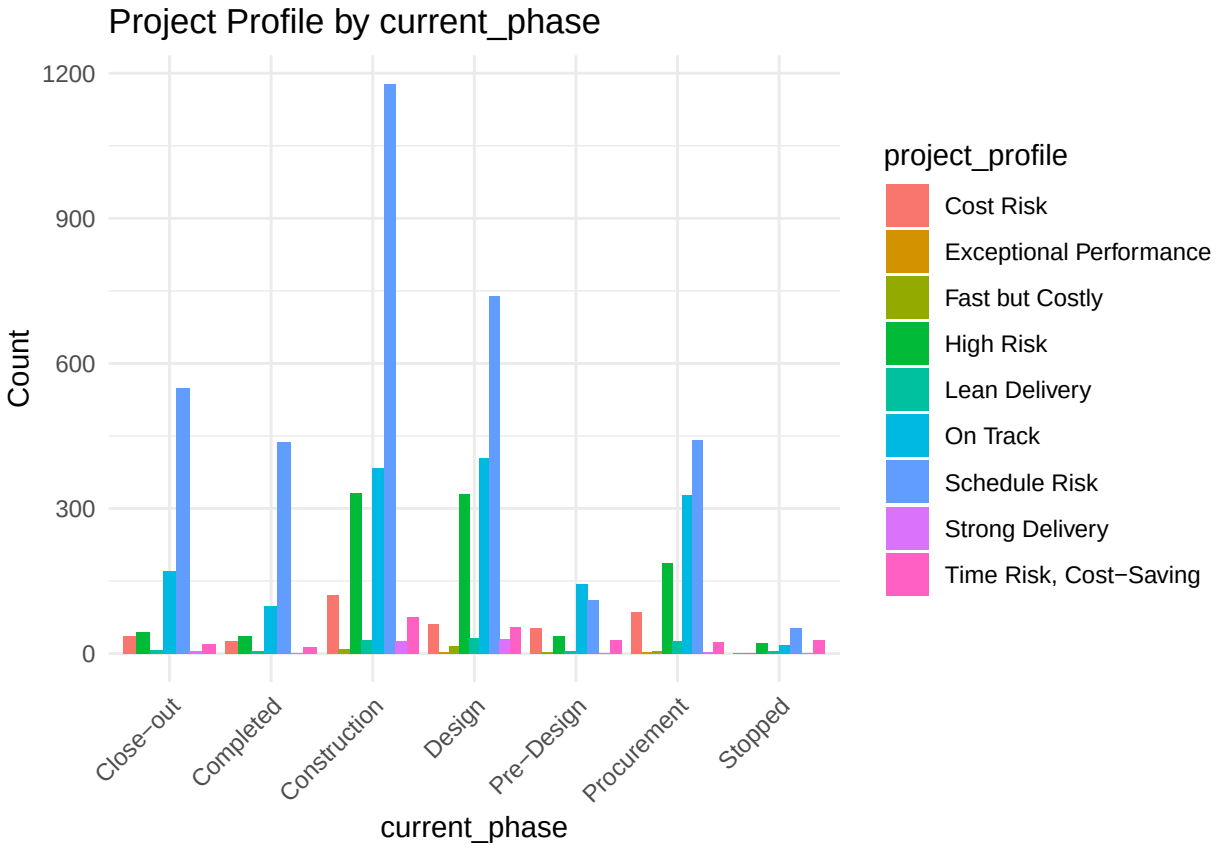
4.6.5 Relationship between project_profile and categorical variables

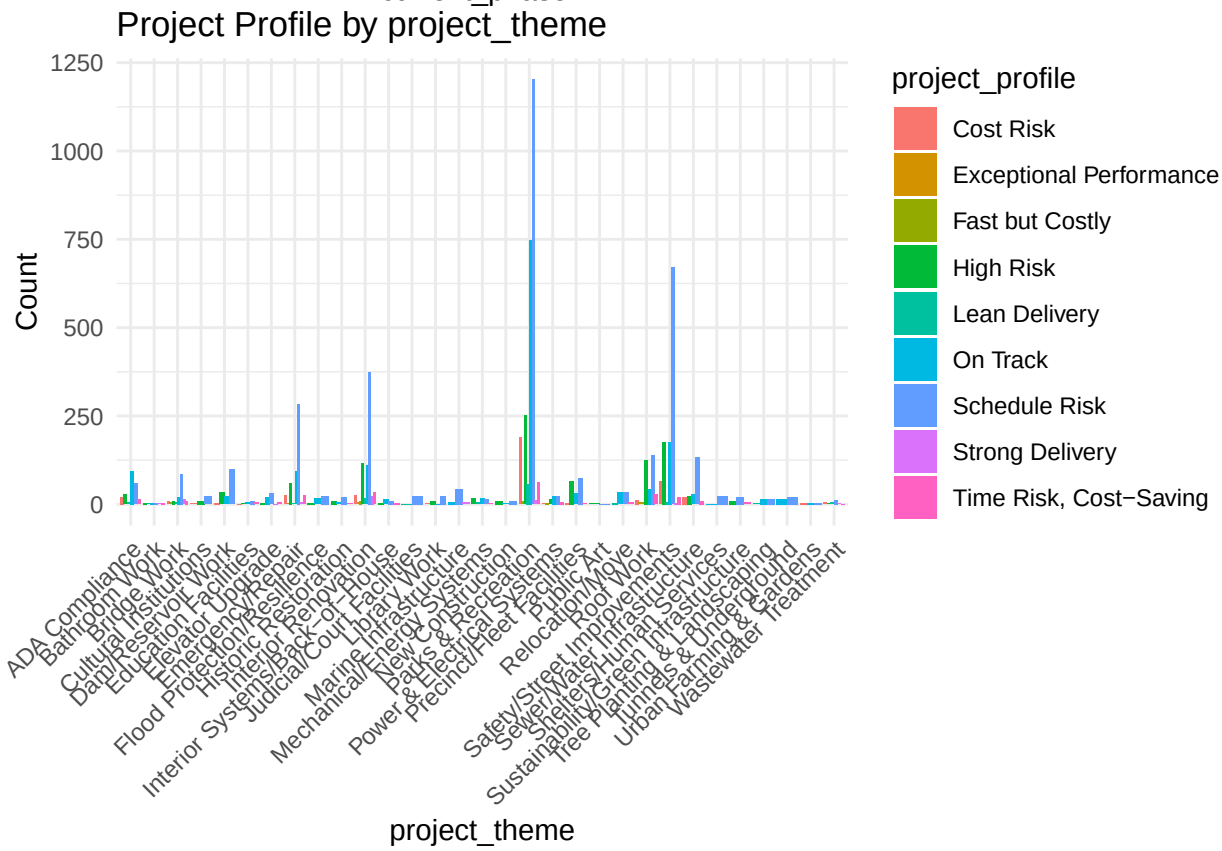
```
library(ggplot2)
library(dplyr)

# Create a list of categorical variables to compare with project_profile
categorical_vars <- c("category_group", "borough", "current_phase", "project_theme", "delay_category")

# Loop through each variable and generate a grouped bar chart
for (var in categorical_vars) {
  p <- ggplot(project_model_data_final_2, aes_string(x = var, fill = "project_profile")) +
    geom_bar(position = "dodge") +
    labs(title = paste("Project Profile by", var), x = var, y = "Count") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  print(p)
}
```

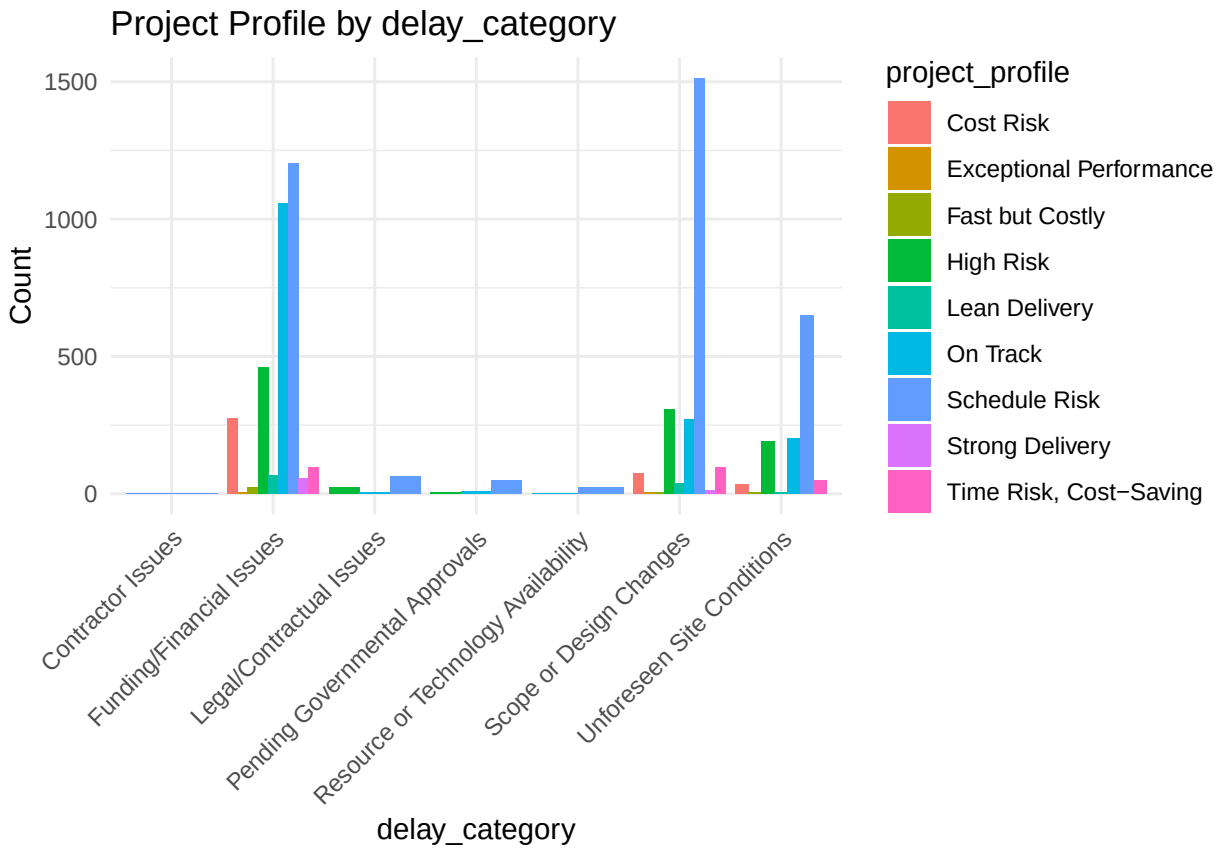
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Project Profile by category_group
Count
1500
1000
500
0
Environmental
Housing
Infrastructure
Miscellaneous / Other
Parks & Recreation
Public Buildings
Transportation
category_group
project_profile
Cost Risk
Exceptional Performance
Fast but Costly
High Risk
Lean Delivery
On Track
Schedule Risk
Strong Delivery
Time Risk, Cost−Saving


Project Profile by borough
Count
750
500
250
0
Bronx
Brooklyn
Citywide
Manhattan
Queens
Staten Island
borough
project_profile
Cost Risk
Exceptional Performance
Fast but Costly
High Risk
Lean Delivery
On Track
Schedule Risk
Strong Delivery
Time Risk, Cost−Saving


Project Profile by current_phase
Count
1200
900
600
300
0
Close−out
Completed
Construction
Design
Pre−Design
Procurement
Stopped
current_phase
project_profile
Cost Risk
Exceptional Performance
Fast but Costly
High Risk
Lean Delivery
On Track
Schedule Risk
Strong Delivery
Time Risk, Cost−Saving


Project Profile by project_theme
Count
1250
1000
750
500
250
0
ADA Compliance
Bathroom Work
Bridge Work
Cultural Institutions
Dam/Reservoir Work
Education Facilities
Elevator Upgrade
Emergency Repair
Flood Protection/Resilience
Historic Restoration
Interior Renovation
Interior Systems/Back-of-House
Judicial/Court Facilities
Library Work
Marine Infrastructure
Mechanical/Energy Systems
New Construction
Parks & Recreation
Power & Electrical Systems
Precinct/Fleet Facilities
Public Art
Relocation/Move
Roof Work
Safety/Street Improvements
Sewer/Water Infrastructure
Shelters/Human Services
Sustainability/Green Infrastructure
Tree Planting & Landscaping
Tunnels & Underground
Urban Farming & Gardens
Wastewater Treatment
project_theme
project_profile
Cost Risk
Exceptional Performance
Fast but Costly
High Risk
Lean Delivery
On Track
Schedule Risk
Strong Delivery
Time Risk, Cost−Saving




```
# Run Chi-Square Test for Independence
for (var in categorical_vars) {
  cat("\n\nChi-Square Test: project_profile vs", var, "\n")
  print(chisq.test(table(project_model_data_final_2[["project_profile"]], project_data[[var]])))
}
```

```
##
##
## Chi-Square Test: project_profile vs category_group

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

##
##  Pearson's Chi-squared test
##
## data:  table(project_model_data_final_2[["project_profile"]], project_data[[var]])
## X-squared = 625.35, df = 48, p-value < 2.2e-16
##
##
##
## Chi-Square Test: project_profile vs borough

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

##
##  Pearson's Chi-squared test
##
```

```
## data:  table(project_model_data_final_2[["project_profile"]], project_data[[var]])
## X-squared = 152, df = 40, p-value = 5.836e-15
##
##
##
## Chi-Square Test: project_profile vs current_phase
## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect
##
##  Pearson's Chi-squared test
##
## data:  table(project_model_data_final_2[["project_profile"]], project_data[[var]])
## X-squared = 696.58, df = 48, p-value < 2.2e-16
##
##
##
## Chi-Square Test: project_profile vs project_theme
## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect
##
##  Pearson's Chi-squared test
##
## data:  table(project_model_data_final_2[["project_profile"]], project_data[[var]])
## X-squared = 1509.3, df = 248, p-value < 2.2e-16
##
##
##
## Chi-Square Test: project_profile vs delay_category
## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect
##
##  Pearson's Chi-squared test
##
## data:  table(project_model_data_final_2[["project_profile"]], project_data[[var]])
## X-squared = 745.4, df = 48, p-value < 2.2e-16
```

4.7 Predicting cost_class and delay_class separately

```
# ---- Setup for Predicting cost_class and delay_class separately ----
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:gridExtra':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
# --- Dataset for cost_class prediction (exclude cost_diff_pct and cost_class) ---
cost_model_data <- project_model_data_final %>%
  select(-c(cost_diff_pct, cost_class, fms_id, latest_budget, latest_spend, cost_diff, orig_start_date,

# Add cost_class back as target
cost_model_data$cost_class <- as.factor(project_model_data_final$cost_class)

# --- Dataset for delay_class prediction (exclude delay_days and delay_class) ---#######
delay_model_data <- project_model_data_final %>%
  select(-c(delay_days, delay_class, fms_id, latest_budget, latest_spend, orig_start_date, orig_end_dat

# Add delay_class back as target
delay_model_data$delay_class <- as.factor(project_model_data_final$delay_class)

# --- Reusable modeling pipeline function ---
run_models <- function(data, target_col) {
  # Set target variable
  data[[target_col]] <- as.factor(data[[target_col]])

  # Convert character to factor
  categorical_vars <- sapply(data, is.character)
```

```r
  categorical_vars <- names(categorical_vars[categorical_vars])
  categorical_vars <- setdiff(categorical_vars, target_col)
  data[categorical_vars] <- lapply(data[categorical_vars], as.factor)

  # Train-test split
  set.seed(123)
  train_index <- createDataPartition(data[[target_col]], p = 0.8, list = FALSE)
  train_data <- data[train_index, ]
  test_data <- data[-train_index, ]

  # --Random Forest:Cross-validated and tuned Random Forest using caret---
  control <- trainControl(method = "cv", number = 5)
  tunegrid <- expand.grid(.mtry = c(2, 4, 6, 8))

  rf_model <- train(
    as.formula(paste(target_col, "~ .")),
    data = train_data,
    method = "rf",
    metric = "Accuracy",
    trControl = control,
    tuneGrid = tunegrid
  )
  print(rf_model)

  rf_preds <- predict(rf_model, newdata = test_data)
  print(confusionMatrix(rf_preds, test_data[[target_col]]))
  print(varImp(rf_model))

  # --- Decision Tree with caret cross-validation---
  dt_control <- trainControl(method = "cv", number = 5)
  dt_model <- train(
    as.formula(paste(target_col, "~ .")),
    data = train_data,
    method = "rpart",
    trControl = dt_control,
    tuneLength = 10
  )
  print(dt_model)
  dt_preds <- predict(dt_model, test_data)
  print(confusionMatrix(dt_preds, test_data[[target_col]]))
  png(filename = paste0("decision_tree_", target_col, ".png"), width = 2400, height = 1600, res = 300)
rpart.plot(dt_model$finalModel)
dev.off()

# Also show the trees in the console
rpart.plot(dt_model$finalModel, box.palette = "Greens")

  # --- XGBoost with cross-validation and early stopping---
  train_matrix <- model.matrix(as.formula(paste(target_col, "~ .")), data = train_data)
  test_matrix <- model.matrix(as.formula(paste(target_col, "~ .")), data = test_data)

  xgb_train <- xgb.DMatrix(data = train_matrix, label = as.numeric(train_data[[target_col]]) - 1)
  xgb_test <- xgb.DMatrix(data = test_matrix, label = as.numeric(test_data[[target_col]]) - 1)
```

```
  xgb_cv <- xgb.cv(
    data = xgb_train,
    nrounds = 100,
    nfold = 5,
    early_stopping_rounds = 10,
    objective = "multi:softmax",
    num_class = length(unique(train_data[[target_col]])),
    verbose = 1
  )
  best_nrounds <- xgb_cv$best_iteration

  xgb_model <- xgboost(
    data = xgb_train,
    nrounds = best_nrounds,
    objective = "multi:softmax",
    num_class = length(unique(train_data[[target_col]])),
    verbose = 0
  )
  xgb_preds <- predict(xgb_model, xgb_test)
  xgb_preds_factor <- factor(xgb_preds + 1, levels = 1:length(levels(train_data[[target_col]])),
                             labels = levels(train_data[[target_col]]))
  print(confusionMatrix(xgb_preds_factor, test_data[[target_col]]))
}

# ---- Run for cost_class and delay_class ----
suppressWarnings({
  run_models(cost_model_data, "cost_class")
  run_models(delay_model_data, "delay_class")
})
```

```
## Random Forest
##
## 5510 samples
##   13 predictor
##    3 classes: 'On Budget', 'Over Budget', 'Under Budget'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4409, 4408, 4408, 4408, 4407
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7500915  0.03969525
##   4     0.8294030  0.44198436
##   6     0.8963743  0.70081764
##   8     0.9205116  0.78059023
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     On Budget Over Budget Under Budget
```

```
##   On Budget         1015          58           32
##   Over Budget          7         223            2
##   Under Budget         1           0           38
##
## Overall Statistics
##
##                Accuracy : 0.9273
##                  95% CI : (0.9123, 0.9405)
##     No Information Rate : 0.7435
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.802
##
##  Mcnemar's Test P-Value : 2.437e-15
##
## Statistics by Class:
##
##                      Class: On Budget Class: Over Budget Class: Under Budget
## Sensitivity                    0.9922             0.7936             0.52778
## Specificity                    0.7450             0.9918             0.99923
## Pos Pred Value                 0.9186             0.9612             0.97436
## Neg Pred Value                 0.9705             0.9493             0.97457
## Prevalence                     0.7435             0.2042             0.05233
## Detection Rate                 0.7376             0.1621             0.02762
## Detection Prevalence           0.8031             0.1686             0.02834
## Balanced Accuracy              0.8686             0.8927             0.76351
## rf variable importance
##
##   only 20 most important variables shown (out of 61)
##
##                                       Overall
## initial_budget                         100.00
## spend_to_date_percent                   81.09
## delay_days                              69.68
## avg_construction_jobs                   55.00
## avg_labor_unemp_rate                    53.58
## avg_temp_anomaly                        45.53
## avg_cyclone_freq                        41.65
## avg_max_wind_speed                      36.80
## category_groupHousing                   13.91
## project_themeRoof Work                  13.48
## project_themeParks & Recreation         12.68
## delay_categoryFunding/Financial Issues  12.00
## category_groupParks & Recreation        11.95
## current_phaseConstruction               11.67
## project_themeSafety/Street Improvements 11.52
## boroughCitywide                         11.17
## category_groupPublic Buildings          11.01
## current_phaseProcurement                10.91
## delay_categoryScope or Design Changes   10.87
## current_phaseDesign                     10.40
## CART
##
## 5510 samples
```

```
##   13 predictor
##    3 classes: 'On Budget', 'Over Budget', 'Under Budget'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4408, 4408, 4408, 4408, 4408
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.003536068  0.7927405  0.34373100
##   0.004243281  0.7822142  0.26165758
##   0.004950495  0.7753176  0.21086676
##   0.005657709  0.7698730  0.17545055
##   0.006836398  0.7696915  0.17423620
##   0.007779349  0.7676951  0.17206321
##   0.008486563  0.7644283  0.15624061
##   0.009193777  0.7629764  0.14770293
##   0.011315417  0.7537205  0.08946937
##   0.028288543  0.7460980  0.03701423
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.003536068.
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     On Budget Over Budget Under Budget
##   On Budget          992         196           63
##   Over Budget         31          85            5
##   Under Budget         0           0            4
##
## Overall Statistics
##
##                Accuracy : 0.7856
##                  95% CI : (0.763, 0.807)
##     No Information Rate : 0.7435
##     P-Value [Acc > NIR] : 0.0001512
##
##                   Kappa : 0.2993
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: On Budget Class: Over Budget Class: Under Budget
## Sensitivity                    0.9697            0.30249            0.055556
## Specificity                    0.2663            0.96712            1.000000
## Pos Pred Value                 0.7930            0.70248            1.000000
## Neg Pred Value                 0.7520            0.84382            0.950437
## Prevalence                     0.7435            0.20422            0.052326
## Detection Rate                 0.7209            0.06177            0.002907
## Detection Prevalence           0.9092            0.08794            0.002907
## Balanced Accuracy              0.6180            0.63481            0.527778
```

```
## [1]  train-mlogloss:0.895754+0.003512    test-mlogloss:0.904977+0.005034
```

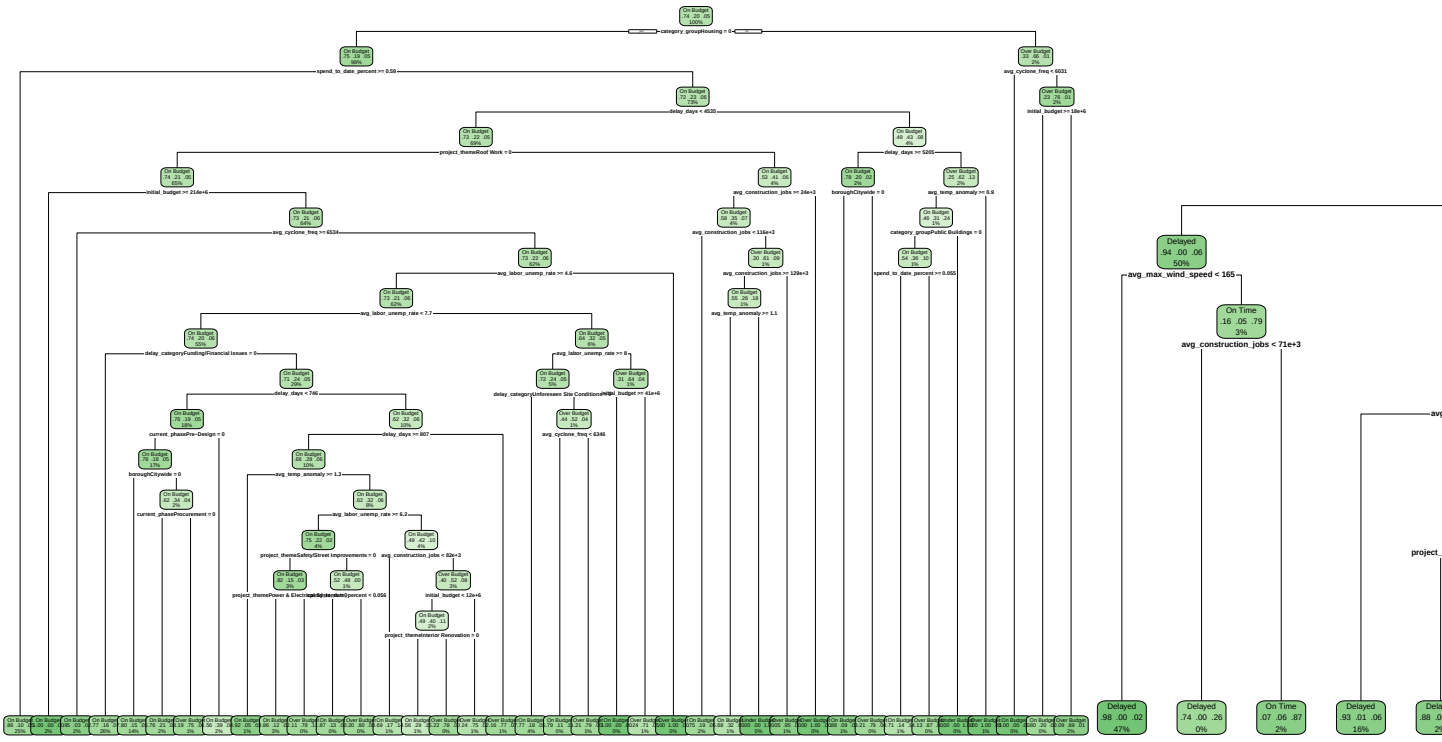
```
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 10 rounds.
##
## [2]  train-mlogloss:0.773842+0.004464    test-mlogloss:0.790680+0.007373
## [3]  train-mlogloss:0.688461+0.008079    test-mlogloss:0.713884+0.008688
## [4]  train-mlogloss:0.624617+0.007638    test-mlogloss:0.656698+0.011656
## [5]  train-mlogloss:0.577682+0.006585    test-mlogloss:0.616102+0.011570
## [6]  train-mlogloss:0.539650+0.006356    test-mlogloss:0.584280+0.011976
## [7]  train-mlogloss:0.509991+0.005949    test-mlogloss:0.558494+0.014894
## [8]  train-mlogloss:0.485720+0.008181    test-mlogloss:0.538675+0.013137
## [9]  train-mlogloss:0.465903+0.008306    test-mlogloss:0.524103+0.013163
## [10] train-mlogloss:0.445666+0.009749    test-mlogloss:0.509308+0.013379
## [11] train-mlogloss:0.431308+0.009563    test-mlogloss:0.497061+0.012076
## [12] train-mlogloss:0.421572+0.008723    test-mlogloss:0.488765+0.010653
## [13] train-mlogloss:0.410141+0.009787    test-mlogloss:0.480121+0.010747
## [14] train-mlogloss:0.398213+0.006507    test-mlogloss:0.471082+0.011424
## [15] train-mlogloss:0.387306+0.007469    test-mlogloss:0.462665+0.009265
## [16] train-mlogloss:0.378729+0.006609    test-mlogloss:0.456088+0.008840
## [17] train-mlogloss:0.370308+0.007002    test-mlogloss:0.449272+0.009193
## [18] train-mlogloss:0.361283+0.006629    test-mlogloss:0.442089+0.009527
## [19] train-mlogloss:0.352697+0.005146    test-mlogloss:0.435507+0.010823
## [20] train-mlogloss:0.343954+0.005259    test-mlogloss:0.428601+0.010849
## [21] train-mlogloss:0.334661+0.003741    test-mlogloss:0.420478+0.009030
## [22] train-mlogloss:0.328326+0.003848    test-mlogloss:0.415077+0.008959
## [23] train-mlogloss:0.321249+0.004008    test-mlogloss:0.409410+0.009066
## [24] train-mlogloss:0.314767+0.004121    test-mlogloss:0.403527+0.010327
## [25] train-mlogloss:0.308162+0.004590    test-mlogloss:0.398323+0.011386
## [26] train-mlogloss:0.299132+0.006690    test-mlogloss:0.391656+0.013217
## [27] train-mlogloss:0.289757+0.004836    test-mlogloss:0.383700+0.012738
## [28] train-mlogloss:0.283406+0.003894    test-mlogloss:0.378463+0.012305
## [29] train-mlogloss:0.277913+0.003336    test-mlogloss:0.374272+0.011284
## [30] train-mlogloss:0.272612+0.000634    test-mlogloss:0.369574+0.009084
## [31] train-mlogloss:0.267601+0.001406    test-mlogloss:0.365598+0.007953
## [32] train-mlogloss:0.259646+0.003086    test-mlogloss:0.358672+0.010140
## [33] train-mlogloss:0.252334+0.003991    test-mlogloss:0.352852+0.009820
## [34] train-mlogloss:0.247326+0.006275    test-mlogloss:0.348201+0.008312
## [35] train-mlogloss:0.239160+0.005932    test-mlogloss:0.340863+0.007325
## [36] train-mlogloss:0.233672+0.004630    test-mlogloss:0.335621+0.007168
## [37] train-mlogloss:0.229579+0.004101    test-mlogloss:0.332359+0.007333
## [38] train-mlogloss:0.224468+0.004842    test-mlogloss:0.327839+0.007906
## [39] train-mlogloss:0.218894+0.005026    test-mlogloss:0.323337+0.008612
## [40] train-mlogloss:0.213578+0.004596    test-mlogloss:0.318552+0.007286
## [41] train-mlogloss:0.207665+0.003534    test-mlogloss:0.313427+0.006439
## [42] train-mlogloss:0.202804+0.004166    test-mlogloss:0.309550+0.005597
## [43] train-mlogloss:0.199274+0.004076    test-mlogloss:0.305495+0.004870
## [44] train-mlogloss:0.195558+0.003979    test-mlogloss:0.302266+0.004027
## [45] train-mlogloss:0.190770+0.003978    test-mlogloss:0.298188+0.004908
## [46] train-mlogloss:0.186865+0.002902    test-mlogloss:0.294702+0.005009
## [47] train-mlogloss:0.183067+0.002336    test-mlogloss:0.291029+0.005723
## [48] train-mlogloss:0.179640+0.002927    test-mlogloss:0.288554+0.005831
## [49] train-mlogloss:0.175594+0.004099    test-mlogloss:0.284694+0.005447
## [50] train-mlogloss:0.170968+0.005250    test-mlogloss:0.280542+0.007053
## [51] train-mlogloss:0.168090+0.005758    test-mlogloss:0.277747+0.007764
## [52] train-mlogloss:0.163454+0.004798    test-mlogloss:0.274056+0.007243
```

```
## [53] train-mlogloss:0.159764+0.004729    test-mlogloss:0.271407+0.006948
## [54] train-mlogloss:0.156817+0.004353    test-mlogloss:0.268765+0.006226
## [55] train-mlogloss:0.155132+0.004204    test-mlogloss:0.267641+0.006322
## [56] train-mlogloss:0.152872+0.003438    test-mlogloss:0.265371+0.005880
## [57] train-mlogloss:0.150463+0.004005    test-mlogloss:0.263170+0.006320
## [58] train-mlogloss:0.147964+0.003833    test-mlogloss:0.260636+0.005963
## [59] train-mlogloss:0.145014+0.003425    test-mlogloss:0.257855+0.005679
## [60] train-mlogloss:0.142172+0.002379    test-mlogloss:0.255308+0.006179
## [61] train-mlogloss:0.139819+0.002567    test-mlogloss:0.253281+0.005601
## [62] train-mlogloss:0.135830+0.002820    test-mlogloss:0.250058+0.005127
## [63] train-mlogloss:0.133927+0.002585    test-mlogloss:0.248507+0.005507
## [64] train-mlogloss:0.131496+0.002874    test-mlogloss:0.246461+0.005014
## [65] train-mlogloss:0.128779+0.004403    test-mlogloss:0.243956+0.006276
## [66] train-mlogloss:0.126818+0.004908    test-mlogloss:0.242317+0.006090
## [67] train-mlogloss:0.124055+0.004991    test-mlogloss:0.239715+0.006318
## [68] train-mlogloss:0.121218+0.005105    test-mlogloss:0.237524+0.005688
## [69] train-mlogloss:0.118432+0.004847    test-mlogloss:0.235665+0.005867
## [70] train-mlogloss:0.116189+0.004530    test-mlogloss:0.233113+0.005602
## [71] train-mlogloss:0.113996+0.005086    test-mlogloss:0.230786+0.005782
## [72] train-mlogloss:0.111882+0.005214    test-mlogloss:0.228700+0.005687
## [73] train-mlogloss:0.110060+0.004783    test-mlogloss:0.226913+0.005926
## [74] train-mlogloss:0.108342+0.005185    test-mlogloss:0.225272+0.005731
## [75] train-mlogloss:0.106351+0.004897    test-mlogloss:0.223529+0.006233
## [76] train-mlogloss:0.103907+0.004438    test-mlogloss:0.221473+0.006503
## [77] train-mlogloss:0.102373+0.004848    test-mlogloss:0.220415+0.006577
## [78] train-mlogloss:0.100175+0.004522    test-mlogloss:0.218774+0.006252
## [79] train-mlogloss:0.098681+0.004406    test-mlogloss:0.217525+0.006259
## [80] train-mlogloss:0.096816+0.004333    test-mlogloss:0.216196+0.006885
## [81] train-mlogloss:0.095349+0.004257    test-mlogloss:0.214925+0.007095
## [82] train-mlogloss:0.094060+0.004505    test-mlogloss:0.213645+0.007045
## [83] train-mlogloss:0.092769+0.004588    test-mlogloss:0.212761+0.007115
## [84] train-mlogloss:0.091126+0.004239    test-mlogloss:0.211144+0.006697
## [85] train-mlogloss:0.089887+0.004401    test-mlogloss:0.209924+0.006848
## [86] train-mlogloss:0.088684+0.004170    test-mlogloss:0.208534+0.006573
## [87] train-mlogloss:0.086955+0.003746    test-mlogloss:0.207191+0.005465
## [88] train-mlogloss:0.085582+0.003744    test-mlogloss:0.205718+0.005792
## [89] train-mlogloss:0.084123+0.003399    test-mlogloss:0.204446+0.005674
## [90] train-mlogloss:0.082738+0.003489    test-mlogloss:0.203404+0.005835
## [91] train-mlogloss:0.081118+0.003394    test-mlogloss:0.201736+0.005579
## [92] train-mlogloss:0.079635+0.002909    test-mlogloss:0.200303+0.005563
## [93] train-mlogloss:0.078387+0.003138    test-mlogloss:0.199048+0.005478
## [94] train-mlogloss:0.076876+0.002945    test-mlogloss:0.197976+0.005702
## [95] train-mlogloss:0.075941+0.003014    test-mlogloss:0.196980+0.006007
## [96] train-mlogloss:0.074457+0.002937    test-mlogloss:0.195497+0.005995
## [97] train-mlogloss:0.073470+0.002615    test-mlogloss:0.194477+0.005833
## [98] train-mlogloss:0.072472+0.002659    test-mlogloss:0.193587+0.005543
## [99] train-mlogloss:0.070912+0.002798    test-mlogloss:0.192327+0.005080
## [100]    train-mlogloss:0.069772+0.002946    test-mlogloss:0.191307+0.005002
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     On Budget Over Budget Under Budget
##   On Budget         1008          50           15
##   Over Budget         14         231            1
```

```
##   Under Budget         1           0           56
##
## Overall Statistics
##
##                Accuracy : 0.9411
##                  95% CI : (0.9274, 0.953)
##     No Information Rate : 0.7435
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8457
##
##  Mcnemar's Test P-Value : 2.526e-07
##
## Statistics by Class:
##
##                      Class: On Budget Class: Over Budget Class: Under Budget
## Sensitivity                    0.9853             0.8221              0.77778
## Specificity                    0.8159             0.9863              0.99923
## Pos Pred Value                 0.9394             0.9390              0.98246
## Neg Pred Value                 0.9505             0.9558              0.98787
## Prevalence                     0.7435             0.2042              0.05233
## Detection Rate                 0.7326             0.1679              0.04070
## Detection Prevalence           0.7798             0.1788              0.04142
## Balanced Accuracy              0.9006             0.9042              0.88851
## Random Forest
##
## 5510 samples
##   13 predictor
##    3 classes: 'Delayed', 'Early', 'On Time'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4409, 4407, 4408, 4408, 4408
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8537238  0.6255054
##   4     0.9535385  0.8927143
##   6     0.9698701  0.9313663
##   8     0.9724128  0.9371988
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Delayed Early On Time
##    Delayed     936     0       9
##    Early         0    20       0
##    On Time      11     2     398
##
## Overall Statistics
##
##                Accuracy : 0.984
```

```
##                  95% CI : (0.9759, 0.99)
##     No Information Rate : 0.6882
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9636
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Delayed Class: Early Class: On Time
## Sensitivity                  0.9884      0.90909         0.9779
## Specificity                  0.9790      1.00000         0.9866
## Pos Pred Value               0.9905      1.00000         0.9684
## Neg Pred Value               0.9745      0.99853         0.9907
## Prevalence                   0.6882      0.01599         0.2958
## Detection Rate               0.6802      0.01453         0.2892
## Detection Prevalence         0.6868      0.01453         0.2987
## Balanced Accuracy            0.9837      0.95455         0.9822
## rf variable importance
##
##   only 20 most important variables shown (out of 61)
##
##                                       Overall
## avg_max_wind_speed                    100.000
## avg_cyclone_freq                       68.682
## avg_temp_anomaly                       66.289
## avg_labor_unemp_rate                   43.141
## initial_budget                         31.389
## spend_to_date_percent                  28.796
## cost_diff                              27.675
## avg_construction_jobs                  25.330
## delay_categoryFunding/Financial Issues 12.756
## delay_categoryScope or Design Changes   6.901
## project_themeParks & Recreation         6.693
## category_groupParks & Recreation        6.427
## project_themeSafety/Street Improvements 5.070
## category_groupTransportation            4.261
## current_phaseDesign                     4.041
## project_themeInterior Renovation        3.988
## category_groupPublic Buildings          3.728
## current_phaseConstruction               3.596
## boroughBrooklyn                         3.586
## boroughManhattan                        3.532
## CART
##
## 5510 samples
##   13 predictor
##    3 classes: 'Delayed', 'Early', 'On Time'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4408, 4408, 4408, 4408, 4408
## Resampling results across tuning parameters:
```

```
##
##   cp           Accuracy   Kappa
##   0.004069767  0.9174229  0.8115343
##   0.005232558  0.9130672  0.8016508
##   0.005813953  0.9121597  0.7993549
##   0.006395349  0.9101633  0.7954344
##   0.008139535  0.9059891  0.7879610
##   0.008720930  0.9059891  0.7879610
##   0.020639535  0.8998185  0.7703854
##   0.052325581  0.8827586  0.7221916
##   0.186627907  0.8395644  0.5907003
##   0.213662791  0.7647913  0.3096616
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.004069767.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Delayed Early On Time
##    Delayed     898     3      51
##    Early         0     3       0
##    On Time      49    16     356
##
## Overall Statistics
##
##                Accuracy : 0.9135
##                  95% CI : (0.8974, 0.9278)
##     No Information Rate : 0.6882
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8004
##
##  Mcnemar's Test P-Value : 0.0002682
##
## Statistics by Class:
##
##                      Class: Delayed Class: Early Class: On Time
## Sensitivity                  0.9483      0.13636         0.8747
## Specificity                  0.8741      1.00000         0.9329
## Pos Pred Value               0.9433      1.00000         0.8456
## Neg Pred Value               0.8844      0.98616         0.9466
## Prevalence                   0.6882      0.01599         0.2958
## Detection Rate               0.6526      0.00218         0.2587
## Detection Prevalence         0.6919      0.00218         0.3060
## Balanced Accuracy            0.9112      0.56818         0.9038
```



```
## [1]  train-mlogloss:0.762908+0.000632    test-mlogloss:0.771086+0.003801
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 10 rounds.
##
## [2]  train-mlogloss:0.567449+0.001552    test-mlogloss:0.582073+0.005420
## [3]  train-mlogloss:0.439701+0.001688    test-mlogloss:0.458803+0.006250
## [4]  train-mlogloss:0.350911+0.001667    test-mlogloss:0.374481+0.007033
## [5]  train-mlogloss:0.287247+0.001908    test-mlogloss:0.313464+0.006620
## [6]  train-mlogloss:0.240723+0.001949    test-mlogloss:0.269804+0.005800
## [7]  train-mlogloss:0.206256+0.001918    test-mlogloss:0.237903+0.006129
## [8]  train-mlogloss:0.179641+0.001879    test-mlogloss:0.213340+0.006300
## [9]  train-mlogloss:0.159393+0.001654    test-mlogloss:0.194690+0.006132
## [10] train-mlogloss:0.143217+0.001397    test-mlogloss:0.179633+0.006081
## [11] train-mlogloss:0.128701+0.001080    test-mlogloss:0.166953+0.007313
## [12] train-mlogloss:0.117505+0.001122    test-mlogloss:0.157115+0.007048
## [13] train-mlogloss:0.107865+0.001147    test-mlogloss:0.148597+0.008043
## [14] train-mlogloss:0.099304+0.001239    test-mlogloss:0.141288+0.009098
## [15] train-mlogloss:0.092613+0.001031    test-mlogloss:0.135228+0.008515
## [16] train-mlogloss:0.086438+0.000418    test-mlogloss:0.130644+0.009326
## [17] train-mlogloss:0.081161+0.000979    test-mlogloss:0.126499+0.009864
## [18] train-mlogloss:0.076374+0.001234    test-mlogloss:0.122470+0.010556
## [19] train-mlogloss:0.071703+0.001788    test-mlogloss:0.118128+0.011180
## [20] train-mlogloss:0.067554+0.001466    test-mlogloss:0.113930+0.010984
## [21] train-mlogloss:0.064368+0.001575    test-mlogloss:0.111571+0.011514
## [22] train-mlogloss:0.060856+0.002221    test-mlogloss:0.108821+0.012384
## [23] train-mlogloss:0.057282+0.001755    test-mlogloss:0.105806+0.011926
## [24] train-mlogloss:0.054833+0.001523    test-mlogloss:0.103757+0.011738
## [25] train-mlogloss:0.052388+0.001298    test-mlogloss:0.101484+0.010920
## [26] train-mlogloss:0.050190+0.001332    test-mlogloss:0.099611+0.011544
## [27] train-mlogloss:0.048436+0.000997    test-mlogloss:0.097936+0.011349
```

```
## [28] train-mlogloss:0.046576+0.000986    test-mlogloss:0.096245+0.011419
## [29] train-mlogloss:0.044392+0.001250    test-mlogloss:0.094621+0.011574
## [30] train-mlogloss:0.043063+0.001418    test-mlogloss:0.093674+0.011737
## [31] train-mlogloss:0.041624+0.001256    test-mlogloss:0.092688+0.012300
## [32] train-mlogloss:0.039984+0.001223    test-mlogloss:0.091297+0.012208
## [33] train-mlogloss:0.038394+0.001126    test-mlogloss:0.090098+0.011596
## [34] train-mlogloss:0.037296+0.001385    test-mlogloss:0.089659+0.011761
## [35] train-mlogloss:0.036014+0.001338    test-mlogloss:0.088539+0.012119
## [36] train-mlogloss:0.034559+0.001274    test-mlogloss:0.087084+0.011576
## [37] train-mlogloss:0.033366+0.000891    test-mlogloss:0.086319+0.011872
## [38] train-mlogloss:0.032147+0.000987    test-mlogloss:0.085483+0.012008
## [39] train-mlogloss:0.031228+0.000784    test-mlogloss:0.084933+0.012406
## [40] train-mlogloss:0.030131+0.000484    test-mlogloss:0.084364+0.012333
## [41] train-mlogloss:0.029099+0.000812    test-mlogloss:0.083791+0.012526
## [42] train-mlogloss:0.028322+0.000914    test-mlogloss:0.083589+0.012695
## [43] train-mlogloss:0.027388+0.000830    test-mlogloss:0.082804+0.012488
## [44] train-mlogloss:0.026370+0.001037    test-mlogloss:0.082055+0.012042
## [45] train-mlogloss:0.025707+0.000790    test-mlogloss:0.081658+0.012058
## [46] train-mlogloss:0.025085+0.000680    test-mlogloss:0.081124+0.012071
## [47] train-mlogloss:0.024459+0.000518    test-mlogloss:0.080746+0.012281
## [48] train-mlogloss:0.023666+0.000477    test-mlogloss:0.079811+0.011799
## [49] train-mlogloss:0.023006+0.000570    test-mlogloss:0.079360+0.011592
## [50] train-mlogloss:0.022381+0.000677    test-mlogloss:0.078983+0.011985
## [51] train-mlogloss:0.021638+0.000690    test-mlogloss:0.078438+0.011939
## [52] train-mlogloss:0.020983+0.000563    test-mlogloss:0.077987+0.011688
## [53] train-mlogloss:0.020407+0.000569    test-mlogloss:0.077532+0.011838
## [54] train-mlogloss:0.019989+0.000524    test-mlogloss:0.077185+0.012075
## [55] train-mlogloss:0.019473+0.000425    test-mlogloss:0.077005+0.012098
## [56] train-mlogloss:0.018943+0.000439    test-mlogloss:0.076412+0.012158
## [57] train-mlogloss:0.018407+0.000528    test-mlogloss:0.075838+0.012018
## [58] train-mlogloss:0.017912+0.000675    test-mlogloss:0.075449+0.011770
## [59] train-mlogloss:0.017532+0.000864    test-mlogloss:0.075194+0.011433
## [60] train-mlogloss:0.017082+0.000905    test-mlogloss:0.074539+0.011478
## [61] train-mlogloss:0.016572+0.000722    test-mlogloss:0.074107+0.011434
## [62] train-mlogloss:0.016123+0.000742    test-mlogloss:0.073814+0.011409
## [63] train-mlogloss:0.015654+0.000752    test-mlogloss:0.073472+0.011441
## [64] train-mlogloss:0.015319+0.000784    test-mlogloss:0.073154+0.011455
## [65] train-mlogloss:0.014809+0.000698    test-mlogloss:0.073072+0.011557
## [66] train-mlogloss:0.014527+0.000734    test-mlogloss:0.072805+0.011377
## [67] train-mlogloss:0.014220+0.000794    test-mlogloss:0.072737+0.011482
## [68] train-mlogloss:0.013966+0.000854    test-mlogloss:0.072614+0.011528
## [69] train-mlogloss:0.013601+0.000855    test-mlogloss:0.072398+0.011803
## [70] train-mlogloss:0.013310+0.000836    test-mlogloss:0.072209+0.011865
## [71] train-mlogloss:0.013002+0.000749    test-mlogloss:0.072182+0.012086
## [72] train-mlogloss:0.012647+0.000717    test-mlogloss:0.071724+0.012097
## [73] train-mlogloss:0.012369+0.000644    test-mlogloss:0.071658+0.012211
## [74] train-mlogloss:0.012132+0.000645    test-mlogloss:0.071791+0.012361
## [75] train-mlogloss:0.011872+0.000646    test-mlogloss:0.071618+0.012326
## [76] train-mlogloss:0.011680+0.000645    test-mlogloss:0.071535+0.012395
## [77] train-mlogloss:0.011409+0.000666    test-mlogloss:0.071454+0.012368
## [78] train-mlogloss:0.011226+0.000661    test-mlogloss:0.071391+0.012488
## [79] train-mlogloss:0.011022+0.000691    test-mlogloss:0.071413+0.012659
## [80] train-mlogloss:0.010796+0.000635    test-mlogloss:0.071179+0.012768
## [81] train-mlogloss:0.010569+0.000641    test-mlogloss:0.070991+0.012741
```

```
## [82] train-mlogloss:0.010350+0.000679    test-mlogloss:0.070924+0.012885
## [83] train-mlogloss:0.010136+0.000698    test-mlogloss:0.070709+0.013048
## [84] train-mlogloss:0.009957+0.000622    test-mlogloss:0.070616+0.013103
## [85] train-mlogloss:0.009743+0.000592    test-mlogloss:0.070668+0.013133
## [86] train-mlogloss:0.009598+0.000622    test-mlogloss:0.070640+0.013149
## [87] train-mlogloss:0.009387+0.000635    test-mlogloss:0.070342+0.013214
## [88] train-mlogloss:0.009217+0.000574    test-mlogloss:0.070264+0.013223
## [89] train-mlogloss:0.009011+0.000496    test-mlogloss:0.070340+0.013233
## [90] train-mlogloss:0.008844+0.000440    test-mlogloss:0.070215+0.013328
## [91] train-mlogloss:0.008710+0.000449    test-mlogloss:0.070283+0.013442
## [92] train-mlogloss:0.008558+0.000442    test-mlogloss:0.070478+0.013375
## [93] train-mlogloss:0.008411+0.000418    test-mlogloss:0.070515+0.013409
## [94] train-mlogloss:0.008287+0.000410    test-mlogloss:0.070624+0.013376
## [95] train-mlogloss:0.008163+0.000374    test-mlogloss:0.070550+0.013237
## [96] train-mlogloss:0.008044+0.000380    test-mlogloss:0.070656+0.013257
## [97] train-mlogloss:0.007921+0.000366    test-mlogloss:0.070681+0.013333
## [98] train-mlogloss:0.007809+0.000350    test-mlogloss:0.070647+0.013491
## [99] train-mlogloss:0.007688+0.000347    test-mlogloss:0.070735+0.013552
## [100]    train-mlogloss:0.007570+0.000345    test-mlogloss:0.070760+0.013460
## Stopping. Best iteration:
## [90] train-mlogloss:0.008844+0.000440    test-mlogloss:0.070215+0.013328
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Delayed Early On Time
##    Delayed     937     0       6
##    Early         0    20       0
##    On Time      10     2     401
##
## Overall Statistics
##
##                Accuracy : 0.9869
##                  95% CI : (0.9794, 0.9922)
##     No Information Rate : 0.6882
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9702
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Delayed Class: Early Class: On Time
## Sensitivity                  0.9894      0.90909         0.9853
## Specificity                  0.9860      1.00000         0.9876
## Pos Pred Value               0.9936      1.00000         0.9709
## Neg Pred Value               0.9769      0.99853         0.9938
## Prevalence                   0.6882      0.01599         0.2958
## Detection Rate               0.6810      0.01453         0.2914
## Detection Prevalence         0.6853      0.01453         0.3001
## Balanced Accuracy            0.9877      0.95455         0.9864
```

4.8 Implementing modeling to predict project_profile

```
# ---Columns to Drop and move forward for modeling---------------------
project_model_data_final_all <- project_model_data_final %>%
  select(-c(
    fms_id,
    latest_budget, latest_spend, cost_diff,
    orig_start_date, orig_end_date, task_end_date,
    cost_class, delay_class, status_combined, delay_days, cost_diff_pct, end_year, start_year
  ))


# ---- Begin Classification Modeling for project_profile ----
library(caret)
library(randomForest)
library(dummy)
```

```
## dummy 0.1.3
```

```
## dummyNews()
```

```
# Convert target and categorical predictors to factors
project_model_data_final_all$project_profile <- as.factor(project_model_data_final_all$project_profile)

# Identify categorical variables (excluding the target)
categorical_vars <- sapply(project_model_data_final_all, is.character)
categorical_vars <- names(categorical_vars[categorical_vars])
categorical_vars <- setdiff(categorical_vars, "project_profile")

# Convert character columns to factors
project_model_data_final_all[categorical_vars] <- lapply(project_model_data_final_all[categorical_vars]

# Partition data into training and testing sets
set.seed(123)
train_index <- createDataPartition(project_model_data_final_all$project_profile, p = 0.8, list = FALSE)
train_data <- project_model_data_final_all[train_index, ]
test_data <- project_model_data_final_all[-train_index, ]

# Cross-validated and tuned Random Forest using caret
control <- trainControl(method = "cv", number = 5, search = "grid")
tunegrid <- expand.grid(.mtry = c(2, 4, 6, 8))

set.seed(123)
rf_model <- train(
  project_profile ~ .,
  data = train_data,
  method = "rf",
  metric = "Accuracy",
  trControl = control,
  tuneGrid = tunegrid,
  importance = TRUE
)

print(rf_model)
```

```
## Random Forest
##
## 5513 samples
##   12 predictor
##    9 classes: 'Cost Risk', 'Exceptional Performance', 'Fast but Costly', 'High Risk', 'Lean Delivery
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4410, 4409, 4410, 4410, 4413
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.6406605  0.3496235
##   4     0.7895831  0.6549575
##   6     0.8699421  0.7962781
##   8     0.8962442  0.8398127
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```r
# Predict on test data
rf_preds <- predict(rf_model, newdata = test_data)

# Evaluate performance
conf_mat <- confusionMatrix(rf_preds, test_data$project_profile)
print(conf_mat)
```

```
## Confusion Matrix and Statistics
##
##                          Reference
## Prediction                Cost Risk Exceptional Performance Fast but Costly
##   Cost Risk                      46                       0               0
##   Exceptional Performance         0                       2               0
##   Fast but Costly                 0                       0               6
##   High Risk                       0                       0               0
##   Lean Delivery                   0                       0               0
##   On Track                       29                       0               0
##   Schedule Risk                   1                       0               0
##   Strong Delivery                 0                       0               0
##   Time Risk, Cost-Saving          0                       0               0
##                          Reference
## Prediction                High Risk Lean Delivery On Track Schedule Risk
##   Cost Risk                       0             0        8             0
##   Exceptional Performance         0             0        0             0
##   Fast but Costly                 0             0        0             0
##   High Risk                     177             0        2             3
##   Lean Delivery                   0             6        0             0
##   On Track                        1            12      291            13
##   Schedule Risk                  16             3        8           685
##   Strong Delivery                 3             0        0             0
##   Time Risk, Cost-Saving          0             0        0             0
##                          Reference
## Prediction                Strong Delivery Time Risk, Cost-Saving
##   Cost Risk                             0                      0
##   Exceptional Performance               0                      0
```

```
##   Fast but Costly                      0                     0
##   High Risk                            0                     2
##   Lean Delivery                        0                     0
##   On Track                             1                     2
##   Schedule Risk                        1                    14
##   Strong Delivery                     11                     0
##   Time Risk, Cost-Saving               0                    30
##
## Overall Statistics
##
##                Accuracy : 0.9133
##                  95% CI : (0.8972, 0.9277)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8666
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Cost Risk Class: Exceptional Performance
## Sensitivity                   0.60526                        1.000000
## Specificity                   0.99383                        1.000000
## Pos Pred Value                0.85185                        1.000000
## Neg Pred Value                0.97726                        1.000000
## Prevalence                    0.05535                        0.001457
## Detection Rate                0.03350                        0.001457
## Detection Prevalence          0.03933                        0.001457
## Balanced Accuracy             0.79955                        1.000000
##                      Class: Fast but Costly Class: High Risk
## Sensitivity                         1.00000           0.8985
## Specificity                         1.00000           0.9940
## Pos Pred Value                      1.00000           0.9620
## Neg Pred Value                      1.00000           0.9832
## Prevalence                          0.00437           0.1435
## Detection Rate                      0.00437           0.1289
## Detection Prevalence                0.00437           0.1340
## Balanced Accuracy                   1.00000           0.9463
##                      Class: Lean Delivery Class: On Track Class: Schedule Risk
## Sensitivity                       0.28571          0.9417               0.9772
## Specificity                       1.00000          0.9455               0.9360
## Pos Pred Value                    1.00000          0.8338               0.9409
## Neg Pred Value                    0.98903          0.9824               0.9752
## Prevalence                        0.01529          0.2251               0.5106
## Detection Rate                    0.00437          0.2119               0.4989
## Detection Prevalence              0.00437          0.2542               0.5302
## Balanced Accuracy                 0.64286          0.9436               0.9566
##                      Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                        0.846154                       0.62500
## Specificity                        0.997794                       1.00000
## Pos Pred Value                     0.785714                       1.00000
## Neg Pred Value                     0.998528                       0.98660
## Prevalence                         0.009468                       0.03496
```

```
## Detection Rate                    0.008012                    0.02185
## Detection Prevalence              0.010197                    0.02185
## Balanced Accuracy                 0.921974                    0.81250
```

```r
# View feature importance
#importance(rf_model)
varImp(rf_model)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 60)
##
##                                      Cost Risk Exceptional Performance
## spend_to_date_percent                    61.10                  23.959
## initial_budget                           82.61                  33.479
## avg_temp_anomaly                         61.97                  24.554
## avg_max_wind_speed                       68.82                  29.555
## avg_labor_unemp_rate                     58.47                  23.388
## avg_construction_jobs                    53.98                  22.891
## avg_cyclone_freq                         57.36                  26.728
## project_themeRoof Work                   29.43                  20.750
## project_themeInterior Renovation         26.34                  22.459
## category_groupHousing                    32.65                   8.475
## project_themeSafety/Street Improvements  38.60                  13.299
## current_phaseConstruction                35.83                  15.690
## boroughManhattan                         35.65                  12.370
## category_groupParks & Recreation         38.07                  20.420
## boroughBrooklyn                          36.38                  17.837
## delay_categoryFunding/Financial Issues   43.75                  22.437
## boroughQueens                            29.18                  18.525
## category_groupPublic Buildings           36.00                  26.845
## category_groupTransportation             31.80                  10.521
## category_groupMiscellaneous / Other      32.47                  10.866
##                                      Fast but Costly High Risk Lean Delivery
## spend_to_date_percent                             30.883     84.00         35.95
## initial_budget                                    38.122     93.01         56.68
## avg_temp_anomaly                                  36.589     77.49         49.76
## avg_max_wind_speed                                41.597     76.75         54.77
## avg_labor_unemp_rate                              38.040     74.72         45.30
## avg_construction_jobs                             28.962     68.41         44.03
## avg_cyclone_freq                                  34.927     66.10         50.47
## project_themeRoof Work                            24.832     63.19         16.18
## project_themeInterior Renovation                  28.468     47.19         33.71
## category_groupHousing                              9.097     55.12         12.45
## project_themeSafety/Street Improvements           17.740     50.28         25.81
## current_phaseConstruction                         23.587     50.02         17.47
## boroughManhattan                                  25.233     49.85         26.62
## category_groupParks & Recreation                  25.855     44.63         31.67
## boroughBrooklyn                                   16.225     48.07         21.29
## delay_categoryFunding/Financial Issues            30.187     48.00         33.20
## boroughQueens                                     19.755     43.47         28.17
## category_groupPublic Buildings                    17.104     47.77         30.20
## category_groupTransportation                      21.241     39.73         22.63
## category_groupMiscellaneous / Other               23.258     44.05         23.96
```

```
##                                       On Track Schedule Risk Strong Delivery
## spend_to_date_percent                    66.49        100.00          41.984
## initial_budget                           78.06         94.80          60.048
## avg_temp_anomaly                         65.95         68.23          44.985
## avg_max_wind_speed                       70.36         69.29          51.106
## avg_labor_unemp_rate                     59.77         69.00          41.994
## avg_construction_jobs                    52.14         65.43          36.950
## avg_cyclone_freq                         60.30         65.57          45.266
## project_themeRoof Work                   32.38         43.58          13.826
## project_themeInterior Renovation         41.74         55.56          30.971
## category_groupHousing                    23.69         34.96           9.453
## project_themeSafety/Street Improvements  40.96         48.94          25.305
## current_phaseConstruction                42.46         47.74          25.319
## boroughManhattan                         40.21         41.69          19.798
## category_groupParks & Recreation         44.29         48.71          30.198
## boroughBrooklyn                          42.79         43.07          25.501
## delay_categoryFunding/Financial Issues   45.11         46.22          37.463
## boroughQueens                            38.73         47.87          23.607
## category_groupPublic Buildings           39.93         45.72          19.101
## category_groupTransportation             39.65         47.66          30.363
## category_groupMiscellaneous / Other      45.34         47.54          16.924
##                                       Time Risk, Cost-Saving
## spend_to_date_percent                                  61.51
## initial_budget                                         77.21
## avg_temp_anomaly                                       71.30
## avg_max_wind_speed                                     66.71
## avg_labor_unemp_rate                                   68.95
## avg_construction_jobs                                  69.69
## avg_cyclone_freq                                       62.76
## project_themeRoof Work                                 45.19
## project_themeInterior Renovation                       40.33
## category_groupHousing                                  19.66
## project_themeSafety/Street Improvements                34.29
## current_phaseConstruction                              37.00
## boroughManhattan                                       31.74
## category_groupParks & Recreation                       41.15
## boroughBrooklyn                                        36.26
## delay_categoryFunding/Financial Issues                 42.38
## boroughQueens                                          39.35
## category_groupPublic Buildings                         46.24
## category_groupTransportation                           27.51
## category_groupMiscellaneous / Other                    36.60
```

```r
# ---- Train Decision Tree and XGBoost Models for Comparison ----

# Decision Tree with caret cross-validation
dt_control <- trainControl(method = "cv", number = 5)
dt_model <- train(
  project_profile ~ .,
  data = train_data,
  method = "rpart",
  trControl = dt_control,
  tuneLength = 10
)
```

```r
print(dt_model)
```

```
## CART
##
## 5513 samples
##   12 predictor
##    9 classes: 'Cost Risk', 'Exceptional Performance', 'Fast but Costly', 'High Risk', 'Lean Delivery
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4410, 4411, 4411, 4409, 4411
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.004062038  0.7046990  0.5137108
##   0.004431315  0.7023413  0.5101816
##   0.005169867  0.6965396  0.4986341
##   0.005723781  0.6914607  0.4865845
##   0.005908419  0.6898273  0.4827424
##   0.006277696  0.6909123  0.4840161
##   0.008677991  0.6767659  0.4487722
##   0.010339734  0.6678780  0.4307800
##   0.022895126  0.6584435  0.4043323
##   0.098350566  0.5650084  0.1536059
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.004062038.
```

```r
predict_dt <- predict(dt_model, test_data)
conf_mat_dt <- confusionMatrix(predict_dt, test_data$project_profile)
print(conf_mat_dt)
```

```
## Confusion Matrix and Statistics
##
##                          Reference
## Prediction                Cost Risk Exceptional Performance Fast but Costly
##   Cost Risk                       0                       0              0
##   Exceptional Performance         0                       0              0
##   Fast but Costly                 0                       0              0
##   High Risk                       0                       0              0
##   Lean Delivery                   0                       0              0
##   On Track                       69                       2              2
##   Schedule Risk                   7                       0              4
##   Strong Delivery                 0                       0              0
##   Time Risk, Cost-Saving          0                       0              0
##                          Reference
## Prediction                High Risk Lean Delivery On Track Schedule Risk
##   Cost Risk                       0             0        0             0
##   Exceptional Performance         0             0        0             0
##   Fast but Costly                 0             0        0             0
##   High Risk                      47             0        2            14
##   Lean Delivery                   0             0        0             0
##   On Track                       16            19      275            46
##   Schedule Risk                 134             2       32           641
```

```
##   Strong Delivery                 0             0        0             0
##   Time Risk, Cost-Saving          0             0        0             0
##                          Reference
## Prediction                 Strong Delivery Time Risk, Cost-Saving
##   Cost Risk                              0                      0
##   Exceptional Performance                0                      0
##   Fast but Costly                        0                      0
##   High Risk                              0                      0
##   Lean Delivery                          0                      0
##   On Track                               9                      6
##   Schedule Risk                          4                     42
##   Strong Delivery                        0                      0
##   Time Risk, Cost-Saving                 0                      0
##
## Overall Statistics
##
##                Accuracy : 0.7014
##                  95% CI : (0.6764, 0.7255)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5012
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Cost Risk Class: Exceptional Performance
## Sensitivity                   0.00000                       0.000000
## Specificity                   1.00000                       1.000000
## Pos Pred Value                    NaN                            NaN
## Neg Pred Value                0.94465                       0.998543
## Prevalence                    0.05535                       0.001457
## Detection Rate                0.00000                       0.000000
## Detection Prevalence          0.00000                       0.000000
## Balanced Accuracy             0.50000                       0.500000
##                      Class: Fast but Costly Class: High Risk
## Sensitivity                         0.00000          0.23858
## Specificity                         1.00000          0.98639
## Pos Pred Value                          NaN          0.74603
## Neg Pred Value                      0.99563          0.88550
## Prevalence                          0.00437          0.14348
## Detection Rate                      0.00000          0.03423
## Detection Prevalence                0.00000          0.04588
## Balanced Accuracy                   0.50000          0.61249
##                      Class: Lean Delivery Class: On Track Class: Schedule Risk
## Sensitivity                       0.00000          0.8900               0.9144
## Specificity                       1.00000          0.8412               0.6652
## Pos Pred Value                        NaN          0.6194               0.7402
## Neg Pred Value                    0.98471          0.9634               0.8817
## Prevalence                        0.01529          0.2251               0.5106
## Detection Rate                    0.00000          0.2003               0.4669
## Detection Prevalence              0.00000          0.3234               0.6307
## Balanced Accuracy                 0.50000          0.8656               0.7898
```

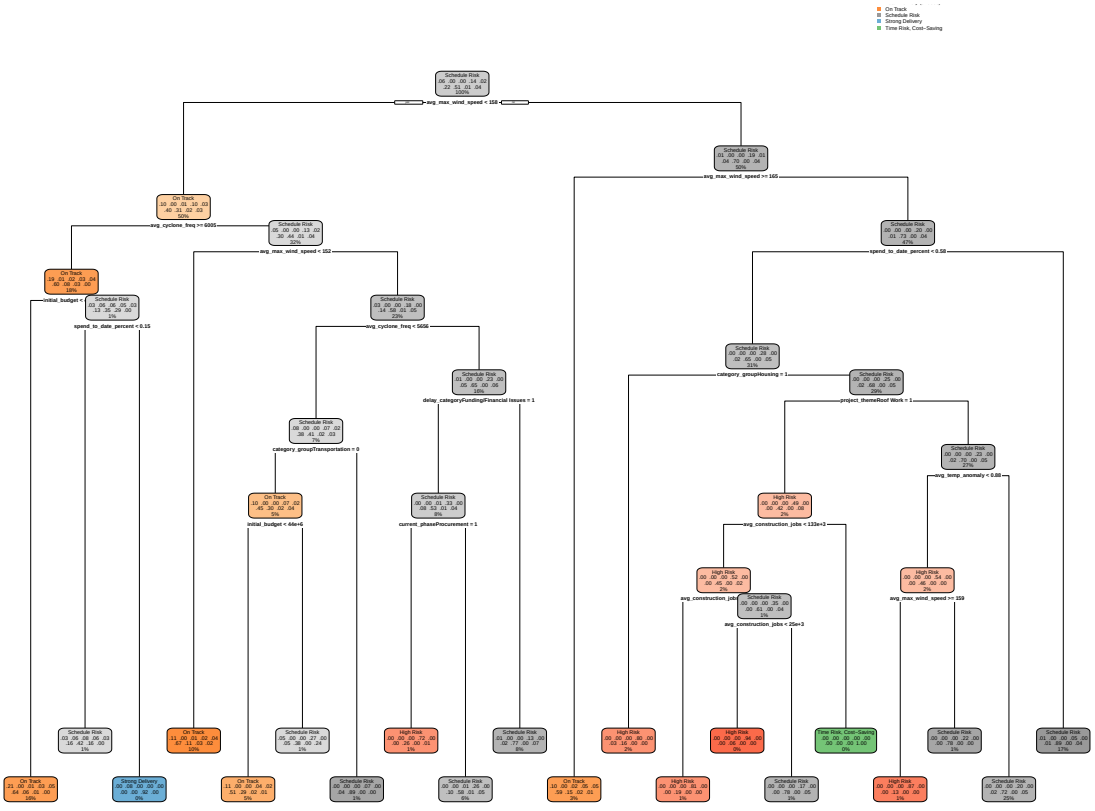
```
##                          Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                            0.000000                       0.00000
## Specificity                            1.000000                       1.00000
## Pos Pred Value                              NaN                           NaN
## Neg Pred Value                         0.990532                       0.96504
## Prevalence                             0.009468                       0.03496
## Detection Rate                         0.000000                       0.00000
## Detection Prevalence                   0.000000                       0.00000
## Balanced Accuracy                      0.500000                       0.50000
```

```
png(filename = "decision_tree_project_profile.png", width = 2400, height = 1600, res = 300)
rpart.plot(dt_model$finalModel)
dev.off()
```

```
## pdf
##   2
```

```
# Plot the decision tree
library(rpart.plot)
rpart.plot(dt_model$finalModel)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
# XGBoost with cross-validation and early stopping
library(xgboost)
library(Matrix)

# Convert data to numeric matrix format
train_matrix <- model.matrix(project_profile ~ . -1, data = train_data)
test_matrix <- model.matrix(project_profile ~ . -1, data = test_data)
```

```r
xgb_train <- xgb.DMatrix(data = train_matrix, label = as.numeric(train_data$project_profile) - 1)
xgb_test <- xgb.DMatrix(data = test_matrix, label = as.numeric(test_data$project_profile) - 1)

xgb_cv <- xgb.cv(
  data = xgb_train,
  nrounds = 100,
  nfold = 5,
  early_stopping_rounds = 10,
  objective = "multi:softmax",
  num_class = length(unique(train_data$project_profile)),
  verbose = 1
)
```

```
## [1]  train-mlogloss:1.538448+0.003184    test-mlogloss:1.573391+0.009930
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 10 rounds.
##
## [2]  train-mlogloss:1.255202+0.002049    test-mlogloss:1.311124+0.012697
## [3]  train-mlogloss:1.068688+0.004032    test-mlogloss:1.141315+0.013592
## [4]  train-mlogloss:0.934396+0.004454    test-mlogloss:1.021040+0.013085
## [5]  train-mlogloss:0.828261+0.006825    test-mlogloss:0.927295+0.014465
## [6]  train-mlogloss:0.745949+0.003785    test-mlogloss:0.857075+0.013639
## [7]  train-mlogloss:0.680689+0.003894    test-mlogloss:0.800316+0.013597
## [8]  train-mlogloss:0.625764+0.004094    test-mlogloss:0.753626+0.013828
## [9]  train-mlogloss:0.582376+0.003866    test-mlogloss:0.716044+0.013671
## [10] train-mlogloss:0.545349+0.004720    test-mlogloss:0.684057+0.015577
## [11] train-mlogloss:0.513356+0.005031    test-mlogloss:0.656567+0.014331
## [12] train-mlogloss:0.487211+0.005040    test-mlogloss:0.633058+0.015555
## [13] train-mlogloss:0.463113+0.004635    test-mlogloss:0.611580+0.015504
## [14] train-mlogloss:0.442049+0.006047    test-mlogloss:0.593196+0.016473
## [15] train-mlogloss:0.422905+0.003640    test-mlogloss:0.576368+0.012568
## [16] train-mlogloss:0.408255+0.002965    test-mlogloss:0.562931+0.012301
## [17] train-mlogloss:0.394709+0.002960    test-mlogloss:0.550212+0.011778
## [18] train-mlogloss:0.381086+0.002616    test-mlogloss:0.538846+0.011323
## [19] train-mlogloss:0.369030+0.002524    test-mlogloss:0.527829+0.011907
## [20] train-mlogloss:0.358033+0.002859    test-mlogloss:0.518567+0.011454
## [21] train-mlogloss:0.348018+0.002388    test-mlogloss:0.510054+0.010090
## [22] train-mlogloss:0.336591+0.003167    test-mlogloss:0.499829+0.011726
## [23] train-mlogloss:0.326574+0.001957    test-mlogloss:0.490893+0.011685
## [24] train-mlogloss:0.316833+0.003165    test-mlogloss:0.481608+0.012679
## [25] train-mlogloss:0.308420+0.002989    test-mlogloss:0.475257+0.012423
## [26] train-mlogloss:0.299926+0.002643    test-mlogloss:0.467687+0.012539
## [27] train-mlogloss:0.291906+0.003121    test-mlogloss:0.460804+0.012580
## [28] train-mlogloss:0.283725+0.001192    test-mlogloss:0.453562+0.010793
## [29] train-mlogloss:0.275775+0.001640    test-mlogloss:0.446862+0.010407
## [30] train-mlogloss:0.269387+0.001591    test-mlogloss:0.441290+0.009482
## [31] train-mlogloss:0.262898+0.002804    test-mlogloss:0.435103+0.008512
## [32] train-mlogloss:0.256170+0.002735    test-mlogloss:0.428946+0.009351
## [33] train-mlogloss:0.250368+0.003113    test-mlogloss:0.424274+0.009146
## [34] train-mlogloss:0.245005+0.002612    test-mlogloss:0.419687+0.009886
## [35] train-mlogloss:0.238470+0.002437    test-mlogloss:0.414502+0.010784
## [36] train-mlogloss:0.233017+0.002105    test-mlogloss:0.410054+0.010526
## [37] train-mlogloss:0.226977+0.001757    test-mlogloss:0.404513+0.011216
```

```
## [38] train-mlogloss:0.220741+0.002382    test-mlogloss:0.399350+0.010674
## [39] train-mlogloss:0.215777+0.003999    test-mlogloss:0.395131+0.011060
## [40] train-mlogloss:0.211142+0.004819    test-mlogloss:0.391334+0.011225
## [41] train-mlogloss:0.205722+0.005115    test-mlogloss:0.386975+0.010011
## [42] train-mlogloss:0.200200+0.003481    test-mlogloss:0.382298+0.009738
## [43] train-mlogloss:0.195697+0.002545    test-mlogloss:0.378784+0.010527
## [44] train-mlogloss:0.191199+0.003185    test-mlogloss:0.375342+0.010387
## [45] train-mlogloss:0.186281+0.003974    test-mlogloss:0.371367+0.009922
## [46] train-mlogloss:0.182137+0.004083    test-mlogloss:0.368068+0.009350
## [47] train-mlogloss:0.178616+0.003642    test-mlogloss:0.365256+0.009597
## [48] train-mlogloss:0.174589+0.003548    test-mlogloss:0.361829+0.010168
## [49] train-mlogloss:0.171141+0.003616    test-mlogloss:0.358817+0.010312
## [50] train-mlogloss:0.166976+0.003629    test-mlogloss:0.355241+0.010685
## [51] train-mlogloss:0.161802+0.003667    test-mlogloss:0.351076+0.011087
## [52] train-mlogloss:0.158060+0.002926    test-mlogloss:0.348199+0.011427
## [53] train-mlogloss:0.154951+0.002921    test-mlogloss:0.346305+0.011273
## [54] train-mlogloss:0.151993+0.002849    test-mlogloss:0.344059+0.011637
## [55] train-mlogloss:0.148380+0.002693    test-mlogloss:0.340795+0.011032
## [56] train-mlogloss:0.144134+0.002386    test-mlogloss:0.337162+0.011562
## [57] train-mlogloss:0.141528+0.002774    test-mlogloss:0.335167+0.011963
## [58] train-mlogloss:0.138442+0.003080    test-mlogloss:0.332765+0.012325
## [59] train-mlogloss:0.135550+0.003058    test-mlogloss:0.330249+0.012342
## [60] train-mlogloss:0.131942+0.002996    test-mlogloss:0.326992+0.012395
## [61] train-mlogloss:0.128192+0.003105    test-mlogloss:0.322949+0.012249
## [62] train-mlogloss:0.125663+0.002892    test-mlogloss:0.320991+0.011916
## [63] train-mlogloss:0.122653+0.002530    test-mlogloss:0.318462+0.012439
## [64] train-mlogloss:0.120014+0.002496    test-mlogloss:0.316119+0.013297
## [65] train-mlogloss:0.117752+0.002738    test-mlogloss:0.314001+0.013377
## [66] train-mlogloss:0.115002+0.002570    test-mlogloss:0.312018+0.013524
## [67] train-mlogloss:0.112626+0.002386    test-mlogloss:0.310416+0.013934
## [68] train-mlogloss:0.110392+0.002361    test-mlogloss:0.308465+0.013888
## [69] train-mlogloss:0.107595+0.002237    test-mlogloss:0.306199+0.014506
## [70] train-mlogloss:0.105628+0.002222    test-mlogloss:0.303993+0.013759
## [71] train-mlogloss:0.103564+0.002234    test-mlogloss:0.302323+0.013928
## [72] train-mlogloss:0.101838+0.002012    test-mlogloss:0.300945+0.013826
## [73] train-mlogloss:0.099436+0.001204    test-mlogloss:0.299182+0.014007
## [74] train-mlogloss:0.097203+0.001381    test-mlogloss:0.297301+0.014013
## [75] train-mlogloss:0.095082+0.001402    test-mlogloss:0.295780+0.014168
## [76] train-mlogloss:0.092955+0.001263    test-mlogloss:0.293750+0.013709
## [77] train-mlogloss:0.090984+0.001313    test-mlogloss:0.292021+0.013624
## [78] train-mlogloss:0.089424+0.001290    test-mlogloss:0.290685+0.013677
## [79] train-mlogloss:0.087460+0.001510    test-mlogloss:0.288624+0.013643
## [80] train-mlogloss:0.085454+0.001183    test-mlogloss:0.286696+0.013784
## [81] train-mlogloss:0.084042+0.001280    test-mlogloss:0.285296+0.013651
## [82] train-mlogloss:0.082207+0.001352    test-mlogloss:0.283344+0.014033
## [83] train-mlogloss:0.080749+0.001666    test-mlogloss:0.282181+0.013783
## [84] train-mlogloss:0.079381+0.001810    test-mlogloss:0.280886+0.013867
## [85] train-mlogloss:0.078057+0.002091    test-mlogloss:0.279990+0.014225
## [86] train-mlogloss:0.076598+0.001976    test-mlogloss:0.278839+0.014038
## [87] train-mlogloss:0.075262+0.001737    test-mlogloss:0.277894+0.014254
## [88] train-mlogloss:0.073988+0.001934    test-mlogloss:0.277128+0.014418
## [89] train-mlogloss:0.072811+0.001765    test-mlogloss:0.276232+0.014103
## [90] train-mlogloss:0.071393+0.001582    test-mlogloss:0.275253+0.014369
## [91] train-mlogloss:0.070063+0.001526    test-mlogloss:0.273780+0.014561
```

```
## [92] train-mlogloss:0.069120+0.001790    test-mlogloss:0.273124+0.014253
## [93] train-mlogloss:0.067805+0.001854    test-mlogloss:0.272103+0.013841
## [94] train-mlogloss:0.066475+0.001722    test-mlogloss:0.271078+0.014042
## [95] train-mlogloss:0.065224+0.001788    test-mlogloss:0.269863+0.014120
## [96] train-mlogloss:0.063851+0.001922    test-mlogloss:0.268727+0.014214
## [97] train-mlogloss:0.062626+0.002156    test-mlogloss:0.267636+0.014582
## [98] train-mlogloss:0.061477+0.002133    test-mlogloss:0.267021+0.014946
## [99] train-mlogloss:0.060311+0.002064    test-mlogloss:0.266012+0.014948
## [100]    train-mlogloss:0.059083+0.001946    test-mlogloss:0.264852+0.015073
```

```r
best_nrounds <- xgb_cv$best_iteration

xgb_model <- xgboost(
  data = xgb_train,
  nrounds = best_nrounds,
  objective = "multi:softmax",
  num_class = length(unique(train_data$project_profile)),
  verbose = 0
)

xgb_preds <- predict(xgb_model, xgb_test)
xgb_preds_factor <- factor(xgb_preds + 1, levels = 1:length(levels(train_data$project_profile)),
                           labels = levels(train_data$project_profile))

conf_mat_xgb <- confusionMatrix(xgb_preds_factor, test_data$project_profile)
print(conf_mat_xgb)
```

```
## Confusion Matrix and Statistics
##
##                           Reference
## Prediction                 Cost Risk Exceptional Performance Fast but Costly
##   Cost Risk                       67                       0              0
##   Exceptional Performance          0                       1              0
##   Fast but Costly                  0                       0              6
##   High Risk                        0                       1              0
##   Lean Delivery                    0                       0              0
##   On Track                         8                       0              0
##   Schedule Risk                    1                       0              0
##   Strong Delivery                  0                       0              0
##   Time Risk, Cost-Saving           0                       0              0
##                           Reference
## Prediction                 High Risk Lean Delivery On Track Schedule Risk
##   Cost Risk                        0             0       13             0
##   Exceptional Performance          0             0        0             0
##   Fast but Costly                  0             0        0             0
##   High Risk                      184             0        4             5
##   Lean Delivery                    0            15        1             0
##   On Track                         2             4      288             9
##   Schedule Risk                   10             2        3           684
##   Strong Delivery                  0             0        0             0
##   Time Risk, Cost-Saving           1             0        0             3
##                           Reference
## Prediction                 Strong Delivery Time Risk, Cost-Saving
##   Cost Risk                              0                      0
##   Exceptional Performance                0                      0
```

```
##   Fast but Costly                      0                     0
##   High Risk                            0                     0
##   Lean Delivery                        0                     0
##   On Track                             0                     2
##   Schedule Risk                        1                    10
##   Strong Delivery                     12                     0
##   Time Risk, Cost-Saving               0                    36
##
## Overall Statistics
##
##                Accuracy : 0.9417
##                  95% CI : (0.928, 0.9535)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9117
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Cost Risk Class: Exceptional Performance
## Sensitivity                   0.88158                       0.5000000
## Specificity                   0.98998                       1.0000000
## Pos Pred Value                0.83750                       1.0000000
## Neg Pred Value                0.99304                       0.9992711
## Prevalence                    0.05535                       0.0014567
## Detection Rate                0.04880                       0.0007283
## Detection Prevalence          0.05827                       0.0007283
## Balanced Accuracy             0.93578                       0.7500000
##                      Class: Fast but Costly Class: High Risk
## Sensitivity                         1.00000           0.9340
## Specificity                         1.00000           0.9915
## Pos Pred Value                      1.00000           0.9485
## Neg Pred Value                      1.00000           0.9890
## Prevalence                          0.00437           0.1435
## Detection Rate                      0.00437           0.1340
## Detection Prevalence                0.00437           0.1413
## Balanced Accuracy                   1.00000           0.9628
##                      Class: Lean Delivery Class: On Track Class: Schedule Risk
## Sensitivity                       0.71429          0.9320                0.9757
## Specificity                       0.99926          0.9765                0.9598
## Pos Pred Value                    0.93750          0.9201                0.9620
## Neg Pred Value                    0.99558          0.9802                0.9743
## Prevalence                        0.01529          0.2251                0.5106
## Detection Rate                    0.01092          0.2098                0.4982
## Detection Prevalence              0.01165          0.2280                0.5178
## Balanced Accuracy                 0.85677          0.9543                0.9678
##                      Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                        0.923077                       0.75000
## Specificity                        1.000000                       0.99698
## Pos Pred Value                     1.000000                       0.90000
## Neg Pred Value                     0.999265                       0.99100
## Prevalence                         0.009468                       0.03496
```

```
## Detection Rate                 0.008740                 0.02622
## Detection Prevalence           0.008740                 0.02913
## Balanced Accuracy              0.961538                 0.87349
```

---

5. Updated Modeling with restructured data (same data as above being WIDER now)

5.1 Little cleaning

```
# Load libraries
library(dplyr)
library(readr)

# Load udpdated project_model_data_final with each line representing unique record of unique project
project_model_data_final_updated <- read_csv("project_model_data_final_updated.csv")
```

```
## Rows: 1941 Columns: 36
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (8): fms_id, cost_class, delay_class, status_combined, borough, project...
## dbl (28): spend_to_date_percent, delay_cat_contractor_issues, delay_cat_fund...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Standardize all numeric variables
#numeric_cols <- sapply(project_model_data_final_updated, is.numeric)
#project_model_data_final_updated[, numeric_cols] <- scale(project_model_data_final_updated[, numeric_c

# Scale weather variables
weather_scaled <- scale(project_model_data_final_updated %>%
  select(avg_cyclone_freq, avg_temp_anomaly, avg_max_wind_speed))

# Run k-means clustering
set.seed(123)
weather_clusters <- kmeans(weather_scaled, centers = 3)

# Assign cluster labels as a categorical feature
project_model_data_final_updated$weather_risk_category <- as.factor(weather_clusters$cluster)

# Optional: Rename levels by risk (based on cluster centroids)
centroids <- weather_clusters$centers
cluster_order <- order(rowMeans(centroids))  # Order clusters by average risk
levels(project_model_data_final_updated$weather_risk_category) <- c("Low", "Moderate", "High")[cluster_

# Check the distribution
#table(project_model_data_final_updated$weather_risk_category)
write_csv(project_model_data_final_updated, "project_model_data_final_updated.csv")
```

5.2 [Same as 4.7] Predicting cost_class and delay_class separately

```r
# ---- Setup for Predicting cost_class and delay_class separately ----
library(caret)
library(randomForest)
library(rpart.plot)
library(xgboost)
library(Matrix)
set.seed(123)

# --- Dataset for cost_class prediction (exclude cost_diff_pct and cost_class) ---
cost_model_data_updated <- project_model_data_final_updated %>%
  select(-c(avg_cyclone_freq, avg_temp_anomaly, avg_max_wind_speed, status_combined, cost_class, fms_id

# Add cost_class back as target
cost_model_data_updated$cost_class <- as.factor(project_model_data_final_updated$cost_class)

# --- Dataset for delay_class prediction (exclude delay_days and delay_class) ---#######
delay_model_data_updated <- project_model_data_final_updated %>%
  select(-c(avg_cyclone_freq, avg_temp_anomaly, avg_max_wind_speed, status_combined, delay_class, fms_i

# Add delay_class back as target
delay_model_data_updated$delay_class <- as.factor(project_model_data_final_updated$delay_class)

# --- Reusable modeling pipeline function ---
run_models <- function(data, target_col) {
  # Set target variable
  data[[target_col]] <- as.factor(data[[target_col]])

  # Convert character to factor
  categorical_vars <- sapply(data, is.character)
  categorical_vars <- names(categorical_vars[categorical_vars])
  categorical_vars <- setdiff(categorical_vars, target_col)
  data[categorical_vars] <- lapply(data[categorical_vars], as.factor)

  # Train-test split
  set.seed(123)
  train_index <- createDataPartition(data[[target_col]], p = 0.8, list = FALSE)
  train_data <- data[train_index, ]
  test_data <- data[-train_index, ]

  # --Random Forest:Cross-validated and tuned Random Forest using caret---
  control <- trainControl(method = "cv", number = 5)
  tunegrid <- expand.grid(.mtry = c(2, 4, 6, 8))

  cat("\n--- RANDOM FOREST RESULTS for", target_col, "---\n")
  rf_model <- train(
    as.formula(paste(target_col, "~ .")),
    data = train_data,
    method = "rf",
    metric = "Accuracy",
    trControl = control,
    tuneGrid = tunegrid
  )
```

```
  print(rf_model)

  rf_preds <- predict(rf_model, newdata = test_data)
  print(confusionMatrix(rf_preds, test_data[[target_col]]))
  print(varImp(rf_model))

  # --- Decision Tree with caret cross-validation---
  dt_control <- trainControl(method = "cv", number = 5)

  cat("\n--- DECISION TREE RESULTS for", target_col, "---\n")
  dt_model <- train(
    as.formula(paste(target_col, "~ .")),
    data = train_data,
    method = "rpart",
    trControl = dt_control,
    tuneLength = 10
  )
  print(dt_model)
  dt_preds <- predict(dt_model, test_data)
  print(confusionMatrix(dt_preds, test_data[[target_col]]))
  png(filename = paste0("decision_tree_", target_col, ".png"), width = 2400, height = 1600, res = 300)
rpart.plot(dt_model$finalModel, box.palette = "Greens")
dev.off()

# Also show the trees in the console
rpart.plot(dt_model$finalModel, box.palette = "Greens")

  # --- XGBoost with cross-validation and early stopping---
  train_matrix <- model.matrix(as.formula(paste(target_col, "~ .")), data = train_data)
  test_matrix <- model.matrix(as.formula(paste(target_col, "~ .")), data = test_data)

  xgb_train <- xgb.DMatrix(data = train_matrix, label = as.numeric(train_data[[target_col]]) - 1)
  xgb_test <- xgb.DMatrix(data = test_matrix, label = as.numeric(test_data[[target_col]]) - 1)

  cat("\n--- XGBOOST RESULTS for", target_col, "---\n")
  xgb_cv <- xgb.cv(
    data = xgb_train,
    nrounds = 100,
    nfold = 5,
    early_stopping_rounds = 30,
    objective = "multi:softmax",
    num_class = length(unique(train_data[[target_col]])),
    verbose = 1
  )
  best_nrounds <- xgb_cv$best_iteration

  xgb_model <- xgboost(
    data = xgb_train,
    nrounds = best_nrounds,
    objective = "multi:softmax",
    num_class = length(unique(train_data[[target_col]])),
    verbose = 0
  )
```

```r
  xgb_preds <- predict(xgb_model, xgb_test)
  xgb_preds_factor <- factor(xgb_preds + 1, levels = 1:length(levels(train_data[[target_col]])),
                             labels = levels(train_data[[target_col]]))
  print(confusionMatrix(xgb_preds_factor, test_data[[target_col]]))
}



# Create and open log file
#log_file <- file("model_output_log.txt", open = "wt")

# Redirect all console output to the file
#sink(log_file)

# Optional: Add a header with timestamp
#cat("==== MODEL RUN START ====\n")
#cat("Timestamp: ", Sys.time(), "\n\n")


# ---- Run for cost_class and delay_class ----
suppressWarnings({
  run_models(cost_model_data_updated, "cost_class")
  run_models(delay_model_data_updated, "delay_class")
})
```

```
##
## --- RANDOM FOREST RESULTS for cost_class ---
## Random Forest
##
## 1554 samples
##   22 predictor
##    3 classes: 'On Budget', 'Over Budget', 'Under Budget'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1243, 1244, 1244, 1243, 1242
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7554725  0.00000000
##   4     0.7606151  0.04092505
##   6     0.7606172  0.07261154
##   8     0.7548211  0.08069591
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     On Budget Over Budget Under Budget
##   On Budget          292          67           23
##   Over Budget          1           4            0
##   Under Budget         0           0            0
##
```

```
## Overall Statistics
##
##                Accuracy : 0.7649
##                  95% CI : (0.7194, 0.8062)
##     No Information Rate : 0.7571
##     P-Value [Acc > NIR] : 0.3871
##
##                   Kappa : 0.0606
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: On Budget Class: Over Budget Class: Under Budget
## Sensitivity                   0.99659            0.05634             0.00000
## Specificity                   0.04255            0.99684             1.00000
## Pos Pred Value                0.76440            0.80000                 NaN
## Neg Pred Value                0.80000            0.82461             0.94057
## Prevalence                    0.75711            0.18346             0.05943
## Detection Rate                0.75452            0.01034             0.00000
## Detection Prevalence          0.98708            0.01292             0.00000
## Balanced Accuracy             0.51957            0.52659             0.50000
## rf variable importance
##
##   only 20 most important variables shown (out of 57)
##
##                                       Overall
## initial_budget                         100.00
## spend_to_date_percent                   96.92
## avg_construction_jobs                   71.57
## avg_labor_unemp_rate                    69.64
## project_ph_design                       49.59
## project_ph_construction                 49.48
## delay_cat_funding_financial_issues      46.18
## project_ph_procurement                  44.64
## delay_cat_scope_or_design_changes       39.69
## project_ph_close_out                    29.27
## project_ph_completed                    25.72
## delay_cat_unforeseen_site_conditions    24.53
## project_ph_pre_design                   20.49
## project_themeParks & Recreation         18.03
## project_themeRoof Work                  15.73
## delay_classOn Time                      14.50
## project_ph_stopped                      14.35
## boroughManhattan                        12.84
## boroughQueens                           12.30
## project_themeSafety/Street Improvements 12.01
##
## --- DECISION TREE RESULTS for cost_class ---
## CART
##
## 1554 samples
##   22 predictor
##    3 classes: 'On Budget', 'Over Budget', 'Under Budget'
```

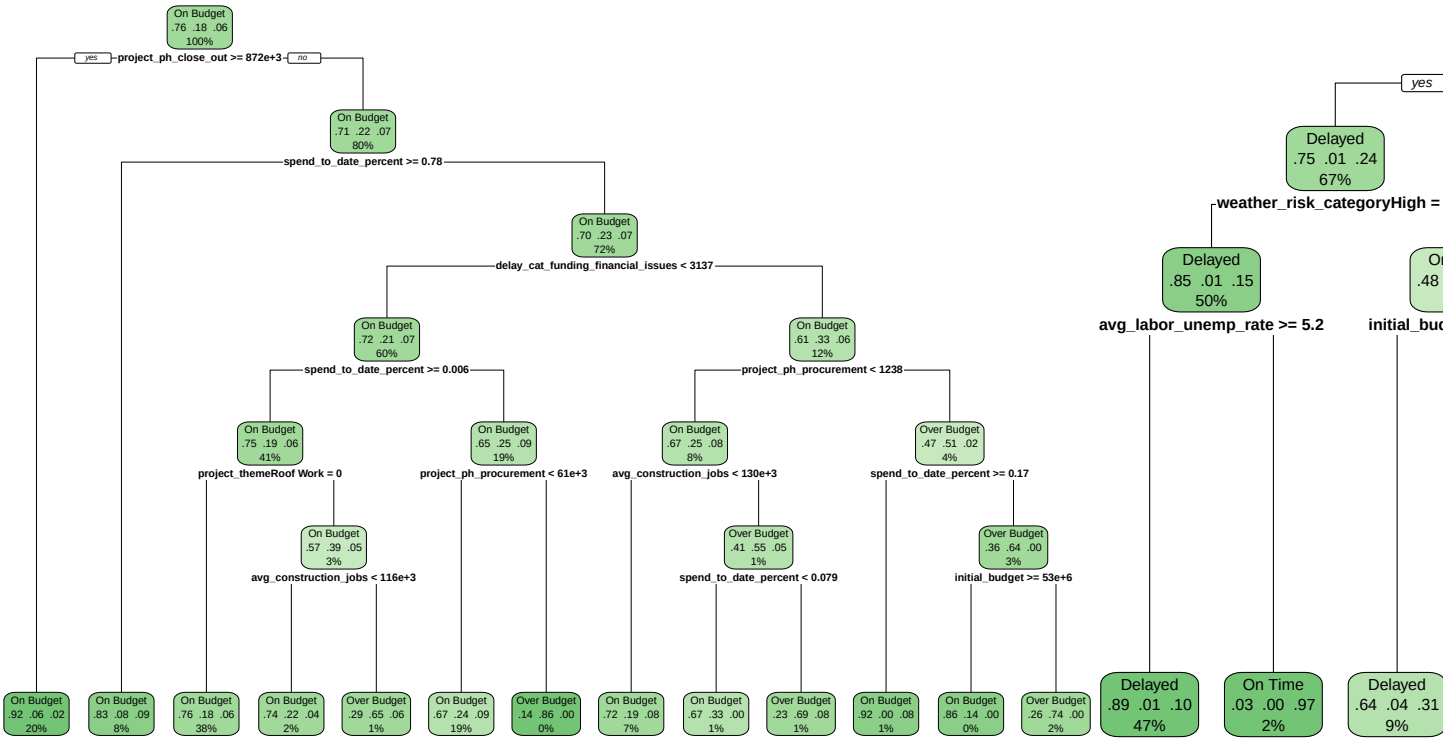
```
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1244, 1244, 1242, 1243, 1243
## Resampling results across tuning parameters:
##
##   cp            Accuracy   Kappa
##   0.0000000000  0.7007539  0.11562306
##   0.0007602339  0.7026894  0.11522456
##   0.0015204678  0.7174866  0.13315633
##   0.0022807018  0.7155677  0.11200572
##   0.0030409357  0.7226562  0.11114512
##   0.0038011696  0.7335928  0.11279268
##   0.0045614035  0.7380924  0.08986051
##   0.0053216374  0.7445337  0.08715169
##   0.0060818713  0.7477491  0.07621737
##   0.0068421053  0.7464650  0.03091202
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.006081871.
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     On Budget Over Budget Under Budget
##   On Budget          289          62           22
##   Over Budget          4           9            1
##   Under Budget         0           0            0
##
## Overall Statistics
##
##                Accuracy : 0.77
##                  95% CI : (0.7248, 0.811)
##     No Information Rate : 0.7571
##     P-Value [Acc > NIR] : 0.2994
##
##                   Kappa : 0.1277
##
##  Mcnemar's Test P-Value : 6.024e-16
##
## Statistics by Class:
##
##                      Class: On Budget Class: Over Budget Class: Under Budget
## Sensitivity                    0.9863            0.12676             0.00000
## Specificity                    0.1064            0.98418             1.00000
## Pos Pred Value                 0.7748            0.64286                 NaN
## Neg Pred Value                 0.7143            0.83378             0.94057
## Prevalence                     0.7571            0.18346             0.05943
## Detection Rate                 0.7468            0.02326             0.00000
## Detection Prevalence           0.9638            0.03618             0.00000
## Balanced Accuracy              0.5464            0.55547             0.50000
##
## --- XGBOOST RESULTS for cost_class ---
## [1]  train-mlogloss:0.887768+0.005812    test-mlogloss:0.923151+0.015179
```

```
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 30 rounds.
##
## [2]  train-mlogloss:0.760878+0.008331    test-mlogloss:0.821889+0.023185
## [3]  train-mlogloss:0.674412+0.011000    test-mlogloss:0.757743+0.029715
## [4]  train-mlogloss:0.612857+0.011904    test-mlogloss:0.714487+0.035379
## [5]  train-mlogloss:0.568406+0.014165    test-mlogloss:0.687226+0.039899
## [6]  train-mlogloss:0.533015+0.013610    test-mlogloss:0.669458+0.045447
## [7]  train-mlogloss:0.502197+0.014371    test-mlogloss:0.659057+0.049157
## [8]  train-mlogloss:0.478489+0.016699    test-mlogloss:0.652590+0.051993
## [9]  train-mlogloss:0.459037+0.014937    test-mlogloss:0.647527+0.054316
## [10] train-mlogloss:0.440933+0.015917    test-mlogloss:0.646326+0.055117
## [11] train-mlogloss:0.425611+0.014096    test-mlogloss:0.645834+0.055861
## [12] train-mlogloss:0.412640+0.014905    test-mlogloss:0.644400+0.057112
## [13] train-mlogloss:0.399750+0.017361    test-mlogloss:0.645088+0.058395
## [14] train-mlogloss:0.388773+0.017261    test-mlogloss:0.644259+0.059464
## [15] train-mlogloss:0.375553+0.020445    test-mlogloss:0.643353+0.059043
## [16] train-mlogloss:0.364704+0.018801    test-mlogloss:0.645274+0.061897
## [17] train-mlogloss:0.353080+0.017515    test-mlogloss:0.647406+0.064097
## [18] train-mlogloss:0.343122+0.017203    test-mlogloss:0.647911+0.065170
## [19] train-mlogloss:0.332505+0.017487    test-mlogloss:0.649251+0.066935
## [20] train-mlogloss:0.324940+0.020022    test-mlogloss:0.648517+0.067603
## [21] train-mlogloss:0.315344+0.018864    test-mlogloss:0.648712+0.068607
## [22] train-mlogloss:0.307646+0.017624    test-mlogloss:0.649752+0.068392
## [23] train-mlogloss:0.300910+0.015950    test-mlogloss:0.652505+0.069176
## [24] train-mlogloss:0.292002+0.015749    test-mlogloss:0.654199+0.071317
## [25] train-mlogloss:0.285772+0.015213    test-mlogloss:0.655914+0.071929
## [26] train-mlogloss:0.280205+0.015469    test-mlogloss:0.656852+0.072328
## [27] train-mlogloss:0.275365+0.015537    test-mlogloss:0.658483+0.074106
## [28] train-mlogloss:0.269195+0.015083    test-mlogloss:0.660956+0.075366
## [29] train-mlogloss:0.263116+0.015895    test-mlogloss:0.662217+0.075736
## [30] train-mlogloss:0.254898+0.014177    test-mlogloss:0.663632+0.075052
## [31] train-mlogloss:0.248984+0.015004    test-mlogloss:0.664849+0.075671
## [32] train-mlogloss:0.241884+0.014176    test-mlogloss:0.667764+0.077089
## [33] train-mlogloss:0.236560+0.015010    test-mlogloss:0.670152+0.077186
## [34] train-mlogloss:0.231822+0.015446    test-mlogloss:0.671251+0.078019
## [35] train-mlogloss:0.225721+0.015070    test-mlogloss:0.672165+0.079058
## [36] train-mlogloss:0.221899+0.015123    test-mlogloss:0.673536+0.081486
## [37] train-mlogloss:0.218094+0.014914    test-mlogloss:0.675164+0.082205
## [38] train-mlogloss:0.214128+0.014216    test-mlogloss:0.675853+0.082869
## [39] train-mlogloss:0.208898+0.014028    test-mlogloss:0.676974+0.082746
## [40] train-mlogloss:0.204170+0.014649    test-mlogloss:0.677111+0.083475
## [41] train-mlogloss:0.199458+0.013241    test-mlogloss:0.678648+0.084290
## [42] train-mlogloss:0.195507+0.013514    test-mlogloss:0.679645+0.084079
## [43] train-mlogloss:0.191264+0.012295    test-mlogloss:0.680673+0.084683
## [44] train-mlogloss:0.187796+0.012197    test-mlogloss:0.682268+0.085344
## [45] train-mlogloss:0.184318+0.011606    test-mlogloss:0.682723+0.084737
## Stopping. Best iteration:
## [15] train-mlogloss:0.375553+0.020445    test-mlogloss:0.643353+0.059043
##
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     On Budget Over Budget Under Budget
```

```
##   On Budget          284          62           23
##   Over Budget          9           8            0
##   Under Budget         0           1            0
##
## Overall Statistics
##
##                Accuracy : 0.7545
##                  95% CI : (0.7085, 0.7966)
##     No Information Rate : 0.7571
##     P-Value [Acc > NIR] : 0.5744
##
##                   Kappa : 0.0905
##
##  Mcnemar's Test P-Value : 1.018e-13
##
## Statistics by Class:
##
##                      Class: On Budget Class: Over Budget Class: Under Budget
## Sensitivity                   0.96928            0.11268             0.000000
## Specificity                   0.09574            0.97152             0.997253
## Pos Pred Value                0.76965            0.47059             0.000000
## Neg Pred Value                0.50000            0.82973             0.940415
## Prevalence                    0.75711            0.18346             0.059432
## Detection Rate                0.73385            0.02067             0.000000
## Detection Prevalence          0.95349            0.04393             0.002584
## Balanced Accuracy             0.53251            0.54210             0.498626
##
## --- RANDOM FOREST RESULTS for delay_class ---
## Random Forest
##
## 1554 samples
##   22 predictor
##    3 classes: 'Delayed', 'Early', 'On Time'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1244, 1243, 1243, 1243, 1243
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7284327  0.2388249
##   4     0.8153241  0.5488512
##   6     0.8346250  0.6059968
##   8     0.8378384  0.6185044
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Delayed Early On Time
##    Delayed     246     3      39
##    Early         0     0       0
##    On Time      14     3      82
```

```
##
## Overall Statistics
##
##                Accuracy : 0.8475
##                  95% CI : (0.8078, 0.8819)
##     No Information Rate : 0.6718
##     P-Value [Acc > NIR] : 3.129e-15
##
##                   Kappa : 0.6371
##
##  Mcnemar's Test P-Value : 0.0004854
##
## Statistics by Class:
##
##                      Class: Delayed Class: Early Class: On Time
## Sensitivity                  0.9462       0.0000         0.6777
## Specificity                  0.6693       1.0000         0.9361
## Pos Pred Value               0.8542          NaN         0.8283
## Neg Pred Value               0.8586       0.9845         0.8646
## Prevalence                   0.6718       0.0155         0.3127
## Detection Rate               0.6357       0.0000         0.2119
## Detection Prevalence         0.7442       0.0000         0.2558
## Balanced Accuracy            0.8077       0.5000         0.8069
## rf variable importance
##
##   only 20 most important variables shown (out of 57)
##
##                                        Overall
## avg_labor_unemp_rate                   100.000
## avg_construction_jobs                   75.186
## spend_to_date_percent                   55.044
## initial_budget                          47.269
## cost_cat_parks_recreation               35.313
## weather_risk_categoryHigh               30.371
## project_ph_design                       23.284
## project_ph_construction                 22.153
## project_ph_procurement                  13.633
## project_ph_close_out                    13.123
## project_ph_completed                    12.525
## cost_cat_public_buildings               11.197
## cost_cat_transportation                 10.908
## weather_risk_categoryModerate            9.784
## project_themeParks & Recreation          8.333
## cost_cat_miscellaneous_other             8.187
## cost_cat_infrastructure                  8.085
## boroughManhattan                         7.995
## project_ph_pre_design                    7.681
## project_themeSafety/Street Improvements  7.112
##
## --- DECISION TREE RESULTS for delay_class ---
## CART
##
## 1554 samples
##   22 predictor
```

```
##    3 classes: 'Delayed', 'Early', 'On Time'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1245, 1242, 1243, 1243, 1243
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.003137255  0.7979694  0.54501570
##   0.003921569  0.7973325  0.54099063
##   0.004411765  0.7954053  0.53488860
##   0.004901961  0.7992639  0.53693097
##   0.006535948  0.7992597  0.53188339
##   0.007843137  0.7992639  0.53157629
##   0.008496732  0.7992639  0.53157629
##   0.017647059  0.7915033  0.50670992
##   0.052941176  0.7264388  0.28474459
##   0.069934641  0.6853224  0.09032251
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.008496732.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Delayed Early On Time
##    Delayed     245     3      61
##    Early         0     0       0
##    On Time      15     3      60
##
## Overall Statistics
##
##                Accuracy : 0.7881
##                  95% CI : (0.744, 0.8278)
##     No Information Rate : 0.6718
##     P-Value [Acc > NIR] : 2.926e-07
##
##                   Kappa : 0.471
##
##  Mcnemar's Test P-Value : 2.139e-07
##
## Statistics by Class:
##
##                      Class: Delayed Class: Early Class: On Time
## Sensitivity                  0.9423       0.0000         0.4959
## Specificity                  0.4961       1.0000         0.9323
## Pos Pred Value               0.7929          NaN         0.7692
## Neg Pred Value               0.8077       0.9845         0.8026
## Prevalence                   0.6718       0.0155         0.3127
## Detection Rate               0.6331       0.0000         0.1550
## Detection Prevalence         0.7984       0.0000         0.2016
## Balanced Accuracy            0.7192       0.5000         0.7141
```




```
##
## --- XGBOOST RESULTS for delay_class ---
## [1]  train-mlogloss:0.826408+0.011008    test-mlogloss:0.863239+0.014101
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 30 rounds.
##
## [2]  train-mlogloss:0.655989+0.013637    test-mlogloss:0.721205+0.016923
## [3]  train-mlogloss:0.541655+0.016292    test-mlogloss:0.628364+0.019927
## [4]  train-mlogloss:0.457313+0.015124    test-mlogloss:0.563795+0.021942
## [5]  train-mlogloss:0.393843+0.016858    test-mlogloss:0.520722+0.023334
## [6]  train-mlogloss:0.348276+0.014736    test-mlogloss:0.491482+0.026111
## [7]  train-mlogloss:0.312556+0.012345    test-mlogloss:0.469159+0.029022
## [8]  train-mlogloss:0.287547+0.011837    test-mlogloss:0.456392+0.030737
## [9]  train-mlogloss:0.264529+0.012230    test-mlogloss:0.441874+0.034598
## [10] train-mlogloss:0.248543+0.012574    test-mlogloss:0.435564+0.036166
## [11] train-mlogloss:0.232302+0.011096    test-mlogloss:0.427123+0.034630
## [12] train-mlogloss:0.217774+0.009507    test-mlogloss:0.420173+0.035817
## [13] train-mlogloss:0.206593+0.008010    test-mlogloss:0.413795+0.033681
## [14] train-mlogloss:0.196954+0.009458    test-mlogloss:0.413855+0.035078
## [15] train-mlogloss:0.187556+0.008513    test-mlogloss:0.411759+0.035869
## [16] train-mlogloss:0.180500+0.008746    test-mlogloss:0.410663+0.037418
## [17] train-mlogloss:0.173841+0.008171    test-mlogloss:0.408923+0.037263
## [18] train-mlogloss:0.168239+0.008199    test-mlogloss:0.407954+0.038058
## [19] train-mlogloss:0.163368+0.009279    test-mlogloss:0.407317+0.038231
## [20] train-mlogloss:0.157510+0.009984    test-mlogloss:0.406603+0.038261
## [21] train-mlogloss:0.151951+0.009545    test-mlogloss:0.406020+0.037694
## [22] train-mlogloss:0.146879+0.009448    test-mlogloss:0.405290+0.037898
## [23] train-mlogloss:0.142311+0.010343    test-mlogloss:0.405295+0.037746
## [24] train-mlogloss:0.137844+0.009609    test-mlogloss:0.403497+0.037133
## [25] train-mlogloss:0.134606+0.008887    test-mlogloss:0.403741+0.037721
```

```
## [26] train-mlogloss:0.130640+0.009301    test-mlogloss:0.404121+0.038896
## [27] train-mlogloss:0.125127+0.008437    test-mlogloss:0.403806+0.040436
## [28] train-mlogloss:0.122008+0.008460    test-mlogloss:0.405078+0.040493
## [29] train-mlogloss:0.118746+0.008753    test-mlogloss:0.406238+0.039601
## [30] train-mlogloss:0.115612+0.008158    test-mlogloss:0.407999+0.040227
## [31] train-mlogloss:0.111936+0.007988    test-mlogloss:0.408885+0.042007
## [32] train-mlogloss:0.108980+0.007750    test-mlogloss:0.409129+0.042310
## [33] train-mlogloss:0.105626+0.007480    test-mlogloss:0.410054+0.041501
## [34] train-mlogloss:0.103129+0.007616    test-mlogloss:0.410937+0.041942
## [35] train-mlogloss:0.100326+0.007097    test-mlogloss:0.411809+0.041893
## [36] train-mlogloss:0.097737+0.007265    test-mlogloss:0.412738+0.042380
## [37] train-mlogloss:0.095429+0.006976    test-mlogloss:0.413188+0.042340
## [38] train-mlogloss:0.092733+0.007875    test-mlogloss:0.413010+0.042383
## [39] train-mlogloss:0.090090+0.007582    test-mlogloss:0.413254+0.042973
## [40] train-mlogloss:0.087713+0.007415    test-mlogloss:0.414115+0.042834
## [41] train-mlogloss:0.084942+0.006611    test-mlogloss:0.413771+0.043059
## [42] train-mlogloss:0.082556+0.006170    test-mlogloss:0.415606+0.043840
## [43] train-mlogloss:0.080572+0.006268    test-mlogloss:0.416480+0.043740
## [44] train-mlogloss:0.078698+0.006212    test-mlogloss:0.417975+0.045144
## [45] train-mlogloss:0.077271+0.005914    test-mlogloss:0.419727+0.046303
## [46] train-mlogloss:0.075690+0.005771    test-mlogloss:0.420733+0.045416
## [47] train-mlogloss:0.073860+0.005404    test-mlogloss:0.422147+0.045235
## [48] train-mlogloss:0.071896+0.005420    test-mlogloss:0.423253+0.044359
## [49] train-mlogloss:0.070443+0.005131    test-mlogloss:0.423242+0.044726
## [50] train-mlogloss:0.068822+0.005515    test-mlogloss:0.424324+0.045030
## [51] train-mlogloss:0.067042+0.004812    test-mlogloss:0.423680+0.045169
## [52] train-mlogloss:0.065159+0.004211    test-mlogloss:0.424942+0.045394
## [53] train-mlogloss:0.063587+0.003853    test-mlogloss:0.426515+0.045568
## [54] train-mlogloss:0.062240+0.003756    test-mlogloss:0.428056+0.045202
## Stopping. Best iteration:
## [24] train-mlogloss:0.137844+0.009609    test-mlogloss:0.403497+0.037133
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Delayed Early On Time
##    Delayed     243     3      28
##    Early         1     0       1
##    On Time      16     3      92
##
## Overall Statistics
##
##                Accuracy : 0.8656
##                  95% CI : (0.8276, 0.898)
##     No Information Rate : 0.6718
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6908
##
##  Mcnemar's Test P-Value : 0.1529
##
## Statistics by Class:
##
##                      Class: Delayed Class: Early Class: On Time
```

```
## Sensitivity                 0.9346     0.000000         0.7603
## Specificity                 0.7559     0.994751         0.9286
## Pos Pred Value              0.8869     0.000000         0.8288
## Neg Pred Value              0.8496     0.984416         0.8949
## Prevalence                  0.6718     0.015504         0.3127
## Detection Rate              0.6279     0.000000         0.2377
## Detection Prevalence        0.7080     0.005168         0.2868
## Balanced Accuracy           0.8453     0.497375         0.8445
```

```
# End sink and close file
#sink()
#close(log_file)
```

5.3 [Same as 4.8] Implementing modeling to predict project_profile

```
# Create and open log file
#log_file <- file("model_output_log_project_profile.txt", open = "wt")

# Redirect all console output to the file
#sink(log_file)

# Optional: Add a header with timestamp
#cat("==== MODELING PROJECT_PROFILE ====\n")
#cat("Timestamp: ", Sys.time(), "\n\n")


cat("\n--- DATA PREPARATION FOR project_profile ---\n")
```

```
##
## --- DATA PREPARATION FOR project_profile ---
```

```
# ---Columns to Drop and move forward for modeling----------------------
project_model_data_final_all_updated <- project_model_data_final_updated %>%
  select(-c(avg_cyclone_freq, avg_temp_anomaly, avg_max_wind_speed, status_combined, cost_class, delay_

# ---- Begin Classification Modeling for project_profile ----
library(caret)
library(randomForest)
library(dummy)
set.seed(123)

# Convert target and categorical predictors to factors
project_model_data_final_all_updated$project_profile <- as.factor(project_model_data_final_updated$proj

# Identify categorical variables (excluding the target)
categorical_vars <- sapply(project_model_data_final_all_updated, is.character)
categorical_vars <- names(categorical_vars[categorical_vars])
categorical_vars <- setdiff(categorical_vars, "project_profile")

# Convert character columns to factors
project_model_data_final_all_updated[categorical_vars] <- lapply(project_model_data_final_all_updated[ca

# Partition data into training and testing sets
set.seed(123)
train_index <- createDataPartition(project_model_data_final_all_updated$project_profile, p = 0.8, list =
```

```r
train_data <- project_model_data_final_all_updated[train_index, ]
test_data <- project_model_data_final_all_updated[-train_index, ]


# === Random Forest ===
cat("\n--- RANDOM FOREST RESULTS for project_profile ---\n")
```

```
##
## --- RANDOM FOREST RESULTS for project_profile ---
```

```r
# Cross-validated and tuned Random Forest using caret
control <- trainControl(method = "cv", number = 5, search = "grid")
tunegrid <- expand.grid(.mtry = c(2, 4, 6, 8))

set.seed(123)
rf_model <- train(
  project_profile ~ .,
  data = train_data,
  method = "rf",
  metric = "Accuracy",
  trControl = control,
  tuneGrid = tunegrid,
  importance = TRUE
)

print(rf_model)
```

```
## Random Forest
##
## 1557 samples
##   28 predictor
##    9 classes: 'Cost Risk', 'Exceptional Performance', 'Fast but Costly', 'High Risk', 'Lean Delivery
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1245, 1247, 1245, 1246, 1245
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.6634333  0.3848114
##   4     0.7552879  0.5818088
##   6     0.7886936  0.6471299
##   8     0.8053933  0.6802281
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```r
# Predict on test data
rf_preds <- predict(rf_model, newdata = test_data)

# Evaluate performance
conf_mat <- confusionMatrix(rf_preds, test_data$project_profile)
print(conf_mat)
```

```
## Confusion Matrix and Statistics
```

```
##
##                          Reference
## Prediction                Cost Risk Exceptional Performance Fast but Costly
##   Cost Risk                       8                       0               0
##   Exceptional Performance         0                       0               0
##   Fast but Costly                 0                       0               0
##   High Risk                       0                       0               0
##   Lean Delivery                   0                       0               0
##   On Track                       15                       0               0
##   Schedule Risk                   1                       0               0
##   Strong Delivery                 0                       0               1
##   Time Risk, Cost-Saving          0                       0               0
##                          Reference
## Prediction                High Risk Lean Delivery On Track Schedule Risk
##   Cost Risk                       0             0        0             0
##   Exceptional Performance         0             0        0             0
##   Fast but Costly                 0             0        0             0
##   High Risk                      16             0        0             1
##   Lean Delivery                   0             2        1             0
##   On Track                        0             5       84             5
##   Schedule Risk                  29             0        4           193
##   Strong Delivery                 0             0        0             0
##   Time Risk, Cost-Saving          0             0        0             0
##                          Reference
## Prediction                Strong Delivery Time Risk, Cost-Saving
##   Cost Risk                             0                     0
##   Exceptional Performance               0                     0
##   Fast but Costly                       0                     0
##   High Risk                             0                     0
##   Lean Delivery                         0                     0
##   On Track                              2                     0
##   Schedule Risk                         1                    14
##   Strong Delivery                       1                     0
##   Time Risk, Cost-Saving                0                     1
##
## Overall Statistics
##
##                Accuracy : 0.7943
##                  95% CI : (0.7503, 0.8336)
##     No Information Rate : 0.5182
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6569
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Cost Risk Class: Exceptional Performance
## Sensitivity                   0.33333                             NA
## Specificity                   1.00000                              1
## Pos Pred Value                1.00000                             NA
## Neg Pred Value                0.95745                             NA
## Prevalence                    0.06250                              0
```

```
## Detection Rate               0.02083                            0
## Detection Prevalence         0.02083                            0
## Balanced Accuracy            0.66667                           NA
##                      Class: Fast but Costly Class: High Risk
## Sensitivity                        0.000000          0.35556
## Specificity                        1.000000          0.99705
## Pos Pred Value                          NaN          0.94118
## Neg Pred Value                     0.997396          0.92098
## Prevalence                         0.002604          0.11719
## Detection Rate                     0.000000          0.04167
## Detection Prevalence               0.000000          0.04427
## Balanced Accuracy                  0.500000          0.67630
##                      Class: Lean Delivery Class: On Track Class: Schedule Risk
## Sensitivity                      0.285714          0.9438                0.9698
## Specificity                      0.997347          0.9085                0.7351
## Pos Pred Value                   0.666667          0.7568                0.7975
## Neg Pred Value                   0.986877          0.9817                0.9577
## Prevalence                       0.018229          0.2318                0.5182
## Detection Rate                   0.005208          0.2188                0.5026
## Detection Prevalence             0.007812          0.2891                0.6302
## Balanced Accuracy                0.641531          0.9261                0.8525
##                      Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                        0.250000                       0.066667
## Specificity                        0.997368                       1.000000
## Pos Pred Value                     0.500000                       1.000000
## Neg Pred Value                     0.992147                       0.963446
## Prevalence                         0.010417                       0.039062
## Detection Rate                     0.002604                       0.002604
## Detection Prevalence               0.005208                       0.002604
## Balanced Accuracy                  0.623684                       0.533333
```

```
# View feature importance
#importance(rf_model)
varImp(rf_model)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 62)
##
##                                    Cost Risk Exceptional Performance
## delay_cat_funding_financial_issues     46.295                   4.455
## delay_cat_scope_or_design_changes      40.534                   6.790
## delay_cat_unforeseen_site_conditions   30.511                   5.800
## cost_cat_parks_recreation              57.572                   4.455
## cost_cat_transportation                47.465                   4.455
## cost_cat_public_buildings              14.630                   9.366
## cost_cat_housing                        6.356                   4.455
## cost_cat_miscellaneous_other            2.625                   4.455
## avg_labor_unemp_rate                   23.076                   7.476
## spend_to_date_percent                  28.798                   6.790
## avg_construction_jobs                  22.315                   7.154
## initial_budget                         24.031                   4.455
## weather_risk_categoryHigh               9.494                   6.790
## project_ph_construction                12.132                   4.455
```

```
## project_ph_completed                   11.645                  6.359
## cost_cat_infrastructure                10.066                  4.455
## cost_cat_environmental                 23.577                  4.455
## project_ph_design                      16.015                  2.550
## delay_cat_legal_contractual_issues     10.193                  4.455
## project_ph_close_out                   12.701                  4.455
##                                    Fast but Costly High Risk Lean Delivery
## delay_cat_funding_financial_issues      14.5228    48.596        24.948
## delay_cat_scope_or_design_changes        9.0212    30.840        23.566
## delay_cat_unforeseen_site_conditions     6.7899    16.969        16.711
## cost_cat_parks_recreation                6.1390    47.083        39.126
## cost_cat_transportation                  0.9972    36.631        14.434
## cost_cat_public_buildings                5.8002    45.872        12.434
## cost_cat_housing                         4.4546    37.513         6.277
## cost_cat_miscellaneous_other             7.7335    33.838        14.157
## avg_labor_unemp_rate                     9.7195    18.254        16.228
## spend_to_date_percent                    8.1677    20.998         3.501
## avg_construction_jobs                    8.6586    13.130        13.331
## initial_budget                           4.0220     8.271         9.233
## weather_risk_categoryHigh                4.8139    16.063         6.685
## project_ph_construction                  7.3280     6.672        12.000
## project_ph_completed                     6.6983    14.239         9.899
## cost_cat_infrastructure                 11.4836    23.867        10.585
## cost_cat_environmental                   4.4546     9.234         6.955
## project_ph_design                        2.6731     8.880        10.573
## delay_cat_legal_contractual_issues       4.4546    13.557         7.126
## project_ph_close_out                     6.7899    16.796        10.100
##                                    On Track Schedule Risk Strong Delivery
## delay_cat_funding_financial_issues    100.00        78.937          28.716
## delay_cat_scope_or_design_changes      91.04        69.036          20.569
## delay_cat_unforeseen_site_conditions   68.44        53.005          13.499
## cost_cat_parks_recreation              42.85        47.551           4.411
## cost_cat_transportation                27.90        27.453           3.681
## cost_cat_public_buildings              28.67        31.911           3.458
## cost_cat_housing                       14.43        21.626           4.455
## cost_cat_miscellaneous_other           23.32        18.133           6.611
## avg_labor_unemp_rate                   33.39        32.541          15.151
## spend_to_date_percent                  28.21        32.197           5.831
## avg_construction_jobs                  30.77        22.707          11.015
## initial_budget                         22.21        29.892           7.959
## weather_risk_categoryHigh              28.24        28.779          10.958
## project_ph_construction                27.74        22.949           2.455
## project_ph_completed                   18.66        25.379           8.598
## cost_cat_infrastructure                15.79        15.783           0.000
## cost_cat_environmental                 17.41         9.847           4.455
## project_ph_design                      22.51        15.903           4.326
## delay_cat_legal_contractual_issues     19.60        22.372           5.800
## project_ph_close_out                   14.93        19.409           4.452
##                                    Time Risk, Cost-Saving
## delay_cat_funding_financial_issues                25.728
## delay_cat_scope_or_design_changes                 24.850
## delay_cat_unforeseen_site_conditions              19.024
## cost_cat_parks_recreation                         40.574
## cost_cat_transportation                           12.321
```

```
## cost_cat_public_buildings                          42.659
## cost_cat_housing                                    7.714
## cost_cat_miscellaneous_other                       20.877
## avg_labor_unemp_rate                               11.873
## spend_to_date_percent                              15.363
## avg_construction_jobs                               9.572
## initial_budget                                     12.248
## weather_risk_categoryHigh                          13.348
## project_ph_construction                             9.532
## project_ph_completed                               12.644
## cost_cat_infrastructure                            12.197
## cost_cat_environmental                              6.359
## project_ph_design                                   8.725
## delay_cat_legal_contractual_issues                  4.455
## project_ph_close_out                                8.667
```

```
# === Decision Tree ===
cat("\n--- DECISION TREE RESULTS for project_profile ---\n")
```

```
##
## --- DECISION TREE RESULTS for project_profile ---
```

```
# ---- Train Decision Tree and XGBoost Models for Comparison ----

# Decision Tree with caret cross-validation
dt_control <- trainControl(method = "cv", number = 5)
dt_model <- train(
  project_profile ~ .,
  data = train_data,
  method = "rpart",
  trControl = dt_control,
  tuneLength = 10
)
print(dt_model)
```

```
## CART
##
## 1557 samples
##   28 predictor
##    9 classes: 'Cost Risk', 'Exceptional Performance', 'Fast but Costly', 'High Risk', 'Lean Delivery
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1247, 1244, 1246, 1245, 1246
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01056803  0.8015896  0.6783756
##   0.01188904  0.7996562  0.6742685
##   0.01254954  0.7983700  0.6715800
##   0.01321004  0.7842138  0.6443353
##   0.01453104  0.7746086  0.6277969
##   0.01585205  0.7630350  0.6060408
##   0.01849406  0.7604791  0.5989634
##   0.02774108  0.7540606  0.5858820
##   0.03963012  0.7366806  0.5515591
```

```
##   0.12351387  0.5957323  0.2076849
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01056803.
```

```
predict_dt <- predict(dt_model, test_data)
conf_mat_dt <- confusionMatrix(predict_dt, test_data$project_profile)
print(conf_mat_dt)
```

```
## Confusion Matrix and Statistics
##
##                          Reference
## Prediction                Cost Risk Exceptional Performance Fast but Costly
##   Cost Risk                       7                       0               0
##   Exceptional Performance         0                       0               0
##   Fast but Costly                 0                       0               0
##   High Risk                       0                       0               0
##   Lean Delivery                   0                       0               0
##   On Track                       17                       0               0
##   Schedule Risk                   0                       0               0
##   Strong Delivery                 0                       0               1
##   Time Risk, Cost-Saving          0                       0               0
##                          Reference
## Prediction                High Risk Lean Delivery On Track Schedule Risk
##   Cost Risk                       0             0        3             0
##   Exceptional Performance         0             0        0             0
##   Fast but Costly                 0             0        0             0
##   High Risk                      11             0        0             9
##   Lean Delivery                   0             3        0             0
##   On Track                        0             4       86             6
##   Schedule Risk                  34             0        0           182
##   Strong Delivery                 0             0        0             0
##   Time Risk, Cost-Saving          0             0        0             2
##                          Reference
## Prediction                Strong Delivery Time Risk, Cost-Saving
##   Cost Risk                             0                      0
##   Exceptional Performance               0                      0
##   Fast but Costly                       0                      0
##   High Risk                             0                      0
##   Lean Delivery                         0                      0
##   On Track                              0                      0
##   Schedule Risk                         0                     12
##   Strong Delivery                       4                      0
##   Time Risk, Cost-Saving                0                      3
##
## Overall Statistics
##
##                Accuracy : 0.7708
##                  95% CI : (0.7255, 0.8119)
##     No Information Rate : 0.5182
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6277
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: Cost Risk Class: Exceptional Performance
## Sensitivity                   0.29167                             NA
## Specificity                   0.99167                              1
## Pos Pred Value                0.70000                             NA
## Neg Pred Value                0.95455                             NA
## Prevalence                    0.06250                              0
## Detection Rate                0.01823                              0
## Detection Prevalence          0.02604                              0
## Balanced Accuracy             0.64167                             NA
##                      Class: Fast but Costly Class: High Risk
## Sensitivity                        0.000000          0.24444
## Specificity                        1.000000          0.97345
## Pos Pred Value                          NaN          0.55000
## Neg Pred Value                     0.997396          0.90659
## Prevalence                         0.002604          0.11719
## Detection Rate                     0.000000          0.02865
## Detection Prevalence               0.000000          0.05208
## Balanced Accuracy                  0.500000          0.60895
##                      Class: Lean Delivery Class: On Track Class: Schedule Risk
## Sensitivity                      0.428571          0.9663               0.9146
## Specificity                      1.000000          0.9085               0.7514
## Pos Pred Value                   1.000000          0.7611               0.7982
## Neg Pred Value                   0.989501          0.9889               0.8910
## Prevalence                       0.018229          0.2318               0.5182
## Detection Rate                   0.007812          0.2240               0.4740
## Detection Prevalence             0.007812          0.2943               0.5938
## Balanced Accuracy                0.714286          0.9374               0.8330
##                      Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                         1.00000                      0.200000
## Specificity                         0.99737                      0.994580
## Pos Pred Value                      0.80000                      0.600000
## Neg Pred Value                      1.00000                      0.968338
## Prevalence                          0.01042                      0.039062
## Detection Rate                      0.01042                      0.007812
## Detection Prevalence                0.01302                      0.013021
## Balanced Accuracy                   0.99868                      0.597290
```

```
png(filename = "decision_tree_project_profile.png", width = 2400, height = 1600, res = 300)
rpart.plot(dt_model$finalModel)
```

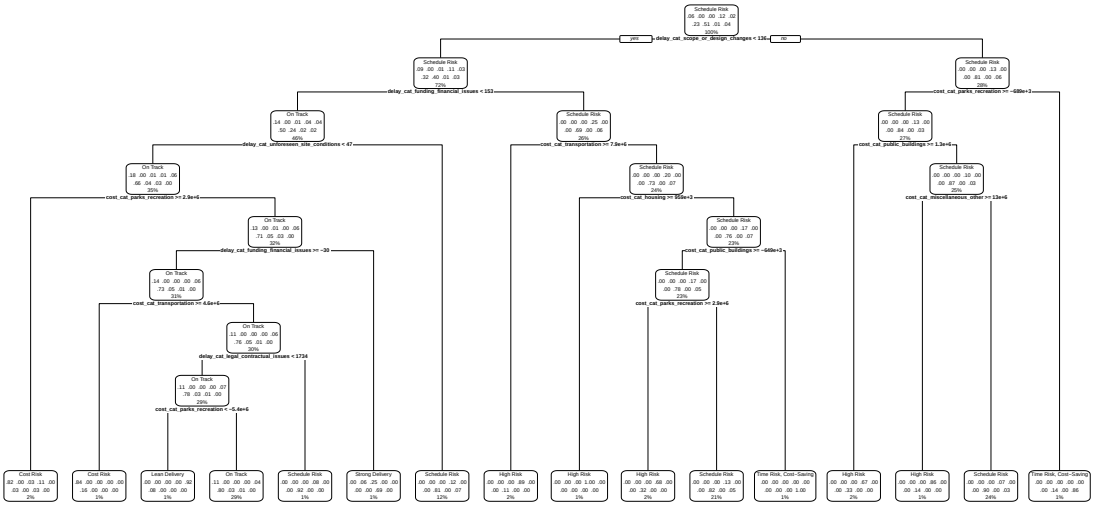
```
## Warning: All boxes will be white (the box.palette argument will be ignored) because
## the number of classes in the response 9 is greater than length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```

```
dev.off()
```

```
## pdf
##   2
```

```
# Plot the decision tree
library(rpart.plot)
rpart.plot(dt_model$finalModel)
```

```
## Warning: All boxes will be white (the box.palette argument will be ignored) because
## the number of classes in the response 9 is greater than length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```



```
# === XGBoost ===
cat("\n--- XGBOOST RESULTS for project_profile ---\n")
```

```
##
## --- XGBOOST RESULTS for project_profile ---
```

```
# XGBoost with cross-validation and early stopping
library(xgboost)
library(Matrix)

# Convert data to numeric matrix format
train_matrix <- model.matrix(project_profile ~ . -1, data = train_data)
test_matrix <- model.matrix(project_profile ~ . -1, data = test_data)

xgb_train <- xgb.DMatrix(data = train_matrix, label = as.numeric(train_data$project_profile) - 1)
xgb_test <- xgb.DMatrix(data = test_matrix, label = as.numeric(test_data$project_profile) - 1)

xgb_cv <- xgb.cv(
  data = xgb_train,
  nrounds = 100,
  nfold = 5,
  early_stopping_rounds = 10,
  objective = "multi:softmax",
  num_class = length(unique(train_data$project_profile)),
  verbose = 1
)
```

```
## [1]  train-mlogloss:1.358658+0.007169    test-mlogloss:1.420773+0.016626
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 10 rounds.
##
## [2]  train-mlogloss:1.030857+0.005699    test-mlogloss:1.135453+0.026113
## [3]  train-mlogloss:0.822586+0.004360    test-mlogloss:0.962589+0.031164
## [4]  train-mlogloss:0.673864+0.004795    test-mlogloss:0.841223+0.035918
## [5]  train-mlogloss:0.563453+0.005594    test-mlogloss:0.751789+0.038810
```

```
## [6]  train-mlogloss:0.476276+0.005508    test-mlogloss:0.684273+0.041543
## [7]  train-mlogloss:0.409583+0.004929    test-mlogloss:0.630329+0.043619
## [8]  train-mlogloss:0.354675+0.006289    test-mlogloss:0.587649+0.044129
## [9]  train-mlogloss:0.312413+0.006592    test-mlogloss:0.556816+0.046408
## [10] train-mlogloss:0.275202+0.007336    test-mlogloss:0.530415+0.048070
## [11] train-mlogloss:0.245219+0.007442    test-mlogloss:0.509333+0.049315
## [12] train-mlogloss:0.221239+0.007023    test-mlogloss:0.492003+0.049535
## [13] train-mlogloss:0.199314+0.006544    test-mlogloss:0.476383+0.049151
## [14] train-mlogloss:0.181274+0.006507    test-mlogloss:0.465784+0.049714
## [15] train-mlogloss:0.165401+0.005807    test-mlogloss:0.455463+0.050190
## [16] train-mlogloss:0.151722+0.005628    test-mlogloss:0.448557+0.053157
## [17] train-mlogloss:0.140643+0.005257    test-mlogloss:0.441595+0.052534
## [18] train-mlogloss:0.129438+0.004499    test-mlogloss:0.437128+0.055184
## [19] train-mlogloss:0.119797+0.004496    test-mlogloss:0.431756+0.056326
## [20] train-mlogloss:0.111570+0.004261    test-mlogloss:0.428919+0.058129
## [21] train-mlogloss:0.103742+0.004475    test-mlogloss:0.426390+0.059203
## [22] train-mlogloss:0.096511+0.004746    test-mlogloss:0.423300+0.059003
## [23] train-mlogloss:0.090776+0.004426    test-mlogloss:0.419870+0.060253
## [24] train-mlogloss:0.085178+0.004742    test-mlogloss:0.419634+0.062748
## [25] train-mlogloss:0.079910+0.003781    test-mlogloss:0.418914+0.062170
## [26] train-mlogloss:0.075259+0.003447    test-mlogloss:0.418335+0.062240
## [27] train-mlogloss:0.071174+0.003485    test-mlogloss:0.416705+0.062259
## [28] train-mlogloss:0.067578+0.003540    test-mlogloss:0.416882+0.063778
## [29] train-mlogloss:0.063848+0.003171    test-mlogloss:0.417880+0.064809
## [30] train-mlogloss:0.060625+0.003023    test-mlogloss:0.418086+0.065371
## [31] train-mlogloss:0.057392+0.002734    test-mlogloss:0.419394+0.066536
## [32] train-mlogloss:0.054432+0.002682    test-mlogloss:0.420969+0.066874
## [33] train-mlogloss:0.051938+0.002420    test-mlogloss:0.421281+0.067218
## [34] train-mlogloss:0.049583+0.002420    test-mlogloss:0.422624+0.068174
## [35] train-mlogloss:0.047432+0.002401    test-mlogloss:0.422538+0.069857
## [36] train-mlogloss:0.045303+0.002341    test-mlogloss:0.423243+0.069687
## [37] train-mlogloss:0.043237+0.002164    test-mlogloss:0.424923+0.070795
## Stopping. Best iteration:
## [27] train-mlogloss:0.071174+0.003485    test-mlogloss:0.416705+0.062259
```

```r
best_nrounds <- xgb_cv$best_iteration

xgb_model <- xgboost(
  data = xgb_train,
  nrounds = best_nrounds,
  objective = "multi:softmax",
  num_class = length(unique(train_data$project_profile)),
  verbose = 0
)

xgb_preds <- predict(xgb_model, xgb_test)
xgb_preds_factor <- factor(xgb_preds + 1, levels = 1:length(levels(train_data$project_profile)),
                           labels = levels(train_data$project_profile))

conf_mat_xgb <- confusionMatrix(xgb_preds_factor, test_data$project_profile)
print(conf_mat_xgb)
```

```
## Confusion Matrix and Statistics
##
##                           Reference
```

```
## Prediction                Cost Risk Exceptional Performance Fast but Costly
##   Cost Risk                      14                        0               0
##   Exceptional Performance         0                        0               0
##   Fast but Costly                 0                        0               0
##   High Risk                       2                        0               0
##   Lean Delivery                   0                        0               0
##   On Track                        8                        0               0
##   Schedule Risk                   0                        0               0
##   Strong Delivery                 0                        0               1
##   Time Risk, Cost-Saving          0                        0               0
##                          Reference
## Prediction                High Risk Lean Delivery On Track Schedule Risk
##   Cost Risk                       0             0        2             0
##   Exceptional Performance         0             0        0             0
##   Fast but Costly                 0             0        0             0
##   High Risk                      29             0        0             9
##   Lean Delivery                   0             3        0             0
##   On Track                        0             4       87             0
##   Schedule Risk                  16             0        0           187
##   Strong Delivery                 0             0        0             0
##   Time Risk, Cost-Saving          0             0        0             3
##                          Reference
## Prediction                Strong Delivery Time Risk, Cost-Saving
##   Cost Risk                             0                      0
##   Exceptional Performance               0                      0
##   Fast but Costly                       1                      0
##   High Risk                             0                      0
##   Lean Delivery                         0                      0
##   On Track                              0                      0
##   Schedule Risk                         0                      8
##   Strong Delivery                       3                      0
##   Time Risk, Cost-Saving                0                      7
##
## Overall Statistics
##
##                Accuracy : 0.8594
##                  95% CI : (0.8205, 0.8926)
##     No Information Rate : 0.5182
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7801
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Cost Risk Class: Exceptional Performance
## Sensitivity                   0.58333                             NA
## Specificity                   0.99444                              1
## Pos Pred Value                0.87500                             NA
## Neg Pred Value                0.97283                             NA
## Prevalence                    0.06250                              0
## Detection Rate                0.03646                              0
## Detection Prevalence          0.04167                              0
```

```
## Balanced Accuracy             0.78889                           NA
##                       Class: Fast but Costly Class: High Risk
## Sensitivity                         0.000000          0.64444
## Specificity                         0.997389          0.96755
## Pos Pred Value                      0.000000          0.72500
## Neg Pred Value                      0.997389          0.95349
## Prevalence                          0.002604          0.11719
## Detection Rate                      0.000000          0.07552
## Detection Prevalence                0.002604          0.10417
## Balanced Accuracy                   0.498695          0.80600
##                       Class: Lean Delivery Class: On Track Class: Schedule Risk
## Sensitivity                       0.428571          0.9775               0.9397
## Specificity                       1.000000          0.9593               0.8703
## Pos Pred Value                    1.000000          0.8788               0.8863
## Neg Pred Value                    0.989501          0.9930               0.9306
## Prevalence                        0.018229          0.2318               0.5182
## Detection Rate                    0.007812          0.2266               0.4870
## Detection Prevalence              0.007812          0.2578               0.5495
## Balanced Accuracy                 0.714286          0.9684               0.9050
##                       Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                         0.750000                       0.46667
## Specificity                         0.997368                       0.99187
## Pos Pred Value                      0.750000                       0.70000
## Neg Pred Value                      0.997368                       0.97861
## Prevalence                          0.010417                       0.03906
## Detection Rate                      0.007812                       0.01823
## Detection Prevalence                0.010417                       0.02604
## Balanced Accuracy                   0.873684                       0.72927
```

```
# End logging
#sink()
#close(log_file)
```

5.4 Some EDA and Figure to include in the Report

```
# Load required libraries
library(ggplot2)
library(dplyr)
library(corrplot)

# Load your dataset
data <- read.csv("project_model_data_final.csv")

plot_cost1 <- ggplot(project_model_data_final_updated, aes(x = cost_class)) +
  geom_bar() +
  labs(title = "Distribution of Cost Class")

# ==== Figure A1: Histogram of cost_diff grouped by fms_id ====
fa1 <- data %>%
  group_by(fms_id, category_group) %>%
  summarise(total_cost_diff = mean(cost_diff, na.rm = TRUE)) %>%
  ggplot(aes(x = total_cost_diff)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  labs(
```

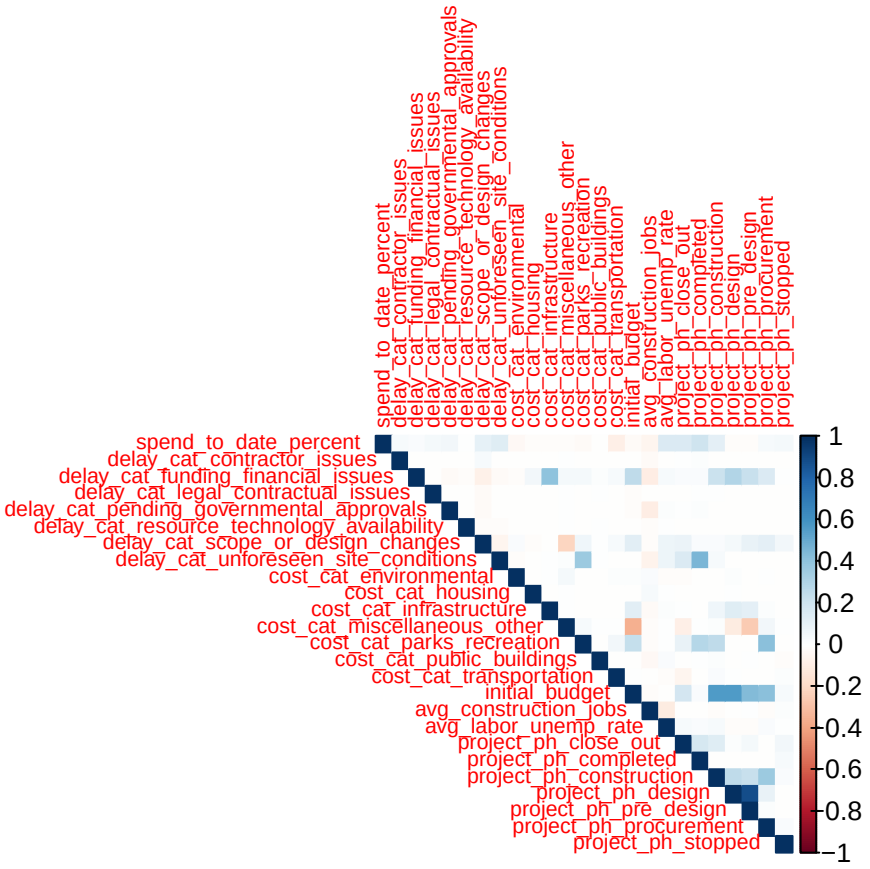
```
    title = "Figure A1. Histogram of Summed Cost Difference per Project",
    x = "Total Cost Difference",
    y = "Frequency"
  ) +
  theme_minimal()
```

```
## `summarise()` has grouped output by 'fms_id'. You can override using the
## `.groups` argument.
```

```
# ==== Figure A2: Boxplot of delay_days by borough ====
fa2 <- data %>%
  group_by(fms_id, delay_category) %>%
  summarise(total_delay_days = mean(delay_days, na.rm = TRUE)) %>%
  ggplot(aes(x = delay_category, y = total_delay_days)) +
  geom_boxplot() +
  labs(
    title = "Boxplot of Total Delay Days by Delay Category",
    x = "Delay Category",
    y = "Total Delay Days"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

```
## `summarise()` has grouped output by 'fms_id'. You can override using the
## `.groups` argument.
```

```
# ==== Figure A3: Correlation matrix of numeric features ====
numeric_data <- project_model_data_final_updated %>% select_if(is.numeric) %>% select(-c(avg_cyclone_fr
corr_matrix <- cor(numeric_data, use = "complete.obs")
corrplot(corr_matrix, method = "color", type = "upper", tl.cex = 0.7)
```



5.5 Plotting 9 modeling accuracy

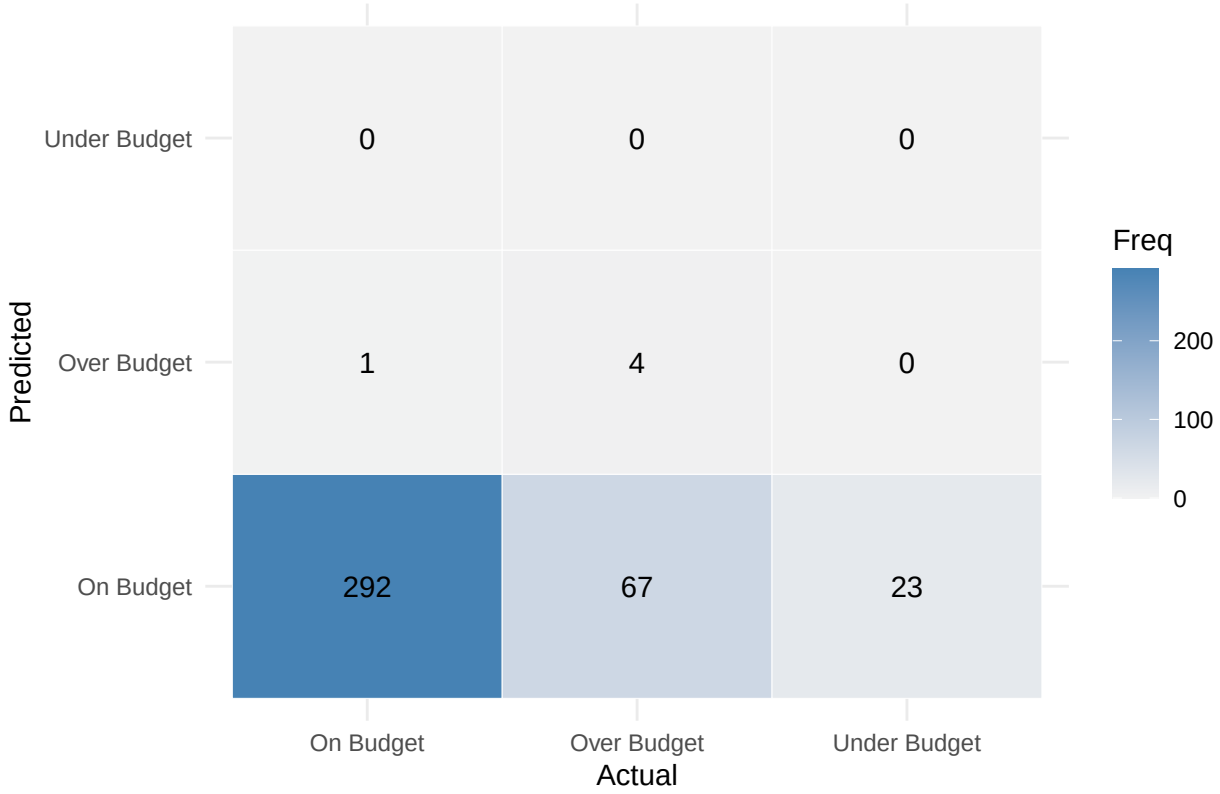
```
library(ggplot2)
library(dplyr)
library(readr)

# Load data
conf_df <- read_csv("Confusion_Matrices_by_Model_and_Task.csv")
```

```
## New names:
## Rows: 81 Columns: 6
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (4): Actual, Predicted, Model, Task dbl (2): ...1, Freq
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
plot_conf_matrix <- function(data, model, task, fig_number) {
  plot_data <- data %>%
    filter(Model == model, Task == task)

  ggplot(plot_data, aes(x = Actual, y = Predicted, fill = Freq)) +
    geom_tile(color = "white") +
    geom_text(aes(label = Freq), color = "black", size = 4) +
    scale_fill_gradient(low = "gray95", high = "steelblue") +
```

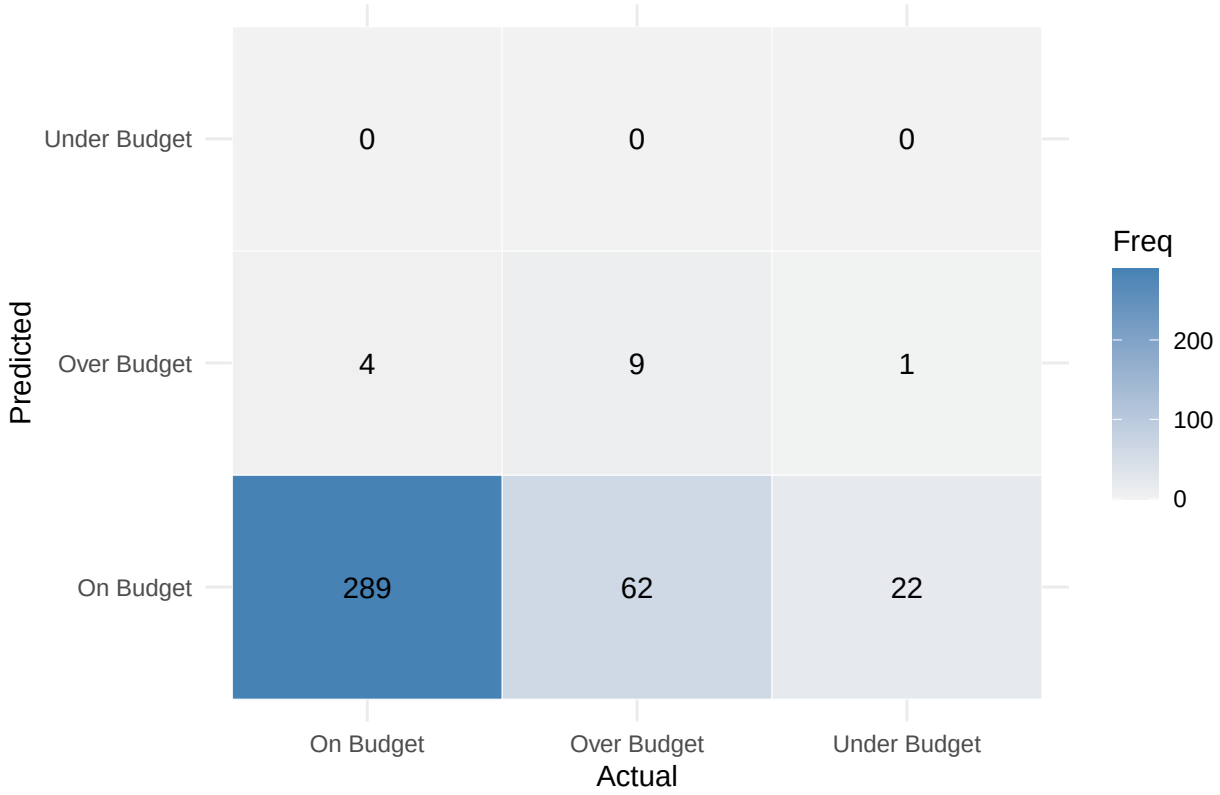
```
    labs(
      title = paste0("Figure ", fig_number, ". ", model, " - ", task, " Classification"),
      x = "Actual",
      y = "Predicted"
    ) +
    theme_minimal()
}


# Cost Classification
plot_conf_matrix(conf_df, "RF", "Cost", 1)
```

Figure 1. RF − Cost Classification



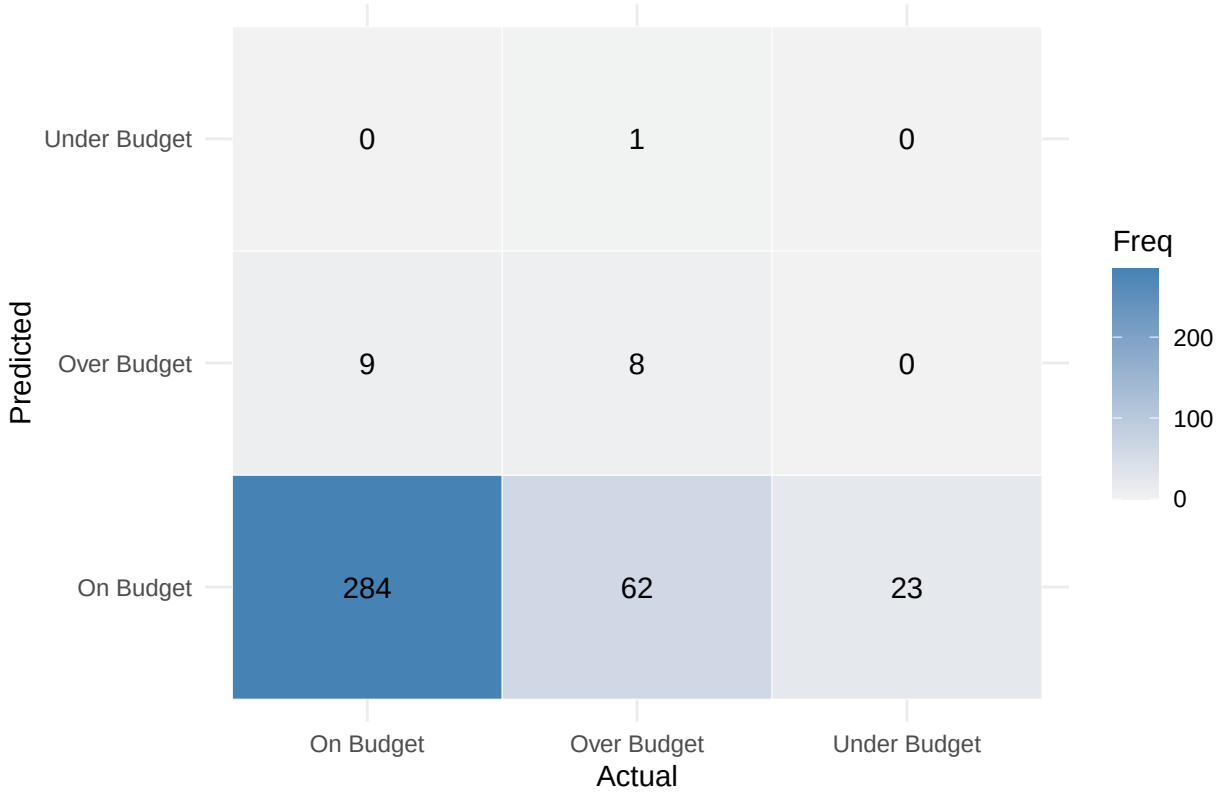

```
plot_conf_matrix(conf_df, "DT", "Cost", "2a")
```

Figure 2a. DT − Cost Classification



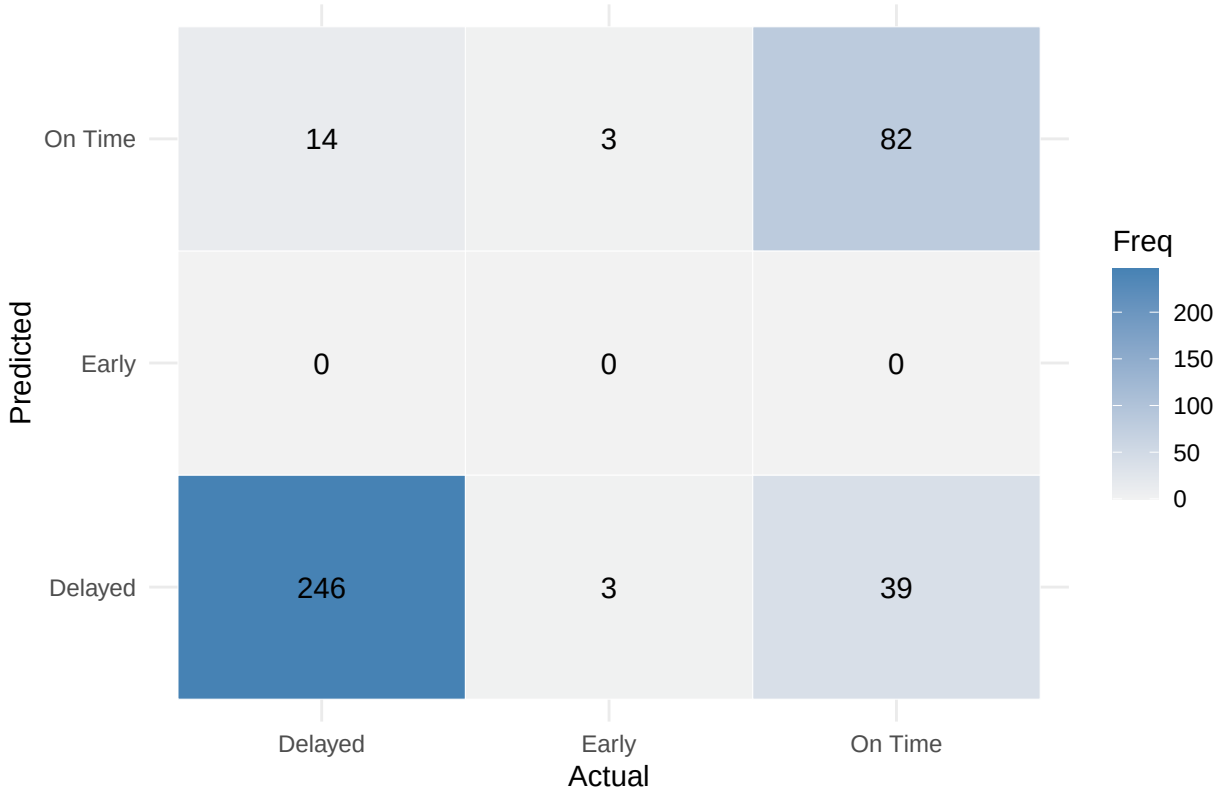

```
plot_conf_matrix(conf_df, "XGB", "Cost", 7)
```

Figure 7. XGB − Cost Classification



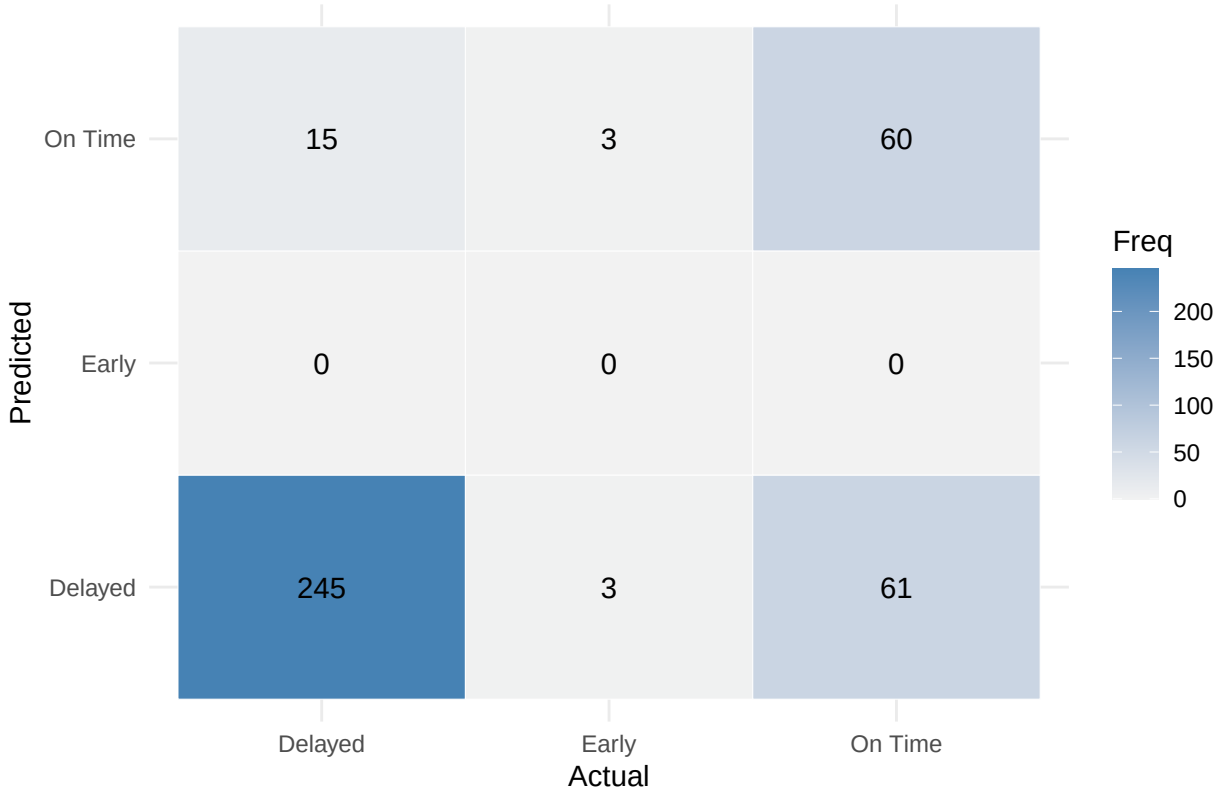

```
# Delay Classification
plot_conf_matrix(conf_df, "RF", "Delay", 10)
```

Figure 10. RF − Delay Classification



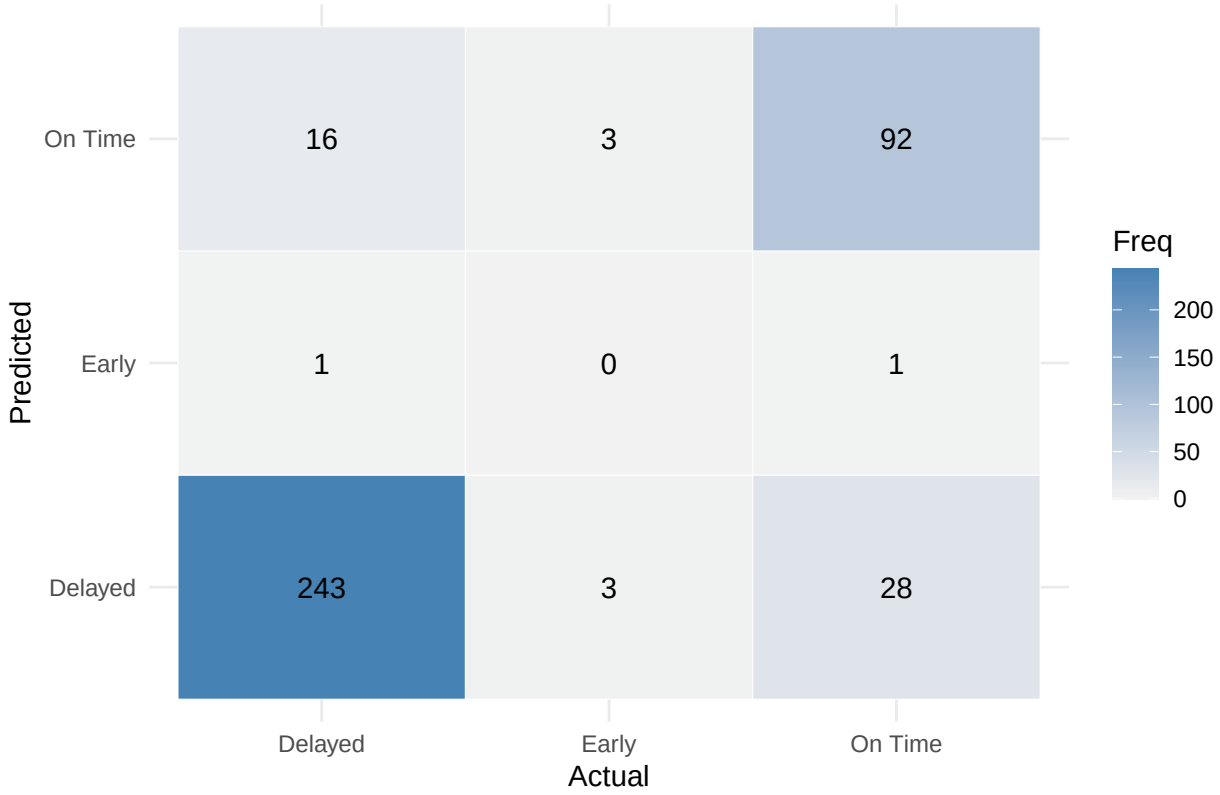

```
plot_conf_matrix(conf_df, "DT", "Delay", 4)
```

Figure 4. DT − Delay Classification



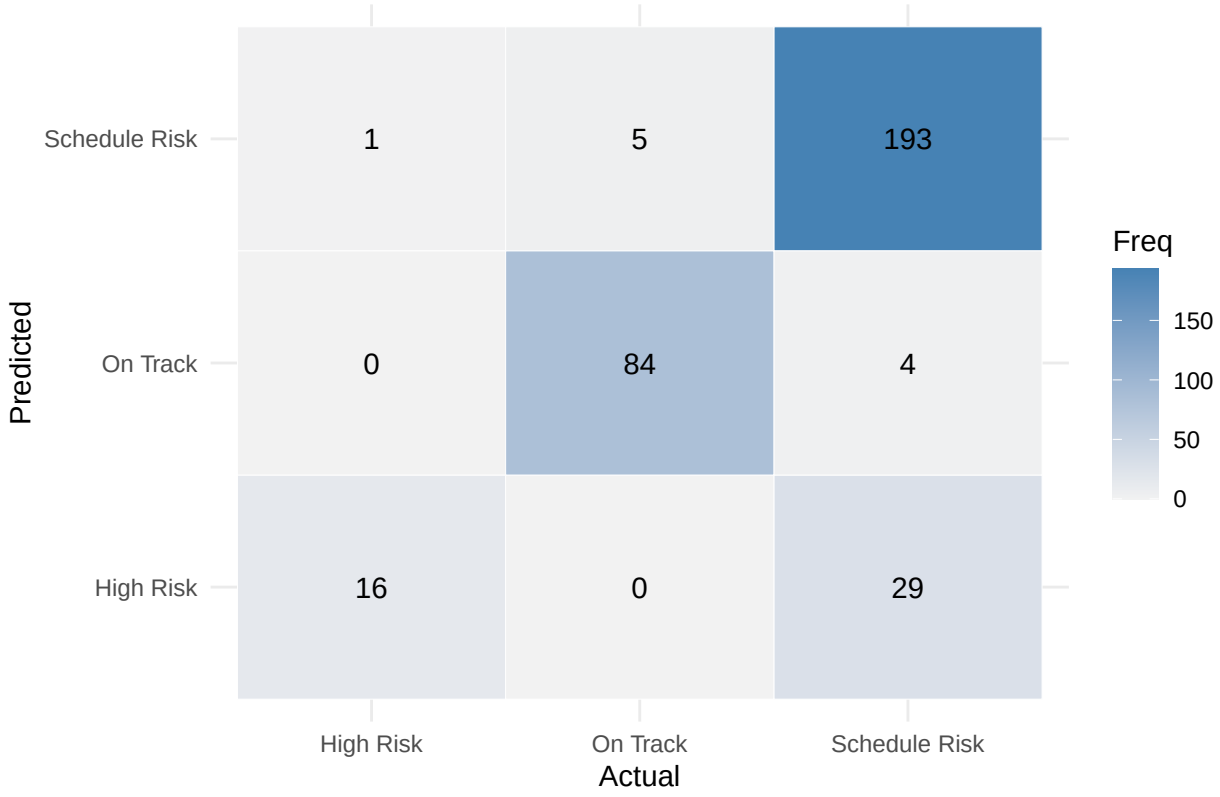

```
plot_conf_matrix(conf_df, "XGB", "Delay", 9)
```

Figure 9. XGB − Delay Classification



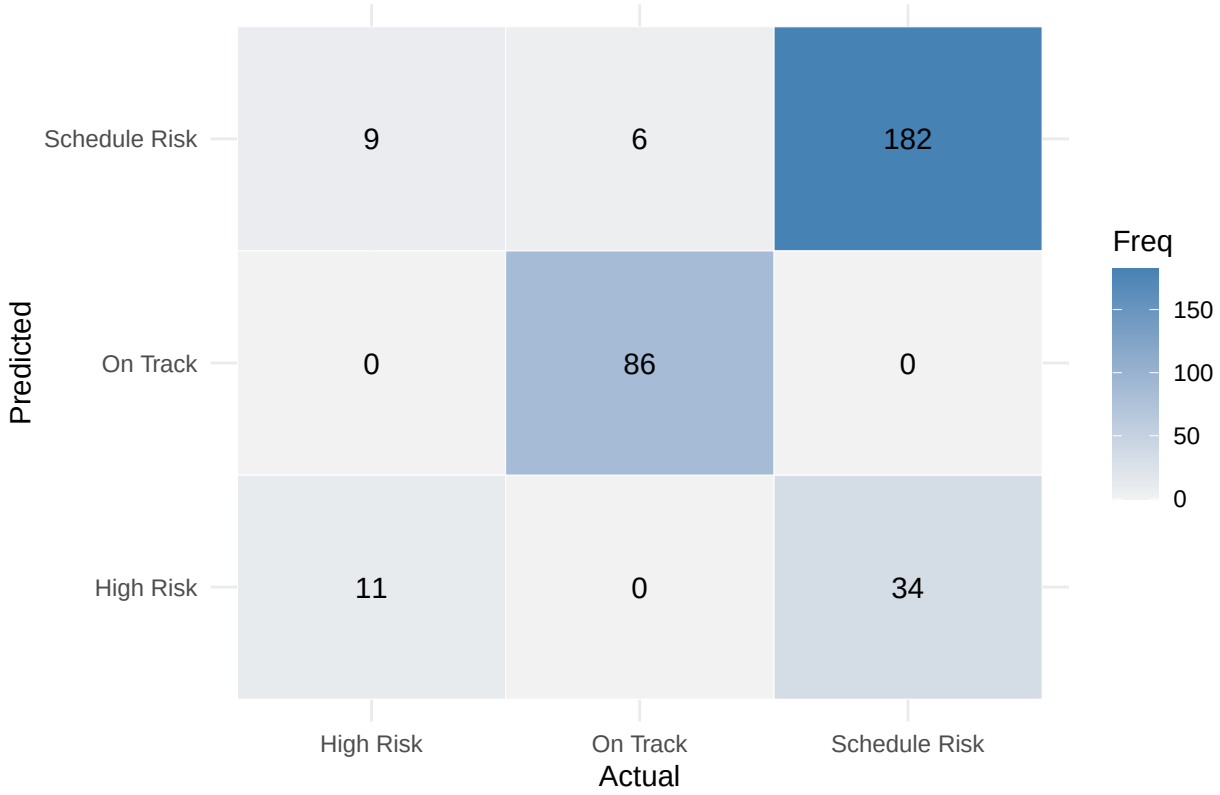

```
# Project Profile Classification
plot_conf_matrix(conf_df, "RF", "Profile", "5a")
```

Figure 5a. RF − Profile Classification




```
plot_conf_matrix(conf_df, "DT", "Profile", "5b")
```

Figure 5b. DT − Profile Classification




```
plot_conf_matrix(conf_df, "XGB", "Profile", 8)
```

Figure 8. XGB − Profile Classification