MATH 154 - Final Project Writeup

due on Thursday, December 12, 2024

Jazelle Saligumba, Kellie Au, Kartika Santoso

Introduction / Motivation (include a visualization maybe)

- q1, q2

Data

We began by using this dataset of Oscar nominees and winners from 1927-2024. This dataset had info on each film, its release year, the category it was nominated for, and whether or not it won.

```
library(tidyverse)
oscar_nominees <- read.csv("the_oscar_award.csv")
```

For each film in this dataset, we used the TMDB API to info on its budget, popularity, revenue, runtime, release date, vote average, vote count, genres, production countries, and spoken languages.

To set up authorization with the TMDB API, make an account and set the API key as an environment variable by creating a `.Renviron` file and listing the API key as below:

`vpw eval=False TMDB_API_KEY=YOURAPIKEY # No spaces or quotation marks!`

Then, run `readRenviron(".Renviron")` in the console.

```
library(httr)
library(jsonlite)

url <- "https://api.themoviedb.org/3/authentication"

personal_authorization <- paste0('Bearer ', Sys.getenv("TMDB_API_KEY"))
```

```
response <- VERB("GET", url, add_headers('Authorization' =  personal_authorization),
                 content_type("application/octet-stream"), accept("application/json"))
```

In order to get the movie data, we first had to make an API call to search for the movie in the TMDB databse and get its TMDB id. We search using the film name and year from the Oscar nominees dataset, then take the result with the highest popularity. We assume here movies that get nominated are much more popular than non-nominated movies.

```
# function to search for a movie by title and year
search_movie <- function(auth, movie_title, year) {
  url <- "https://api.themoviedb.org/3/search/movie"

  # parameters for the query
  queryString <- list(
    query = movie_title,
    year = as.character(year)
  )

  response <- VERB("GET", url, query = queryString,
                   add_headers('Authorization' = auth),
                   content_type("application/octet-stream"),
                   accept("application/json"))

  # check for response status
  if (http_error(response)) {
    stop("Failed to retrieve movie data. HTTP Status: ", status_code(response))
  }

  # parse json
  response_content <- content(response, "text")
  json_data <- fromJSON(response_content)

  # return the top movie hit
  if (length(json_data$results) > 0) {
    return(
      as.data.frame(json_data$results |>
        arrange(desc(popularity)) |> # choose based on popularity
        head(1)
    ))
  } else { # modify to search +/- 1
    return(NULL) # No results found
```

```
  }
}
```

```
# Example: Search for Parasite
parasite <- search_movie(personal_authorization, "Parasite", 2020)
print(parasite)
```

```
  adult                    backdrop_path  genre_ids     id original_language
1 FALSE /hiKmpZMGZsrkA3cdce8a7Dpos1j.jpg 35, 53, 18 496243                ko
  original_title
1

1 All unemployed, Ki-taek's family takes peculiar interest in the wealthy and glamorous Parks
  popularity                      poster_path release_date    title video
1     96.484 /7IiTTgloJzvGI1TAYymCfbfl3vT.jpg   2019-05-30 Parasite FALSE
  vote_average vote_count
1        8.505      18352
```

Now that we have the movie's TMDB id, we can make another API call to get its full details.

```
search_movie_extra <- function(auth, movie_id, fields) {
  parse_field <- function(json_data, field) {
    return(ifelse(!is.null(json_data[[field]]), json_data[[field]], NA))
  }

  url <- paste0("https://api.themoviedb.org/3/movie/", movie_id)

  # Make the GET request with the Authorization header and JSON accept header
  response <- GET(url,
                  add_headers('Authorization' = auth),
                  content_type("application/octet-stream"),
                  accept("application/json"))

  # Check for response status
  if (http_error(response)) {
    stop("Failed to retrieve movie data. HTTP Status: ", status_code(response))
  }

  # Parse the JSON response
  response_content <- content(response, "text")
  json_data <- fromJSON(response_content)
```

```
  result <- data.frame(matrix(ncol = length(fields), nrow = 1))
  colnames(result) <- fields

  # Safely extract all the available fields
  if (length(json_data) > 0) {
    for (field in fields) {
      result[[field]] = parse_field(json_data, field)
    }

    return(result)
  } else {
    return(data.frame())
  }
}
```

```
# Define fields to search for
input_fields <- c("adult", "backdrop_path", "belongs_to_collection", "budget", "genres", "hom
```

```
# Example: Get details for Fightclub
fightclub <- search_movie_extra(personal_authorization, 550, input_fields)
print(fightclub)
```

```
  adult                     backdrop_path belongs_to_collection   budget genres
1 FALSE /hZkgoQYus5vegHoetLkCJzb17zJ.jpg                    NA 63000000     18
                                  homepage  id   imdb_id original_language
1 http://www.foxmovies.com/movies/fight-club 550 tt0137523                en
  original_title
1     Fight Club

1 A ticking-time-bomb insomniac and a slippery soap salesman channel primal male aggression t
  popularity                      poster_path      production_companies
1     91.499 /pB8BM7pdSp6B6Ih7QZ4DrQ3PmJK.jpg 711, 508, 4700, 25, 20555
  production_countries release_date   revenue runtime spoken_languages   status
1               DE, US   1999-10-15 100853753     139          English Released
                 tagline      title video vote_average vote_count
1 Mischief. Mayhem. Soap. Fight Club FALSE        8.437      29484
```

Once we have all of the details for each of the movies, we can merge this data with our original Oscar nominees data.

Visualizations

- q4, q5

Random Forest

- q4, q5

As we saw in the above visualizations, there are definitely trends in what kind of movies get nominated for Oscars. The next question we wanted to answer was whether we could train a random forest model to be able to predict which of those nominees actually wins the Oscar.

To use our data, we first had to do some preprocessing. In particular, we wanted to predict on the information in the `genres`, `production_countries`, and `spoken_languages` columns, but these columns were in a list format, so we widened our dataset by turning those columns into multiple binary columns. We created a new binary column for each genre, and for the top 5 production countries and top 5 spoken languages. We also included an `other` column for movies that had a non-top 5 production country or spoken language.

```
# function for unnesting genre list columns
wider_by_genre <- function(df) {
  # make sure all values are vectors
  df$genre_ids <- map(df$genre_ids, ~ if (is.list(.)) unlist(.) else .)

  # now, change all the number values to be genres(char)
  new_df <- df |>
    unnest(genre_ids) |>
    distinct() |>
    mutate(genre_ids = genres_dict[as.character(genre_ids)]) |>
    filter(!is.na(genre_ids)) |>
    mutate(valid_id = 1) |> # create binary value for the pivot_wider to use
    pivot_wider(names_from = genre_ids, values_from = valid_id,
                values_fill = 0) # use names_prefix if want "genre_"

  return(new_df)
}

# more generic flattening of lists
flatten_lists <- function(df, feature, top_vector, prefix) {
  # make sure all values in the specified column are vectors
  df[[feature]] <- map(df[[feature]], ~ if (is.list(.)) unlist(.) else .)
  df$feature[is.null(df$feature)] <- NA
```

```
  # now, change all the number values
  new_df <- df |>
    unnest({{feature}}) |>  # unnest the specified column
    mutate(valid_value = 1, # create binary value for pivot_wider
           temp_feature = case_when(
             .data[[feature]] %in% top_vector ~ as.character(.data[[feature]]),
             TRUE ~ "Other")) |>
    select(-{{feature}}, -feature) |>
    distinct() |>
    pivot_wider(names_from = temp_feature, values_from = valid_value,
                names_prefix = prefix, values_fill = 0)

  return(new_df)
}
top_countries <- c("US", "GB", "FR",  "DE", "IT", "CA")
top_languages <- c("English", "French", "German", "Spanish", "Italian")
top_companies <- c(174, 4, 21, 25, 5)
```

```
# preprocess data for nominees since 1975
rf_data <- all_data |>
  # first make df wider (flatten all data)
  mutate(genre_ids = genres) |>
  select(-genres) |>
  wider_by_genre() |>
  flatten_lists("production_countries", top_countries, "country_") |>
  flatten_lists("spoken_languages", top_languages, "lang_") |>
  select(-belongs_to_collection, -production_companies) |>
  filter(year_ceremony >= 1975) # filter for last 50 years
```

```
rf_data <- rf_data |>
  mutate(won_at_least_one = as.factor(won_at_least_one))

rf_data <- rf_data |>
  ungroup() |>
  distinct(film, .keep_all = TRUE) |>
  select(-year_film, -year_ceremony, -ceremony, -category, -name, -film, -winner,
         -adult, -backdrop_path, -homepage, -id, -imdb_id, -original_title, -overview, -poste
```

Now that our data has been preprocessed, we are ready to train a model. We also perform cross-validation on the `mtry`, `trees`, and `min_n` parameters.

```r
library(tidymodels)

# Partition data into train / test sets
set.seed(47)
movie_split <- initial_split(rf_data)
movie_train <- training(movie_split)
movie_test <- testing(movie_split)

# Recipe
recipe <-
  recipe(won_at_least_one ~ . ,
         data = movie_train)

# Model
movie_rf <- rand_forest(mtry = tune(),
                        trees = tune(),
                        min_n = tune()) |>
  set_engine("ranger", importance = "permutation") |>
  set_mode("classification")

# Workflow
movie_rf_wflow <- workflow() |>
  add_model(movie_rf) |>
  add_recipe(recipe)

# CV
movie_folds <- vfold_cv(movie_train, v = 4)

# Tuning parameters
movie_grid <- grid_regular(mtry(range = c(1,40)),
                           trees(range = c(10,500)),
                           min_n(range = c(1,20)),
                           levels = 5)

# Tune (CV)
movie_rf_tune <-
  movie_rf_wflow |>
  tune_grid(resamples = movie_folds,
            grid = movie_grid)

select_best(movie_rf_tune, metric = "accuracy")
```

After performing CV, we can select the best model and finalize it.

```
movie_rf_best <- finalize_model(
  movie_rf,
  select_best(movie_rf_tune, metric = "accuracy"))

movie_rf_final <-
  workflow() |>
  add_model(movie_rf_best) |>
  add_recipe(recipe) |>
  fit(data = movie_train)

movie_rf_final
```

Now that we have finalized our model, we can predict on our test data to see how well our model does on unseen data.

```
# Test on testing data
preds <- movie_rf_final |>
  predict(new_data = movie_test) |>
  cbind(movie_test)

preds |>
  metrics(truth = won_at_least_one, estimate = .pred_class) |>
  filter(.metric == "accuracy")

conf_matrix <- conf_mat(preds, truth = won_at_least_one, estimate = .pred_class)
conf_matrix
```

We observe that our model has a 72% accuracy rate on our test data. From our confusion matrix, we see that the true positive rate is 74% and the true negative rate is 94%. Note that there are much fewer movies that win Oscars than are nominated, so as seen in the confusion matrix, there are much more "False" observations in our data than "True" observations, causing the skew towards "False".

Ethical Considerations

- q6

Conclusion

- q7

References

- cite packages, kaggle dataset (need to cite source of the kaggle data)

1. Why should anyone care about this?

Evaluating the media industry and the quality/replicability/authenticity of the media we consume

recent years in hollywood, (ex. writers' strike led to more reality tv, covid), is there more oscar baiting than usual, - putting money into movies (expensive sound/visual effects, cgi, alist actors, etc), want to get it back… how do you get that money back? - is there a trend in what kind of movies people are making nowadays to get the money back (ex. more period pieces) - when are the "best" movies made? and what are the "best" movies - what movies are better - how much money do oscar nominees/wins make back? (in comparison to the budget) - how much is a win worth compared to a nomination? - classification model of who wins and who doesn't - predicting 2025 oscar winners / nominees

overarching questions: who wins the oscars? (oscar formula and international movies), what is an oscar worth? (motivation), how true is our current idea of the oscar formula? (period pieces, big budget, released late in the year)

- quantifying the Oscar formula
- seeing trends in genres of winners/nominees since 1940s
- sentiment analysis on descriptions to see if more positive/negative themes are more likely to win
- genres
- budget / revenue
- release date (is it late in the year)
- runtime
- vote average / vote count
- popularity
- blockbuster vs oscar
- run a random forest on who will win oscars (by category?)
  - ∗ sentiment analysis on overviews?
  - preprocessing: split release date into year/month
  - how many years of data do we wanna use and which categories?
  - shiny app to make an interface where ppl can "predict" the winners(?)
  - slider graphs

- insights on budget/revenue/popularity/voteAve–> motivates why people wanna win an oscar

2. What is the project about? Do not assume that your readers have any domain knowledge! The burden of explanation as to what you are talking about is on you! For example, if your project involves phylogenetic trees, do not assume that your audience has anything other than a basic, lay understanding of genetics.

-

2. Where did your data come from? What kind of data was it? Is there a link to the data or some other way for the reader to follow up on your work? Be sure to cite / reference where your data came from!!

-

3. What are your findings? What kind of statistical computations (if any) have you done to support those conclusions? Again, while the R code will show how the calculation was performed, it is up to you to interpret, in English sentences, the results of these calculations. Do not forget about units, axis labels, etc.

- Oscar bait is lowkey real!
  - release day is later in the year
  - genres to be released are often genre, historical pieces

4. What are the limitations of your work? Be clear so that others do not misinterpret your findings.

- Our work is limited by the fact that we do not account for movies that were not recognized as an Oscar nominee
  - hence, we don't have a third dataset/column that tells us qualities of movies that are not nominated.
  - so our RF, that is predicting movies that have yet to be nominated is heavily limited.
- API has some corrupted values (e.g. budget may be incorrect and genre is interpreted) since it is user informed
-

5. To what population do your results apply? Do they generalize? Could your work be extended with more data or computational power or time to analyze? How could your study be improved? Suggesting plausible extensions don't weaken your work – they strengthen it by connecting it to future work.

- Population = Movies that are nominated for an Oscar Award, no as of now they do not generalize

- With more data, we'd be able to understand more aspects of movies and come up with better model
  - ability to investigate more aspects of the movie (e.g. data on the actors/directors/producers involved in the film)
  - gathering data on movies that were NOT nominated would also be really helpful
-

Our study can be improved:

6. What ethical considerations that came about during the project. How were you or weren't you able to address those issues?

- questioning the authenticity of the media we consume and the media we consider to be "so good"
- what does the academy think is a good movie?
  - the academy gets to dictate what the masses watch!
- what movie gets to be guilded?

7. What did you learn beyond what we've covered in class?

- how to work with live data from API, API calls
- how to create a fully publishable shiny app that works dynamically with data and API
-