MDML Assignment 6

Trent Yu

Due: December 5, 2024. Total Points: 100.

Instructions

1. Fill in the `author` field at the top of this .Rmd file and indicate who you worked with, if anyone. Note that you must turn in your own work (see below).
2. Read the rest of these instructions and the "Grading" section.
3. Starting with the "Assignment" section, complete the rest of this R Markdown file from top to bottom. You will provide R code and some written responses in this Markdown file, and you will fill in the function outlines in the provided `assignment6.R` script. Do **not** write any code outside of the functions included in `assignment6.R` and do **not** add additional arguments to any function in that script. Do **not** edit this Markdown file except as directed in the instructions. After you have filled in a function in `assignment6.R`, you can run the corresponding code block in this Markdown file to progress through the assignment. When you finish the assignment, knit this file to obtain a pdf.
4. You must **only** use the packages loaded in this Markdown file in this assignment. Do not install or load any other packages.
5. I strongly suggest that you create and test your code in a "scratch" R script (don't turn that script in) before putting your answers in this file and in `assignment6.R`. That is, you should consider this file and the R script as the place to put your final answers for each question, not where you figure out how to do each question.
6. None of your functions should take more than a few minutes to run. That is, even if you believe your function answers a question correctly, you will lose points if we cannot run it in a reasonable amount of time (under five minutes); if this occurs, you should write more efficient code for your function.
7. Your submission for this assignment should consist only of these three files; do **not** change the file names:
    - `assignment6_workflow.Rmd`
    - `assignment6_workflow.pdf`
    - `assignment6.R`

Tips

1. Start early! In the past, students have found assignments in this class to be quite time-consuming; there is a reason you are given two weeks for each one.
2. Running code on large datasets can take a long time. You can save time by sampling a small subset of rows to confirm your code works before running it on the whole dataset. Similarly, you may find it helpful to use `dplyr::slice_sample()` to select a random subset of points to plot if your plots are taking a long time to load.

Grading

Submit your assignment files on Brightspace. Assignments submitted after 12:30pm on the due date are considered late. Please see the syllabus for the late policy.

You may discuss this assignment with your professor, course assistant, and classmates, but you must turn in your own work. In particular, *you may not directly copy anyone else's code*; that would constitute plagiarism. You may **not** use ChatGPT or similar generative AI tools when doing your homework unless explicitly instructed to do so.

You will be graded on the following: how accurately you followed instructions; correctness, completeness, and clarity of your code and written answers; creativity (when applicable); and quality of visualizations (when applicable). Also, note that passing a test (implemented by a test function in the R scripts) does **not** mean that you have done a problem correctly; **failing** a test means that the autograder will not be able to grade that part of your script.

Assignment

For this assignment, you will assume the role of a data scientist working for New York City's Department of Health and Mental Hygiene. Your supervisor is in charge of restaurant inspections across the city, and due to hiring restrictions she has been forced to reassign inspectors to other tasks. She has tasked you with building a predictive model to help prioritize where the remaining inspectors go, and would like you to use open data to help identify which restaurants are likely to have a high "inspection score" (a higher score is usually worse).

Start by reading the following documentation about NYC's restaurant inspection process:

- Restaurant scoring and grading.
- Dataset column descriptions. (the descriptions given here may not exactly match what's in the data)
- links to reports about the grading system.

Setup

Create a folder on your computer for this assignment and put this R Markdown file and `assignment6.R` in that folder. If you are going to create a "scratch" R script to work in, put that in the folder as well. Make sure this folder is your working directory (either using the `setwd()` command or by going to Session –> Set Working Directory in the RStudio menu). Next, within this folder, create a subdirectory called `data`.

Download `DOHMH_New_York_City_Restaurant_Inspection_Results.csv` from Brightspace and move it into your `data/` subdirectory. Uncomment and run the code block below to load the packages you'll use in this assignment (install the `ranger` package first if necessary), and the restaurant inspection data. Note that the dataset consists of one row for each *violation*, not each *inspection* (i.e., there are often multiple rows for the same inspection; individual inspections are generally characterized by a restaurant id and date).

install.packages("ranger") library(tidyverse) library(ranger) library(ROCR) setwd("/Users/trentyu/Desktop/Messy Data Machine Learning/Assignment 6")

all_data <- read_csv('data/DOHMH_New_York_City_Restaurant_Inspection_Results.csv', na = c("","NA", "N/A"))

Part A: Processing the data. [60 points]

Question A1: Clean the data [25 points]

In `assignment6.R`, complete the `clean_restaurant_data()` function, which takes the `all_data` tibble as input. Inside the function, you should:

1. Drop all columns except the following 10 **columns** from `all_data`:

    - `CAMIS`
    - `BORO`
    - `CUISINE DESCRIPTION`
    - `ACTION`
    - `VIOLATION CODE`
    - `CRITICAL FLAG`
    - `SCORE`
    - `GRADE`
    - `INSPECTION TYPE`
    - `INSPECTION DATE`.

2. Rename the following 9 **columns** (`new_name`: `ORIGINAL_NAME`):

    - `id`: `CAMIS`
    - `borough`: `BORO`
    - `cuisine`: `CUISINE DESCRIPTION`
    - `action`: `ACTION`
    - `code`: `VIOLATION CODE`
    - `critical`: `CRITICAL FLAG`
    - `score`: `SCORE`
    - `grade`: `GRADE`
    - `inspection_type`: `INSPECTION TYPE`.

3. Create the following columns, then drop the original `INSPECTION DATE` column:

    - `inspection_date`: a date object created using `INSPECTION DATE`.
    - `inspection_year`: the year from `inspection_date`.

4. Find the columns that contain the values below, and shorten the following **values** for legibility (`new_value`: `original_value`):

    - `closed`: `Establishment Closed by DOHMH. Violations were cited in the following area(s) and those requiring immediate action were addressed.`
    - `re-closed`: `Establishment re-closed by DOHMH`
    - `re-opened`: `Establishment re-opened by DOHMH`
    - `no violations`: `No violations were recorded at the time of this inspection.`
    - `violations`: `Violations were cited in the following area(s).`
    - `cycle-compliance`: `Cycle Inspection / Compliance Inspection`
    - `cycle-initial`: `Cycle Inspection / Initial Inspection`
    - `cycle-re-inspection`: `Cycle Inspection / Re-inspection`
    - `cycle-reopening`: `Cycle Inspection / Reopening Inspection`
    - `cycle-second-compliance`: `Cycle Inspection / Second Compliance Inspection`
    - `pre-permit-nonop-compliance`: `Pre-permit (Non-operational) / Compliance Inspection`
    - `pre-permit-nonop-initial`: `Pre-permit (Non-operational) / Initial Inspection`
    - `pre-permit-nonop-re-inspection`: `Pre-permit (Non-operational) / Re-inspection`

- `pre-permit-nonop-second-compliance`: `Pre-permit (Non-operational) / Second Compliance Inspection`
- `pre-permit-op-compliance`: `Pre-permit (Operational) / Compliance Inspection`
- `pre-permit-op-initial`: `Pre-permit (Operational) / Initial Inspection`
- `pre-permit-op-re-inspection`: `Pre-permit (Operational) / Re-inspection`
- `pre-permit-op-reopening`: `Pre-permit (Operational) / Reopening Inspection`
- `pre-permit-op-second-compliance`: `Pre-permit (Operational) / Second Compliance Inspection`

5. Remove all rows that have missing borough information, as well as rows that correspond to restaurants that haven't been inspected, inspections with missing or negative scores, or one of the following inspection types:

    - `"Calorie Posting / Re-inspection"`
    - `"Inter-Agency Task Force / Re-inspection"`
    - `"Smoke-Free Air Act / Re-inspection"`
    - `"Administrative Miscellaneous / Re-inspection"`
    - `"Trans Fat / Re-inspection"`
    - `"Inter-Agency Task Force / Initial Inspection"`.

Then, since some restaurants received different scores for the same inspection on the same day, replace all scores for any inspection for a given restaurant on a given day by the maximum score.

Your `clean_restaurant_data()` function should return the `cleaned_data` tibble. After completing `clean_restaurant_data()`, uncomment and run the code block below.

setwd("/Users/trentyu/Desktop/Messy Data Machine Learning/Assignment 6/data") source('assignment6.R') cleaned_data <- clean_restaurant_data(all_data) test_clean_restaurant_data()

Question A2: Restrict to initial cycle inspections [10 points]

In `assignment6.R`, complete the `make_initial_cycle_data()` function, which takes the `cleaned_data` tibble you just generated as input, and outputs a tibble called `initial_cycle_data` which has 6 columns: `id`, `inspection_date`, `borough`, `cuisine`, `inspection_year`, and `outcomes`.

Your function should create `initial_cycle_data` from `cleaned_data` by first restricting to initial cycle inspections conducted in 2017, 2018, and 2019. Then, ensure that each row in `initial_cycle_data` corresponds to a **unique** initial cycle inspection for a restaurant on a given date. Since `cleaned_data` may contain several rows that correspond to the same initial cycle inspection on a particular date, reconcile values in `borough`, `cuisine`, `inspection_type`, and `inspection_year` by choosing the value from the first row for that inspection; construct the binary `outcomes` variable to be `TRUE` if the score for any row corresponding to that initial cycle inspection was 28 or higher, and `FALSE` otherwise.

To illustrate, suppose that there are 30 rows in `cleaned_data` for "Ravi's Pizza," corresponding to five different inspections on different dates, each of which had 6 violations per inspection. Suppose that one initial cycle inspection occurred in 2017, one in 2018, and two in 2019 (the other inspection was either in another year or wasn't an initial cycle inspection). Then, there should be exactly four rows in `restaurant_data` corresponding to the four initial cycle inspections of "Ravi's Pizza."

After completing `make_initial_cycle_data()`, uncomment and run the code block below.

source('assignment6.R') initial_cycle_data <- make_initial_cycle_data(cleaned_data) test_make_initial_cycle_data()

Question A3: Create features. [25 points]

In `assignment6.R`, complete the `make_features()` function, which takes two arguments, `initial_cycle_data`, and `cleaned_data`, and outputs a tibble called `restaurant_data`, which you will use to fit a predictive model to prioritize initial cycle inspections in Part B, next. Importantly, we should only use or construct features that could be available *before* a given initial cycle inspection takes place. Inside `make_features()`, do the following:

1. Rename `initial_cycle_data` as `restaurant_data` and add features called `month` and `weekday` to `restaurant_data`, which encode the month and day of week of the initial cycle inspection as abbreviated character vectors (e.g., "Jan" or "Sun").
2. Add the following four features to each initial cycle inspection in `restaurant_data`, constructed from the historical inspection records for that restaurant contained in `cleaned_data`:
    - `num_previous_low_inspections`: The number of previous inspections with `score < 14`.
    - `num_previous_med_inspections`: The number of previous inspections with both `score >= 14` and `score < 28`.
    - `num_previous_high_inspections`: The number of previous inspections with `score >= 28`.
    - `num_previous_closings`: The number of previous inspections resulting in closing or re-closing.

Hint: one way to do this is to left join `restaurant_data` with `cleaned_data` on `id`, then filter to retain rows where the inspection date from `cleaned_data` is before the corresponding inspection date from `restaurant_data` (you may get a warning when you do this left join). Group by `id` and the inspection date from `restaurant_data` and use `mutate` to create the four features above, then join these features back to `restaurant_data`. Finally, replace `NA` values of features with 0 for restaurants that had no prior inspections.

After completing `make_features()`, uncomment and run the code block below.

source('assignment6.R') restaurant_data <- make_features(initial_cycle_data, cleaned_data) test_make_features()

Part B: Predict outcomes of future restaurant inspections [40 points]

Question B1: Fit predictive models. [10 points]

In `assignment6.R`, complete the `fit_models()` function, which takes as input
the `restaurant_data` tibble generated by `make_features()` in Question A3. Inside `fit_models()`, do the following:

1. Make sure that the `month`, `weekday`, and `outcomes` variables are (unordered) factors.
2. Create a training set called `train` consisting of all initial cycle inspections in 2017 and 2018, and a testing set called `test` consisting of all initial cycle inspections in 2019.
3. Fit a logistic regression model on `train`, predicting `outcomes` as a function of only `cuisine`, `borough`, `month`, and `weekday`. Generate predicted probabilities of `outcomes == TRUE` for all initial cycle inspections in `test`. Return these predicted probabilities as a vector named `logistic_predictions`. Then, compute the AUC of this model (a number between 0 and 1) on `test`, using the ROCR package. Return the AUC as a vector named `auc_logistic` that has a single element.

4. Fit a random forest model on `train` using the `ranger` package, predicting `outcomes` as a function of `cuisine`, `borough`, `month`, `weekday`, **and** the four historical features created in Question A3. Use 1000 trees, and make sure that both `respect.unordered.factors` and `probability` are `TRUE`, but other settings can have default values. Generate predicted probabilities of `outcomes == TRUE` for all initial cycle inspections in `test`. Return these predicted probabilities as a vector named `rf_predictions`. Return the AUC of this model on `test` as a vector named `auc_rf` that has a single element.

`fit_models()` should return a named list with the following elements:

- `outcomes`, the true outcomes variable for all initial cycle inspections in `test`
- `logistic_predictions`
- `rf_predictions`
- `auc_logistic`
- `auc_rf`

After completing `fit_models()`, uncomment and run the code block below.

source('assignment6.R') output <- fit_models(restaurant_data) outcomes <- output[['outcomes']] logistic_predictions <- output[['logistic_predictions']] rf_predictions <- output[['rf_predictions']] auc_logistic <- output[['auc_logistic']] auc_rf <- output[['auc_rf']] test_fit_models()

Question B2: Create a precision-at-k plot. [15 points]

Create one plot with two precision-at-k curves for `test` (one for each model), using the true outcomes in `outcomes` and the predicted probabilities from your logistic regression model (`logistic_predictions`) and from your random forest model (`rf_predictions`). Your x-axis should display the number of restaurants, ranked from highest model-estimated probability of the outcomes to lowest model-estimated probability of the outcomes, and your y-axis should display the corresponding model precision. For example, if you were to use the random forest model to rank all restaurants from those most likely to have the outcomes to those least likely to have the outcomes, a point like (200, 0.3) would indicate that among the 200 highest-ranked restaurants, 30% of them had the outcomes.

I suggest you restrict the x-axis to be a reasonable range, say, excluding the first 100 inspections, so your visualization looks more sensible. (This is just like the recall-at-k% plot you might have made in Assignment 3, except the x-axis should be on the absolute scale (not percent scale), and the y-axis should display precision instead of recall.)

Include your code for the plot where indicated, and the plot itself below.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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(ranger)
```

```
## Warning: package 'ranger' was built under R version 4.3.3
```

```r
library(ROCR)
setwd("/Users/trentyu/Desktop/Messy Data Machine Learning/Assignment 6")

all_data <- read_csv('data/DOHMH_New_York_City_Restaurant_Inspection_Results.csv',
                     na = c("", "NA", "N/A"))
```

```
## Rows: 394917 Columns: 26
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (18): DBA, BORO, BUILDING, STREET, PHONE, CUISINE DESCRIPTION, INSPECTIO...
## dbl  (8): CAMIS, ZIPCODE, SCORE, Latitude, Longitude, Community Board, BIN, BBL
##
## 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
clean_restaurant_data <- function(all_data) {
  all_data %>%
    select(CAMIS, BORO, `CUISINE DESCRIPTION`, ACTION, `VIOLATION CODE`,
           `CRITICAL FLAG`, SCORE, GRADE, `INSPECTION TYPE`, `INSPECTION DATE`) %>%
    rename(
      id = CAMIS,
      borough = BORO,
      cuisine = `CUISINE DESCRIPTION`,
      action = ACTION,
      code = `VIOLATION CODE`,
      critical = `CRITICAL FLAG`,
      score = SCORE,
      grade = GRADE,
      inspection_type = `INSPECTION TYPE`
    ) %>%
    mutate(
      inspection_date = lubridate::mdy(`INSPECTION DATE`),
      inspection_year = lubridate::year(inspection_date)
    ) %>%
    select(-`INSPECTION DATE`) %>%
    mutate(
      action = recode(action,
        `Establishment Closed by DOHMH. Violations were cited in the following area(s) and those requir
        `Establishment re-closed by DOHMH` = "re-closed",
        `Establishment re-opened by DOHMH` = "re-opened",
        `No violations were recorded at the time of this inspection.` = "no violations",
        `Violations were cited in the following area(s).` = "violations"
      ),
      inspection_type = recode(inspection_type,
        `Cycle Inspection / Compliance Inspection` = "cycle-compliance",
        `Cycle Inspection / Initial Inspection` = "cycle-initial",
        `Cycle Inspection / Re-inspection` = "cycle-re-inspection",
        `Cycle Inspection / Reopening Inspection` = "cycle-reopening",
```

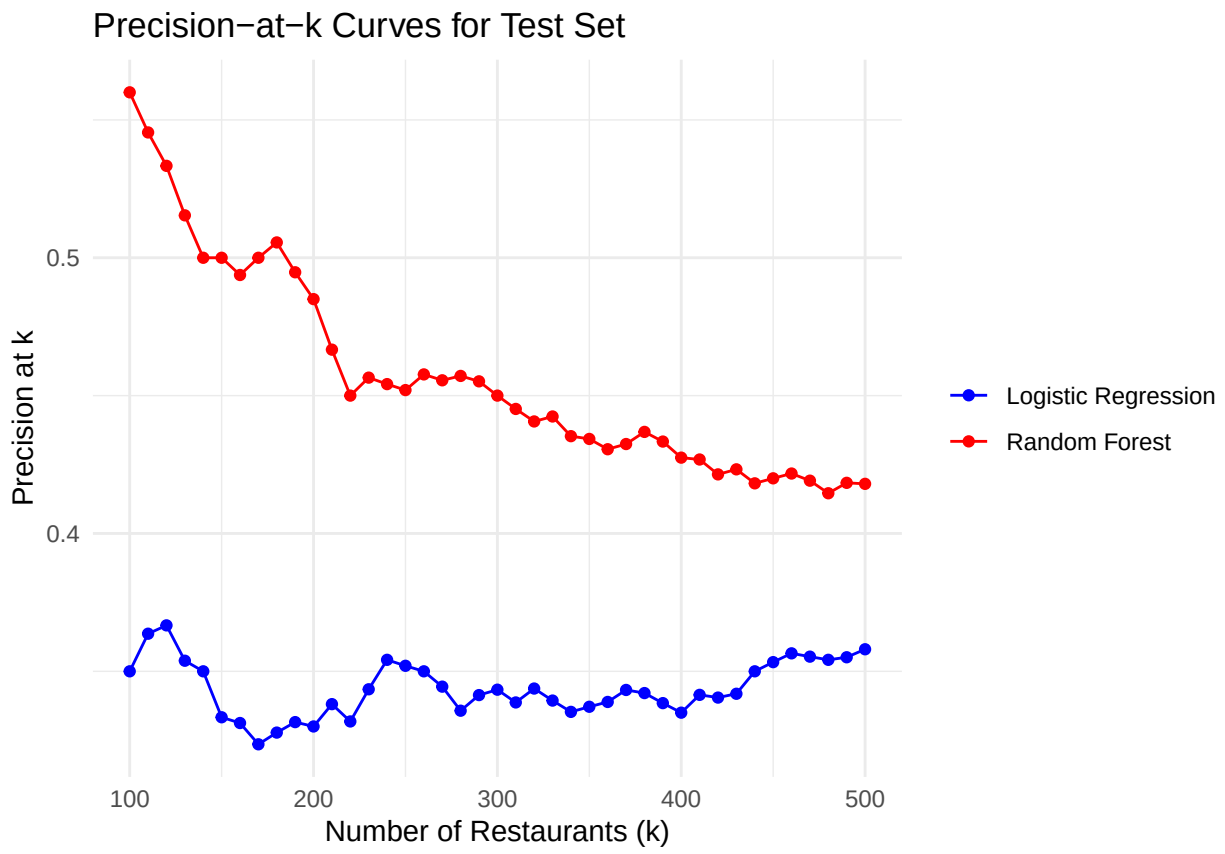
```
        `Cycle Inspection / Second Compliance Inspection` = "cycle-second-compliance",
        `Pre-permit (Non-operational) / Compliance Inspection` = "pre-permit-nonop-compliance",
        `Pre-permit (Non-operational) / Initial Inspection` = "pre-permit-nonop-initial",
        `Pre-permit (Non-operational) / Re-inspection` = "pre-permit-nonop-re-inspection",
        `Pre-permit (Non-operational) / Second Compliance Inspection` = "pre-permit-nonop-second-compli
        `Pre-permit (Operational) / Compliance Inspection` = "pre-permit-op-compliance",
        `Pre-permit (Operational) / Initial Inspection` = "pre-permit-op-initial",
        `Pre-permit (Operational) / Re-inspection` = "pre-permit-op-re-inspection",
        `Pre-permit (Operational) / Reopening Inspection` = "pre-permit-op-reopening",
        `Pre-permit (Operational) / Second Compliance Inspection` = "pre-permit-op-second-compliance"
      )
    ) %>%
    filter(
      !is.na(borough),
      !is.na(score),
      score >= 0,
      !inspection_type %in% c(
        "Calorie Posting / Re-inspection",
        "Inter-Agency Task Force / Re-inspection",
        "Smoke-Free Air Act / Re-inspection",
        "Administrative Miscellaneous / Re-inspection",
        "Trans Fat / Re-inspection",
        "Inter-Agency Task Force / Initial Inspection"
      )
    ) %>%
    group_by(id, inspection_date) %>%
    mutate(score = max(score, na.rm = TRUE)) %>%
    ungroup()
}


make_initial_cycle_data <- function(cleaned_data){
  if(isFALSE(tibble::is_tibble(cleaned_data))){
    stop("cleaned_data should be a tibble")
  }

  initial_cycle_data <- cleaned_data %>%
    filter(
      inspection_type == "cycle-initial",
      inspection_year %in% c(2017, 2018, 2019)
    ) %>%

    group_by(id, inspection_date) %>%

    summarize(
      borough = first(borough),
      cuisine = first(cuisine),
      inspection_year = first(inspection_year),
      outcome = any(score >= 28, na.rm = TRUE),
      .groups = "drop"
    )

  return(initial_cycle_data)
```

```
}


make_features <- function(initial_cycle_data, cleaned_data){
  if(isFALSE(tibble::is_tibble(initial_cycle_data))){
    stop("initial_cycle_data should be a tibble")
  }
  if(isFALSE(tibble::is_tibble(cleaned_data))){
    stop("cleaned_data should be a tibble")
  }
  # Rename initial_cycle_data to restaurant_data
  restaurant_data <- initial_cycle_data %>%
    mutate(
      month = month(inspection_date, label = TRUE, abbr = TRUE), # Add month feature
      weekday = wday(inspection_date, label = TRUE, abbr = TRUE) # Add weekday feature
    )

  historical_features <- cleaned_data %>%
    left_join(restaurant_data, by = "id", suffix = c("_historical", "_initial")) %>%
    filter(inspection_date_historical < inspection_date_initial) %>%
    group_by(id, inspection_date_initial) %>%
    summarize(
      num_previous_low_inspections = sum(score < 14, na.rm = TRUE),
      num_previous_med_inspections = sum(score >= 14 & score < 28, na.rm = TRUE),
      num_previous_high_inspections = sum(score >= 28, na.rm = TRUE),
      num_previous_closings = sum(action %in% c("closed", "re-closed"), na.rm = TRUE),
      .groups = "drop"
    )

  restaurant_data <- restaurant_data %>%
    left_join(historical_features, by = c("id", "inspection_date" = "inspection_date_initial")) %>%
    mutate(
      num_previous_low_inspections = replace_na(num_previous_low_inspections, 0),
      num_previous_med_inspections = replace_na(num_previous_med_inspections, 0),
      num_previous_high_inspections = replace_na(num_previous_high_inspections, 0),
      num_previous_closings = replace_na(num_previous_closings, 0)
    )

  return(restaurant_data)
}


fit_models <- function(restaurant_data){
  if(isFALSE(tibble::is_tibble(restaurant_data))){
    stop("restaurant_data should be a tibble")
  }
    restaurant_data <- restaurant_data %>%
    mutate(
      month = as.factor(month),
      weekday = as.factor(weekday),
      outcome = as.factor(outcome)
    )
```

```r
  train <- restaurant_data %>% filter(inspection_year %in% c(2017, 2018))
  test <- restaurant_data %>% filter(inspection_year == 2019)

  outcomes <- test$outcome

  logistic_model <- glm(
    outcome ~ cuisine + borough + month + weekday,
    data = train,
    family = binomial()
  )

  logistic_predictions <- predict(logistic_model, newdata = test, type = "response")

  pred_logistic <- prediction(logistic_predictions, as.numeric(outcomes) - 1)
  auc_logistic <- performance(pred_logistic, measure = "auc")@y.values[[1]]

  rf_model <- ranger(
    outcome ~ cuisine + borough + month + weekday +
      num_previous_low_inspections + num_previous_med_inspections +
      num_previous_high_inspections + num_previous_closings,
    data = train,
    num.trees = 1000,
    respect.unordered.factors = TRUE,
    probability = TRUE
  )

  rf_predictions <- predict(rf_model, data = test)$predictions[, 2]

  pred_rf <- prediction(rf_predictions, as.numeric(outcomes) - 1)
  auc_rf <- performance(pred_rf, measure = "auc")@y.values[[1]]

  return(list(
    outcomes = as.numeric(outcomes) - 1,
    logistic_predictions = logistic_predictions,
    rf_predictions = rf_predictions,
    auc_logistic = auc_logistic,
    auc_rf = auc_rf
  ))
}


cleaned_data <- clean_restaurant_data(all_data)
initial_cycle_data <- make_initial_cycle_data(cleaned_data)
restaurant_data <- make_features(initial_cycle_data, cleaned_data)
```

```
## Warning in left_join(., restaurant_data, by = "id", suffix = c("_historical", : Detected an unexpecte
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 65296 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
model_results <- fit_models(restaurant_data)

outcomes <- model_results$outcomes
logistic_predictions <- model_results$logistic_predictions
rf_predictions <- model_results$rf_predictions

precision_at_k <- function(outcomes, predictions, k) {
  ranked <- outcomes[order(-predictions)]
  top_k <- ranked[1:k]
  mean(top_k)
}

k_values <- seq(100, 500, by = 10)

logistic_precision <- sapply(k_values, function(k) precision_at_k(outcomes, logistic_predictions, k))
rf_precision <- sapply(k_values, function(k) precision_at_k(outcomes, rf_predictions, k))

precision_data <- data.frame(
  k = rep(k_values, 2),
  precision = c(logistic_precision, rf_precision),
  model = rep(c("Logistic Regression", "Random Forest"), each = length(k_values))
)

ggplot(precision_data, aes(x = k, y = precision, color = model)) +
  geom_line() +
  geom_point() +
  labs(
    title = "Precision-at-k Curves for Test Set",
    x = "Number of Restaurants (k)",
    y = "Precision at k"
  ) +
  theme_minimal() +
  theme(legend.title = element_blank()) +
  scale_color_manual(values = c("Logistic Regression" = "blue", "Random Forest" = "red"))
```



PLOT GOES HERE.

Question B3: Analyze your models. [15 points]

Answer the following three questions in **1-2** sentences each where indicated below:

1. Based on the AUCs you computed in steps B1 and B2, would you choose one model over the other?

```
I would choose the Random Forest model over logistic regression based on the higher AUC because
it shows better overall performance in distinguishing between positive and negative outcomes.
```

2. What about based on the precision plot?

```
Based on the precision plot, the Random Forest model appears preferable
since it maintains higher precision for the top-ranked inspections compared to
the Logistic Regression mode
```

3. Explain how two different models can both have similar (low) AUC scores, but different precision scores on the highest ranked inspections.

```
AUC measures the overall discriminatory ability of a model across all thresholds,
while precision focuses on the proportion of true positives among the top ranked predictions.
```

WRITTEN RESPONSES GO HERE.

Then, answer the three following questions using **3-6** sentences each where indicated below.

1. Do you think the fields in the data are accurately recorded? If some might be more prone to human bias (implicit or explicit) in how they are recorded, which might they be, and why? In particular, do you think that the outcomes we are trying to predict is objectively well-measured?

The fields in the data are likely be influenced by some degree of human bias. Some criteria like critical flag and action involve subjective assessments. Inspectors may interpret violations differently, and implicit biases. Additionally, the outcomes may not always be an objective measure of health risks, as it depends on the thoroughness and consistency of the inspection process, which can vary across inspectors and inspections.

2. Does it make sense to prioritize inspections based on whether or not they have a high score? Can you think of other criteria that one might use to prioritize inspections instead?

   If the goal is to address restaurants with the most severe violation, prioritizing inspections based on high scores makes sense. However, it may overlook emerging risks in restaurants with no prior history of high scores. Other criteria could include the frequency of prior inspections, reports of customer complaints. prioritizing based on trends in inspection scores over time could help identify restaurants with decline compliance.

3. Can you think of other data that could be collected or other oversight processes that could be put into place to make sure that you are prioritizing restaurant inspections in an "accurate and fair" manner?

Additional data, such as customer complaints, health department hotline calls, or customer reviews from online platforms like Yelp or Tripadvisor, could provide signals to prioritize inspections. Oversight processes could include periodic audits of inspectors to ensure consistency, using multiple inspectors for cross-validation in ambiguous cases to identify restaurants at risk while decreasing biases. Using external factors like local disease outbreaks related to food could also help in making prioritizzation both accurate and fair.