Capstone Proposal

Noah Anderson

2023-10-21

Introduction

Divvy, a bike share program owned by the City of Chicago and in collaboration with Lyft, has seen a marked increase in usage since its inception in 2013. Notably, the majority of Divvy bikes are electric, necessitating a return to docking stations after each use. A prevalent issue with such bike-sharing systems is the potential imbalance at stations, often characterized by a higher number of departing trips than arriving ones at specific stations. To address this, Divvy invests in manual re-balancing of bikes by employing staff, ensuring that the system operates smoothly.

To enhance this operational efficiency, this project aims to employ predictive analytics. Leveraging predictors like weather conditions, the population density of the station's vicinity, and historical trip data, the goal is to accurately forecast the total number of "to" and "from" trips for each station. Subsequent sections will provide insights into the sample data, detailing its cleansing process, and will also shed light on prospective methods for data interpretation and modeling.

The Data

The City of Chicago maintains a comprehensive database that is readily accessible through the RSocrata package, an initiative in which the City itself played a significant role during its early development. For our study's demographic dimension, we'll harness population metrics derived from census data. This information is seamlessly integrated using the tidycensus package in R, which is specially designed to fetch census data while ensuring compatibility with tidyverse functions. To further enhance our predictive capabilities, weather data encompassing the entire Chicago region will be sourced via the riem package.

Loading Libraries

```
library(RSocrata)
library(tidycensus)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## 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
```

```
library(sf)
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(riem)
```

Divvy Data

For our analysis, we utilize a tailored query to extract the most recent 10,000 rows from the 2019 Divvy bike trip dataset. Although the dataset encompasses 20 columns, our focus will primarily be on eight of them, given their relevance to our project objectives. The pivotal columns in question include start_time, stop_time, from_station_id, to_station_id, and the associated coordinates.

```
# Create URL that queries the last 10,000 rows of the 2019 Divvy data
divvy_url <- "https://data.cityofchicago.org/resource/fg6s-gzvg.json?$where=start_time >= '2019-01-01' A

# Read in divvy data
divvy_df <- read.socrata(divvy_url)
```

Upon retrieval, the Divvy dataset undergoes a transformation to produce a consolidated data frame. This frame captures each station's daily 'from' and 'to' trips. It's essential to highlight that the coordinates undergo modification to adopt the sfc_POINT class, a designation within the sf package. This particular class empowers our dataset with capabilities for enhanced spatial visualizations and seamless data joining.

```
# Convert to and from times into date format
divvy_df$date_from <- as.Date(divvy_df$start_time)
divvy_df$date_to <- as.Date(divvy_df$stop_time)

# Compute daily trips from each station, including coordinates
from_trips_daily <- divvy_df %>%
  group_by(date_from, from_station_id, from_latitude, from_longitude) %>%
  summarize(daily_from_trips = n(), .groups = "drop")

# Compute daily trips to each station, including coordinates
to_trips_daily <- divvy_df %>%
  group_by(date_to, to_station_id, to_latitude, to_longitude) %>%
  summarize(daily_to_trips = n(), .groups = "drop")

# Merge, compute net trips, and streamline columns
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),
```

```
    daily_to_trips = replace_na(daily_to_trips, 0),
    net_trips = daily_from_trips - daily_to_trips
  ) %>%
  st_as_sf(coords = c("longitude", "latitude"), crs = 4269)
```

Census Data

For our demographic insights, we leverage the tidycensus package to extract census data. We primarily focus on the variable P001001, which denotes the total population. It's imperative to point out that the data is requisitioned at the tract level, ensuring a granular understanding of population distribution.

```
# Import total population data for Cook County IL from 2010 Census
cook_population <- get_decennial(geography = "tract",
                                 variables = "P001001",
                                 year = 2010,
                                 state = "IL",
                                 county = "Cook County",
                                 geometry = TRUE)
```

```
## Getting data from the 2010 decennial Census

## Downloading feature geometry from the Census website.  To cache shapefiles for use in future sessions

## Using Census Summary File 1

##   |                                                                              |
```

To achieve a confluence of our daily trips data with demographic data, we deploy the sf package. This methodology ensures that if a station's location aligns with a particular census tract, it inherits the population value attributed to that tract, providing an integrated dataset that amalgamates trip dynamics with population density.

```
daily_trips_pop <-  st_join(daily_trips, cook_population, join = st_within) %>%
  select(-c(NAME, variable, GEOID)) %>%
  rename(Population = value)
```

Weather

Leveraging the riem package, we source weather data specifically from O'Hare airport, which provides updates every five minutes. A limitation we must acknowledge is the absence of a more detailed weather dataset that encompasses the entirety of the City of Chicago. Given the city's diverse microclimates, this granularity would have been advantageous. Nevertheless, our current approach ensures that the daily average weather metrics for the city play a significant role in reflecting the overall magnitude of trips.

Subsequent to data retrieval, we organize the weather metrics, capturing the daily high and low temperatures in Fahrenheit, cumulative precipitation, and the day's peak wind speed.

```r
# Import O'Hare weather data for the daily_trips time range
weather.Data <-
  riem_measures(station = "ORD", date_start = as.character(min(daily_trips$date)), date_end = as.charac
  select(valid, tmpf, p01i, sknt)

# Aggeregate data to give daily summaries
weather.Panel <-
  weather.Data %>%
  mutate(date = as.Date(valid) ) %>%
  group_by(date) %>%
  summarize(HighTemp = max(tmpf, na.rm = T),
              LowTemp = min(tmpf, na.rm = T),
              Percipitation = sum(p01i, na.rm = T),
              Wind_Speed = max(sknt, na.rm = T))
```

Final Data Set

In culmination, our finalized dataset amalgamates pertinent attributes across our diverse data sources, yielding a structured and cohesive data frame. This consolidated frame encapsulates the population statistics based on the census tract corresponding to each station's location, daily weather data, geographical data indicating station locations, and metrics detailing the volume of trips originating from and destined for each station, as well as the net movement of trips.

It's worth highlighting that, at this juncture, the dataset doesn't incorporate 'lags' – data points that capture trip metrics from preceding days. Such lags can act as robust predictive indicators when developing a time series model.

```r
# Join daily_trips with weather.Panel
daily_trips_full <- left_join(daily_trips_pop, weather.Panel)
```

```
## Joining with `by = join_by(date)`
```

```r
head(daily_trips_full)
```

```
## Simple feature collection with 6 features and 10 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -87.67844 ymin: 41.87184 xmax: -87.62872 ymax: 41.90327
## Geodetic CRS:  NAD83
## # A tibble: 6 x 11
##   date       station_id daily_from_trips daily_to_trips net_trips
##   <date>     <chr>                 <int>          <int>     <int>
## 1 2019-12-28 106                       3              0         3
## 2 2019-12-28 107                       2              0         2
## 3 2019-12-28 108                       1              0         1
## 4 2019-12-28 110                       1              0         1
## 5 2019-12-28 111                       1              0         1
## 6 2019-12-28 130                       1              1         0
## # i 6 more variables: geometry <POINT [°]>, Population <dbl>, HighTemp <dbl>,
## #   LowTemp <dbl>, Percipitation <dbl>, Wind_Speed <dbl>
```

Next Steps

Predictors and Outcome Variables

A pivotal decision in this analysis pertains to the selection of the outcome variable. The options at our disposal include daily_from_trips, daily_to_trips, and net_trips. While an exhaustive approach would entail developing distinct models for both daily_from_trips and daily_to_trips—from which net_trips could subsequently be deduced—it's imperative to consider the computational implications.

Given the objective of aiding in the bike re-balancing process, my inclination is to prioritize daily_to_trips as the chief outcome variable for this phase of the project. This serves as a representative demonstration of a system that, with further augmentation, could encompass daily_from_trips as well. An open question remains whether directly forecasting net_trips is methodologically sound or if it's more prudent to derive it from individual to and from trip predictions.

The predictor variables under consideration encapsulate a station's corresponding census tract population, daily meteorological conditions, and preceding trip counts, referred to as 'lags'. The exact modality of integrating lags remains an area of exploration, but fortunately, a rich tapestry of academic literature on this subject can provide guidance. Furthermore, in a bid to enhance the predictive capability of the model, I am contemplating the integration of the distance to transit stops as an additional predictor, a metric readily accessible through RSocrata.

Size of Data

In the quest to harness the Divvy data for meaningful insights, several challenges arise. Foremost among them is the sheer volume of the dataset. While the demonstration in this document is based on a subset of 10,000 data frames extracted from the Divvy logs, it's crucial to note that this represents only a snapshot—specifically, two days in 2019. When one considers the entire gamut of Divvy data, spanning from 2013 to 2020, we're confronted with a staggering 21.2 million rows.

This considerable dataset, even when pared down by grouping by date and station, poses a formidable computational challenge, especially given the limitations of my dual-core processor. While transitioning to cloud-based computing is an avenue worth exploring, it isn't a panacea. Delving into data that sprawls across seven years brings to the fore a plethora of concerns and conundrums.

Key among the considerations is the delineation of the dataset. If the ambition is to unearth patterns that reflect seasonality, the scope of the data becomes crucial. Moreover, choices around which chronological slices to spotlight and which geographic regions to prioritize come into play. These strategic decisions, vital for shaping the trajectory and depth of the analysis, will be contemplated and crystallized in the forthcoming draft.

Potential Models

In the pursuit of understanding and predicting patterns within the Divvy data, a methodological, phased approach to modeling will be adopted. This strategy, moving from simplicity to intricacy, allows for a robust exploration of the dataset while being mindful of computational constraints and interpretability.

1. **Linear Model**: This will serve as the foundational model, offering a baseline understanding. Given its straightforward nature, a linear model can provide quick insights, especially on direct relationships between predictors and the response variable.
2. **Random Forest**: Should the data exhibit signs of confounding variables, or interdependencies between predictors, a random forest might be more adept at capturing these nuances. For instance, areas with higher populations might experience more pronounced changes in trips during adverse weather,

especially if these regions are proximate to transit hubs. Such interactions, where the effect of one predictor depends on the level of another, are where random forests often shine.

3. **Neural Network**: Venturing into the realm of deep learning, a neural network represents the zenith of complexity in this modeling hierarchy. While computationally intensive and demanding more data, neural networks have the potential to detect intricate patterns, especially from non-linear relationships, which might elude the more traditional models. Given the multifaceted nature of the Divvy data, a neural network might unveil hidden relationships and trends, offering a more holistic picture.

As this journey of modeling unfolds, each step will not only offer insights into the data but also inform decisions about the subsequent stages. The goal is to strike a balance between model complexity, interpretability, and predictive accuracy, ensuring that the insights drawn can be effectively harnessed to enhance the efficiency of the Divvy bike-sharing program.