Capstone Proposal

Noah Anderson

2023-10-21

Introduction

Divvy, a bike share program owned by the City of Chicago in partnership with Lyft, has observed a significant rise in usage since its launch in 2013. A notable feature of Divvy bikes is that most of them are electric, necessitating a return to docking stations after each trip. A common challenge faced by such bike-sharing systems is the potential imbalance at stations, which can be seen when there are more departing trips than arriving ones at specific locations. To mitigate this, Divvy has been actively involved in the manual re-balancing of bikes through its workforce, ensuring seamless operation.

To further enhance this operational efficiency, the current project leverages the power of predictive analytics. Drawing on predictors such as weather conditions, population density near the stations, and historical trip data, the objective is to precisely predict the number of "to" and "from" trips at each station. The following sections delve into the data, detailing its preparation and offering insights into potential techniques for its analysis and modeling (City of Chicago, 2023).

In recent literature, the book "Public Policy Analytics: Code & Context for Data Science in Government" by Ken Steif (2021) stands out. Steif employed R to create a spatial time series linear model that predicts Uber ridership for each census tract, using lag values and weather data as predictors. Drawing inspiration from this work, this project modifies and adapts Steif's model to the specifics of Divvy, with an emphasis on higher geographic granularity – at the station level. Furthermore, this research intends to experiment with advanced modeling techniques such as random forests or neural networks to enhance predictive accuracy.

It's worth noting that predicting bike share usage patterns is an evolving research domain, with multiple methodologies proposed by various scholars. For instance, Fishman et al., representing the Data Science for Global Good foundation, proposed a model using Poisson regression to anticipate hourly bike station demands for Washington D.C.'s bike share system. However, when applied to Chicago's Divvy system, the model by DSGG faced limitations due to a purported lack of sufficient historical data. Addressing this, the present study aims to incorporate lag terms, offering a solution to overcome the constraint of limited historical data. The final model will predict the total number of trips to given Divvy stations daily.

Methodology

Data Acquisition and Integration:

- **Demographic Data:** For the demographic aspect of our study, we will extract population data at the tract level using the `tidycensus` package in R. This tool is engineered to fetch census data seamlessly while ensuring compatibility with tidyverse functions (U.S. Census Bureau, 2010).
- **Weather Data:** Weather data, covering the entire Chicago expanse, will be sourced through the `riem` package in R, specifically from the station "ORD" (O'Hare International Airport) (Riem API, 2023).

- **Proximity to Train Stops:** Grasping alternate transportation modes can be instrumental. The distance to the nearest train stop, sourced from the RSocrata database, will be factored in as a predictor to ascertain its impact on the total riders per station (City of Chicago, 2023).
- **Lag Terms:** A significant component of our model involves the inclusion of lag terms for each station. This pertains to the total number of trips to and from the station on prior days.

Modeling Approach:

1. **Linear Model:** Serving as our foundational comparison metric, this basic model will guide us in evaluating the performance of more intricate models.
2. **Random Forest:** Random forest's selection is influenced by its competence in addressing confounding variables. For instance, the proximity to a transit stop might be marginally impactful on a day with favorable weather. However, during inclement conditions, riders might favor the sheltered comfort of trains. This model is adept at discerning such nuanced interactions.

In conclusion, this project merges various data streams to develop a model that predicts bike share patterns in Chicago. By employing a systematic methodological approach, we aspire to propose enhancements that can elevate the Divvy bike-sharing system.

References

1. **Divvy Bike Share Data**
    - City of Chicago. (2023). Divvy bike share data. RSocrata Database. Retrieved October 29, 2023, from https://data.cityofchicago.org/resource/fg6s-gzvg.json?
2. **Chicago Rail Stops Data**
    - City of Chicago. (2023). Chicago rail stops. RSocrata Database. Retrieved October 29, 2023, from https://data.cityofchicago.org/resource/8pix-ypme.json
3. **Cook County Population Data**
    - U.S. Census Bureau. (2010). Decennial census data for Cook County, Illinois. Retrieved October 29, 2023, via the `tidycensus` package in R.
4. **Weather Data**
    - Riem API. (2023). Weather data for station "ORD" (O'Hare International Airport). Retrieved October 29, 2023, from Riem API database.

Appendix

Setting Paramters and Loading Libraries

```r
library(RSocrata)
library(tidycensus)
library(tidyverse)
library(sf)
library(riem)
```

```r
# Define parameters
start_date <- "2019-01-01"
end_date <- "2020-01-01"
order_by <- "start_time DESC"

# Set NULL for no limit
limit <- 10000
```

Importing Data

```r
# Base URL for fetching Divvy bike share data from the City of Chicago's database
divvy_url <- "https://data.cityofchicago.org/resource/fg6s-gzvg.json?"

# Construct the full URL for data retrieval
# Checks if 'limit' is null, and constructs the URL accordingly
divvy_url_full <- ifelse( is.null(limit),
                          # Construct URL without limit parameter
                          paste0(divvy_url,"$where=start_time >= '", start_date,
                                 "' AND start_time < '", end_date,
                                 "'&$order=", order_by),
                          # Construct URL with limit parameter
                          paste0(divvy_url,"$where=start_time >= '", start_date,
                                 "' AND start_time < '", end_date,
                                 "'&$order=", order_by,
                                 "&$limit=", limit)
)

# Fetch Divvy data using the constructed URL and store in a dataframe
divvy_df <- read.socrata(divvy_url_full)

# Base URL for fetching rail stops data from the City of Chicago's database
rail_stops_url <- "https://data.cityofchicago.org/resource/8pix-ypme.json"

# Fetch rail stops data using the provided URL
rail_stops <- read.socrata(rail_stops_url)

# Retrieve Cook County's 2010 decennial population data at the tract level
# Using the 'tidycensus' package
cook_population <- get_decennial(geography = "tract",
                                 variables = "P001001",
                                 year = 2010,
                                 state = "IL",
                                 county = "Cook County",
                                 geometry = TRUE)

# Retrieve weather data for ORD station between given start and end dates
# Filters specific columns (valid, tmpf, p01i, sknt) from the fetched data
weather.Data <-
  riem_measures(station = "ORD", date_start = start_date, date_end = end_date) %>%
  select(valid, tmpf, p01i, sknt)
```

Cleaning Data

```r
# Convert 'start_time' and 'stop_time' columns to Date format
divvy_df$date_from <- as.Date(divvy_df$start_time)
divvy_df$date_to <- as.Date(divvy_df$stop_time)

# Group the data by 'from' station and date, then calculate the daily count of 'from' trips
from_trips_daily <- divvy_df %>%
  group_by(date_from, from_station_id, from_latitude, from_longitude) %>%
  summarize(daily_from_trips = n(), .groups = "drop")

# Group the data by 'to' station and date, then calculate the daily count of 'to' trips
to_trips_daily <- divvy_df %>%
  group_by(date_to, to_station_id, to_latitude, to_longitude) %>%
  summarize(daily_to_trips = n(), .groups = "drop")

# Merge 'from' and 'to' trips data by matching station IDs and dates
daily_trips <- left_join(from_trips_daily, to_trips_daily,
                         by = c("date_from" = "date_to",
                                "from_station_id" = "to_station_id")) %>%
  transmute(
    date = date_from,
    station_id = from_station_id,
    latitude = from_latitude,
    longitude = from_longitude,
    daily_from_trips = replace_na(daily_from_trips, 0),  # Replace NA values with 0
    daily_to_trips = replace_na(daily_to_trips, 0),
    net_trips = daily_to_trips - daily_from_trips,       # Calculate net daily trips
    day = day(date),
    month = month(date)
  ) %>%
  st_as_sf(coords = c("longitude", "latitude"), crs = 4269)  # Convert to spatial data frame

# Convert rail stops data to spatial data frame using 'location' columns
rail_stops_sf <- rail_stops %>%
  st_as_sf(coords = c("location.longitude", "location.latitude"),
           crs = 4269)

# Calculate the minimum distance from each Divvy station to the nearest rail stop
daily_trips$distance_to_rail <- apply(st_distance(daily_trips, rail_stops_sf), 1, min)

# Spatially join Divvy stations data with population data by checking if points are within census tract
daily_trips_pop <-  st_join(daily_trips, cook_population, join = st_within) %>%
  select(-c(NAME, variable, GEOID)) %>%
  rename(Population = value)  # Rename population column for clarity

# Summarize the weather data to get daily statistics
weather.Panel <-
  weather.Data %>%
  mutate(date = as.Date(valid)) %>%
  group_by(date) %>%
  summarize(HighTemp = max(tmpf, na.rm = T),             # Maximum temperature for the day
            LowTemp = min(tmpf, na.rm = T),              # Minimum temperature for the day
            Percipitation = sum(p01i, na.rm = T),        # Total precipitation for the day
            Wind_Speed = max(sknt, na.rm = T))           # Maximum wind speed for the day

# Merge Divvy trips data (with population) and the summarized weather data
daily_trips_full <- left_join(daily_trips_pop, weather.Panel)
```