FinalProject

January 14, 2020

1 COGS 108 - Final Project

2 Overview

We set out to find what types of conditions result in more accidents. We initially defined condition with indicators like weather, street construction, level of street lights, cultural conditions (think west-coast chill vs east-coast rush) and the street quality such as number of potholes. After careful consideration, we decided to focus our study on San Diego and a single indicator, overall street condition (OCI), to measure the aggregate cracking, potholes and drivability. Finding a correlation between street condition and accidents could help us alleivate and better inform drivers mitigating the number of crashes.

3 Names

Brian Chi William Xu Nancy Morales Sydney Blouin Ana Selvaraj

4 Revised Research Question

After narrowing our focus for this project, we decided to specify our research question as follows:

What is the effect of poor street conditions on the number of traffic collisions that occurred in San Diego in 2015?

We defined poor street conditions to reflect lower OCI ratings, consistent with the ratings provided by the City of San Diego.

5 Background and Prior Work

Traffic collisions are the leading cause of death among people ages 15 - 29 (1). In particular, America leads the developed world on traffic deaths (2). Our group wanted to understand the factors behind the average of 6 million car accidents that happen every year in America (3).

A factor that determines the need for street repair is the Overall Condition Index (OCI), a 100-point rating system that measures cracking, potholes, and drivability to assess the overall quality of a given road (4). OCI values of 0-39 represent “Poor” conditions, 40-69 represent “Fair” conditions, and 70-100 represent “Good” conditions. Street repair projects are more urgent on roads that have OCI values are closer to 0.

These projects can cause obstruction on roadways, encourage rubbernecking and increase traffic volume, potentially causing skirmishes between cars and/or construction equipment. However, "If [roads are] not adequately treated, all streets will deteriorate and develop large cracks and potholes" (5). When drivers evade potholes, they could change lanes abruptly and drive unpredictably, thus increasing the chances of collisions.

We decided to focus our study on San Diego to limit the variation of geographical/weather conditions that could potentially impact traffic incidents.

References: 1) Goodyear, Sarah. "The Keys to Designing Cities With Fewer Traffic Fatalities." CityLab. 2) Schmitt, Angie, and Stephen Miller. "Why the U.S. Leads the Developed World on Traffic Deaths." Streetsblog USA 3) "Car Accident Statistics in the U.S.: Driver Knowledge." Driver Knowledge, www.driverknowledge.com/car-accident-statistics/. 4) City of San Diego Open Data Portal. "Streets Overall Condition Index" 5) Eugene. "How Pavement Conditions Are Rated"

6 Hypothesis

In San Diego, do traffic accidents occur more often on roads with "worse" driving conditions in 2015?

We define the conditions of a street with the following: * Street Cracking * Potholes * Drivability

These three give us the overall condition index (OCI) that gives us a score to determine the condition of a street.

Hypothesis:

Streets that have poorer conditions will be more likely to have traffic accidents than those that have better conditions. We believe that this will be the case because drivers will be focused on avoiding potholes leading to more dangerous driving conditions.

7 Datasets

All of our data comes from the open datasets of San Diego.

- Dataset Name: **Traffic Collisions**
- Link to the dataset: https://data.sandiego.gov/datasets/police-collisions/
- Number of observations: 29,443
- Description: This dataset contains the list of traffic collisions. It includes information such as the address and intersecting street at which the collision occured.
- Variables: "report_id","date_time","police_beat","address_number_primary","address_pd_primary","ac
-
- Dataset Name: **Streets Overall Condition Index (OCI)**
- Link to the dataset: https://data.sandiego.gov/datasets/police-collisions/
- Number of observations: 30,713

- Description: This dataset contains the list of street conditions in which conditions are defined by the street cracking, potholes, and drivability. Each data contains the start and end segment.
- Variables: “seg_id”,“oci”,“street”,“street_from”,“street_to”,“seg_length_ft”,“seg_width_ft”,“func_class”

We are using the traffic collision and the street OCI dataset because these directly address our goals. We plan to combine these two datasets by analyzing the location at which a traffic collision happened and figuring out the OCI by looking at the street segment.

8 Setup

The following code imports all libaries in addition to setting up optional parameters that we will use. This also sets up our google maps API key and our baseline datasets (Collisions and OCI). We will later use the google maps API to calculate roughly 30,000~ accident coordinates and 30,000~ street segments.

[1]:
```
%matplotlib inline
import geopandas as gpd
import descartes
import matplotlib.colors
from shapely.geometry import Point, Polygon
import calendar
import matplotlib.colors as colors
from collections import defaultdict

import googlemaps
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import time
from haversine import haversine, Unit

pd.options.display.max_rows = 10
pd.set_option('precision', 5)
pd.set_option("mode.chained_assignment", None)


# If you want to run, uncomment two lines below and insert API key
api_key = 'AIzaSyBDKT3wNKReDeGJ0zj1cMzPUhZDls6HvZk'
gmaps = googlemaps.Client(key=api_key)

df_roads = pd.read_csv("../datasets/oci_2015_datasd.csv")
df_accidents = pd.read_csv("../datasets/pd_collisions_datasd_v1.csv")
```

9 Data Cleaning

The data cleaning process was the most strenuous part of this project. We had several problems with our dataset. The first being that our Accident dataset contained freeways which our Segment OCI dataset did not include freeways posing a problem of us mapping an OCI to an Accident. Secondly, our Segment OCI dataset contained alleys while none of our Accidents were in alleys introducing the issue of noise. Moreover, the segment OCI dataset contained the START/END of a road however we do not know what entails the beginning of a road thus these were dropped too. Lastly, both the Accident and Segment OCI dataset only contained street names and no coordinates leading to an issue of mapping Accidents to the OCI.

The process composed of 5 different parts:

1. Remove all alleys from the Segment OCI dataset
2. Remove all highways from the Accident dataset
3. For each accident, given it's street, find the respective latitude and longitude. This gives us one coordinate of where the accident is.
4. For each road segment, given the street and it's intersection, find the respective latitude and longitude. A segment is composed of two points specifically the start and end. Thus we will have two coordinates for each road segment.
5. Clean out all accidents and road segments that are not in San Diego. e.g. We have coordinates in Big Bear. This is unfortunately a downfall of the google API when there are ambigious number of streets with the same name.

After we have grabbed our latitude and longitude for every single possible location. Our goal is to map each accident to the street quality OCI. We note the caveat that locations are simply an estimate however we believe the best estimate location will give us a good metric as streets that are near each other are likely to have to similar conditions. To map each accident to an OCI, our process is as follows:

1. Find a midpoint given a street segment since we know the start and end of a segment.
2. Given an accident, create a bounding box around that accident and find all midpoints that are within the bounding box. If we find no points within our bounding box, increase the size of the bounding box. For example, find the people that are standing within five meters of you. (Why are we doing this? We do this because there are roughly 30,000 entries and we want to use speedy np arrays to do quick arithmetic.)
3. Now that we have a set of midpoints and accidents, we use haversine to find the midpoint segment that is closest to the accident. Haversine is a distance formula.

The folowing code introduces what we have explained but requires a few hours to finish 30,000 queries. To introduce what we have done, we include a subset of 10 items for demostration purposes.

We have commented every block of code prefixing with explanation.

[2]:
```
### EXPLANATION: Filter out both dataset and create subset for demostration␣
↪purposes

# First, cleaning the roads data
# drop alleys
df_roads = df_roads[df_roads['func_class'].str.upper() != 'ALLEY']
```

```
# drop rows with begin and end
df_roads = df_roads[df_roads.street_from.str.upper() != 'BEGIN']
df_roads = df_roads[df_roads.street_from.str.upper() != 'BEGINNING']
df_roads = df_roads[df_roads.street_to.str.upper() != 'END']

# Removing any numbers that have a start of 0
df_roads["street"] = df_roads["street"].str.lstrip("0")
df_roads["street_to"] = df_roads["street_to"].str.lstrip("0")
df_roads["street_from"] = df_roads["street_from"].str.lstrip("0")

df_roads_subset = df_roads[:10]

# Next, cleaning the accidents data
df_accidents = df_accidents[~df_accidents.address_road_primary.str.lower().str.
↪contains("i-|ca-")]
df_accidents["address_road_primary"] = df_accidents["address_road_primary"].str.
↪lstrip("0")

df_accidents_subset = df_accidents[:10]
```

[3]:
```
### EXPLANATION: Returns the latitude, longitude of streets
def getLatLong(street):
    street = street.strip()
    geo_result = gmaps.geocode(address=street)
    time.sleep(0.02)

    # Checking for error condition
    if len(geo_result) == 0 or geo_result is None:
        print("cannot find", street)
        return (0, 0)

    actual_results = geo_result[0]
    geometry = actual_results.get("geometry")
    location = geometry.get("location")
    lat = location.get("lat")
    lng = location.get("lng")

    return lat, lng
```

[4]:
```
### EXPLANATION: Helper function that returns that latitude and longitude␣
↪specifically for San Diego
def getIntersectionLatLong(streets):
    intersection = streets[0] + " & " + streets[1] + ", San Diego, CA"

    lat, lng = getLatLong(intersection)

    # Prints the intersection, lat, lng for testing purposes
```

```
    print(intersection, lat, lng)

    return lat, lng
```

[5]:
```
### EXPLANATION: Gets the lat and lng of each of the intersections, and appends␣
↪it to the dataset
startLatList = []
startLngList = []
endLatList = []
endLngList = []

for index, st, stf, stt in df_roads_subset[["street", "street_from",␣
↪"street_to"]].itertuples():

    startLat, startLng = getIntersectionLatLong([st, stf])
    endLat, endLng = getIntersectionLatLong([st, stt])

    startLatList.append(startLat)
    startLngList.append(startLng)
    endLatList.append(endLat)
    endLngList.append(endLng)

df_roads_subset["start_lat"] = startLatList
df_roads_subset["start_long"] = startLngList
df_roads_subset["end_lat"] = endLatList
df_roads_subset["end_long"] = endLngList
```

```
1ST AV & MONTECITO WY, San Diego, CA 32.7528342 -117.1641088
1ST AV & ARBOR DR, San Diego, CA 32.7537898 -117.1641572
1ST AV & LEWIS ST, San Diego, CA 32.7467767 -117.1639326
1ST AV & MONTECITO WY, San Diego, CA 32.7528342 -117.1641088
1ST AV & W WASHINGTON ST, San Diego, CA 32.7499659 -117.1639162
1ST AV & LEWIS ST, San Diego, CA 32.7467767 -117.1639326
1ST AV & UNIVERSITY AV, San Diego, CA 32.7196476 -117.1637294
1ST AV & W WASHINGTON ST, San Diego, CA 32.7499659 -117.1639162
1ST AV & ROBINSON AV, San Diego, CA 32.7467767 -117.1639326
1ST AV & UNIVERSITY AV, San Diego, CA 32.7196476 -117.1637294
1ST AV & PENNSYLVANIA AV, San Diego, CA 32.7451259 -117.1638978
1ST AV & ROBINSON AV, San Diego, CA 32.7467767 -117.1639326
1ST AV & BROOKES AV, San Diego, CA 32.7434478 -117.16382
1ST AV & PENNSYLVANIA AV, San Diego, CA 32.7451259 -117.1638978
1ST AV & W WALNUT AV, San Diego, CA 32.7414575 -117.1639691
1ST AV & BROOKES AV, San Diego, CA 32.7434478 -117.16382
1ST AV & UPAS ST, San Diego, CA 32.7407892 -117.1640177
1ST AV & W WALNUT AV, San Diego, CA 32.7414575 -117.1639691
1ST AV & THORN ST, San Diego, CA 32.7395581 -117.1640691
1ST AV & UPAS ST, San Diego, CA 32.7407892 -117.1640177
```

[6]:

```
### EXPLANATION: Remove all entries in the intersections that are out of bounds
minLat = 32
maxLat = 34
maxLng = -116
minLng = -118

df_roads_subset = df_roads_subset[~(df_roads_subset["start_lat"] < minLat) &␣
↪~(df_roads_subset["start_lat"] > maxLat) &
                        ~(df_roads_subset["start_long"] < minLng) &␣
↪~(df_roads_subset["start_long"] > maxLng) &
                        ~(df_roads_subset["end_lat"] < minLat) &␣
↪~(df_roads_subset["end_lat"] > maxLat) &
                        ~(df_roads_subset["end_long"] < minLng) &␣
↪~(df_roads_subset["end_long"] > maxLng)]
```

[7]:

```
### EXPLANATION: We are appending the latitude and longitude back into accidents
location_arr = []

for index, row in df_accidents_subset[['report_id', 'address_number_primary',
                                'address_road_primary', 'address_sfx_primary']].
↪iterrows():

    # Store information about the collision
    report_id, number, road, sfx = row

    # Check address primary
    number = int(number)
    # Replace with empty string if 0
    if number == 0:
        number = ""

    street = "{0} {1} {2}, San Diego, CA".format(number, road, sfx)
    lat, lng = getLatLong(street)
    location_arr.append((report_id, lat, lng))

    print(street, lat, lng)

df_loc = pd.DataFrame(location_arr, columns=['report_id', 'lat', 'lng'])
df_accidents_subset = df_accidents_subset.merge(df_loc)
```

```
5500 VALERIO TRAIL, San Diego, CA 32.9624494 -117.2014782
8300 CAM DEL ORO  , San Diego, CA 32.8568988 -117.2568618
6400 CRAWFORD STREET, San Diego, CA 32.7898003 -117.0938746
8100 ROYAL GORGE DRIVE, San Diego, CA 32.8147757 -117.0512292
1000 A STREET, San Diego, CA 32.7188763 -117.155663
1000 11TH AVENUE, San Diego, CA 32.7157761 -117.1547177
2600 RAMFOS PLACE, San Diego, CA 32.6799953 -117.0425026
```

```
3900 DE LA VALLE  , San Diego, CA 32.9832079 -117.2267559
9100 SYDNEY COURT, San Diego, CA 32.8729365 -117.2021869
1600 HORNBLEND STREET, San Diego, CA 32.798875 -117.240545
```

[8]:
```
### EXPLANATION: Showcasing our new segment dataframe with start/end coordinates
df_roads_subset
```

[8]:
```
         seg_id     oci  street      street_from        street_to  \
2056  SS-000002  84.978  1ST AV     MONTECITO WY         ARBOR DR
2057  SS-000003  77.616  1ST AV         LEWIS ST     MONTECITO WY
2058  SS-000004  78.776  1ST AV  W WASHINGTON ST         LEWIS ST
2059  SS-000005  62.010  1ST AV    UNIVERSITY AV  W WASHINGTON ST
2060  SS-000006  50.856  1ST AV      ROBINSON AV    UNIVERSITY AV
2061  SS-000007  87.952  1ST AV  PENNSYLVANIA AV      ROBINSON AV
2062  SS-000008  81.938  1ST AV       BROOKES AV  PENNSYLVANIA AV
2063  SS-000009  48.836  1ST AV      W WALNUT AV       BROOKES AV
2064  SS-000010  66.306  1ST AV          UPAS ST      W WALNUT AV
2065  SS-000011  67.272  1ST AV         THORN ST          UPAS ST

      seg_length_ft  seg_width_ft func_class    pvm_class  area_sq_ft  \
2056       352.0000          40.0  Collector  AC Improved  14080.0000
2057       357.0000          40.0  Collector  AC Improved  14280.0000
2058       718.0000          42.0  Collector  AC Improved  30156.0000
2059       607.0000          42.0  Collector  AC Improved  25494.0000
2060       475.0000          42.0  Collector  AC Improved  19950.0000
2061       660.0007          42.0  Collector  AC Improved  27720.0294
2062       659.0190          42.0  Collector  AC Improved  27678.7980
2063       676.2468          42.0  Collector  AC Improved  28402.3656
2064       253.0000          52.0  Collector  AC Improved  13156.0000
2065       378.0012          52.0  Collector  AC Improved  19656.0624

     oci_desc       oci_wt  start_lat  start_long   end_lat   end_long
2056     Good  1.19649e+06   32.75283  -117.16411  32.75379 -117.16416
2057     Good  1.10836e+06   32.74678  -117.16393  32.75283 -117.16411
2058     Good  2.37557e+06   32.74997  -117.16392  32.74678 -117.16393
2059     Fair  1.58088e+06   32.71965  -117.16373  32.74997 -117.16392
2060     Fair  1.01458e+06   32.74678  -117.16393  32.71965 -117.16373
2061     Good  2.43803e+06   32.74513  -117.16390  32.74678 -117.16393
2062     Good  2.26795e+06   32.74345  -117.16382  32.74513 -117.16390
2063     Fair  1.38706e+06   32.74146  -117.16397  32.74345 -117.16382
2064     Fair  8.72322e+05   32.74079  -117.16402  32.74146 -117.16397
2065     Fair  1.32230e+06   32.73956  -117.16407  32.74079 -117.16402
```

[9]:
```
### EXPLANATION: Showcasing the accidents with latitude and longitude
df_accidents_subset
```

[9]:
```
  report_id            date_time  police_beat  address_number_primary  \
0    170082  2017-01-01 00:01:00          935                    5500
1    170166  2017-01-01 00:01:00          124                    8300
2    170101  2017-01-01 00:01:00          322                    6400
3    170218  2017-01-01 00:01:00          325                    8100
4    170220  2017-01-01 01:00:00          524                    1000
5    170097  2017-01-01 01:00:00          521                    1000
6    170153  2017-01-01 01:18:00          437                    2600
7    170035  2017-01-01 01:53:00          935                    3900
8    170044  2017-01-01 01:58:00          115                    9100
9    170042  2017-01-01 02:00:00          122                    1600

  address_pd_primary address_road_primary address_sfx_primary  \
0                                 VALERIO               TRAIL
1                             CAM DEL ORO
2                                CRAWFORD              STREET
3                             ROYAL GORGE               DRIVE
4                                       A              STREET
5                                    11TH              AVENUE
6                                  RAMFOS               PLACE
7                             DE LA VALLE
8                                  SYDNEY               COURT
9                               HORNBLEND              STREET

  address_pd_intersecting address_name_intersecting address_sfx_intersecting  \
0
1
2
3
4
5
6
7
8
9

  violation_section violation_type  \
0          MISC-HAZ             VC
1          MISC-HAZ             VC
2          MISC-HAZ             VC
3             22107             VC
4          MISC-HAZ             VC
5             22107             VC
6             22107             VC
7             22107             VC
8             22107             VC
9             22107             VC
```

```
                                        charge_desc  injured  killed  \
0  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
1  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
2  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
3              TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
4  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
5              TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
6              TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
7              TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
8              TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
9              TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0

  hit_run_lvl       lat        lng
0  MISDEMEANOR  32.96245 -117.20148
1  MISDEMEANOR  32.85690 -117.25686
2  MISDEMEANOR  32.78980 -117.09387
3  MISDEMEANOR  32.81478 -117.05123
4  MISDEMEANOR  32.71888 -117.15566
5  MISDEMEANOR  32.71578 -117.15472
6          NaN  32.68000 -117.04250
7  MISDEMEANOR  32.98321 -117.22676
8  MISDEMEANOR  32.87294 -117.20219
9  MISDEMEANOR  32.79888 -117.24054
```

9.1 The previous was an example of data cleaning. We will now load in our official dataset that we have cleaned

[10]:
```python
### EXPLANATION: Load in the official dataset
df_roads_cleaned = pd.read_csv("../datasets/intersections.csv")
df_accidents_cleaned = pd.read_csv("../datasets/accidents.csv")
```

[11]:
```python
### EXPLANATION: Add the midpoints to the dataset for intersections
def add_midpoint_lng(row):
    start_lng = row["start_long"]
    end_lng = row["end_long"]
    return_lng = (start_lng + end_lng) / 2
    return return_lng

def add_midpoint_lat(row):
    start_lat = row["start_lat"]
    end_lat = row["end_lat"]
    return_lat = (start_lat + end_lat) / 2
    return return_lat

df_roads_cleaned["midpoint_lat"] = df_roads_cleaned.apply(lambda row:␣
 ↪add_midpoint_lat(row), axis = 1)
```

```
df_roads_cleaned["midpoint_lng"] = df_roads_cleaned.apply(lambda row:␣
↪add_midpoint_lng(row), axis = 1)
```

[12]:
```
### EXPLANATION: Converting the columns into arrays for quick computation
midpoint_lat = df_roads_cleaned['midpoint_lat'].values
midpoint_lng = df_roads_cleaned['midpoint_lng'].values
midpoint_oci = df_roads_cleaned['oci'].values
```

[13]:
```
### EXPLANATION: Matching each accident to the nearest road segment, and␣
↪assigning the oci of the nearest segment
start_time = time.time()

dataPerReport = dict() # (lat, lng, oci)
for index, (report_id, lat, lng) in df_accidents_cleaned[['report_id', 'lat',␣
↪'lng']].iterrows():
    alpha = 0.005 # Decrease once we have 30,000 segments
    seg_lats = []
    while (len(seg_lats) == 0):
        accident_loc = (lat, lng)
        bbox = (lat - alpha, lat + alpha, lng - alpha, lng + alpha) # min_lat,␣
↪max_lat, min_lng, max_lng
        plat = (midpoint_lat > bbox[0]) & (midpoint_lat < bbox[1])
        plng = (midpoint_lng > bbox[2]) & (midpoint_lng < bbox[3])
        insect = plat & plng # intersection

        seg_lats = midpoint_lat[insect]
        seg_lngs = midpoint_lng[insect]
        seg_ocis = midpoint_oci[insect]

        alpha += 0.005

    if index < 10:
        print("We have found ", len(seg_lats), "segments for ", report_id)

    assert len(seg_lats) == len(seg_lngs) == len(seg_ocis)

    n = len(seg_lats)

    min_dist = 200
    best_index = -1
    for i in range(n):
        seg_loc = (seg_lats[i], seg_lngs[i])
        dist = haversine(accident_loc, seg_loc) # in KM
        if dist < min_dist:
            min_dist = dist
            best_index = i
```

```
    dataPerReport[report_id] = (seg_lats[best_index], seg_lngs[best_index],␣
 ↪seg_ocis[best_index])

print("--- %s seconds ---" % (time.time() - start_time))
```

```
We have found  19 segments for  170082
We have found  60 segments for  170166
We have found  121 segments for  170101
We have found  37 segments for  170218
We have found  159 segments for  170220
We have found  147 segments for  170097
We have found  111 segments for  170153
We have found  2 segments for  170035
We have found  11 segments for  170044
We have found  81 segments for  170042
--- 42.289937257766724 seconds ---
```

[14]:
```
### EXPLANATION: Creating the report dataframe to append to accidents
report_df = pd.DataFrame.from_dict(dataPerReport, orient = "index", columns =␣
 ↪["lat", "lng", "oci"])
report_df["report_id"] = report_df.index
report_df.reset_index(level = 0, inplace = True)
report_df = report_df[["report_id", "oci"]]
```

[15]:
```
### EXPLANATION: Merge the report dataframe with the accident dataframe.
df_accidents_final = pd.merge(df_accidents_cleaned, report_df, on = "report_id")
df_accidents_final.drop(["Unnamed: 0", "Unnamed: 0.1"], axis = 1, inplace =␣
 ↪True)
```

[16]:
```
df_accidents_final
```

[16]:
```
      report_id            date_time  police_beat  address_number_primary  \
0        170082  2017-01-01 00:01:00          935                    5500
1        170166  2017-01-01 00:01:00          124                    8300
2        170101  2017-01-01 00:01:00          322                    6400
3        170218  2017-01-01 00:01:00          325                    8100
4        170220  2017-01-01 01:00:00          524                    1000
…           …                    …            …                      …
28741  19205512  2019-11-11 17:18:00          242                    9500
28742  19205516  2019-11-11 19:10:00          434                    8500
28743  19205514  2019-11-11 19:18:00          112                    4200
28744  19205515  2019-11-11 19:55:00          432                     300
28745  19205517  2019-11-11 20:40:35          524                    1100

      address_pd_primary address_road_primary address_sfx_primary  \
0                                     VALERIO               TRAIL
1                                 CAM DEL ORO
```

```
2                                    CRAWFORD              STREET
3                                 ROYAL GORGE               DRIVE
4                                           A              STREET
…                  …                  …                 …
28741                              GOLD COAST               DRIVE
28742                                 POTRERO              STREET
28743                                 GENESEE              AVENUE
28744                                  THRUSH              STREET
28745                                     5TH              AVENUE

      address_pd_intersecting address_name_intersecting  \
0
1
2
3
4
…                        …                         …
28741
28742
28743
28744
28745

      address_sfx_intersecting violation_section violation_type  \
0                                       MISC-HAZ             VC
1                                       MISC-HAZ             VC
2                                       MISC-HAZ             VC
3                                          22107             VC
4                                       MISC-HAZ             VC
…                         …                …              …
28741                                      22350             VC
28742                                      22107             VC
28743                                     21456B             VC
28744                                      22107             VC
28745                                      22106             VC

                                              charge_desc  injured  killed  \
0       MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
1       MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
2       MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
3                   TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
4       MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
…                                                   …      …       …
28741                UNSAFE SPEED (BASIC SPEED LAW) (I)        1       0
28742         TURNS:UNSAFE TURN AND/OR NO TURN SIGNAL (I)        0       0
28743   PEDESTRIAN CROSS AGAINST NO WALK SIGN/SIGNAL (I)        1       0
28744         TURNS:UNSAFE TURN AND/OR NO TURN SIGNAL (I)        0       0
```

```
28745                              UNSAFE START ON HWY (I)        1        0

       hit_run_lvl       lat         lng     oci
0      MISDEMEANOR  32.96245 -117.20148  84.368
1      MISDEMEANOR  32.85690 -117.25686  28.270
2      MISDEMEANOR  32.78980 -117.09387  56.994
3      MISDEMEANOR  32.81478 -117.05123  46.900
4      MISDEMEANOR  32.71888 -117.15566  46.942
…              …         …          …      …
28741          NaN  32.90536 -117.12050  68.686
28742  MISDEMEANOR  32.71490 -117.02320  71.576
28743          NaN  32.81939 -117.18204  54.116
28744  MISDEMEANOR  32.70762 -117.07221  91.616
28745  MISDEMEANOR  32.71689 -117.16036  82.204

[28746 rows x 19 columns]
```

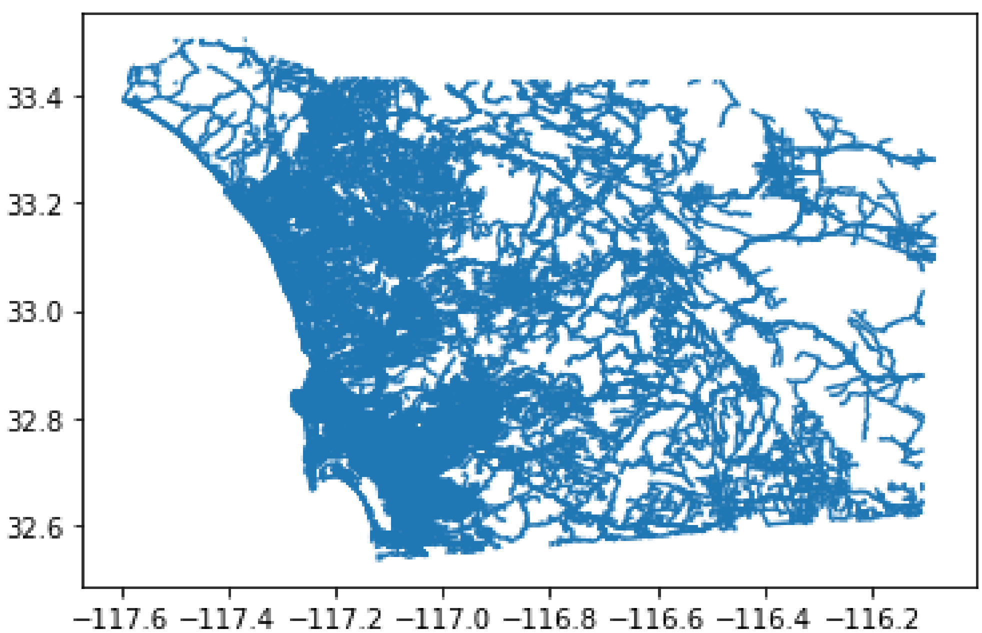
10 Data Analysis & Results

10.0.1 Map of San Diego

X is longitude, Y is latitude

[17]:
```
### EXPLANATION: Creating a street map of San Diego
street_map = gpd.read_file('../mapData/tl_2019_06073_roads.shp')
street_map.plot()
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcfd0d814d0>



[18]:
```python
### EXPLANATION: Getting rid of any potential out of bounds points
indexNames = df_accidents_final[(df_accidents_final['lat'] < 31) |␣
 ↪(df_accidents_final['lat'] > 34) | (df_accidents_final['lng'] > -116) |␣
 ↪(df_accidents_final['lng'] < -118)].index
df_accidents_final.drop(indexNames , inplace = True)
df_accidents_final.head()
```

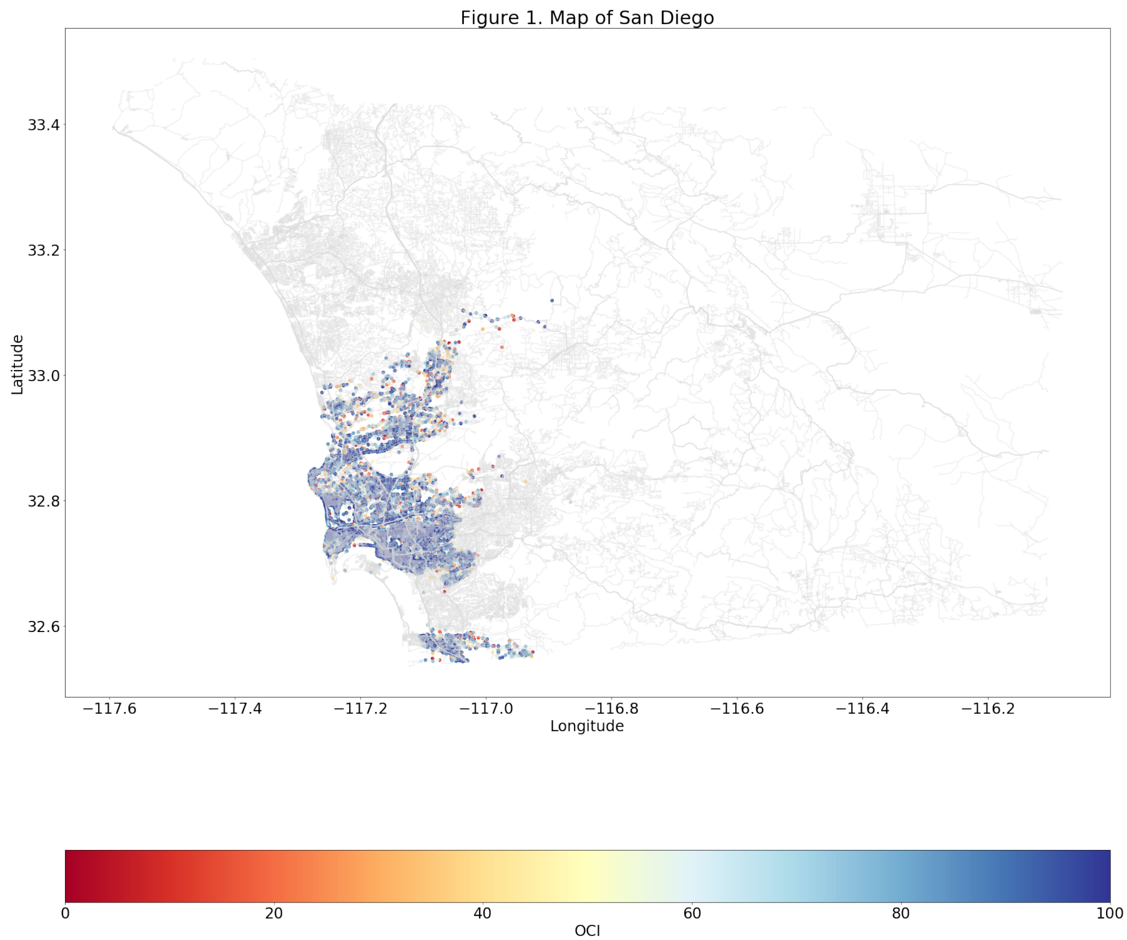
[18]:
```
  report_id            date_time  police_beat  address_number_primary  \
0    170082  2017-01-01 00:01:00          935                    5500
1    170166  2017-01-01 00:01:00          124                    8300
2    170101  2017-01-01 00:01:00          322                    6400
3    170218  2017-01-01 00:01:00          325                    8100
4    170220  2017-01-01 01:00:00          524                    1000

  address_pd_primary address_road_primary address_sfx_primary  \
0                                 VALERIO               TRAIL
1                            CAM DEL ORO
2                                CRAWFORD              STREET
3                            ROYAL GORGE               DRIVE
4                                       A              STREET

  address_pd_intersecting address_name_intersecting address_sfx_intersecting  \
0
1
2
3
4

  violation_section violation_type  \
0          MISC-HAZ             VC
1          MISC-HAZ             VC
2          MISC-HAZ             VC
3             22107             VC
4          MISC-HAZ             VC

                                      charge_desc  injured  killed  \
0  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
1  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
2  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0
3             TURNING MOVEMENTS AND REQUIRED SIGNALS        0       0
4  MISCELLANEOUS HAZARDOUS VIOLATIONS OF THE VEHI…        0       0

   hit_run_lvl       lat         lng     oci
0  MISDEMEANOR  32.96245 -117.20148  84.368
1  MISDEMEANOR  32.85690 -117.25686  28.270
```

```
2  MISDEMEANOR  32.78980 -117.09387  56.994
3  MISDEMEANOR  32.81478 -117.05123  46.900
4  MISDEMEANOR  32.71888 -117.15566  46.942
```

[19]:
```python
### EXPLANATION: Set up variables for plotting
crs = {'init': 'espg:4326'}
geometry = [Point(xy) for xy in zip(df_accidents_final["lng"],␣
 ↪df_accidents_final["lat"])]
```

[20]:
```python
geo_df = gpd.GeoDataFrame(df_accidents_final, crs=crs, geometry=geometry)
```

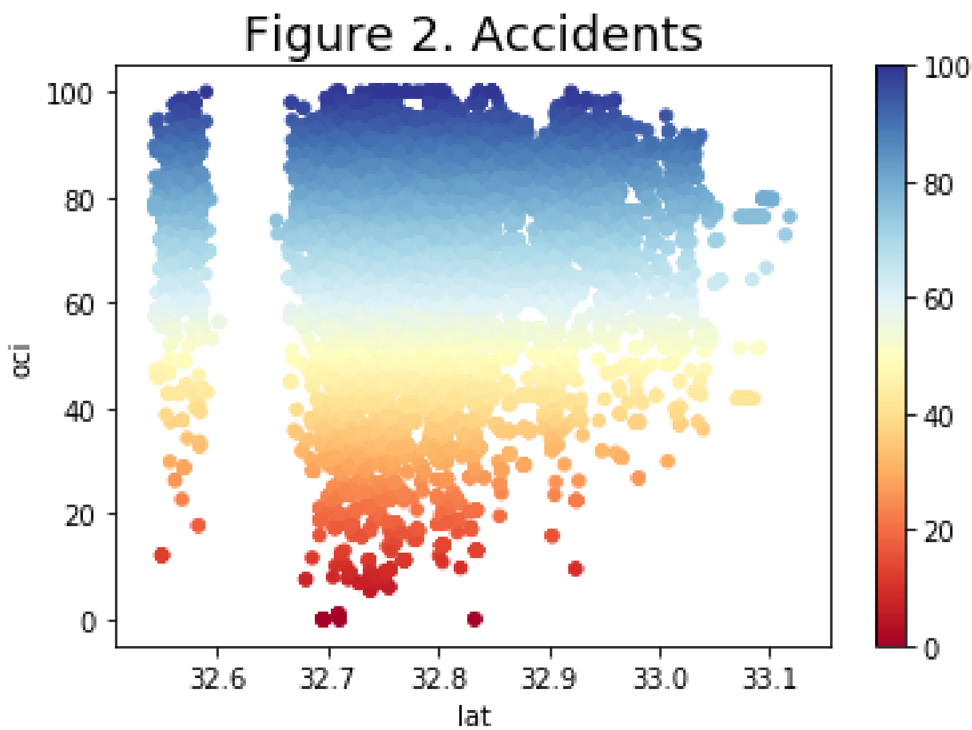
[21]:
```python
fig, ax = plt.subplots(figsize=(30, 30))
ax.set_title("Figure 1. Map of San Diego", fontsize=30)
ax.set_ylabel("Latitude", fontsize=24)
ax.set_xlabel("Longitude", fontsize=24)
street_map.plot(ax=ax, alpha=0.4, color="#DCDCDC")
geo_df.plot(ax=ax, markersize=20, c = df_accidents_final.oci, cmap='RdYlBu')

norm = colors.Normalize(vmin=df_accidents_final.oci.min(),␣
 ↪vmax=df_accidents_final.oci.max())
cbar = plt.cm.ScalarMappable(norm=norm, cmap='RdYlBu')
ax.tick_params(labelsize=24)
# # add colorbar
ax_cbar = fig.colorbar(cbar, ax=ax, orientation='horizontal')
ax_cbar.ax.tick_params(labelsize=24)
# add label for the colorbar
ax_cbar.set_label('OCI', fontsize=24)
```



The San Diego Map above shows a series of dots, each representing a car crash. Each dot (accident) is colored with a gradient ranging from Red (worst roads) to Blue (best roads). As the gradient shadow becomes more concentrated, there are more car crashes on that specific road. At first glance of this map, we do not see a large blotches of accidents on poor roads (red dots) thus accidents are not concentrated on poor roads.

[22]:
```
fig, ax = plt.subplots()
ax.set_title("Figure 2. Accidents", fontsize=18)
df_accidents_final.plot(kind='scatter', x='lat', y='oci', c=df_accidents_final.
 ↪oci, cmap='RdYlBu', ax=ax)
```

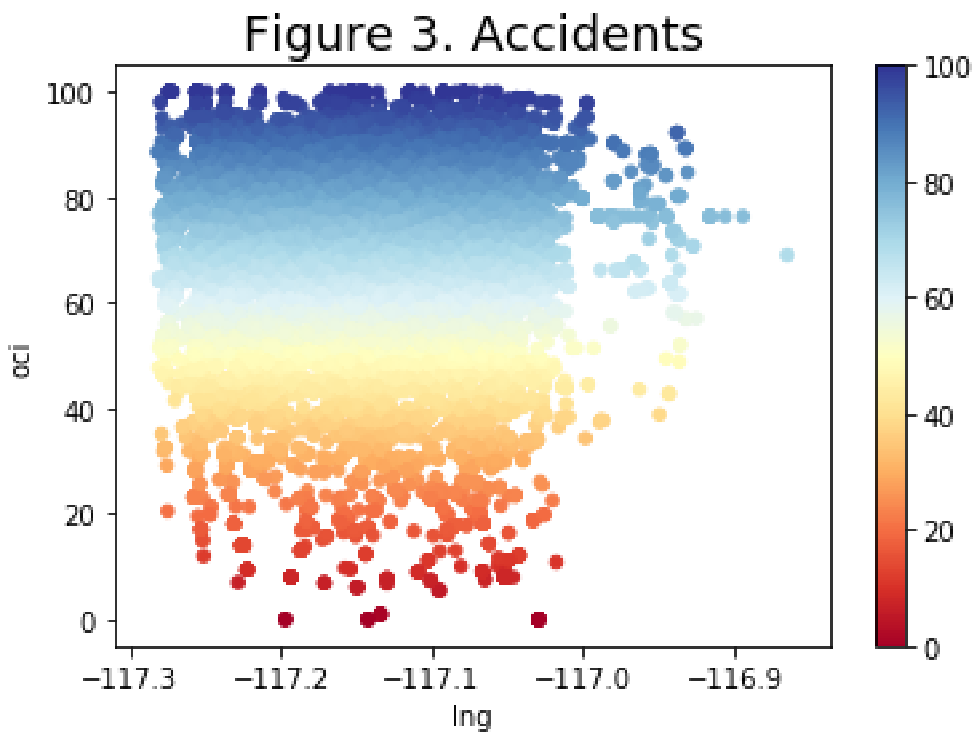
[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfcf721690>
```



The scatter plot above shows all accidents of OCI against latitude. OCI is colored from red to blue representing respecitvely thte worst and best roads. Looking at this plot, we can see that there are more accidents with good roads (high OCI) than poor roads (low OCI) thus concluding that there is a higher chance of having an accident on high OCI than a low one.

[23]:
```
fig, ax = plt.subplots()
ax.set_title("Figure 3. Accidents", fontsize=18)
df_accidents_final.plot(kind='scatter', x='lng', y='oci', c =␣
 ↪df_accidents_final.oci, cmap='RdYlBu', ax=ax)
```

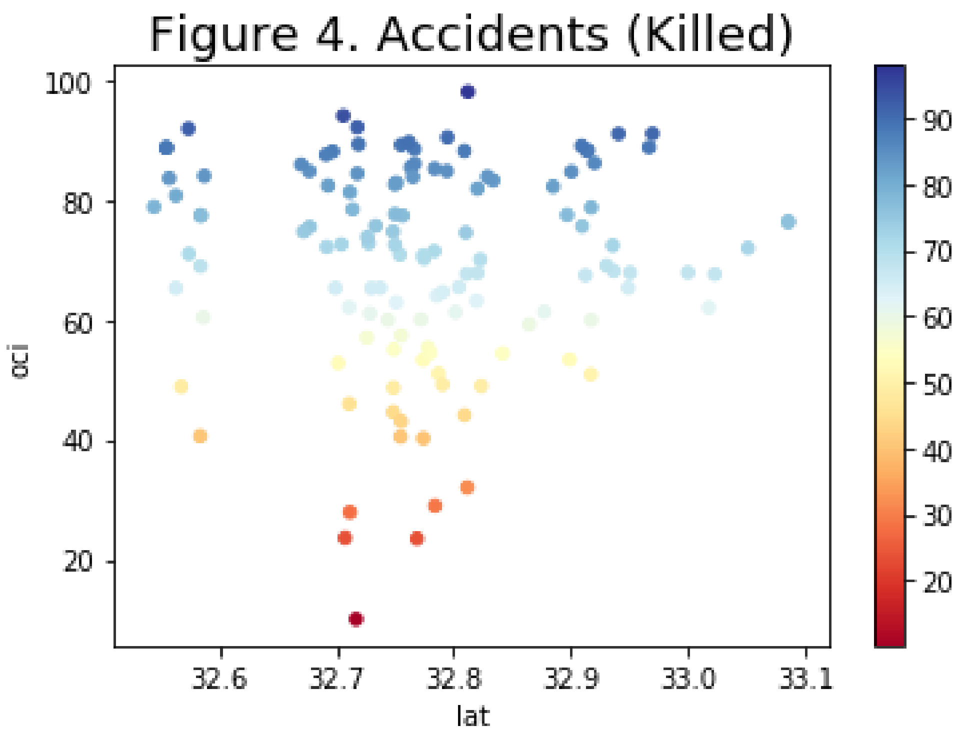
[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcfce9749d0>



The scatter plot above shows all accidents of OCI against longitude. Similar to before, we can conclude that there are more accidents on good roads (high OCI) than poor roads (low OCI).

[24]:
```
fig, ax = plt.subplots()
ax.set_title("Figure 4. Accidents (Killed)", fontsize=18)

fatal_df = df_accidents_final[(df_accidents_final.killed > 0)]
fatal_df.plot(kind='scatter', x='lat', y='oci', c = fatal_df.oci,␣
 ↪cmap='RdYlBu', ax=ax)
```

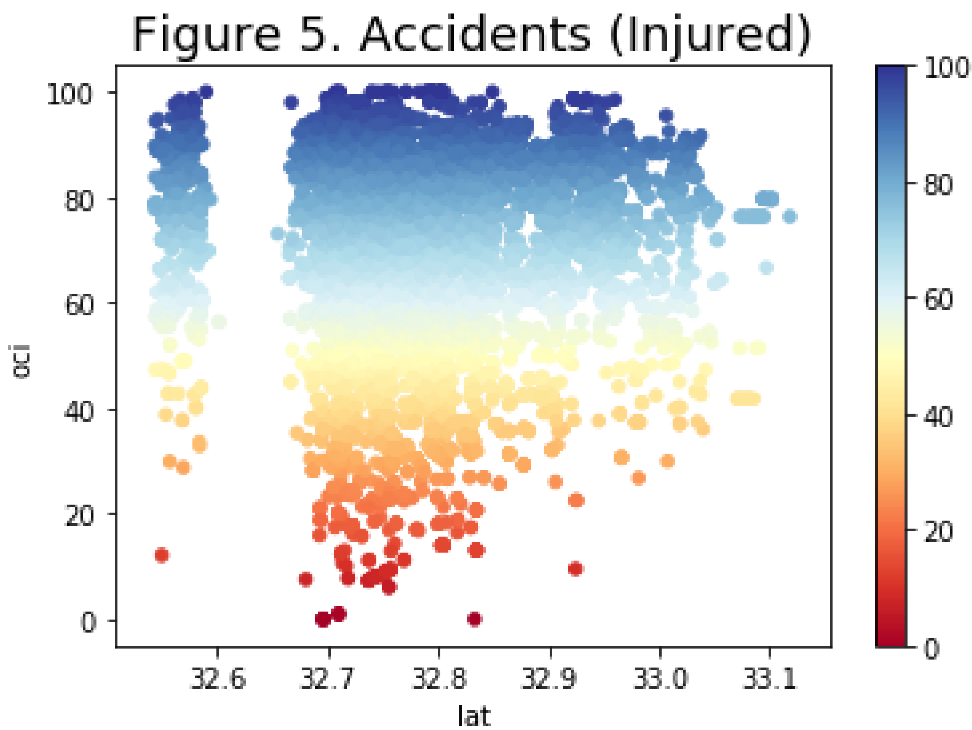
[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcfce8b9a10>



The scatter plot above shows all accidents with fatalities of OCI against latitude. Although the subset of data is small for fataltiies, we can see that there are more fatalities on good roads than poor roads. This is likely because there are more good roads than poor roads meaning there is an equal chance of a fatality happening on any type of road.

[25]:
```
fig, ax = plt.subplots()
ax.set_title("Figure 5. Accidents (Injured)", fontsize=18)

injury_df = df_accidents_final[(df_accidents_final.injured > 0)]
injury_df.plot(kind='scatter', x='lat', y='oci', c = injury_df.oci,␣
↪cmap='RdYlBu', ax=ax)
```

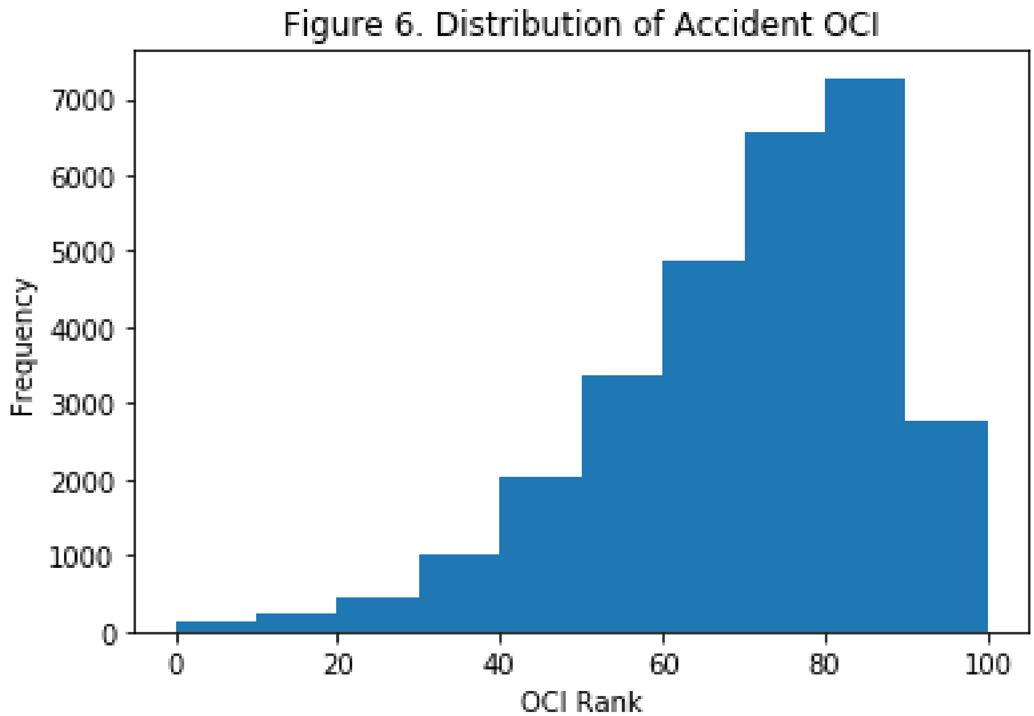
[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcfce882250>



The scatter plot above shows all accidents that resulted in an injury of OCI against latitude. Because there are more good roads than poor roads and there are more accidents on good roads than poor roads, we can conclude that there is an equal chance of an accident with injury occuring on any road.

[26]:
```
# Plotting the distribution of accidents based on OCI
plt.hist(df_accidents_final['oci'])
plt.title('Figure 6. Distribution of Accident OCI')
plt.xlabel('OCI Rank')
plt.ylabel('Frequency')
```

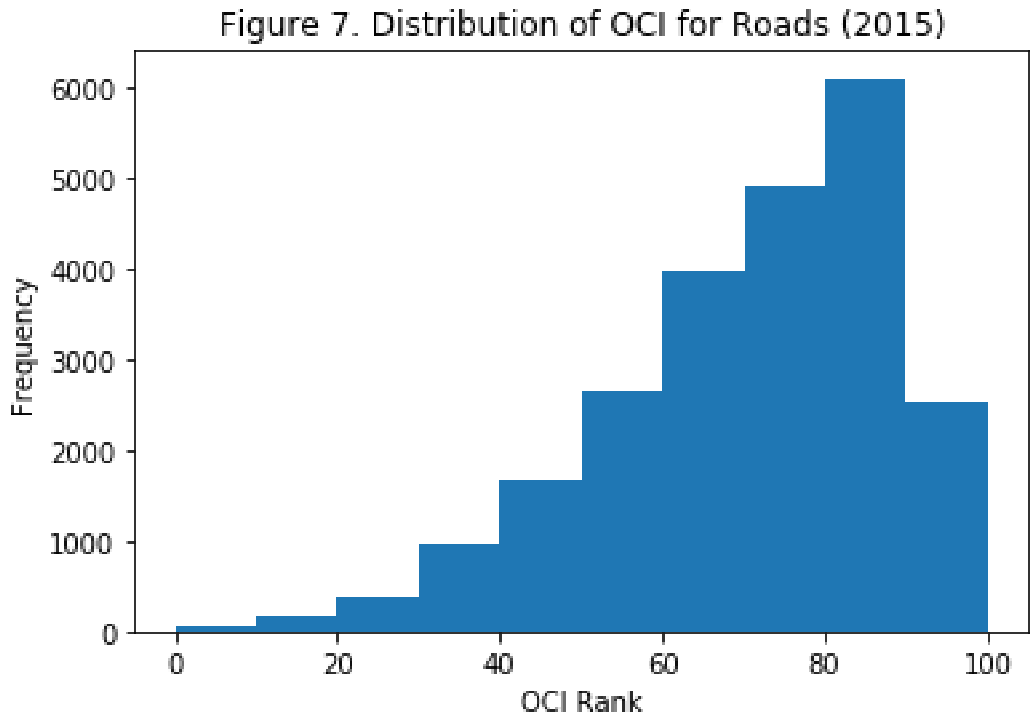
[26]:
```
Text(0, 0.5, 'Frequency')
```



The histogram above represents the frequency of OCIs for accidents. We can see that this is left skewed and the number of accidents increasing as OCI increases. We will compare and contrast this to the next histogram that has the frequency of OCIs for road segments.

[27]:
```
# Plotting the distribution the OCI of the original dataset
plt.hist(df_roads['oci'])
plt.title('Figure 7. Distribution of OCI for Roads (2015)')
plt.xlabel('OCI Rank')
plt.ylabel('Frequency')
```

[27]:
```
Text(0, 0.5, 'Frequency')
```



The histogram above represents the frequency of OCIs of road segments. The histogram of OCIs for road segments is also left skewed and is very similar to the histogram of OCIs for accidents. From this, we can conclude that accidents are equally likely to occur on any type of road because the shape of the two histograms are the same.

[43]:
```
print("Mean for Accidents:", df_accidents_final['oci'].mean())
print("Mean for Road Segments:", df_roads['oci'].mean())

print("The means for the accident and road segments are roughly similar.\
This rejects our initial hypothesis that roads with poor OCI lead to more car␣
 ↪crashes.\
We can see that accidents are equa")
```

```
Mean for Accidents: 70.20951650721865
Mean for Road Segments: 70.5519920665387
The means for the accident and road segments are roughly similar.    This
rejects our initial hypothesis that road segments
```

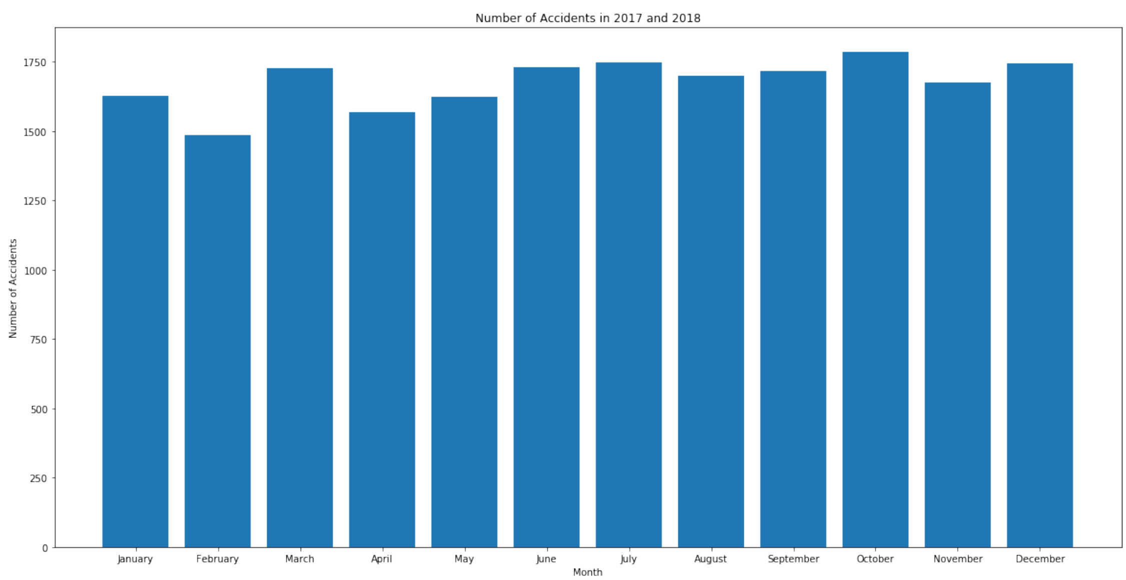
[28]:
```
# Extracting number of accidents in each month of a given year (2017)
months = []
for entry in df_accidents_final.iterrows():
    date_time = entry[1]['date_time']
```

```
    date = date_time.split(' ')[0]
    date_split = date.split('-')
    year = int(date_split[0])
    month = int(date_split[1])
    if year == 2017 or year == 2018:
        months.append(month)
months = pd.Series(months)
months = [months.value_counts()[i] for i in range(1,13)]
months
```

[28]: [1626, 1486, 1725, 1569, 1624, 1730, 1747, 1700, 1718, 1785, 1674, 1745]

[29]:
```
# Plotting number of accidents per month in 2017 and 2018
ax = plt.figure(figsize=(20,10))
plt.bar(x=calendar.month_name[1:13], height=months)
plt.title('Number of Accidents in 2017 and 2018')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
```

[29]: Text(0, 0.5, 'Number of Accidents')

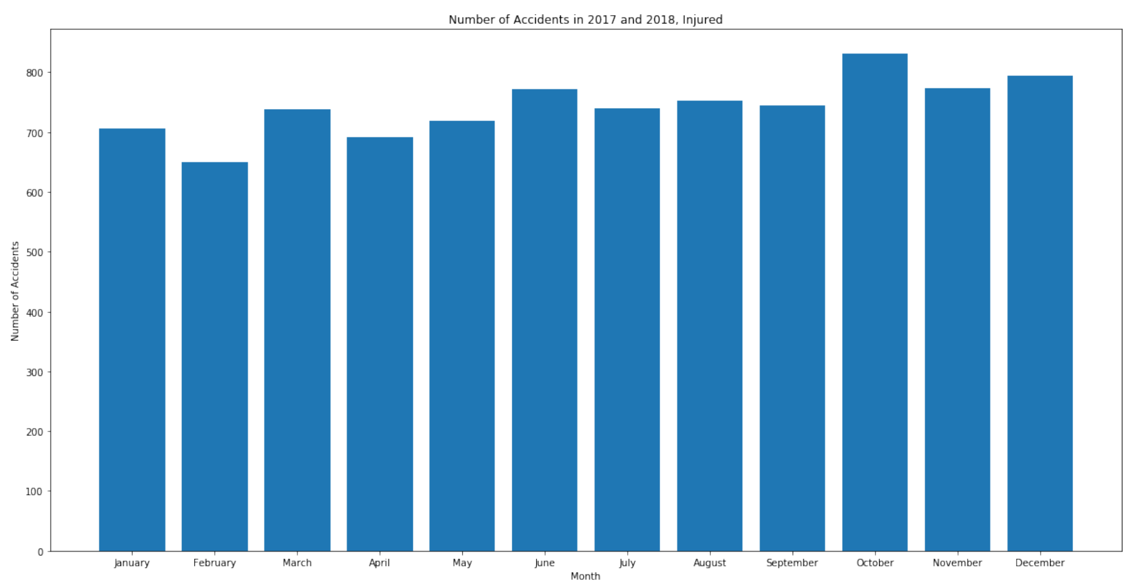


The bar chart above shows the distribution of accidents between months for the years 2017 and 2018. We were looking for a correlation between the raining months and car accidents however we found that accidents are equally likely to occur regardless of the month. We did not include 2019 data because 2019 has an incomplete dataset.

[30]:
```
injured = df_accidents[df_accidents['injured'] > 0]
```

```
months = []
for entry in injured.iterrows():
    date_time = entry[1]['date_time']

    date = date_time.split(' ')[0]
    date_split = date.split('-')
    year = int(date_split[0])
    month = int(date_split[1])
    if year == 2017 or year == 2018:
        months.append(month)
months = pd.Series(months)
months = [months.value_counts()[i] for i in range(1,13)]

ax = plt.figure(figsize=(20,10))
plt.bar(x=calendar.month_name[1:13], height=months)
plt.title('Number of Accidents in 2017 and 2018, Injured')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
```

[30]: Text(0, 0.5, 'Number of Accidents')



The bar chart above shows the distribution of accidents between months for the years 2017 and 2018 given that someone was injured. We had the data for 2019 but we did not include since it is incomplete and 2019 is not finished. The bar chart looks approximately uniform and there is no correlation between month and injury.

[31]:
```
# Looking at the roads with the most accidents
df_most = df_accidents['address_road_primary'].value_counts()
```

```
df_most = df_most.rename_axis("Street_Name").reset_index(name =␣
↪"Number_of_Accidents")
df_most
```

[31]:

```
          Street_Name  Number_of_Accidents
0          UNIVERSITY                  714
1            EL CAJON                  709
2      CLAIREMONT MESA                 380
3              FRIARS                  363
4              GARNET                  362
…                 …                    …
2997    TIERRA GRANDE                    1
2998         LONGFORD                    1
2999      BLOCK S 28TH                   1
3000            WYATT                    1
3001     CARMEL GROVE                    1

[3002 rows x 2 columns]
```

Looking at the dataframe of accident frequency for a given road, we see that roughly 30,000~ of the accidents happened on 3,002 roads in San Diego. The top three roads that we see are University, Clairemont Mesa, and El Cajon. To further solidify our results, our dataset matches closely to another study by Kindley lawyers (https://www.kindleylawyers.com/the-five-most-dangerous-intersections-in-san-diego/). Furthermore, we have a San Diego native discussion of the most popular roads that includes University and El Cajon (https://www.reddit.com/r/sandiego/comments/3zq96a/most_popular_san_diego_street/) supporting a possible conclusion that roads with more traffic are likely to have more accidents.

[35]:

```
counter = 0

def getMean(row):

    street = row[0]
    global counter

    phrase = "(?i)" + street
    df_ocis = df_roads[(df_roads["street"].str.contains(street, case = False,␣
↪regex = False)) |
                        (df_roads["street_from"].str.contains(street, case =␣
↪False, regex = False)) |
                        (df_roads["street_to"].str.contains(street, case =␣
↪False, regex = False))]

    mean = df_ocis["oci"].mean()
    counter += 1
```

```
    return mean
```

[36]:
```
df_most["Avg_OCI"] = df_most.apply(lambda row: getMean(row), axis = 1)
df_most = df_most.dropna()
df_most.head(10)
```

[36]:

	Street_Name	Number_of_Accidents	Avg_OCI
0	UNIVERSITY	714	68.47331
1	EL CAJON	709	68.08060
2	CLAIREMONT MESA	380	67.93088
3	FRIARS	363	73.86050
4	GARNET	362	70.67851
5	BALBOA	362	67.27500
6	GENESEE	347	74.69566
7	IMPERIAL	346	70.35849
8	MIRA MESA	331	79.36714
9	EUCLID	322	66.69812

Looking at the dataframe above, we see the ten streets with the highest number of accidents. We see that a majority of these roads have an high OCI rating that is within the range of fairly good to good. None of these roads are in poor condition.

11 Ethics & Privacy

The OCI and traffic collisions datasets are publicly available and anonymized. Since there is no individual identifier, it is unlikely that there will be any instances where the data could potentially be harmful to us or those included in the dataset.

Although our study could not be utilized to identify individuals involved in the traffic collisions, the data could identify locations in San Diego where poor street conditions resulted in more traffic considerations. If our hypothesis was supported, possible ethical considerations could include taking measures to improve these road conditions in order to prevent future traffic collisions.

12 Conclusion & Discussion

12.1 Limitations:

Data Cleaning:
In terms of data cleaning, we realized that occasionally the google api would give us abnormal results that were still in San Diego. For example, searching up a given intersection may not give us the lat, long pair of the correct intersection, but actually an intersection a few blocks away. Since there were not many of these cases, and since it would be nearly impossible to cover all these cases without looking at every single data point in our dataset, these remain in our dataset.

Additionally, in matching accidents to the nearest road segment, we used an approximation. We took the midpoint of each road segment after figuring out the latitude and longitude pairs that defined the start and end of that road segment, and we used that midpoint to geolocate the road segment. This made running the algorithm to match each accident to a given road segment more reasonable, but it also meant introducing a little bit of inaccuracy since it is a heuristic algorithm.

Lastly, road segments and accidents don't necessarily show up on the same street, meaning we are also approximating the road condition of the accident based on the nearest road, which may be inaccurate since the nearest road that we have a measurement on may be "good", but the accident happened on a really "bad" road.

Location:
Our analysis is only limited to San Diego, and cannot be generalized to locations outside of San Diego. Many factors besides road condition play into accidents, and there may be many more that are introduced when we move to other cities / counties.

12.2 Results:

After looking at our analysis of accidents and the conditions of the roads around them, we came to the conclusion that there is no correlation between lower OCI and higher likelihood for accidents to occur on that given road. In an attempt to find other correlations in our dataset, we decided to look at accidents by month and filtering by accidents that result in injuries, both showing no correlation in either case. Lastly, we decided to look at the roads with the most accidents, and realized that on average the OCIs for those roads were fair / good, further confirming our conclusion that OCI does not correlate with more accidents for a given road. We did realize that the roads with the more accidents encounter a lot of traffic, so it is more likely that larger amounts of traffic on a given road result in more accidents, and that is potentially something to look into in the future.

[ ]:

```

```