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

Souleymane Doumbia

2025-05-11

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"
    )
```

```r
  ) %>%
  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

```r
# 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

```r
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",
```

```r
    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", "Equipmen
    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|UPGRAD
      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)) ~ "R
      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 POUND
      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
```

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

      # === NEW CATEGORIES (FOR UNMATCHED PROJECTS) ===
      str_detect(agency_project_description, regex("MUSEUM|ZOO|AQUARIUM|BOTANICAL GARDEN|CULTURAL CENTE
      str_detect(agency_project_description, regex("SHELTER|TRANSITIONAL HOUSING|FAMILY RESIDENCE|HOMEL
      str_detect(agency_project_description, regex("FLOOD PROTECTION|MITIGATION|BULKHEAD|LEVEE|SHORELIN
      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", ignor
      str_detect(agency_project_description, regex("TUNNEL|SHAFT|CSO|STORAGE|UNDERGROUND|CONNECTION CHA
      str_detect(agency_project_description, regex("TREE|REFORESTATION|PLANTING|HORTICULTURE|GARDEN", i
      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", ignor
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 C
      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", ig
      str_detect(delay_desc, regex("PENDING APPROVAL OF NECESSARY PERMITS|STATE REQ CONTRACT", ignore_c
      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|
      str_detect(ten_year_plan_category, regex("WATER|TUNNEL|MAIN REPLACEMENT|PLANT|FILTER|CITY TUNNEL|
      str_detect(ten_year_plan_category, regex("SHELTER|HOUSING|HOMELESS|LOW TO MODERATE INCOME|PUBLIC 
      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_ca
      str_detect(ten_year_plan_category, regex("SUSTAINABILITY|GREEN INFRASTRUCTURE|ENVIRONMENT|WATER P
      TRUE ~ "Miscellaneous / Other"
```

```
    )
  )



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

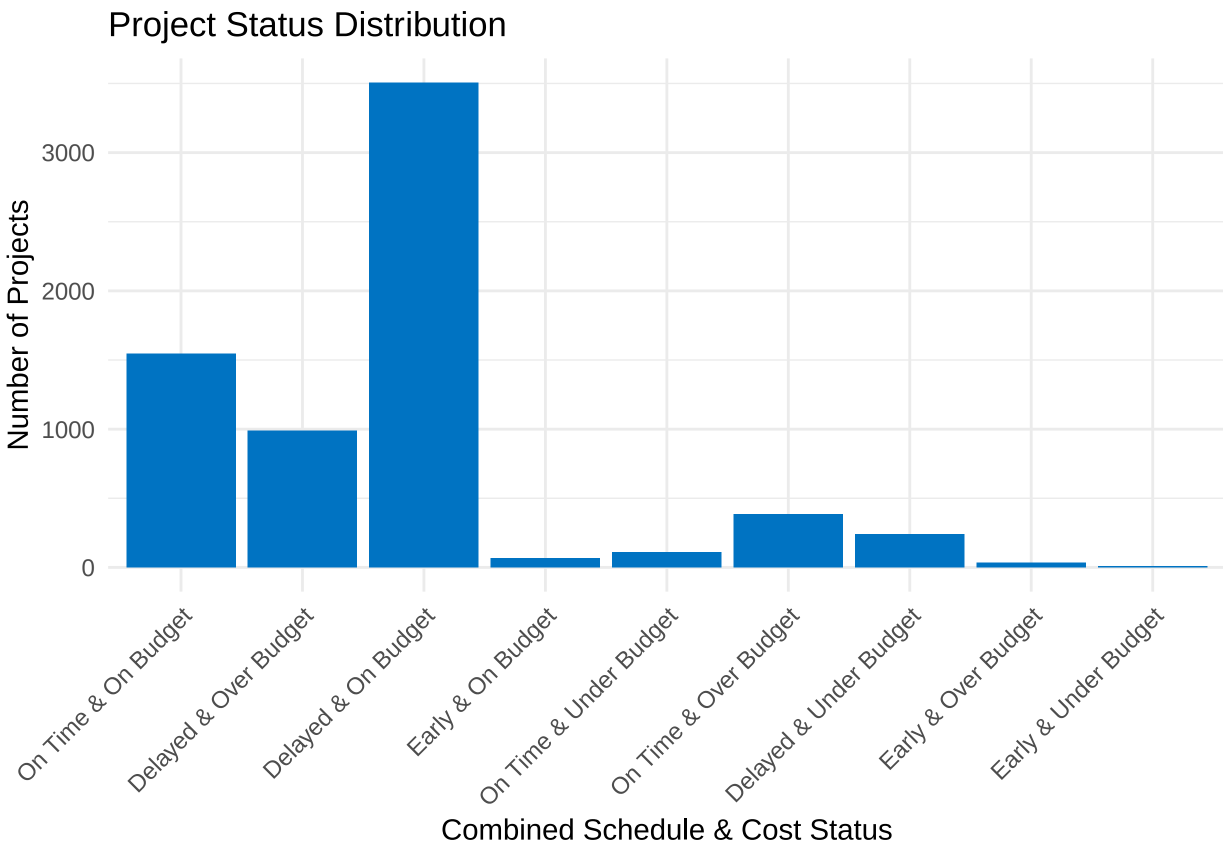
3.4 Summaise table of Project distibution

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

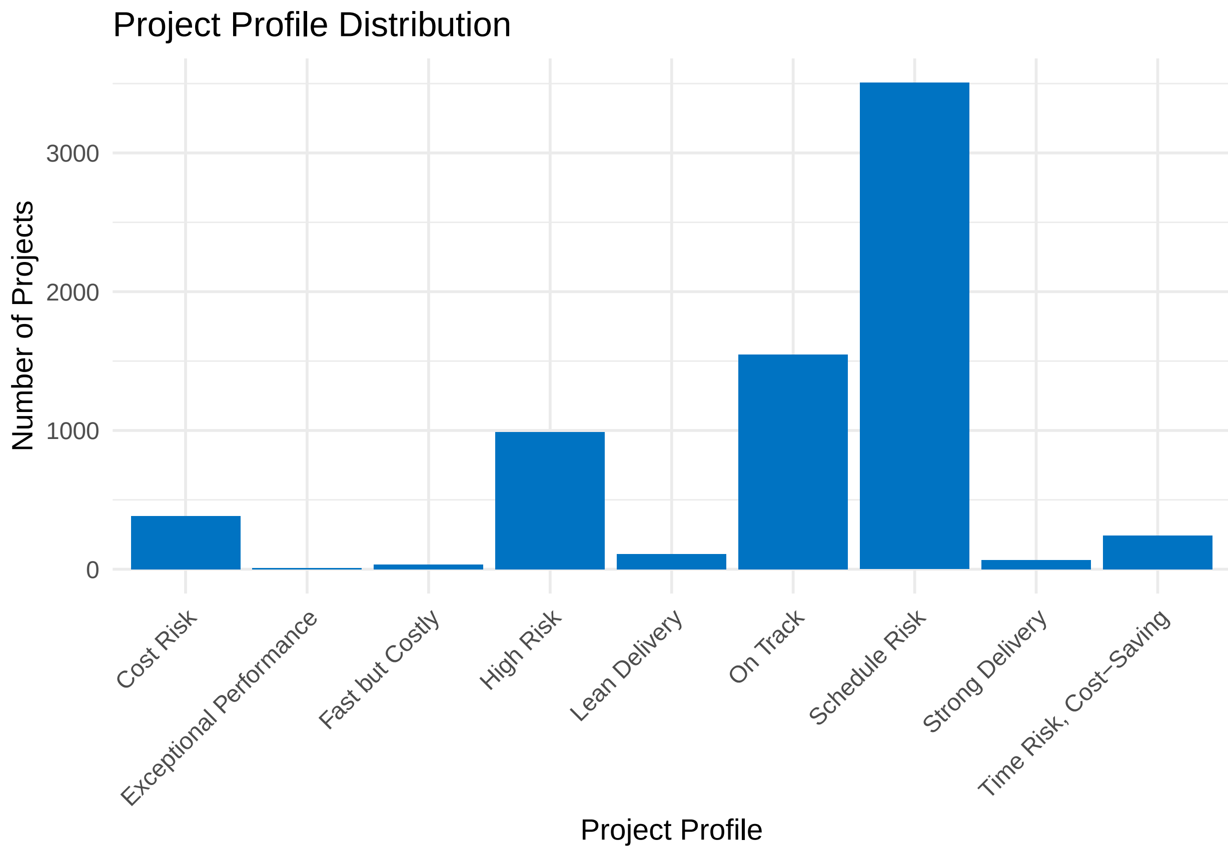
```
##
##              On Budget Over Budget Under Budget
##    Delayed        3506          989          242
##    Early            67           33           10
##    On Time        1546          384          109
```

3.5 EDA by current_phase

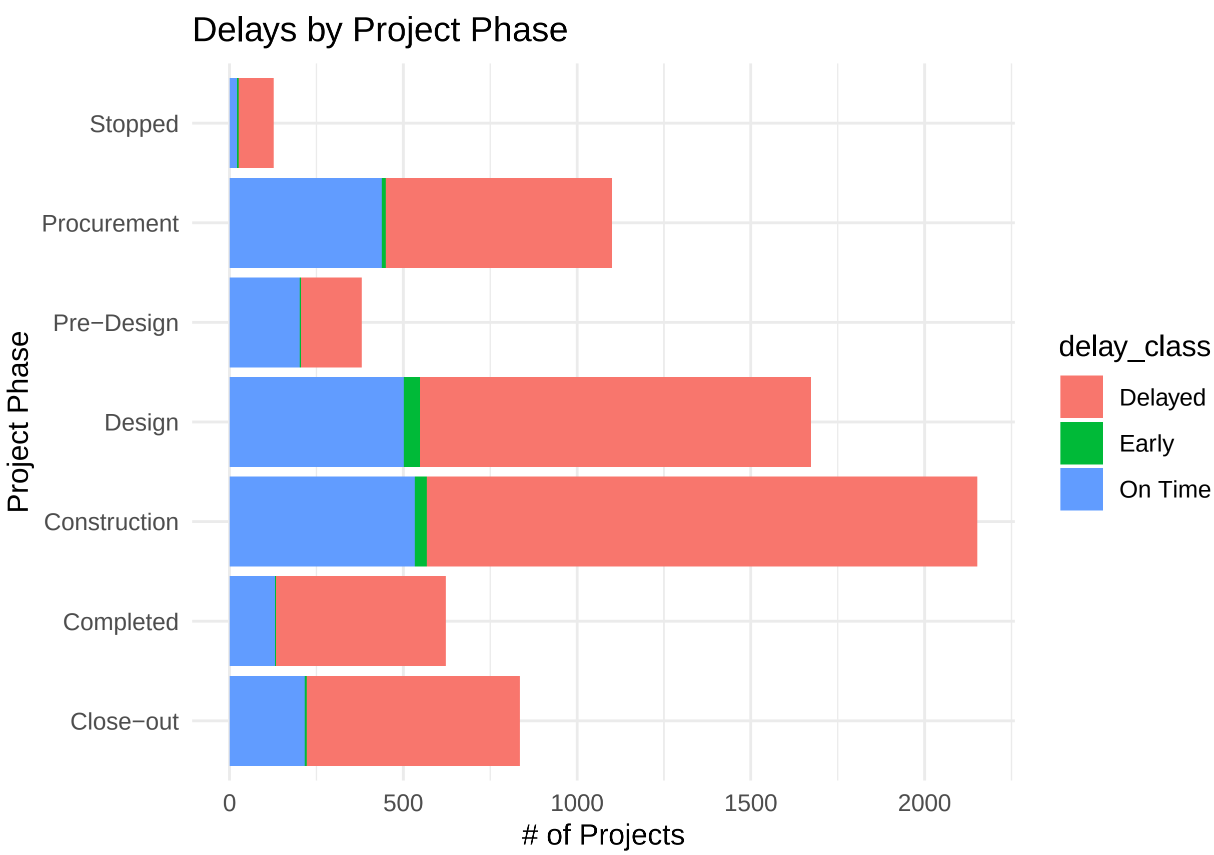
```
# Reorder factor levels by frequency
project_status_clean <- project_status_clean %>%
  mutate(status_combined = fct_infreq(status_combined))

# Bar plot with ordered categories
ggplot(project_status_clean, aes(x = status_combined)) +
  geom_bar(fill = "#0073C2FF") +
  labs(title = "Project Status Distribution",
       x = "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))
```



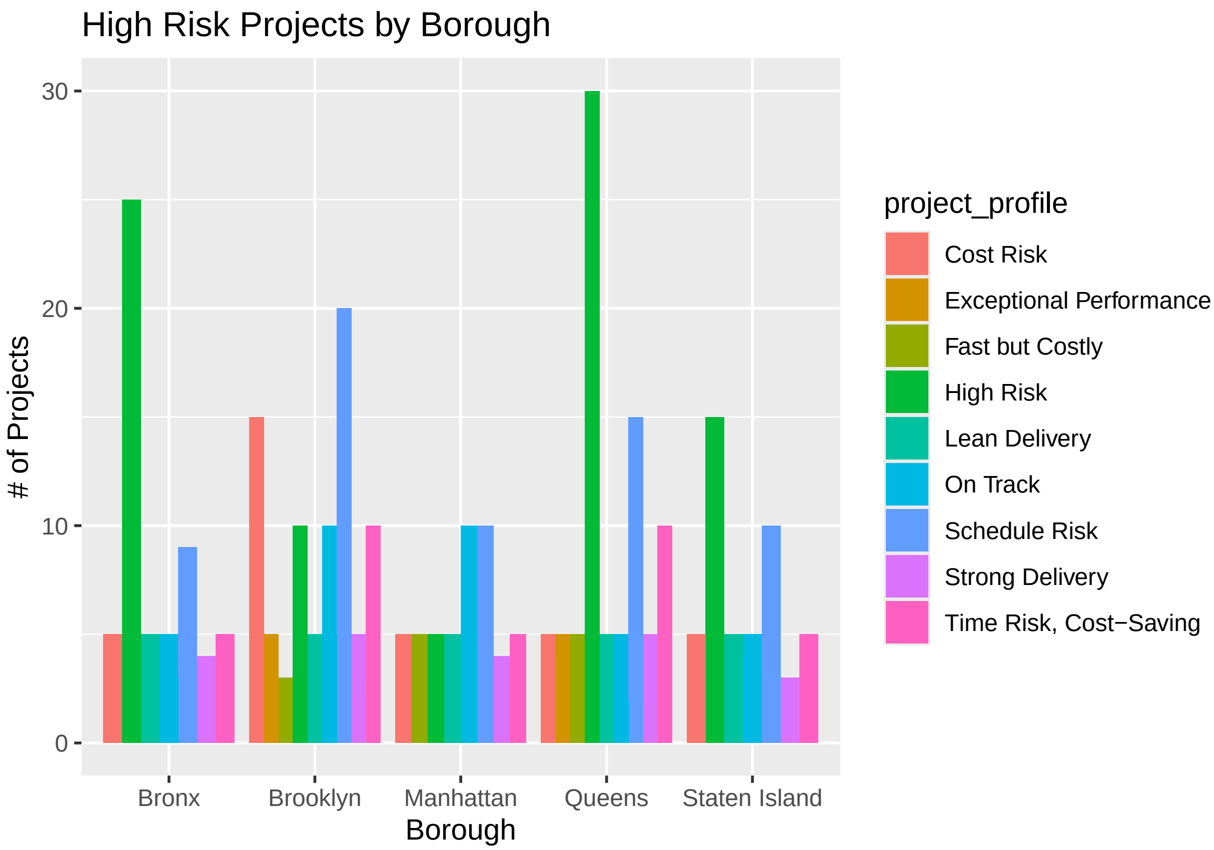

```r
# 2. Add High Risk flag
#project_status_clean <- project_status_clean %>%
#  mutate(
#    high_risk = ifelse(delay_class == "Delayed" & cost_class == "Over Budget", "Yes", "No")
#  )

# 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 = 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()
```

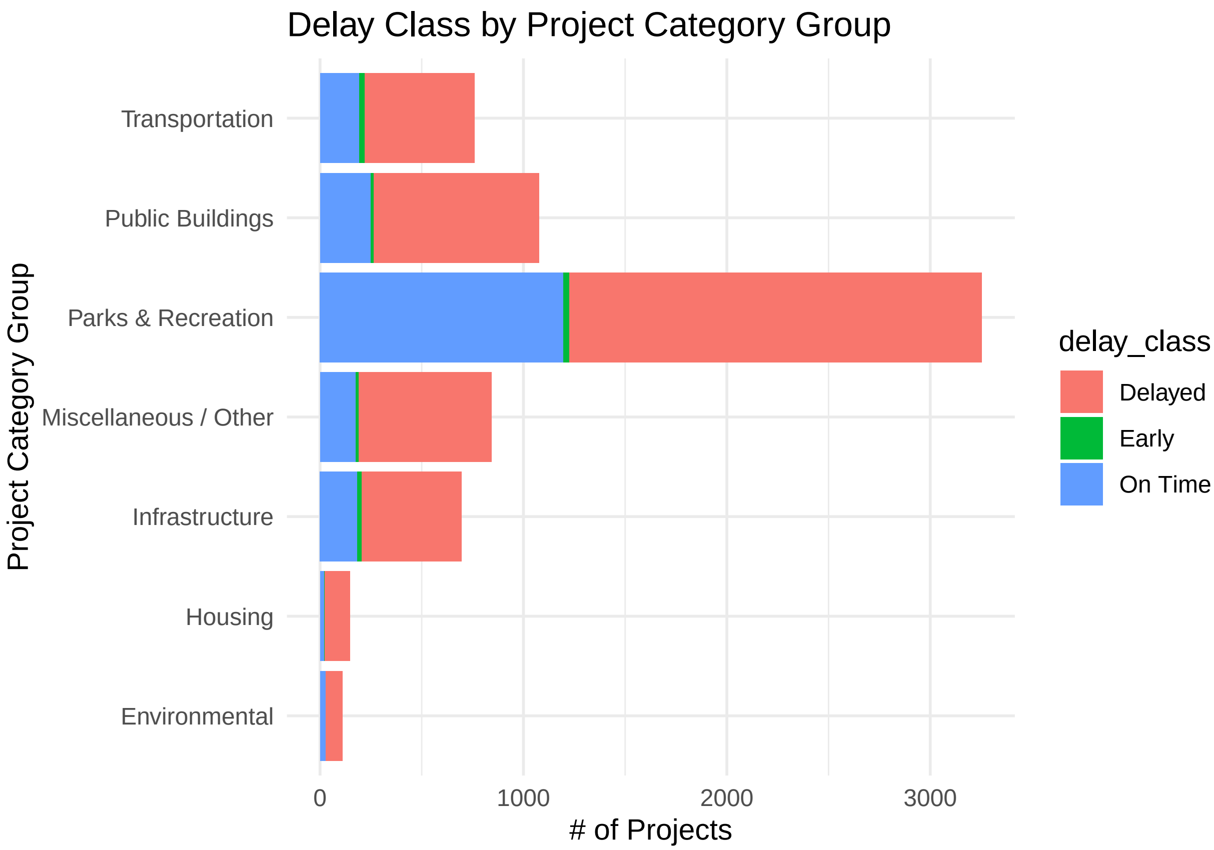
3.6 EDA by borough

```
## 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")
```

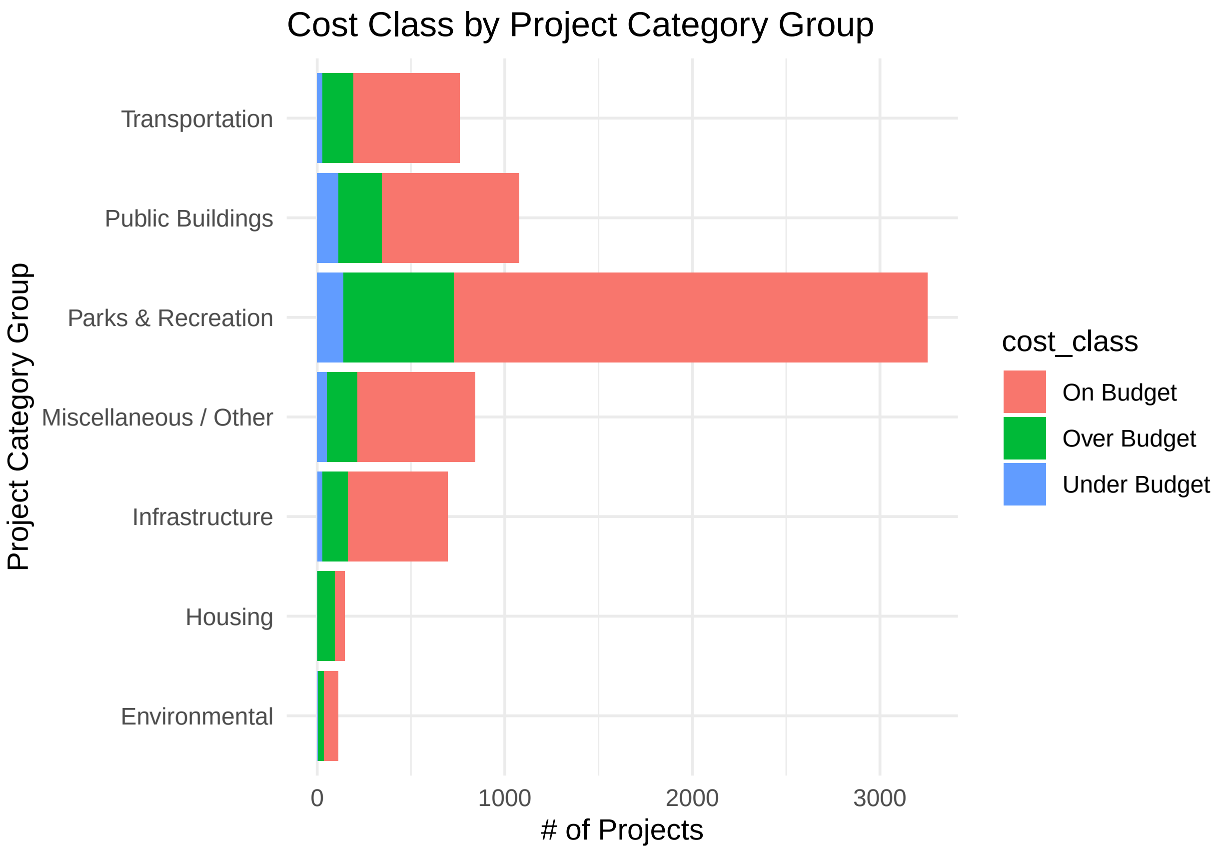


3.7 EDA by "ten_year_plan_category", "Cost Class" and by "high_risk Flag"

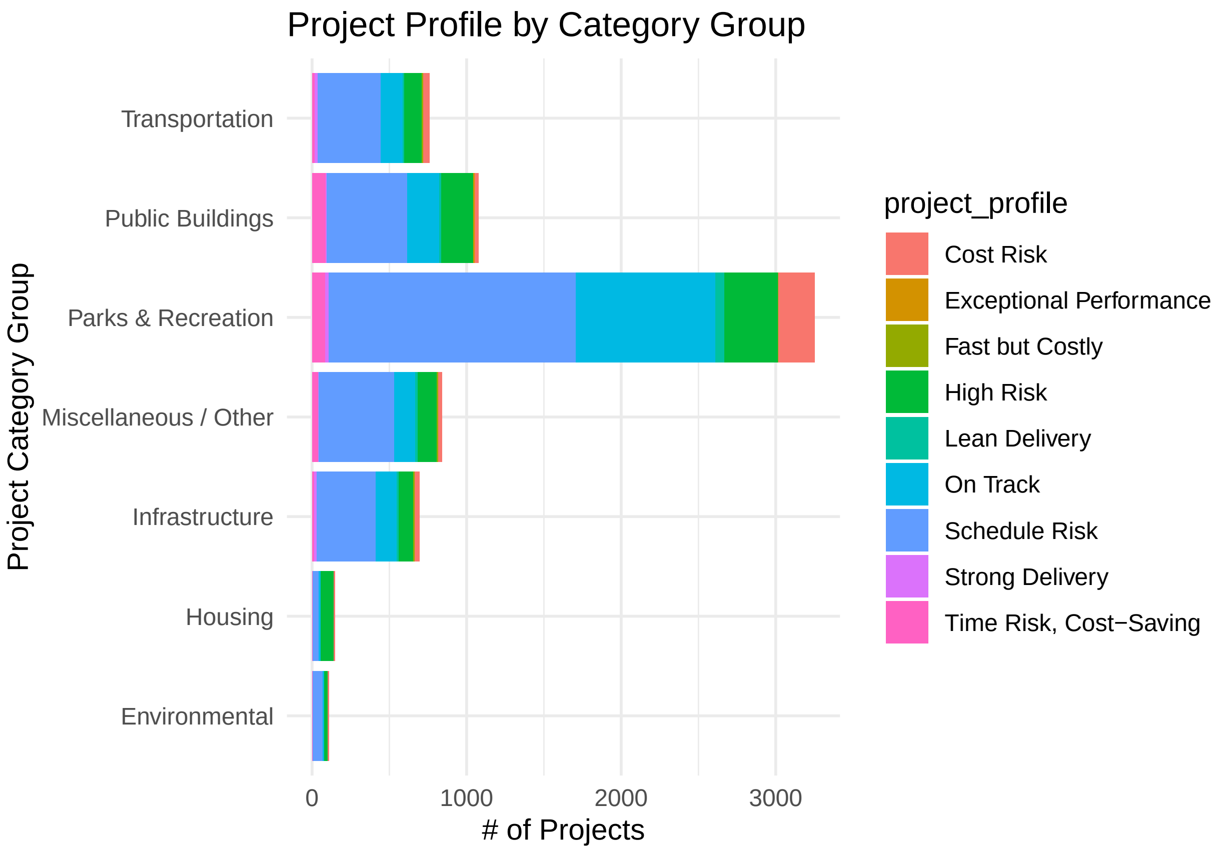
```
library(dplyr)
library(ggplot2)
library(stringr)


# 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()
```



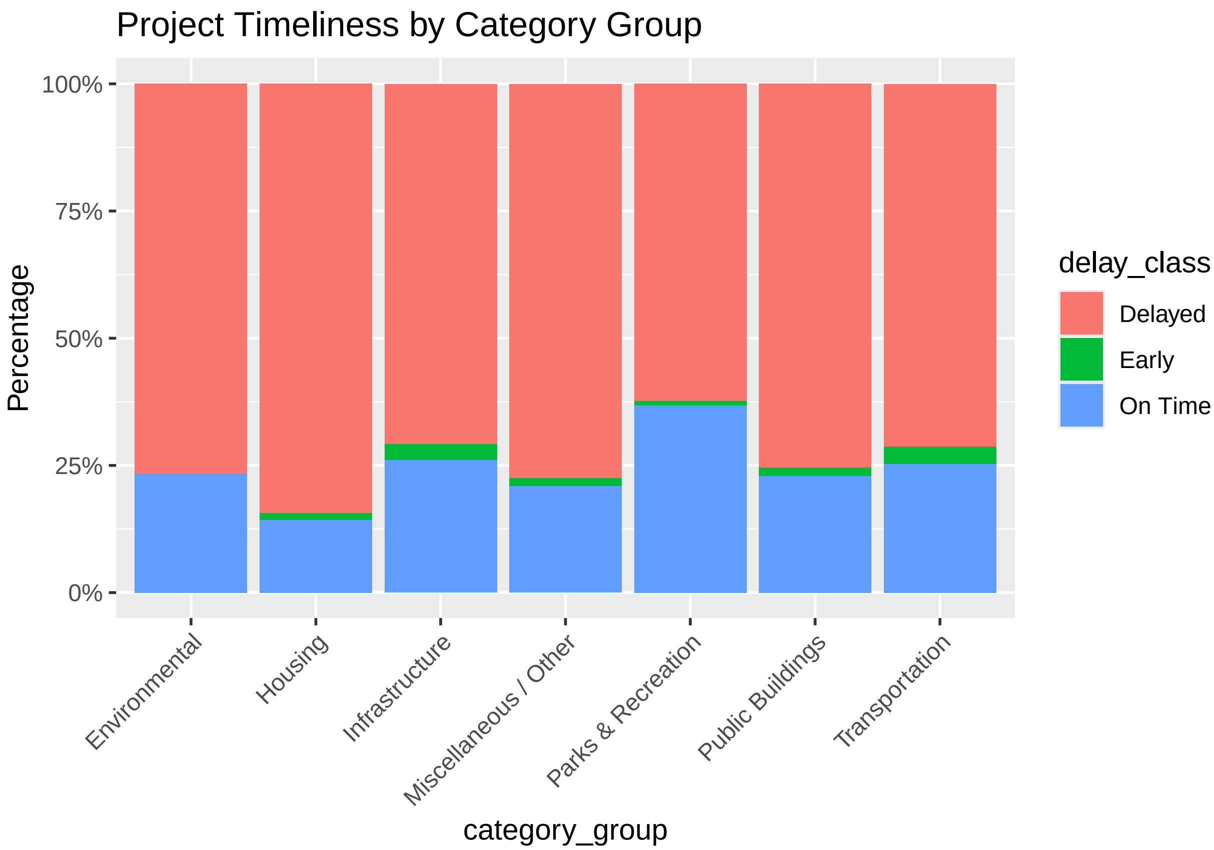

```r
# EDA: Cost Class
project_status_clean %>%
  count(category_group, cost_class) %>%
  ggplot(aes(x = category_group, y = n, fill = cost_class)) +
  geom_col(position = "stack") +
  labs(title = "Cost Class by Project Category Group",
       x = "Project Category Group",
       y = "# of Projects")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




```r
# 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

```r
library(tidyverse)
```

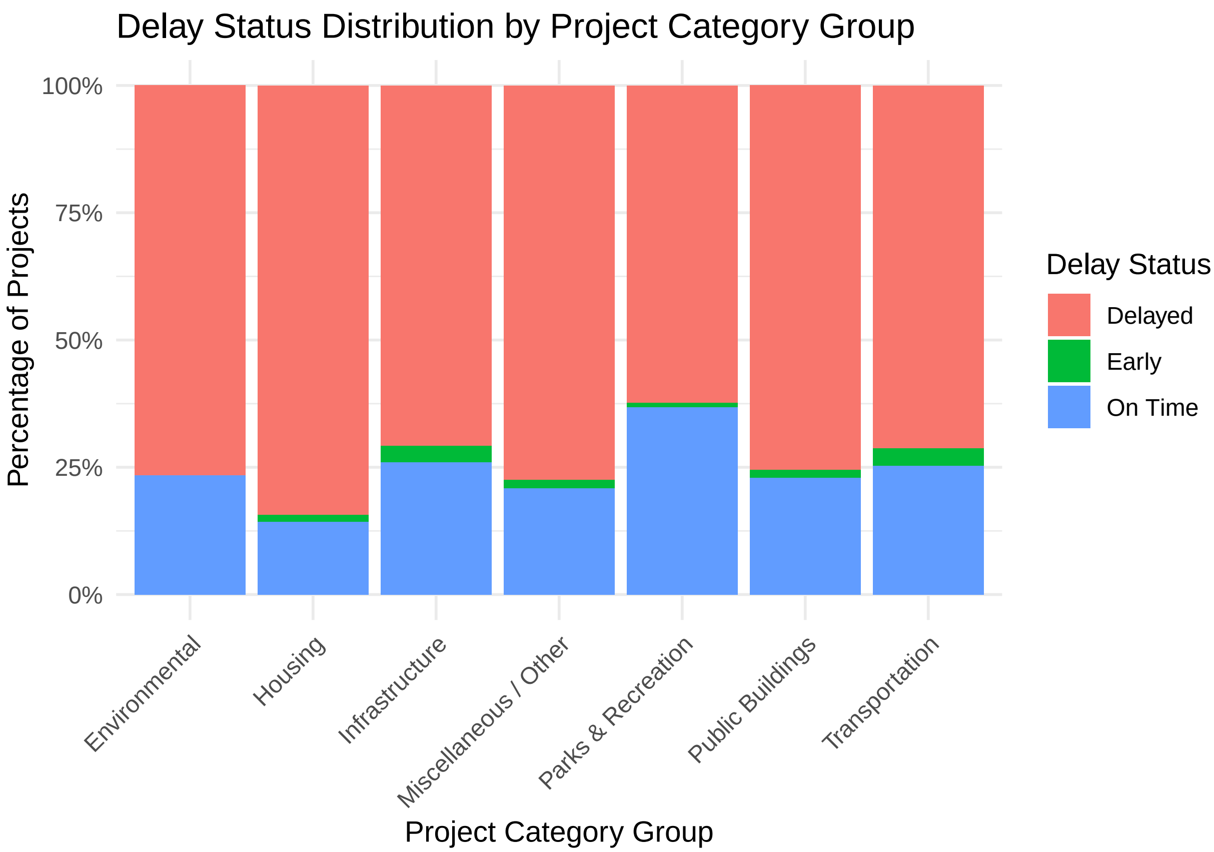
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v purrr   1.0.2     v tidyr   1.3.0
## v tibble  3.2.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

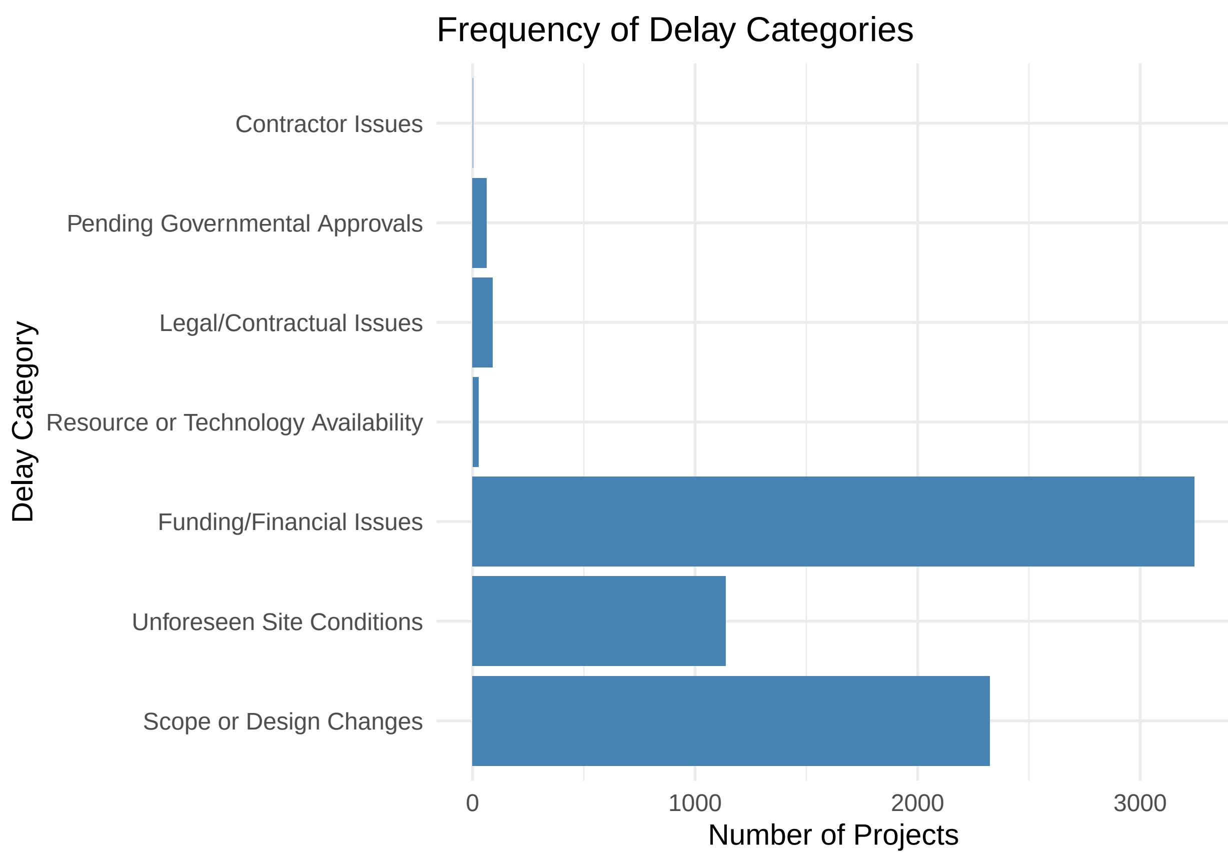
```r
library(scales)
```

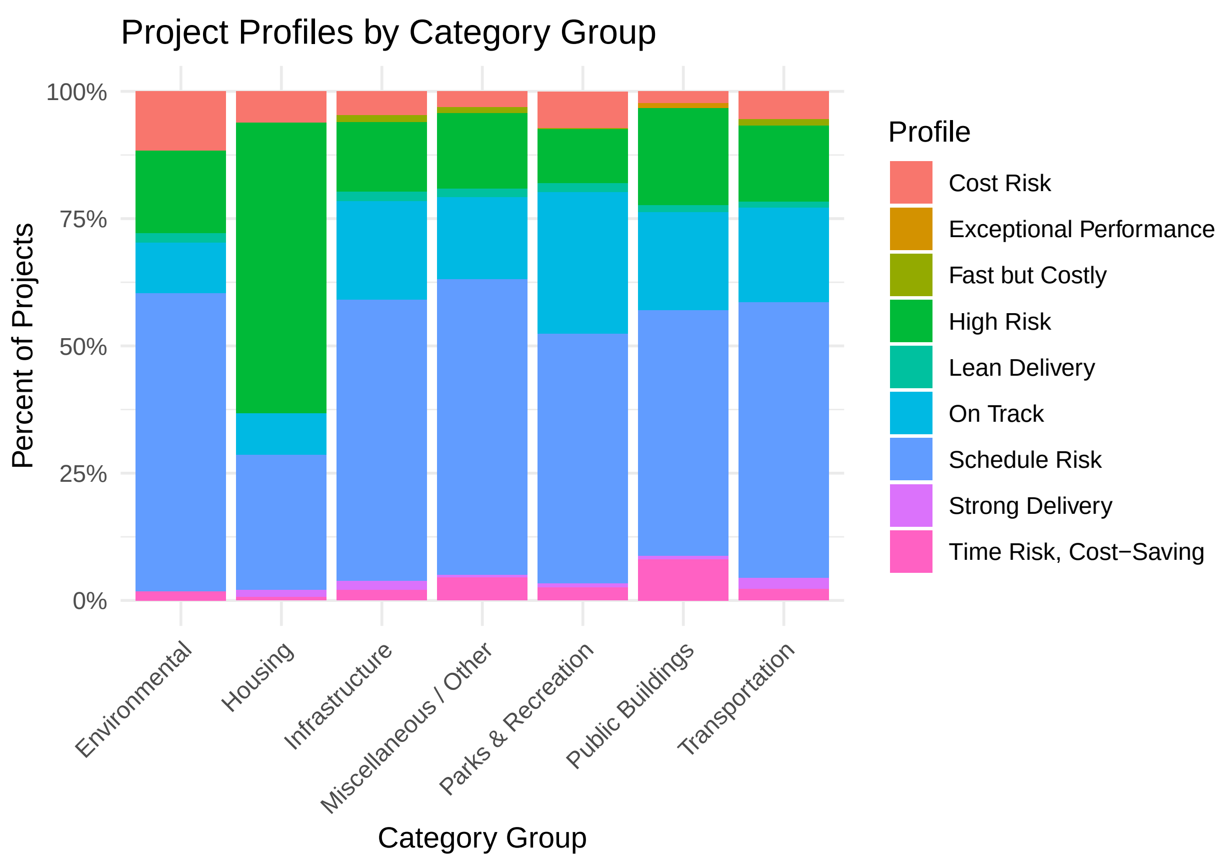
```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

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

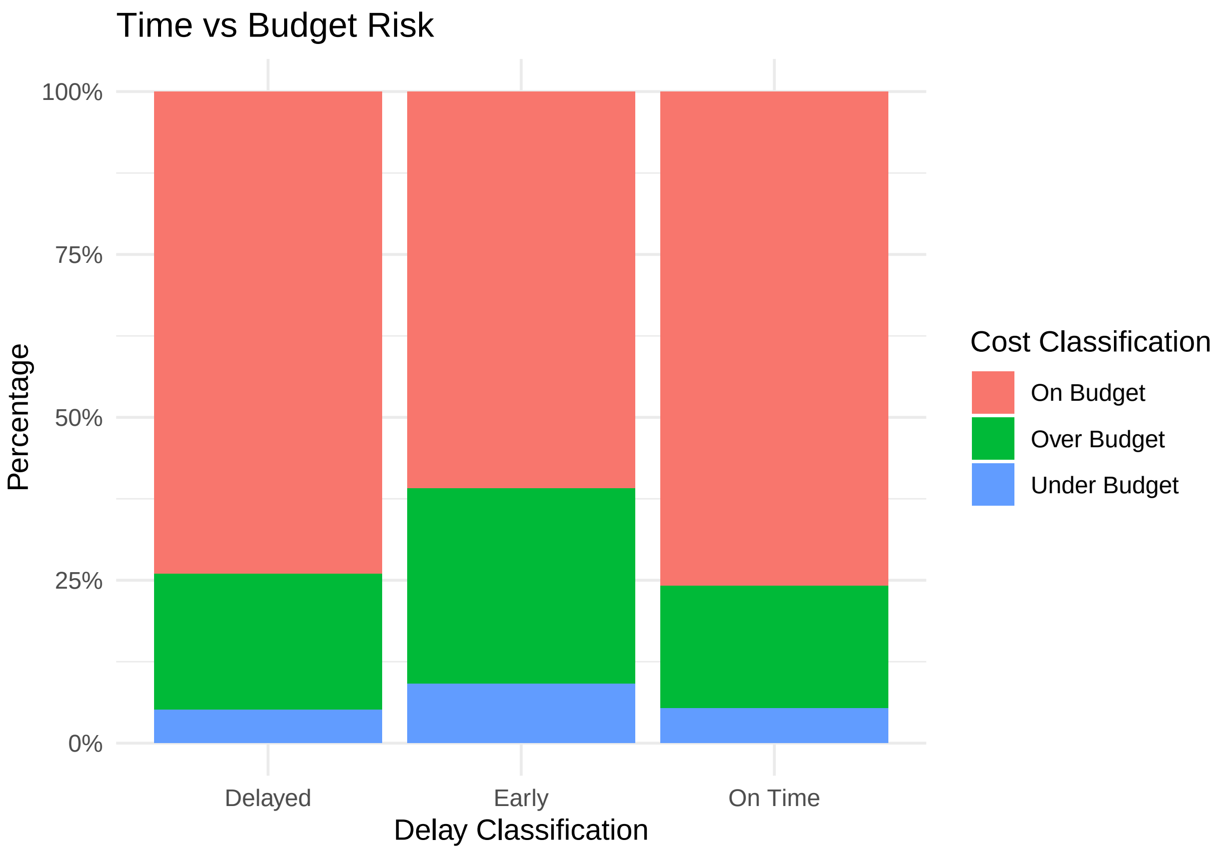
```
group_by(category_group) %>%
mutate(pct = count / sum(count)) %>%
ggplot(aes(x = category_group, y = pct, fill = delay_class)) +
geom_bar(stat = "identity", position = "fill") +
scale_y_continuous(labels = percent_format()) +
labs(
  title = "Delay Status Distribution by Project Category Group",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



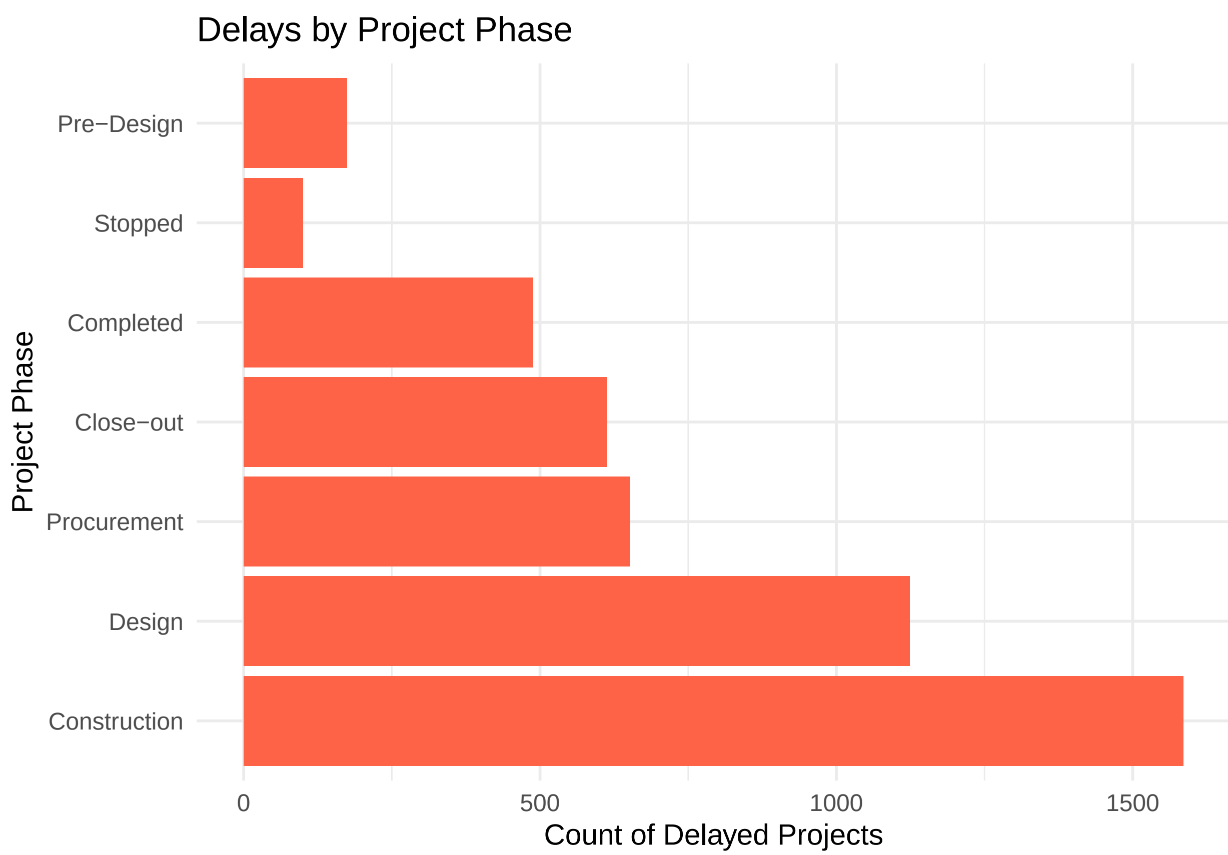

Frequency of Delay Categories
Delay Category
Contractor Issues
Pending Governmental Approvals
Legal/Contractual Issues
Resource or Technology Availability
Funding/Financial Issues
Unforeseen Site Conditions
Scope or Design Changes
0
1000
2000
3000
Number of Projects


Project Profiles by Category Group
Percent of Projects
100%
75%
50%
25%
0%
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

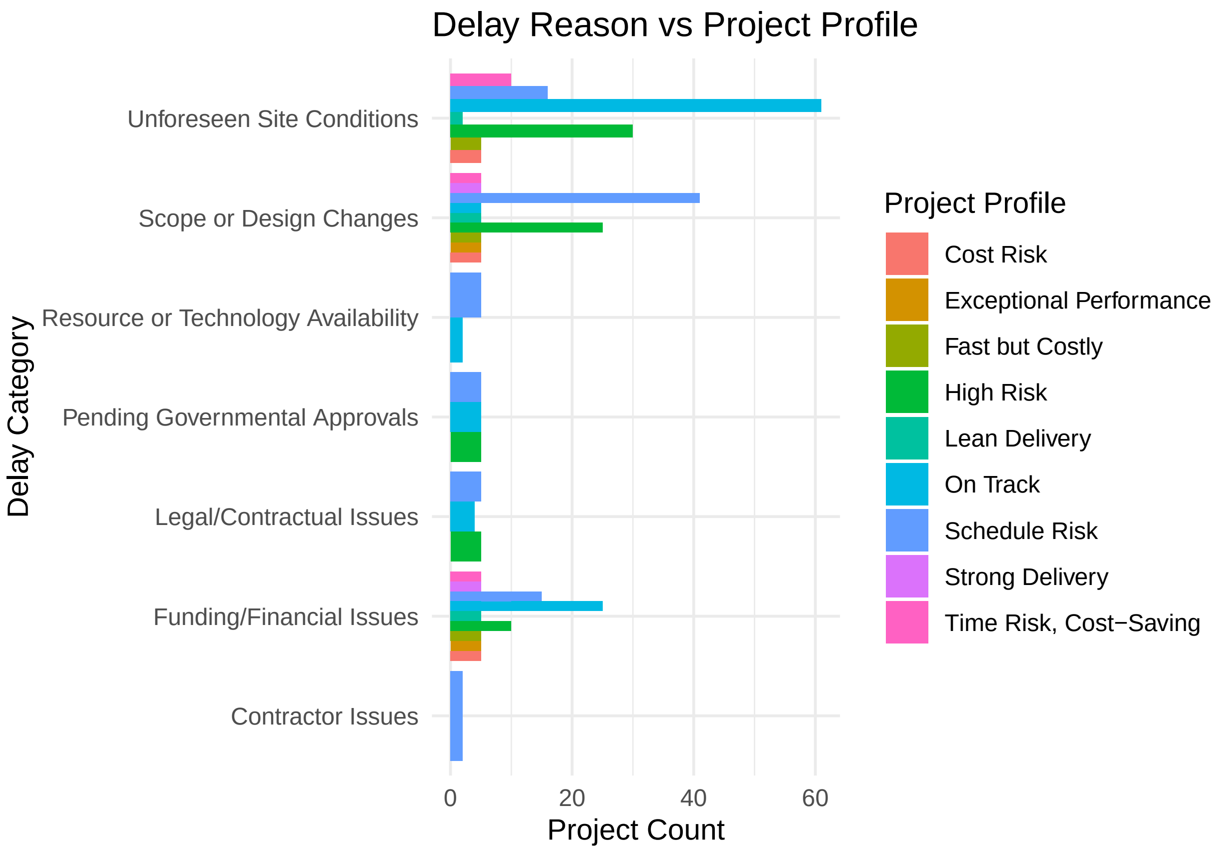
```r
### 1. Time vs Budget Risk: Delay Class vs Cost Class
library(ggplot2)

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 = "tomato") +
  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.
```

```r
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.
```

```r
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.
```

```r
#-------------------------------------------------------------------------------

# 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_cyclone
    avg_temp_anomaly = ifelse(is.na(avg_temp_anomaly), mean(avg_temp_anomaly, na.rm = TRUE), avg_temp_an
    avg_max_wind_speed = ifelse(is.na(avg_max_wind_speed), mean(avg_max_wind_speed, na.rm = TRUE), avg_m
  )
##-----------------------------------------------------------------------------------------------

# 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 Predicting cost_class and delay_class separately

```r
# ---- 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
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

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

```
##     combine
```

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

```
## Warning: package 'rpart.plot' was built under R version 4.3.2
## Loading required package: rpart
```

```r
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.3.3
##
## Attaching package: 'xgboost'
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(Matrix)
```

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

```r
# --- 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)
  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, ]
```

```r
  # --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)

  # --- 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
  )
```

```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[[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
```

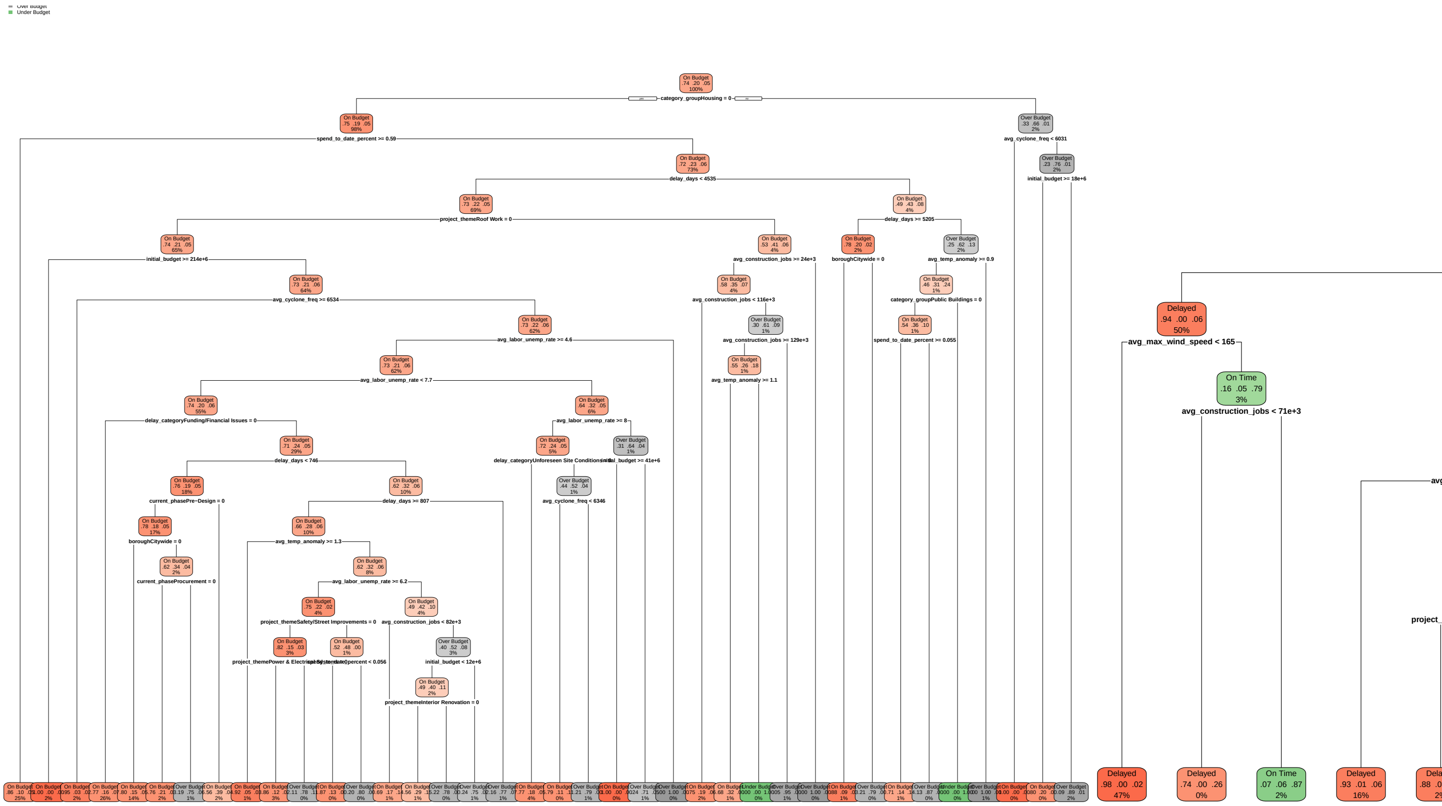
```
## [9]  train-mlogloss:0.465903+0.008306    test-mlogloss:0.524103+0.013163
## [10] train-mlogloss:0.445666+0.009749    test-mlogloss:0.509308+0.013379
## [11] train-mlogloss:0.431308+0.009563    test-mlogloss:0.497061+0.012076
## [12] train-mlogloss:0.421572+0.008723    test-mlogloss:0.488765+0.010653
## [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
```

4.7 Implementing modeling to predict project_profile

```r
# ---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()
```

```r
# 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
```

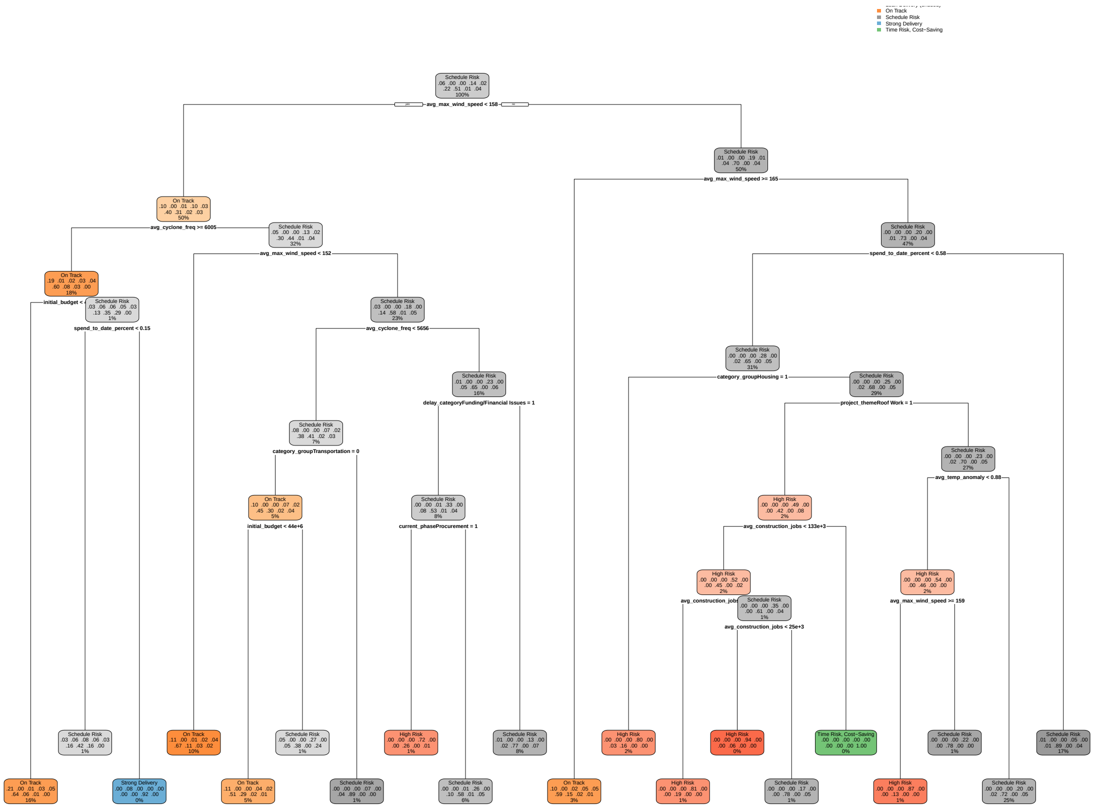
```
##                         Class: Strong Delivery Class: Time Risk, Cost-Saving
## Sensitivity                          0.000000                       0.00000
## Specificity                          1.000000                       1.00000
## Pos Pred Value                            NaN                           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
```

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

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

```r
# 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
```



```r
# 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
```