Analysis of Tesla Supercharger locations

Final Project

CS X Programming Python

Rashmi Pai

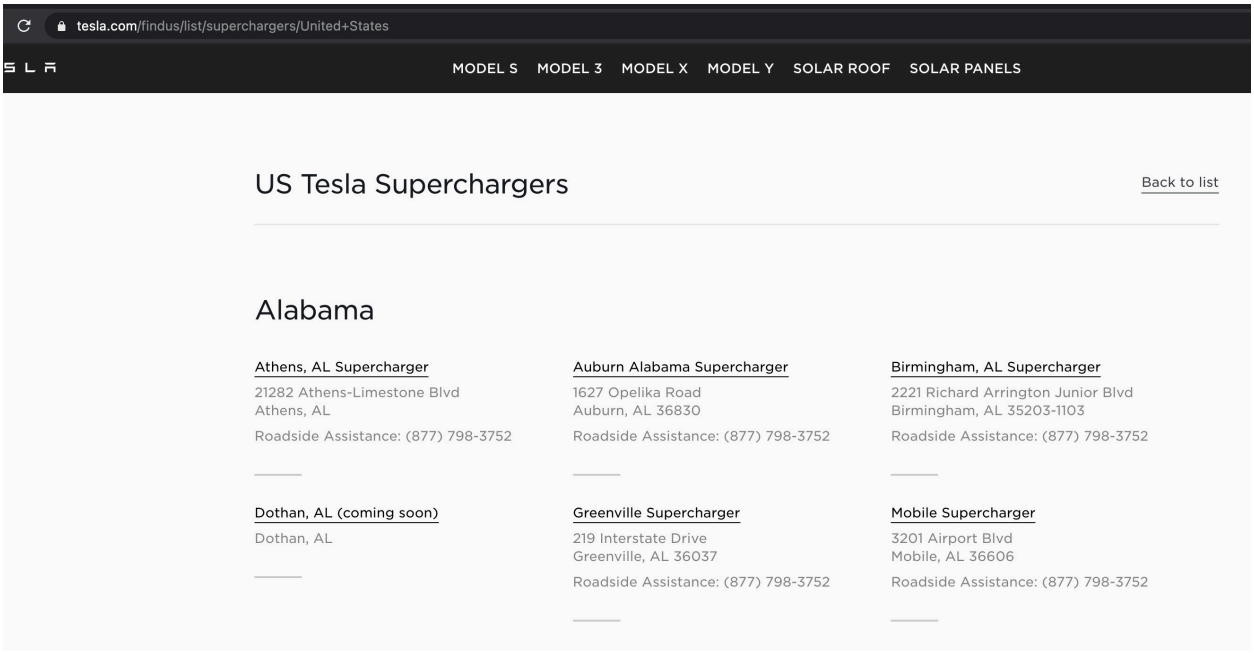
Table of Contents

Data

Source

For my project I considered two types of data sources.

1. HTML data from Tesla's official website



2. json data from supercharge.info (open world community maintaining metadata of Tesla Superchargers)

Limitations

There were couple of limitations of the dataset I'm working with. They are listed below.

- *Incomplete data:* Particularly when dealing with HTML parsing of data, I found many datasets which were incomplete. For the scope of this project I'd enough data to analyze so I'd the leverage to leave out such datasets
- *Duplicate data:* Both the above data sources have some data which is duplicated

Range

For the scope of my analysis, I've limited the range of the data source to only include United States.

Analysis

Scope of Analysis

Tesla is becoming not only the fastest selling car company in EV segment but fastest selling across all segments. Due to the rise in the popularity of its fleet, Tesla is also expanding its Supercharger network. As noted on their website, they plan to install even more superchargers as they move forward to establish a true network on interconnected highways with conveniently placed superchargers to make long distance EV commutes hassle free.

The reason I picked this data to analyze was because I'm intrigued by how Tesla chooses a spot for installing its supercharger stations. I also want to see how many superchargers are located in each state as well as how many superchargers are installed every year.

In addition to these insights, I want to find some deeper insights into the data like which month of the year Tesla opens more of these superchargers and whats the frequency distribution of the superchargers by the number of stalls (stations) each one has onsite.

Tier-1 Analysis

As part of some basic analysis, I've tried to calculate the number of entries in the original data source, number of entries in the cleansed dataset after eliminating all duplicates and leaving out all incomplete datasets.

	Value
Total number of entries in original HTML data	878
Total number of entries in Sqlite table for HTML data after cleansing	836
Total number of entries in original json data	888
Total number of entries in Sqlite table for json after data cleansing	844

TOTAL NUMBER OF ROWS AFTER IMPORTING CLEAN DATA INTO SQLITE TABLES

```
c.execute('''
select count(*) from superchargers_html
''')
rows = c.fetchall()
for row in rows:
    print(row)
```

```
(836,)
```

```
c.execute('''
select count(*) from superchargers_json''')
rows = c.fetchall()
for row in rows:
    print(row)
```

```
(844,)
```

TOTAL NUMBER OF ENTRIES IN JSON DATA

```
#get the data from another source, filtering only for country=USA
# https://supercharge.info/about
base_url = 'https://supercharge.info/service/supercharge/allSites'
superchargers=[]
plain_json = requests.get(base_url).json()
for i in plain_json:
    if i['address']['country']=='USA':
        superchargers.append(i)
print('look at first object within json list:\n')
# lets look at total number of superchargers
print(f'\ntotal number of superchargers from json data: \
{len(superchargers)}')

# lets look at total number of superchargers that are currently
# open using a filter function
print(f'\nnumber of open superchargers from json data: \
{len(list(filter(lambda x: "OPEN" in x["status"], superchargers)))}')
pp.pprint(superchargers[0])
```

```
look at first object within json list:


total number of superchargers from json data: 1032

number of open superchargers from json data: 888
{'address': {'city': 'San Juan Capistrano',
             'country': 'USA',
             'countryId': 100,
             'region': 'North America',
             'regionId': 100,
             'state': 'CA',
             'street': '31971 Camino Capistrano',
             'zip': '92675'},
 'battery': False,
 'counted': True,
 'dateOpened': '2014-05-06',
 'elevationMeters': 30,
 'gps': {'latitude': 33.498458, 'longitude': -117.6632},
 'id': 122,
 'locationId': 'sanjuancapistranosupercharger',
 'name': 'San Juan Capistrano, CA',
 'powerKilowatt': 0,
 'solarCanopy': False,
 'stallCount': 7,
 'status': 'OPEN',
 'statusDays': 0,
 'urlDiscuss': True}
```

TOTAL NUMBER OF ENTRIES IN PARSED HTML DATA

In [5]:
```
coming_soon=[]
already_open=[]

# a function that maps html data into SuperchargerData objects
def extract_data_from_vcard(vcard):
    name=vcard.find('a').text
    locality=vcard.find('span', {'class': 'locality'}).text
    splits = locality.split(',')
#     print(f"splits: {splits}")
    if len(splits)<2:
        return None
    city=splits[0]
    szplit=splits[1].split(' ')
    if len(szplit)<3:
        return None
    state=szplit[1]
    zipcode=szplit[2]
    streetAddress=vcard.find('span', {'class': 'street-address'}).text
    return SuperchargerData(name,city,state,zipcode,streetAddress)

for state in states:
    for vcard in state.find_all('address', {'class': 'vcard'}):
        try:
            if not '(coming soon)' in vcard.text:
                data=extract_data_from_vcard(vcard)
                if data is not None:
                    already_open.append(data)
            else:
                data=extract_data_from_vcard(vcard)
                if data is not None:
                    coming_soon.append(data)
        except:
            pass
#print(f'already_open: {already_open}')
# print(f'coming_soon: {coming_soon}')
# some basic tier 1 analysis
print(len(already_open))
list(map(lambda x:str(x.name),filter(lambda x:x.state == 'CA',already_open)))
```

```
878
```

Tier 2 analysis

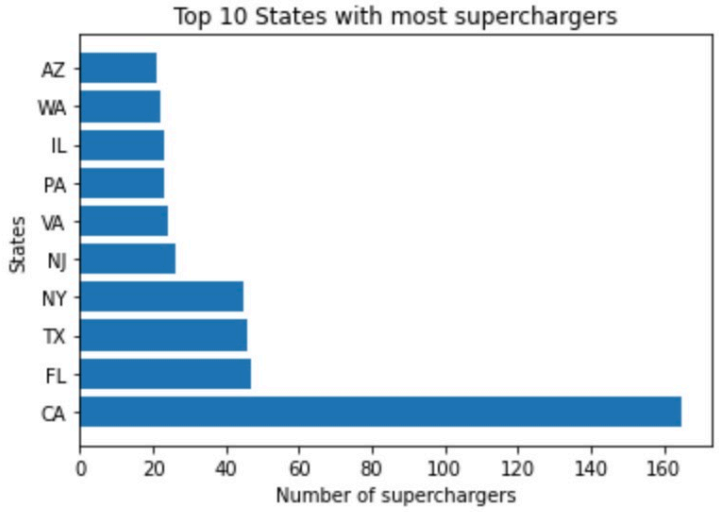
For Tier-2 analysis, I tried to analyze

1. *Top 10 states with most number of superchargers* — I ran a group by aggregation on state and ordered the results based on count in descending order finally limiting the results to top 10
2. *Total number of superchargers opened every year* — For this I used a strftime date function of sqlite to extract year from the dateOpened column and did a group by aggregation on it to get the result on it.
3. *Distribution of superchargers installed by month* — The query was similar to above, instead of extracting year, I extracted the month as part of the subquery and then ran a sum aggregation on the result of the inner query.

1. TOP 10 STATES WITH MOST NUMBER OF SUPERCHARGERS

The state of California has the highest number of superchargers, almost more than sum of the next three or four states on the list of top 10 states with supercharger sites.

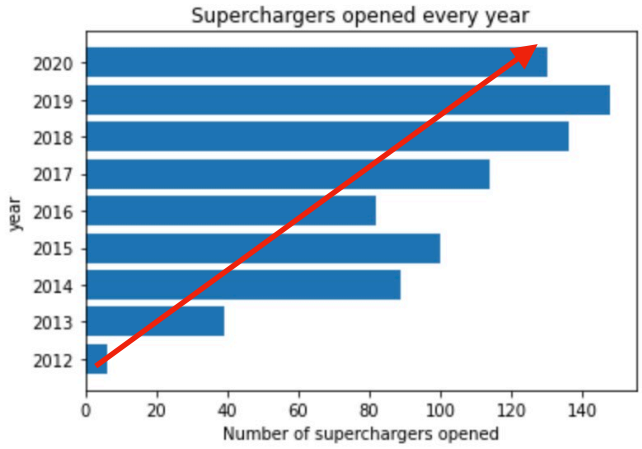
```
# top 10 states with most superchargers
res = execute_query('''
select count(*) as c, state from superchargers_html
group by state order by c desc limit 10''', db)
data=[]
for row in res:
    cf={}
    cf['count']=row[0]
    cf['state']=row[1]
    data.append(cf)
# https://matplotlib.org/3.2.1/gallery/lines_bars_and_markers/barh.html
x=[i['state'] for i in data]
x_pos = [i for i, _ in enumerate(x)]
y = [i['count'] for i in data]
plt.barh(x_pos, y)
plt.ylabel("States")
plt.xlabel("Number of superchargers")
plt.title("Top 10 States with most superchargers")
plt.yticks(x_pos, x)
plt.show()
```



2. TOTAL NUMBER OF SUPERCHARGERS OPENED EVERY YEAR

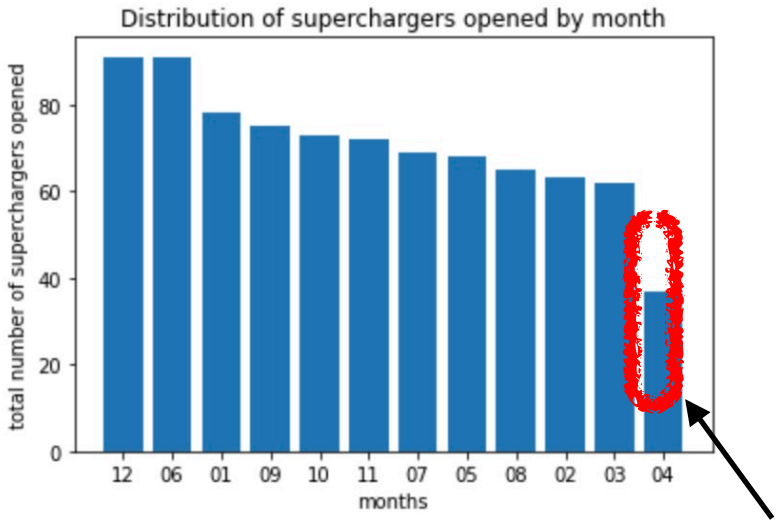
Since the first superchargers opened in 2012, Tesla has continuously made an effort to open more superchargers every year thereafter with the exception of 2016.

```
# use helper function to run a query that gives number of superchargers opened every yea
res = execute_query('''
select count(*), strftime('%Y',dateOpened) as year
from superchargers_json
group by year''',db)
data=[]
for row in res:
    cf={}
    cf['count']=row[0]
    cf['year']=row[1]
    data.append(cf)
x=[i['year'] for i in data]
x_pos = [i for i, _ in enumerate(x)]
facilities = [i['count'] for i in data]
plt.barh(x_pos, facilities)
plt.ylabel("year")
plt.xlabel("Number of superchargers opened")
plt.title("Superchargers opened every year")
plt.yticks(x_pos, x)
plt.show()
```



3. *DISTRIBUTION OF SUPERCHARGERS INSTALLED BY MONTH*

```
res = execute_query('''
select mo,sum(c) as total
from (select count(*) as c, strftime('%m',dateOpened) as mo
        from superchargers_json
        group by mo) as subquery
group by mo
order by total desc''',db)
data = [x[1] for x in res]
months = [x[0] for x in res]
plt.bar(months, data)
plt.xlabel('months')
plt.ylabel('total number of superchargers opened')
plt.title('Distribution of superchargers opened by month')
plt.show()
```



April is not a popular month on Tesla Supercharger opening calendar

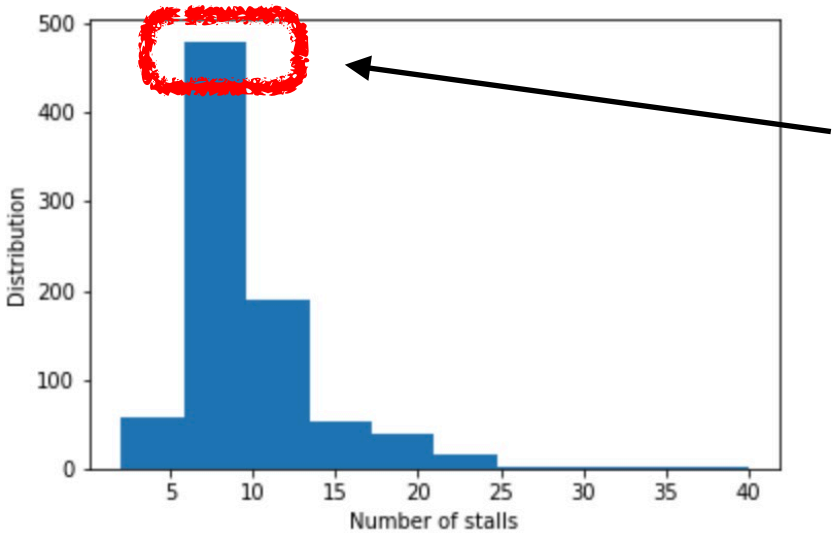
Tier 3 analysis

For Tier 3 analysis, I mainly tried —

1. Find the frequency distribution of total number superchargers by the number of stalls each supercharger site has available.
2. State with minimum number of superchargers
3. For every year, which state ranked the highest for the total number of superchargers sites opened
4. Find the city with the maximum number of superchargers by zipcode using self-joins

1. FREQUENCY DISTRIBUTION OF TOTAL NUMBER OF SUPERCHARGERS VS NUMBER OF STALLS AT EACH SUPERCHARGER

```
# ordered list of superchargers by most number of stalls available
res=execute_query('''
select stallCount,name,state
from superchargers_json
group by name,state order by stallCount desc
''',db)
stalls=[x[0] for x in res]
# https://stackoverflow.com/questions/33203645/how-to-plot-a-histogram-
import matplotlib.pyplot as plt
import numpy as np
plt.hist(stalls, density=False)  # `density=False` would make counts
plt.ylabel('Distribution')
plt.xlabel('Number of stalls');
plt.show()
```



Most superchargers have <10 stalls

2. STATE WITH MINIMUM NUMBER OF SUPERCHARGERS

```
#time to run some tier 2 analysis/queries!
# state with minimum superchargers

res=execute_query('''
select min(num_sc), state
from (select count(*) as num_sc,state from superchargers_html group by state) as subquery
''',db)

print(res)
```

```
[(1, 'DC')]
```

3. FOR EVERY YEAR, WHICH STATE HAS MOST NUMBER OF SUPERCHARGER SITES OPENED

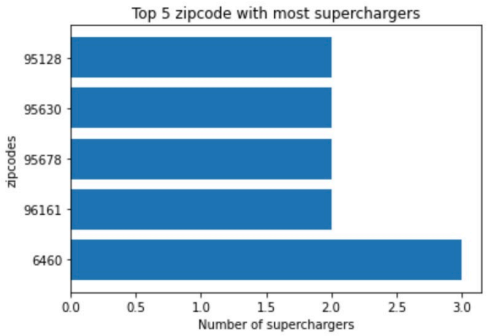
As can be seen, Tesla has been heavily investing in installing their supercharger infrastructure in the state of California more than any other state as it consistently tops each year in terms of number of supercharging sites opened.

```
# run a query to get states with max number of superchargers opened every year
res = execute_query('''
select year,max(c),state
from (select count(*) as c, strftime('%Y',dateOpened) as year, state
        from superchargers_json
        group by year, state) as subquery
group by year''',db)
from prettytable import PrettyTable
t = PrettyTable(['Year', 'superchargers opened', 'state'])
for r in res:
    t.add_row(list(r))
print(t)
```

```
+------+----------------------+-------+
| Year | superchargers opened | state |
+------+----------------------+-------+
| 2012 |          5           |   CA  |
| 2013 |          7           |   CA  |
| 2014 |          9           |   AZ  |
| 2015 |          16          |   CA  |
| 2016 |          16          |   CA  |
| 2017 |          11          |   TX  |
| 2018 |          31          |   CA  |
| 2019 |          46          |   CA  |
| 2020 |          27          |   CA  |
+------+----------------------+-------+
```

4. FIND THE CITY WITH THE MAXIMUM NUMBER OF SUPERCHARGERS BY ZIPCODE USING JOINS

```
# top 10 zipcode with most superchargers
res = execute_query('''
select count(*) as c, zipcode from superchargers_html
group by zipcode
order by c desc limit 5''', db)
data=[]
for row in res:
    cf={}
    cf['count']=row[0]
    cf['zipcode']=row[1]
    data.append(cf)
x=[i['zipcode'] for i in data]
x_pos = [i for i, _ in enumerate(x)]
facilities = [i['count'] for i in data]
plt.barh(x_pos, facilities)
plt.ylabel("zipcodes")
plt.xlabel("Number of superchargers")
plt.title("Top 5 zipcode with most superchargers")
plt.yticks(x_pos, x)
plt.show()
# print the supercharger location info with maximum superchargers by zipcode
res=execute_query('''
select name,streetAddress,city,state,zipcode from superchargers_html join
(select count(*) as total, zipcode as max_z from superchargers_html
group by zipcode order by total desc limit 1) as subqueryB
where zipcode=max_z
''',db)
print(res)
```




```
[('Milford - Boston Post Road Supercharger', '1201 Boston Post Road', 'Milford', 'CT', 6460), ('Milford, CT North Sup
ercharger', 'Milford Travel Plaza', 'Milford', 'CT', 6460), ('Milford, CT South Supercharger', 'Travel Plaza', 'Milfo
rd', 'CT', 6460)]
```

Future enhancements

- We could join this data with Foursquare/Yelp data to find out POI (Places of interest) near SuperChargers
- Future locations for chargers based on current usage if we had live data
- Cost to charge at a supercharger based on power tariffs (would make for interesting data)