Analysis

December 22, 2023

1 Urban Transit Data Analytics and Strategy at San Diego

1.1 Question(s) propose to address:

1.1.1 What do you expect to find?

Our project originally attempted to identify traffic patterns for UCSD students, but due to the lack of live traffic data around the university, we decided to focus on areas that can be accessed by public transportation.

Shopping without a car can be a challenge in San Diego, especially for UCSD students. Our project aims to identify the most frequently used public transportation routes and nearby stops to determine if they can be accessed by taking public transportation. By examining the relationship between public transportation and nearby businesses, the project may uncover the proportion of shops that are accessible by public transportation in San Diego.

This project is essential because it provides an opportunity for city officials who are interested in public transportation routing to gain insight into the areas with infrequent public transportation. Hopefully, the project can improve the overall quality of life for San Diego residents who do not have access to a car.

1.1.2 Why is this question important?

The significance of traffic patterns and bus routes in San Diego can be attributed to several reasons:

- Access to transportation is critical, as a significant portion of San Diego's population relies on public transportation to commute to work, school, and other activities. Identifying the most frequently used bus routes and their nearby stops can inform people of the businesses they can reliably access, thus improving their access to transportation.
- Promoting the use of public transportation instead of personal vehicles can contribute to urban development. By reducing traffic, cities can reap several benefits, including improved air quality and reduced emissions.
- The presence of nearby shopping options at bus stops can also drive business growth. Business owners can make informed decisions about promoting their hours of operation by understanding where buses stop and their frequency.

1.1.3 What is the business case for solving the question(s) you posed above?

The business case for resolving the question of traffic patterns and bus routes in San Diego is essential for several reasons:

- Economic Development: Identifying the most frequently used bus routes and nearby stops can help businesses strategically locate themselves to attract more foot traffic and potentially increase revenue.
- Increased Public Transportation Ridership: Understanding the needs and preferences of public transportation users can assist city officials in increasing the frequency and routes to areas with less coverage, thereby enhancing ridership.
- Environmental Responsibility: By promoting the use of public transportation, the city can reduce car ownership, leading to a less pollutive environment, and demonstrate environmental responsibility.

1.1.4 Who is the expected audience and how would they benefit from what you find?

The expected audience for the project investigating the traffic patterns in San Diego is broad and includes a range of stakeholders who could benefit from the findings, such as:

- City Officials: City officials can benefit from the project by using them to plan future public transportation routes.
- Business Owners: Business owners, can benefit from the project by gaining insights into the most frequent stop and its frequency.
- Residents: Residents can benefit from the project by having a better understanding of the most frequent route and business it can access.

1.2 Background and literature:

San Diego is a city located in Southern California. With over 1.4 million residents, San Diego is the second-largest city in California. Visit https://www.california-demographics.com/cities_by_population for more information.

San Diego has a comprehensive public transportation system that serves the needs of residents and visitors. The San Diego Metropolitan Transit System (MTS) is responsible for operating and managing the public transportation system throughout the city. The MTS offers a wide range of services, including buses, trolleys, and commuter trains, with over 88 million annual passenger trips or over 300,000 trips each weekday. Visit https://www.sdmts.com/about/about-mts for more information.

In addition to the MTS, the North County Transit District (NCTD) manages the public transportation system in the northern part of San Diego. The NCTD offers a range of services, including buses, COASTER commuter trains, and SPRINTER light rail with approximately 1.4 million passengers annually. Visit https://gonctd.com/wp-content/uploads/2021/12/FY2021-NCTD-ACFR.pdf for more information.

The MTS and NCTD public transportation systems provide a convenient and cost-effective way for residents and visitors to access San Diego's many attractions, including the world-famous San Diego Zoo, SeaWorld San Diego, and Balboa Park. Furthermore, public transportation serves as an essential mode of transportation for many San Diego residents who rely on it to commute to work, school, and other activities. Visit https://www.sdmts.com/getting-around/popular-destinations for more information.

To ensure that the public transportation system continues to meet the needs of the community, the MTS and NCTD regularly assess and update their services and routes. This includes identifying the most frequently used routes and nearby stops to make informed decisions about route planning and public transportation funding. They also heavily invest in many feature projects such as onboard Wi-Fi, and clean air vehicles to make the ride more enjoyable. Visit https://www.sdmts.com/inside-mts/current-projects for more information.

1.3 Python libraries or ArcGIS modules you plan to use and why:

For this project, we plan to leverage several powerful Python libraries and ArcGIS modules to conduct our analysis, including:

- ArcGIS: We will use ArcGIS, a geospatial analysis, data visualization, and data management tool, to perform geo enrichment. Geoenrichment involves adding demographic and other data to geographic datasets, allowing us to analyze the walkable distance around the most frequented bus stops.
- GeoPandas: We will use GeoPandas, a Python library that provides tools for geospatial data analysis and visualization, to conduct much of the data processing. This will include tasks such as replacing missing values, dropping unnecessary columns, replacing values as needed, and conducting exploratory data analysis.
- Scikit-learn: We will use Scikit-learn, a Python library that provides fundamental tools for supervised machine learning, such as regression and classification, to conduct a basic classification of nearby stops and their associated businesses.
- XGBoost: We will use XGBoost for gradient boosting in our classification model.

By utilizing these tools, we aim to analyze traffic patterns and bus routes in San Diego to gain insights into the most frequently used routes and nearby stops. This information will be critical for identifying areas with inadequate public transportation coverage, promoting the use of public transportation over personal vehicles, and supporting economic development and environmental responsibility in the city.

1.4 Data sources:

To effectively analyze traffic patterns and bus routes in San Diego, we will be utilizing two primary data sources that will serve as the foundation for our analysis.

The first data source is a zip file available at http://www.sdmts.com/google_transit_files/google_transit.zip. This dataset used Google Maps to route a bus route for MTS and NCTD. The zip file contains all the general transit feed specifications (GTFS). GTFS is a data specification that allows public transit agencies to publish their transit data in a format that can be consumed by a wide variety of software applications. It contains all the information about bus routes, frequency, bus stops, and their locations in a geo json formate. With this data, we will be able to identify the most frequently used bus routes in San Diego and the locations of nearby bus stops.

The second data source is a dataset available at https://mygeodata.cloud/data/download/osm/shopping-centers-and-department-stores/united-states-of-america–california/san-diego-county/san-diego. MyGeodata Cloud is a gis data storage that offers users to access to a wide range of the spatial dataset. This dataset provides information on all shopping centers and department stores in San

Diego County. We will use this data to understand the availability of shopping options near bus stops. The data contains the shop name, location, and types of shops in a geo json formate.

1.4.1 Data and module import:

[1]:
```
# should you need to
#!pip install xgboost
```

[2]:
```
# ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
# modules to import
import arcgis
from arcgis.gis import GIS
from arcgis import geometry
from arcgis.features import GeoAccessor, GeoSeriesAccessor,␣
 ↪FeatureLayerCollection
import geopandas as gpd
import pandas as pd
import numpy as np
from arcgis import features
from arcgis.gis import GIS
from arcgis.geoenrichment import *
from arcgis.geocoding import geocode
from arcgis.geometry import lengths, areas_and_lengths, project
from arcgis.geometry import Point, Polyline, Polygon, Geometry
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
import xgboost as xgb
```

[4]:
```
# login
gis = GIS(username='dsc170wi23_5', password='PA8d%eLmuD*6WV')
```

1.4.2 Data cleaning and visualization:

Data visualizatoin: read the GeoJSON file (shop)

[5]:
```
shop_gdf = gpd.read_file('./shop_general_point.geojson')
shop_gdf.head()
```

[5]:

	shop	name	addr:housenumber	addr:street	\
0	department_store	Bloomingdale's	None	None	
1	department_store	Macy's	None	None	
2	department_store	Marshalls	None	None	

```
3  department_store            Marshalls             None        None
4  department_store  Mitsuwa Marketplace             None        None

  addr:postcode addr:city phone website addr:country opening_hours  …  \
0          None      None  None    None         None          None  …
1          None      None  None    None         None          None  …
2          None      None  None    None         None          None  …
3          None      None  None    None         None          None  …
4          None      None  None    None         None          None  …

  brand:wikipedia addr:state operator wheelchair amenity diet:gluten_free  \
0            None       None     None       None    None             None
1            None       None     None       None    None             None
2    en:Marshalls       None     None       None    None             None
3    en:Marshalls       None     None       None    None             None
4            None       None     None       None    None             None

  diet:vegan organic ref:walmart                     geometry
0       None    None        None  POINT (-117.16454 32.76860)
1       None    None        None  POINT (-117.14685 32.76888)
2       None    None        None  POINT (-117.23129 32.86501)
3       None    None        None  POINT (-117.14710 32.91451)
4       None    None        None  POINT (-117.15028 32.81896)

[5 rows x 22 columns]
```

read the GeoJSON file (MTS/NCTD route)

[6]:
```
transit_route_gdf = gpd.read_file("./data/transit_routes_datasd.geojson")
transit_route_gdf.head()
```

[6]:
```
   objectid route_id   shape_id     rteshpname short_name  \
0         1      105   105_0_98   105_0_98_105        105
1         2      105   105_0_99   105_0_99_105        105
2         3      105  105_1_100  105_1_100_105        105
3         4      105  105_1_101  105_1_101_105        105
4         5      105   105_1_97   105_1_97_105        105

                   long_name  route_type rte_type_t agency_id  \
0  Old Town - University City           3        Bus       MTS
1  Old Town - University City           3        Bus       MTS
2  Old Town - University City           3        Bus       MTS
3  Old Town - University City           3        Bus       MTS
4  Old Town - University City           3        Bus       MTS

                                          route_url hex_color  \
0  https://www.sdmts.com/schedules-real-time?frag…    FFFFFF
```

```
1  https://www.sdmts.com/schedules-real-time?frag…    FFFFFF
2  https://www.sdmts.com/schedules-real-time?frag…    FFFFFF
3  https://www.sdmts.com/schedules-real-time?frag…    FFFFFF
4  https://www.sdmts.com/schedules-real-time?frag…    FFFFFF

                                              geometry
0  LINESTRING (-117.19921 32.75467, -117.19913 32…
1  LINESTRING (-117.19921 32.75467, -117.19913 32…
2  LINESTRING (-117.20610 32.82880, -117.20609 32…
3  LINESTRING (-117.20097 32.81123, -117.20097 32…
4  LINESTRING (-117.21382 32.86910, -117.21381 32…
```

show the transit route dictionary

[7]:
```
transit_route_gdf_dict = pd.read_csv('./data/transit_routes_dictionary_datasd.
  ↪csv')
transit_route_gdf_dict
```

[7]:
```
        field field_type                                          description  \
0    objectid    Integar                                                  NaN
1    route_id     String  The route_id field contains an ID that uniquel…
2    shape_id     String                                  Shape identifier
3  rteshpname     String                                  Route shape name
4  short_name     String  The short name of a route. This will often be …
5   long_name     String  The full name of a route. This name is general…
6  route_type    Integar        The type of transportation used on a route
7  rte_type_t     String                        Text value for route type
8   route_url     String  The URL of a web page about that particular route
9   hex_color     String                                   Color for route

                                  possible_values
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN
5                                             NaN
6  0 - Tram, Streetcar, Light rail. Any light rai…
7                                             NaN
8                                             NaN
9                                             NaN
```

read the GeoJSON file (MTS/NCTD stop)

[8]:
```
transit_stop_gdf = gpd.read_file("./data/transit_stops_datasd.geojson")
transit_stop_gdf.head()
```

[8]:
```
   objectid   stop_uid stop_id  stop_code                       stop_name  \
0         1  MTS_10001   10001      10001     Cabrillo National Monument
1         2  MTS_10003   10003      10003           Pearl St & Draper Av
2         3  MTS_10004   10004      10004              Pearl St & Fay Av
3         4  MTS_10006   10006      10006  Torrey Pines Rd & Exchange Pl
4         5  MTS_10007   10007      10007  Torrey Pines Rd & Princess St

    stop_lat    stop_lon stop_agncy  wheelchair       intersec stop_place  \
0  32.674458 -117.240426        MTS           1  M-Special_Loc     cabmon
1  32.839578 -117.276187        MTS           1          N-E/B       None
2  32.840128 -117.273810        MTS           1          N-E/B       None
3  32.845847 -117.268673        MTS           1          N-N/B       None
4  32.848826 -117.262585        MTS           0          N-E/B       None

  parent_sta        lat         lng                      geometry
0       None  32.674453 -117.240414  POINT (-117.24041 32.67445)
1       None  32.839574 -117.276175  POINT (-117.27617 32.83957)
2       None  32.840124 -117.273798  POINT (-117.27380 32.84012)
3       None  32.845842 -117.268661  POINT (-117.26866 32.84584)
4       None  32.848821 -117.262574  POINT (-117.26257 32.84882)
```

read each MTS trip detail

[9]:
```
mts_trips = pd.read_csv('./data/MTS/trips.txt')
mts_trips.head()
```

[9]:
```
  route_id       service_id   trip_id trip_headsign  direction_id  \
0      AIR  76360-1111111-0  16282358           NaN             1
1      AIR  76360-1111111-0  16282359           NaN             1
2      AIR  76360-1111111-0  16282360           NaN             1
3      AIR  76360-1111111-0  16282361           NaN             1
4      AIR  76360-1111111-0  16282362           NaN             1

     direction_name block_id   shape_id  wheelchair_accessible  \
0  Counterclockwise        1  AIR_9_245                      1
1  Counterclockwise        2  AIR_9_245                      1
2  Counterclockwise        3  AIR_9_245                      1
3  Counterclockwise        1  AIR_9_245                      1
4  Counterclockwise        2  AIR_9_245                      1

   trip_bikes_allowed trip_headsign_short
0                   2                NaN
1                   2                NaN
2                   2                NaN
3                   2                NaN
4                   2                NaN
```

read each NCTD trip detail

[10]:
```
nctd_trips = pd.read_csv('./data/NCTD/trips.txt')
nctd_trips.head()
```

[10]:

	route_id	service_id	trip_id	trip_headsign	direction_id	\
0	399	NC2210-399-Weekday-03	16790593	Escondido	0	
1	399	NC2210-399-Weekday-03	16790594	Escondido	0	
2	399	NC2210-399-Weekday-03	16790595	Escondido	0	
3	399	NC2210-399-Weekday-03	16790596	Escondido	0	
4	399	NC2210-399-Weekday-03	16790597	Escondido	0	

	direction_name	block_id	shape_id	wheelchair_accessible	\
0	East	E06	3990004	1	
1	East	E08	3990004	1	
2	East	E10	3990004	1	
3	East	E12	3990004	1	
4	East	E14	3990004	1	

	trip_bikes_allowed	trip_short_name
0	2	Escondido
1	2	Escondido
2	2	Escondido
3	2	Escondido
4	2	Escondido

Expected data cleaning, if any: We expected to clean and filter the following data set:

For `transit_route`, we will need to replace None with NCTD in `transit_route_gdf['agency_id']`, fill `Nan` with its respected short_name in `transit_route_gdf['long_name']` and assign a color to Bus, Rail, and Tram in `transit_route_gdf['route_type']`

[11]:
```
# replace None with NCTD
transit_route_gdf['agency_id'] = transit_route_gdf['agency_id'].fillna('NCTD')
# fill Non` with its respaected short_name
transit_route_gdf.loc[transit_route_gdf['long_name'] == 'COASTER',␣
 ↪'short_name'] = \
    transit_route_gdf.loc[transit_route_gdf['long_name'] == 'COASTER',␣
 ↪'short_name'].fillna('COASTER')

transit_route_gdf.loc[transit_route_gdf['long_name'] == 'SPRINTER',␣
 ↪'short_name'] = \
    transit_route_gdf.loc[transit_route_gdf['long_name'] == 'SPRINTER',␣
 ↪'short_name'].fillna('SPRINTER')

transit_route_gdf.loc[transit_route_gdf['long_name'] == 'Old Town to Airport␣
 ↪Shuttle', 'short_name'] = \
```

```
    transit_route_gdf.loc[transit_route_gdf['long_name'] == 'Old Town to␣
 ↪Airport Shuttle', 'short_name'].fillna('AIR')

# assign color to Bus, Rail, and Tram
transit_route_gdf.loc[transit_route_gdf['route_type'] == 0, 'hex_color'] =␣
 ↪'#00FF00'
transit_route_gdf.loc[transit_route_gdf['route_type'] == 2, 'hex_color'] =␣
 ↪'#0000FF'
transit_route_gdf.loc[transit_route_gdf['route_type'] == 3, 'hex_color'] =␣
 ↪'#FF0000'
```

For `mts_routes`, we will need to get the most frequent MTS route in `mts_trips[['route_id', 'service_id']]` and get the most frequenct by counting the number of bus each hour. The formula will be 12 bus each hour for 12 hours for both direction (back and front), totaling 288 bus for the line each day.

[12]:
```
# get the most frequent MTS route
mts_routes_count = mts_trips[['route_id', 'service_id']].groupby('route_id').
 ↪count()

mts_trips_count = 12 * 12 * 2 # 12 bus each hour for 12 hours for both␣
 ↪direction (back and front)
mts_frequent_route = mts_routes_count[mts_routes_count['service_id'] >␣
 ↪mts_trips_count].index.to_numpy()
mts_frequent_route
```

[12]:
```
array(['1', '10', '11', '12', '120', '13', '2', '201', '202', '215',
       '225', '235', '3', '30', '35', '41', '43', '44', '5', '510', '520',
       '530', '6', '7', '709', '8', '815', '852', '9', '904', '906',
       '907', '929', '932', '955', '962', '992'], dtype=object)
```

For `nctd_routes`, we will a similiary things, we need to get the most frequentNCTD route in `nctd_trips[['route_id', 'service_id']]` and get the most frequenct by counting the number of bus each hour. The formula will be 12 bus each hour for 12 hours for both direction (back and front), totaling 288 bus for the line each day.

[13]:
```
# get the most frequent NCTD route
nctd_routes_count = nctd_trips[['route_id', 'service_id']].groupby('route_id').
 ↪count()

nctd_trips_count = 12 * 12 * 2 # 12 bus each hour for 12 hours for both␣
 ↪direction (back and front)
nctd_frequent_route =  nctd_routes_count[nctd_routes_count['service_id'] >␣
 ↪nctd_trips_count].index.to_numpy(dtype=str)
nctd_frequent_route
```

[13]:
```
array(['303', '305'], dtype='<U21')
```

Finally, we wil add the the information of most frequent rout bacl to its origional data frame by creating a new column namd `frequent_route` and assign 1 if it is a frequent route and vice versa.

[14]:
```
# add the information back to transit_gdf
frequent_route = np.append(mts_frequent_route, nctd_frequent_route)
transit_route_gdf['frequent_route'] = np.where(transit_route_gdf['route_id'].
 ↪isin(frequent_route), 1, 0)
```

[15]:
```
transit_stop_gdf
```

[15]:

```
      objectid   stop_uid stop_id  stop_code  \
0            1  MTS_10001   10001      10001
1            2  MTS_10003   10003      10003
2            3  MTS_10004   10004      10004
3            4  MTS_10006   10006      10006
4            5  MTS_10007   10007      10007
…          …         …     …          …
6215      6216      NCTD_    None          0
6216      6217      NCTD_    None          0
6217      6218      NCTD_    None          0
6218      6219      NCTD_    None          0
6219      6220      NCTD_    None          0

                            stop_name   stop_lat    stop_lon stop_agncy  \
0           Cabrillo National Monument  32.674458 -117.240426        MTS
1                 Pearl St & Draper Av  32.839578 -117.276187        MTS
2                    Pearl St & Fay Av  32.840128 -117.273810        MTS
3       Torrey Pines Rd & Exchange Pl  32.845847 -117.268673        MTS
4       Torrey Pines Rd & Princess St  32.848826 -117.262585        MTS
…                                 …        …         …        …
6215  Plaza Camino Real Transit Center  33.178276 -117.336525       NCTD
6216      San Luis Rey Transit Center  33.254410 -117.298129       NCTD
6217             Solana Beach Station  32.992937 -117.271067       NCTD
6218          Sorrento Valley Station  32.902813 -117.225088       NCTD
6219             Vista Transit Center  33.203240 -117.245070       NCTD

      wheelchair       intersec stop_place parent_sta        lat         lng  \
0              1  M-Special_Loc     cabmon       None  32.674453 -117.240414
1              1          N-E/B       None       None  32.839574 -117.276175
2              1          N-E/B       None       None  32.840124 -117.273798
3              1          N-N/B       None       None  32.845842 -117.268661
4              0          N-E/B       None       None  32.848821 -117.262574
…          …             …        …          …        …         …
6215           0           None       None       None  33.178271 -117.336513
6216           0           None       None       None  33.254405 -117.298117
6217           0           None       None       None  32.992932 -117.271055
6218           0           None       None       None  32.902808 -117.225076
```

```
6219           0           None       None       None  33.203235 -117.245058

                          geometry
0     POINT (-117.24041 32.67445)
1     POINT (-117.27617 32.83957)
2     POINT (-117.27380 32.84012)
3     POINT (-117.26866 32.84584)
4     POINT (-117.26257 32.84882)
…                             …
6215  POINT (-117.33651 33.17827)
6216  POINT (-117.29812 33.25441)
6217  POINT (-117.27106 32.99293)
6218  POINT (-117.22508 32.90281)
6219  POINT (-117.24506 33.20324)

[6220 rows x 15 columns]
```

We will then create a buffer of 15 metre, equivalent of 49.21 feet around the most frequent bus route. Then we will check if the bus stop is within the buffer and assignm a new column call `frequent_stop`

[16]:
```
# get the frequent route gdf
frequent_route_gdf = transit_route_gdf[transit_route_gdf['frequent_route'] == 1]
# set crs to metre
frequent_route_gdf = frequent_route_gdf.to_crs('EPSG:32610')
# create a buffer of 15 metre ~ 49.21 feet around the route
frequent_route_buffer = frequent_route_gdf.buffer(15).to_crs('EPSG:4326').
 ↪unary_union
# check if the stop is within frequent_route_buffer
transit_stop_gdf['frequent_stop'] = transit_stop_gdf.geometry.apply(lambda p: 1␣
 ↪if frequent_route_buffer.contains(p) else 0)
```

1.5 Descriptive statistics for the data

we will combine all the informateion and do a quick visualization, we see the san diego actually hav a great coverage at a glance, but it that actual the case?

[17]:
```
# combine and visualize all route and stop
combined_gdf = gpd.GeoDataFrame(pd.concat([transit_route_gdf,␣
 ↪transit_stop_gdf], ignore_index=True), crs=transit_route_gdf.crs)
combined_gdf.explore(column='rte_type_t')
```

[17]:
```
<folium.folium.Map at 0x7f32515e4f70>
```

publish to the transit route to ArcGIS, if not published

[18]:
```
# publish to ArcGIS, if not published
query_result = gis.content.search(query='title:transit_route owner:dsc170wi23_5␣
 ↪type:Feature')
if not query_result:
    transit_route_sedf = GeoAccessor.from_geodataframe(transit_route_gdf,␣
 ↪column_name='geometry')
    transit_route_sedf.spatial.to_featurelayer(title='transit_route', gis=gis)
    transit_route_sedf = gis.content.search(query='title:transit_route owner:
 ↪dsc170wi23_5 type:Feature')[0]
else:
    transit_route_sedf = query_result[0]
transit_route_sedf.share(org=True)
transit_route_sedf
```

[18]: <Item title:"transit_route" type:Feature Layer Collection owner:dsc170wi23_5>

publish the transit stop to ArcGIS, if not published

[19]:
```
# publish to ArcGIS, if not published
query_result = gis.content.search(query='title:transit_stop owner:dsc170wi23_5␣
 ↪type:Feature')
if not query_result:
    transit_stop_sedf = GeoAccessor.from_geodataframe(transit_stop_gdf,␣
 ↪column_name='geometry')
    transit_stop_sedf.spatial.to_featurelayer(title='transit_stop', gis=gis)
    transit_stop_sedf = gis.content.search(query='title:transit_stop owner:
 ↪dsc170wi23_5 type:Feature')[0]
else:
    transit_stop_sedf = query_result[0]
transit_stop_sedf.share(org=True)
transit_stop_sedf
```

[19]: <Item title:"transit_stop" type:Feature Layer Collection owner:dsc170wi23_5>

1.6 Analysis

query the most frequent route stop

[20]:
```
# query the most frequent route stop
transit_stop_sedf_layer = transit_stop_sedf.layers[0]
transit_stop_feature_set = transit_stop_sedf_layer.query(where='frequent_s=1')
transit_stop_geometries = [Point(feature.geometry) for feature in␣
 ↪transit_stop_feature_set.features]
```

query the most frequent route stop

[21]:
```
# query the most frequent route stop
transit_route_sedf_layer = transit_route_sedf.layers[0]
transit_route_feature_set = transit_route_sedf_layer.query(where='frequent_r=1')
transit_route_geometries = [Point(feature.geometry) for feature in␣
 ↪transit_route_feature_set.features]
```

Enrich it with walking time of 0.25, 0.5, 1 min
Note that (for somerason it default to driving_time no matter what the pormimity_type is)
Also, it take a while to do! So I save the result

[22]:
```
# Enrich it with walking time of 0.25, 0.5, 1 min (for somerason it default to␣
 ↪driving_time no matter what the pormimity_type is)
# take a while to do!
# transit_stop_walkable_distance = enrich(study_areas=transit_stop_geometries,
#                                         proximity_value=[0.25, 0.5, 1],
#                                         proximity_type='walking_time',
#                                         proximity_metric='minutes')
# transit_stop_walkable_distance

transit_stop_walkable_distance = pd.read_csv('./data/enrich.csv')
transit_stop_walkable_distance
```

[22]:

	FID	source_cou	area_type	buffer_uni	buffer_u_1	\
0	1	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
1	2	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
2	3	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
3	4	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
4	5	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
…	…	…	…	…	…	
5794	5795	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
5795	5796	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
5796	5797	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
5797	5798	USA	NetworkServiceArea	Minutes	Drive Time Minutes	
5798	5799	USA	NetworkServiceArea	Minutes	Drive Time Minutes	

	buffer_rad	aggregatio	\
0	1.00	BlockApportionment:US.BlockGroups;PointsLayer:…	
1	0.25	BlockApportionment:US.BlockGroups;PointsLayer:…	
2	0.50	BlockApportionment:US.BlockGroups;PointsLayer:…	
3	1.00	BlockApportionment:US.BlockGroups;PointsLayer:…	
4	0.25	BlockApportionment:US.BlockGroups;PointsLayer:…	
…	…	…	
5794	1.00	BlockApportionment:US.BlockGroups;PointsLayer:…	
5795	0.25	BlockApportionment:US.BlockGroups;PointsLayer:…	
5796	0.50	BlockApportionment:US.BlockGroups;PointsLayer:…	
5797	1.00	BlockApportionment:US.BlockGroups;PointsLayer:…	

```
5798        0.25  BlockApportionment:US.BlockGroups;PointsLayer:…

      population  apportionm  has_data  totpop   tothh  avghhsz  totmales  \
0          2.191       2.576         1  2153.0  1050.0     2.05    1058.0
1          2.191       2.576         1   400.0   227.0     1.74     196.0
2          2.191       2.576         1   810.0   436.0     1.84     400.0
3          2.191       2.576         1  1683.0   816.0     2.05     842.0
4          2.191       2.576         1   120.0    50.0     2.40      60.0
…          …          …       …     …     …       …        …
5794       2.191       2.576         1   393.0   165.0     2.38     162.0
5795       2.191       2.576         0     0.0     0.0     0.00       0.0
5796       2.191       2.576         0     0.0     0.0     0.00       0.0
5797       2.191       2.576         1   772.0   213.0     3.60     394.0
5798       2.191       2.576         0     0.0     0.0     0.00       0.0

      totfemales    Shape__Area  Shape__Length  \
0         1095.0  828973.105469    5271.614024
1          204.0  127182.031250    1495.295859
2          410.0  289692.535156    2632.585860
3          841.0  809571.585938    4907.274088
4           60.0  111560.421875    1283.371128
…          …            …              …
5794       231.0  336566.351562    3260.338809
5795         0.0   77723.902344    1099.650280
5796         0.0  155075.320312    1760.644416
5797       378.0  435332.003906    3879.124915
5798         0.0   76194.714844    1127.658765

                                                  SHAPE
0     {'rings': [[[-13055365.2563, 3873875.2146], [-…
1     {'rings': [[[-13055215.257, 3872506.584], [-13…
2     {'rings': [[[-13055240.2569, 3872714.8437], [-…
3     {'rings': [[[-13055165.2571, 3873071.8686], [-…
4     {'rings': [[[-13055115.2574, 3872149.5762], [-…
…                                                   …
5794  {'rings': [[[-13061265.2293, 3925374.327], [-1…
5795  {'rings': [[[-13057815.2451, 3928751.9146], [-…
5796  {'rings': [[[-13057765.2453, 3928961.178], [-1…
5797  {'rings': [[[-13057715.2455, 3929319.9239], [-…
5798  {'rings': [[[-13051665.2732, 3923491.6726], [-…

[5799 rows x 18 columns]
```

publish to ArcGIS, if not published

[23]:
```
# publish to ArcGIS, if not published
```

```
query_result = gis.content.search(query='title:walkable_distance_combine owner:
 ↪dsc170wi23_5 type:Feature')
if not query_result:
    gdf = gpd.GeoDataFrame(transit_stop_walkable_distance, geometry='SHAPE').
 ↪set_crs(3857).to_crs(4326)
    walkable_distance = GeoAccessor.from_geodataframe(gdf,␣
 ↪column_name='geometry')
    walkable_distance.spatial.to_featurelayer(title='walkable_distance_combine␣
 ↪', gis=gis)
    walkable_distance_sedf = gis.content.search(query= \
                                                'title:
 ↪walkable_distance_combine  owner:dsc170wi23_5 type:Feature')[0]
else:
    walkable_distance_sedf = query_result[0]
walkable_distance_sedf.share(org=True)
walkable_distance_sedf
```

[23]:
```
<Item title:"walkable_distance_combine " type:Feature Layer Collection
owner:dsc170wi23_5>
```

merge all the walkable into on polygon

[24]:
```
# merge all the walkable into on polygon
walkable_gdf = (gpd.GeoDataFrame(walkable_distance_sedf.layers[0].query().sdf,␣
 ↪geometry='SHAPE')
                .set_crs(3857)
                .to_crs(4326)
                .unary_union)
merge_gdf = gpd.GeoDataFrame(geometry=[walkable_gdf])
```

check if shop is within that polygon

[25]:
```
# check if shop is within that polygon
shop_gdf['frequent'] = np.where(shop_gdf.within(merge_gdf.geometry[0]), 1, 0)
```

publish to ArcGIS, if not published

[26]:
```
# publish to ArcGIS, if not published
query_result = gis.content.search(query='title:shop owner:dsc170wi23_5 type:
 ↪Feature')
if not query_result:
    shop_sedf = GeoAccessor.from_geodataframe(shop_gdf, column_name='geometry')
    shop_sedf.spatial.to_featurelayer(title='shop', gis=gis)
    shop_sedf = gis.content.search(query='title:shops owner:dsc170wi23_5 type:
 ↪Feature')[0]
else:
    shop_sedf = query_result[0]
```

```
shop_sedf.share(org=True)
shop_sedf
```

[26]: <Item title:"shop" type:Feature Layer Collection owner:dsc170wi23_5>

To get a better understanding of the proportion of frequent routes and stops in San Diego, we can perform a quick statistical analysis. We can calculate the percentage of routes and stops that meet our definition of frequent, which will provide us with a clearer picture of the overall usage of public transportation in the area.

[27]:
```
len(transit_stop_feature_set.sdf) / len(transit_stop_sedf_layer.query().sdf)
```

[27]: 0.31077170418006433

[28]:
```
len(transit_route_feature_set.sdf) / len(transit_route_sedf_layer.query().sdf)
```

[28]: 0.32894736842105265

After analyzing the data and applying our criteria for defining frequent routes and stops, we have found that approximately 30% of the bus routes and stops in San Diego meet our definition of frequent.

To further understand and visualize this data, we will use ESRI. Using ESRI, we can create interactive maps and visualizations that will allow us to explore the data in greater detail.

[29]:
```
map_widget = gis.map('San Deigo, CA')
```

we will define the render information

[ ]:
```
# Defind render
cmap = ["#76b7b2", "#2ca02c", "#b3cde3", "#00FF00", "#F0F000","#FFFF00"]
unique_values = list(set(shop_sedf.layers[0].query().sdf['shop'].values))
symbols = []
for i, value in enumerate(unique_values):
    color = cmap[i]
    symbol = {"type": "esriSMS", "style": "esriSMSCircle", "color": color,␣
 ↪"size": 5}
    symbols.append(symbol)

# Define a unique value renderer with the symbols and unique values
shop_sedf_renderer = {"type": "uniqueValue", "field1": "shop",␣
 ↪"uniqueValueInfos": []}
for i in range(len(unique_values)):
    info = {"value": unique_values[i], "symbol": symbols[i]}
    shop_sedf_renderer["uniqueValueInfos"].append(info)

shop_sedf.layers[0].renderer = shop_sedf_renderer
```

In addition to the steps outlined earlier, we will also consider different walking distances of 0.25,

0.5, and 1 mile respectively. By doing this, we can identify the bus stops that fall within these walking distances and gain further insights into public transportation usage patterns in San Diego.

To achieve this, we will create three buffers around each bus stop, each with a radius corresponding to the walking distance. For instance, the buffer for a walking distance of 0.25 miles. Noted the before it said driving time and that has been already encounter of. We will assume that 1 minute is equal to 1 mile.

[ ]:
```
# walk 0.25 miles
query_result = gis.content.search(query='title:walkable_distance_25 owner:
 ↪dsc170wi23_5 type:Feature')
if not query_result:
    gdf = gpd.GeoDataFrame(walkable_distance_sedf.layers[0].
 ↪query(where='buffer_rad=0.25').sdf, geometry='SHAPE').set_crs(3857).
 ↪to_crs(4326).unary_union
    merge_gdf = gpd.GeoDataFrame(geometry=[gdf])
    merge_gdf['name'] = 'walking time'
    walkable_distance = GeoAccessor.from_geodataframe(merge_gdf,␣
 ↪column_name='geometry')
    walkable_distance.spatial.to_featurelayer(title='walkable_distance_25',␣
 ↪gis=gis)
    walkable_distance_25_sedf = gis.content.search(query= \
                                                   'title:walkable_distance_25␣
 ↪owner:dsc170wi23_5 type:Feature')[0]
else:
    walkable_distance_25_sedf = query_result[0]


# walk 0.5 miles
query_result = gis.content.search(query='title:walkable_distance_50 owner:
 ↪dsc170wi23_5 type:Feature')
if not query_result:
    gdf = gpd.GeoDataFrame(walkable_distance_sedf.layers[0].
 ↪query(where='buffer_rad=0.50').sdf, geometry='SHAPE').set_crs(3857).
 ↪to_crs(4326).unary_union
    merge_gdf = gpd.GeoDataFrame(geometry=[gdf])
    merge_gdf['name'] = 'walking time'
    walkable_distance = GeoAccessor.from_geodataframe(merge_gdf,␣
 ↪column_name='geometry')
    walkable_distance.spatial.to_featurelayer(title='walkable_distance_50',␣
 ↪gis=gis)
    walkable_distance_50_sedf = gis.content.search(query= \
                                                   'title:walkable_distance_50␣
 ↪owner:dsc170wi23_5 type:Feature')[0]
else:
    walkable_distance_50_sedf = query_result[0]
```

```
# walk 1.0 miles
query_result = gis.content.search(query='title:walkable_distance_100 owner:
 ↪dsc170wi23_5 type:Feature')
if not query_result:
    gdf = gpd.GeoDataFrame(walkable_distance_sedf.layers[0].
 ↪query(where='buffer_rad=1.00').sdf, geometry='SHAPE').set_crs(3857).
 ↪to_crs(4326).unary_union
    merge_gdf = gpd.GeoDataFrame(geometry=[gdf])
    merge_gdf['name'] = 'walking time'
    walkable_distance = GeoAccessor.from_geodataframe(merge_gdf,␣
 ↪column_name='geometry')
    walkable_distance.spatial.to_featurelayer(title='walkable_distance_100',␣
 ↪gis=gis)
    walkable_distance_100_sedf = gis.content.search(query= \
                                                    'title:walkable_distance_100␣
 ↪owner:dsc170wi23_5 type:Feature')[0]
else:
    walkable_distance_100_sedf = query_result[0]
```

This is just for the ESRI map infromation

[ ]:
```
# set the render information
walkable_distance_25_renderer = {
    "type": "simple",
    "symbol": {
        "type": "esriSFS",
        "color": [128, 128, 128, 128, 32],
        "outline": {
            "type": "esriSLS",
            "color": [0, 0, 0],
            "width": 0
        }
    }
}

walkable_distance_25_sedf.layers[0].renderer =walkable_distance_25_renderer


walkable_distance_50_renderer = {
    "type": 'simple',
    "symbol": {
        "type": "esriSFS",
        "color": [128, 128, 128, 128, 64],
        "outline": {
            "type": "esriSLS",
            "color": [0, 0, 0, 0],
            "width": 0
```

```
        }
    }
}

walkable_distance_50_sedf.layers[0].renderer = walkable_distance_50_renderer


walkable_distance_100_renderer = {
    "type": 'simple',
    "symbol": {
        "type": "esriSFS",
        "color": [128, 128, 128, 128, 128],
        "outline": {
            "type": "esriSLS",
            "color": [0, 0, 0, 128],
            "width": 1
        }
    }
}

walkable_distance_100_sedf.layers[0].renderer = walkable_distance_100_renderer
```

We add the layer we need back to the map and set a larget highe for visualizaion.

[ ]:
```
map_widget.add_layer(shop_sedf)

map_widget.add_layer(walkable_distance_25_sedf)
map_widget.add_layer(walkable_distance_50_sedf)
map_widget.add_layer(walkable_distance_100_sedf)

map_widget.add_layer(transit_route_sedf.layers[0].query(where='frequent_r=1'))
map_widget.add_layer(transit_stop_sedf.layers[0].query(where='frequent_s=1'))
```

[ ]:
```
map_widget.layout.height = '900px'
map_widget.legend = True
```

The resullt will be the following: (the color for shop does nto appear correctly for some reason)

[ ]:
```
map_widget
```

and the proportion of shops that are within frequent bus stops

[ ]:
```
len(shop_sedf.layers[0].query(where='frequent=1.0').sdf) / len(shop_sedf.
↪layers[0].query().sdf)
```

After analyzing the data, we have discovered that even with a mere 30% coverage of bus stops in San Diego, approximately half of the shops in the area can still be accessed. This indicates that well-planned bus routes and frequent stops are crucial in facilitating easy access to shopping centers

and department stores for both residents and visitors.

Interestingly, when looking at the map, we might assume that the coverage would extend to more than 55% of the area based on walkable distance. However, our analysis reveals that the actual coverage is lower than expected. We believe that other factors, such as physical barriers or inadequate pedestrian infrastructure, may contribute to the limited accessibility of shops through public transportation.

Our analysis has also revealed that one noticeable area where there is a lack of coverage is Mira Mesa, where the frequent bus routes and stops fail to cover most of the shops in the area. This suggests that there is a need to improve the public transportation infrastructure in this part of San Diego to ensure that residents have better access to shopping centers and department stores.

Machine Larning Analysis We will perform a basic classification using the RandomForest algorithm. Random Forest is a machine learning algorithm that combines multiple decision trees to improve prediction accuracy and stability, making it suitable for handling large, noisy, and high-dimensional datasets. We will use the variables of `frequent_bus`, `frequent_route`, and the `shop` type to determine whether a location is within a walkable distance of `frequent`.

[ ]:
```
# X data bus
walkable_gdf = (gpd.GeoDataFrame(walkable_distance_sedf.layers[0].
 ↪query(where='buffer_rad=0.25').sdf, geometry='SHAPE')
                .set_crs(3857)
                .to_crs(4326)
                .unary_union)
merge_gdf = gpd.GeoDataFrame(geometry=[walkable_gdf])

# check if shop is within that polygon
shop_gdf['frequent_bus'] = np.where(shop_gdf.within(merge_gdf.geometry[0]), 1,␣
 ↪0)
```

[ ]:
```
# bus route route
walkable_gdf = (transit_route_gdf[transit_route_gdf['frequent_route'] == 1]
                .to_crs(3857)
                .buffer(400)
                .to_crs(4326)
                .unary_union)
merge_gdf = gpd.GeoDataFrame(geometry=[walkable_gdf])

# check if shop is within that polygon
shop_gdf['frequent_route'] = np.where(shop_gdf.within(merge_gdf.geometry[0]),␣
 ↪1, 0)
```

To use the `shop` variable as a feature in our Random Forest classification, we will need to perform one hot encoding on the categorical data.

[ ]:
```
X = shop_gdf[['frequent_bus', 'frequent_route', 'shop']]

X = OneHotEncoder(drop='first').fit_transform(X)
```

```
y = shop_gdf[['frequent']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

Here is out accuracy score. Accuracy score is a metric used to evaluate the performance of a classification model. It represents the proportion of correctly classified instances over the total number of instances.

[ ]:
```
# Random Forest
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
accuracy_score(y_train, clf.predict(X_train))
```

[ ]:
```
# XGBoost
model = xgb.XGBClassifier()
model.fit(X_train, y_train)
accuracy_score(y_test, model.predict(X_test))
```

1.7 Summary of products and results:

Our analysis of frequent bus routes and stops in San Diego has provided valuable insights into the relationship between public transportation and access to shopping centers and department stores in the area. Our findings reveal that even with just 30% coverage of bus stops, frequent bus routes and stops can reach approximately 55% of the shops in San Diego. Additionally, using a simple classification technique, we were able to achieve an accuracy rate of approximately 80% in determining whether a shop is considered frequent based on its proximity to a frequent bus route and stop.

These findings highlight the importance of investing in frequent bus services in San Diego to improve access to shopping centers and department stores for residents and visitors. While walkable distances may be relatively short, factors such as physical barriers or inadequate pedestrian infrastructure may hinder access to public transportation and shops. This can lead to people choosing to drive rather than take public transportation, which can negatively impact traffic congestion and air quality in the area.

Our analysis also highlights the need to address gaps in public transportation coverage, particularly in areas such as Mira Mesa where there is a lack of frequent bus routes and stops. Improving access to shopping centers and department stores in this area is particularly important for UCSD students, as popular Asian markets such as 99 Ranch and H Mart are located there.

Overall, our analysis underscores the importance of continued investment in public transportation infrastructure in San Diego to improve accessibility, promote economic growth and development, and create a more sustainable and livable community for all residents.

1.8 Discussion:

Our analysis of frequent bus routes and stops in San Diego provides new insights into the relationship between public transportation and access to shopping centers and department stores in

the area. We were able to confirm that San Diego does have a comprehensive network of public transportation and with only a small portion of the frequent bus route. San Diegan can access most of the needed by taking its publication. However, one new thing that our analysis provides is the lack of public transportation at Mira Mesa, this is especially true for UCSD students as it is one of the major shopping places for them. As this might be one of the major factors why the car owner ship amount UCSD student is so high.

In terms of trade-offs and decision points, we had to use the driving distance to do a reverse calculation of the walking distance for our geo enrichment. As we are unable to get the walking time no matter how we adjust the parameter of the geo-enrichment function. The map also did not display its color correctly. We also can be choosing better machine learning and its data to provide a more accurate result of our prediction.

1.9 Conclusions and future work:

Our initial research question was to analyze the relationship between frequent bus routes and stops and is walkable distance to shop in San Diego. Our analysis provided the proportion of shops that can be reached. With only 30% coverage of frequent bus routes and stops, it has been able to cover more than 50% of the shop.

In conclusion, while our analysis provides valuable insights into the relationship between frequent bus routes and stops and accessibility to shopping centers and department stores in San Diego, there are additional data and analysis steps that can be taken to further address the research question. By gathering additional data on travel time and cost, as well as incorporating live traffic data, we can provide more accurate and comprehensive insights into the benefits of public transportation and the impact on accessibility to different places in San Diego.

Moreover, our approach can be extended to other areas or topics, such as analyzing the accessibility of different services or facilities in San Diego, including hospitals, restaurants, libraries, police stations, and more. By using additional datasets and analysis techniques, we can provide more informative insights into the relationship between public transportation and accessibility.

The results of our analysis can be used by city planners, transportation authorities, and local businesses to inform decision-making related to public transportation infrastructure and economic development.

[ ]: