Exploratory Data Analysis and Visualizations

Exploring and Predicting EV Registration Data

Daniel Gieseke | August 2023

The electric vehicle (EV) industry is booming at every level, and companies in the industry are aiming to understand the different factors that affect EV adoption, and the geographic spread of EVs such that they can adequately prepare and market themselves. This project aims to take in real world data on EV registrations by zip code along with several socioeconomic factors to find respective variance and correlation, and ultimately build a machine learning model that can predict EV registration by zip code; with these models we aim to understand the factors that affect EV adoption.

About this notebook

This notebook contains visualizations and some exploratory data analysis (EDA) for this project. For information on the data cleaning, joining, and feature engineering, please see my "Data Preparation" notebook and for the models created in conjunction with this project, please visit my "Models" notebook.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set_style("darkgrid", {"font.family": "Helvetica"})
import matplotlib.pyplot as plt
from uszipcode import SearchEngine
import plotly.figure_factory as ff
from sklearn.manifold import TSNE
import mpu
import zipcodes
import folium
import json
import addfips
import warnings
warnings.filterwarnings("ignore")
```

```
/Users/danielgieseke/anaconda3/envs/learn-env/lib/python3.8/site-packages/fuzzywuzzy/fuzz.py:11: UserWa
rning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warn
ing')
```

Importing EV Registration Data and Performing Light Cleaning

In [2]:

```
# Importing Data
df = pd.read_csv('Cleaned_EV_Reg_Data')
df.head()
```

Out[2]:

	Unnamed: 0	ZIP Code	Vehicle Count	NY	NJ	CT	Median Family Household Income	Population	Percent Pop. Bachelors Deg. - 35-44	Total Pop. Bachelors Deg. - 35-44	...	Percent Pop. Age 25-29	Percent Pop. Age 30-34	Pe 3
0	0	6001	351	0.0	0.0	1.0	166235.0	19262.0	80.8	2069.0	...	2.8	4.4	
1	1	6002	126	0.0	0.0	1.0	97917.0	21579.0	37.0	907.0	...	5.0	7.4	
2	2	6010	200	0.0	0.0	1.0	91605.0	60748.0	39.7	3228.0	...	6.7	6.3	
3	3	6013	99	0.0	0.0	1.0	156442.0	9519.0	60.4	890.0	...	4.0	4.3	
4	4	6016	29	0.0	0.0	1.0	118345.0	6273.0	57.5	385.0	...	8.1	9.0	

5 rows × 27 columns

In [3]:

```
# Dropping Unneeded Row
df = df.drop(columns='Unnamed: 0')

# Converting ZIP Column to STR
df['ZIP Code'] = df['ZIP Code'].astype(str)
```

In [4]:

```
# Cleaning ZIP Column
dirty_zips = df['ZIP Code']
clean_zips = []

for code in dirty_zips:
    if len(code) < 5:
        new_code = '0' + code
        clean_zips.append(new_code)
    else:
        clean_zips.append(code)

df['ZIP Code'] = clean_zips
```

In [5]:

```
# Adding State Column
conditions = [
    df['NY'] == 1,
    df['NJ'] == 1,
    df['CT'] == 1,
]
```

In [6]:

```
outputs = [
    'New York', 'New Jersey', 'Connecticut'
]
```

In [7]:

```
df['State'] = np.select(conditions, outputs)
```

In [8]:

```
# Adding FIPS Column to df
# Dataframe for FIPS (gets all unique combinations)
fips = pd.DataFrame(df.groupby(['State', 'County']).size().reset_index().iloc[:,[0,1]])

# Add FIPS code to fips df
af = addfips.AddFIPS()
for index, row in fips.iterrows():
    fips.at[index,'FIPS']=af.get_county_fips(fips.at[index,'County'],fips.at[index,'State'])

# Add FIPS to data
df_maps = fips.merge(df, how='inner', on=['County', 'State'])
```

In [9]:

```
df_maps.to_csv('EV_Reg_Maps_Data.csv')
```

In [10]:

```
# Creating "TIME" df by Joining 2022 data
df_2022 = pd.read_csv('Cleaned_EV_Reg_Data_2022')
```

In [11]:

```
# Converting ZIP Column to STR
df_2022['ZIP Code'] = df_2022['ZIP Code'].astype(str)

# Cleaning ZIP Column
dirty_zips = df_2022['ZIP Code']
clean_zips = []

for code in dirty_zips:
    if len(code) < 5:
        new_code = '0' + code
        clean_zips.append(new_code)
    else:
        clean_zips.append(code)

df_2022['ZIP Code'] = clean_zips
```

In [12]:

```
df_time = df_maps.merge(df_2022, on='ZIP Code', how='left')
```

In [13]:

```
# Removing Connecticut
df_time = df_time.dropna()
```

In [14]:

```
df_time
```

Out[14]:

	State	County	FIPS	ZIP Code	Vehicle Count_x	NY	NJ	CT	Median Family Household Income	Population	...	NYC Suburb	Young_Liberal	Ir
288	New Jersey	Atlantic County	34001	08037	131	0.0	1.0	0.0	97126.0	23814.0	...	0	4.7	
289	New Jersey	Atlantic County	34001	08201	45	0.0	1.0	0.0	75114.0	10647.0	...	0	6.2	
290	New Jersey	Atlantic County	34001	08203	67	0.0	1.0	0.0	107138.0	7725.0	...	0	5.9	
291	New Jersey	Atlantic County	34001	08205	173	0.0	1.0	0.0	85737.0	28666.0	...	0	7.8	
292	New Jersey	Atlantic County	34001	08215	71	0.0	1.0	0.0	85625.0	14092.0	...	0	5.8	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3177	New York	Yates County	36123	14507	4	1.0	0.0	0.0	77143.0	1298.0	...	0	0.0	
3178	New York	Yates County	36123	14527	61	1.0	0.0	0.0	77297.0	12716.0	...	0	0.0	
3179	New York	Yates County	36123	14544	13	1.0	0.0	0.0	66292.0	2035.0	...	0	0.0	
3180	New York	Yates County	36123	14837	14	1.0	0.0	0.0	68389.0	5345.0	...	0	0.0	
3181	New York	Yates County	36123	14842	3	1.0	0.0	0.0	63333.0	797.0	...	0	0.0	

2862 rows × 31 columns

In [15]:

```
# Feature Engineering Time Data
df_time['EVs Added in 2022'] = df_time['Vehicle Count_x'] - df_time['Vehicle Count_y']
```

In [16]:

```
df_time['EVs Added in 2022'].median()
```

Out[16]:

```
16.0
```

In [17]:

```
df_time['EV Growth per Capita'] = df_time['EVs per capita_x'] - df_time['EVs per capita_y']
```

In [18]:

```
df_time['EV Growth per Capita'].median()
```

Out[18]:

```
163.97849462365588
```

Maps Illustrating Spread of Data

In [19]:

```
FIPS_VehicleCount = df_maps[['FIPS', 'Vehicle Count']]
```

In [20]:

```
FIPS_VehicleCount = FIPS_VehicleCount.groupby('FIPS').sum()
FIPS_VehicleCount = FIPS_VehicleCount.reset_index()
```

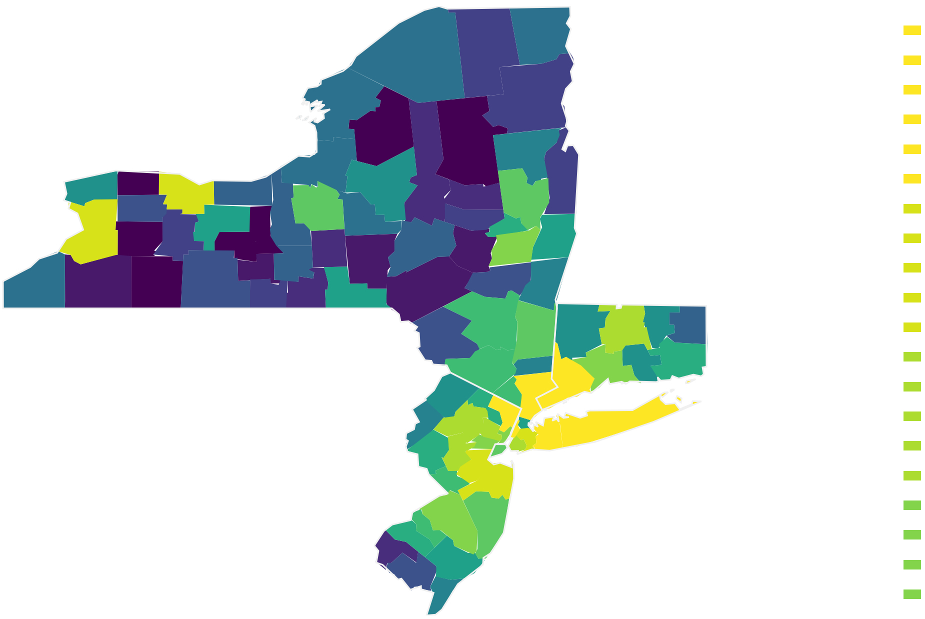
In [21]:

```
fips = FIPS_VehicleCount['FIPS'].astype(str)
```

In [22]:

```
fig = ff.create_choropleth(fips=fips, values=FIPS_VehicleCount['Vehicle Count'],
                           scope=['NY', 'NJ', 'CT'], round_legend_values=True,
                           title='Electric Vehicle (EV) Registration by County (2023)')
fig.layout.template = None
fig.show()
```

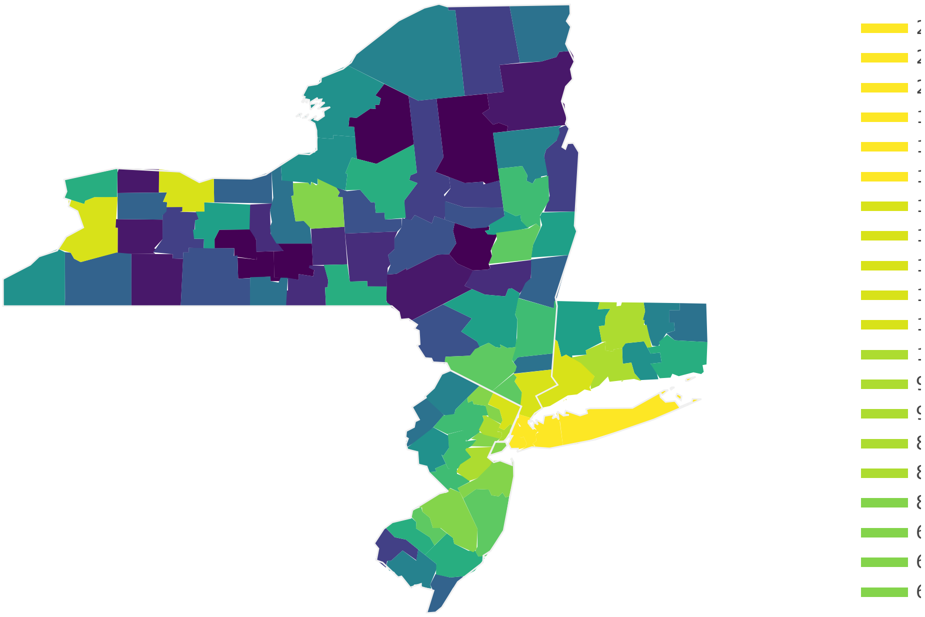
Electric Vehicle (EV) Registration by County (2023)



```
In [23]: FIPS_Population = df_maps[['FIPS', 'Population']]
         FIPS_Population = FIPS_Population.groupby('FIPS').sum()
         FIPS_Population = FIPS_Population.reset_index()
```

```
In [24]: fig = ff.create_choropleth(fips=FIPS_Population['FIPS'], values=FIPS_Population['Population'],
                                    scope=['NY', 'NJ'], round_legend_values=True,
                                    title='Population by County')
         fig.layout.template = None

         fig.show()
```

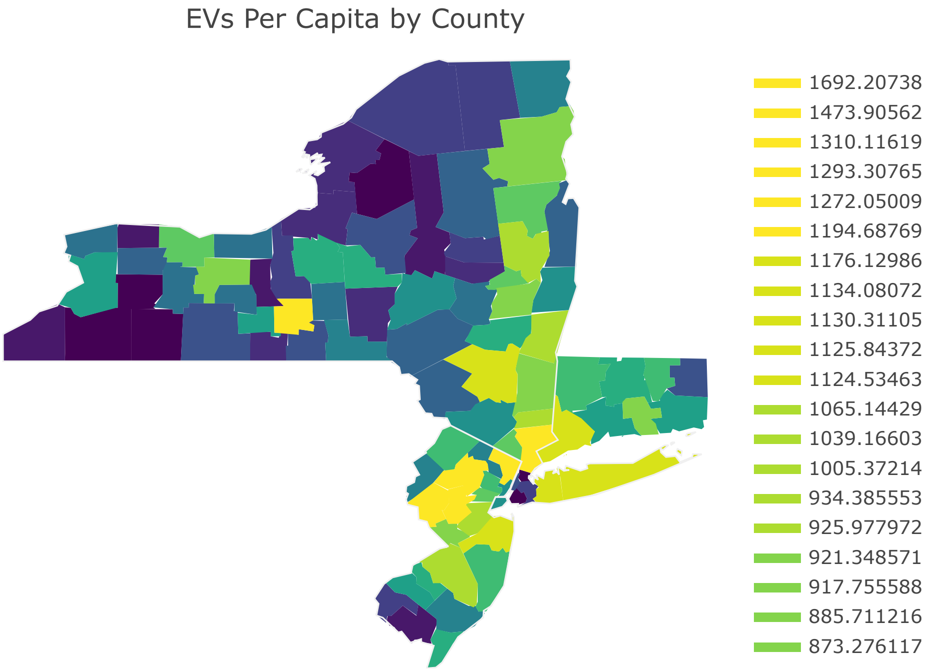
Population by County



```
In [25]: FIPS_EVs_Capita = FIPS_Population.merge(FIPS_VehicleCount, on='FIPS', how='left')
         FIPS_EVs_Capita['EVs Per Capita'] = (FIPS_EVs_Capita['Vehicle Count']/FIPS_EVs_Capita['Population'])*1
```

In [26]:

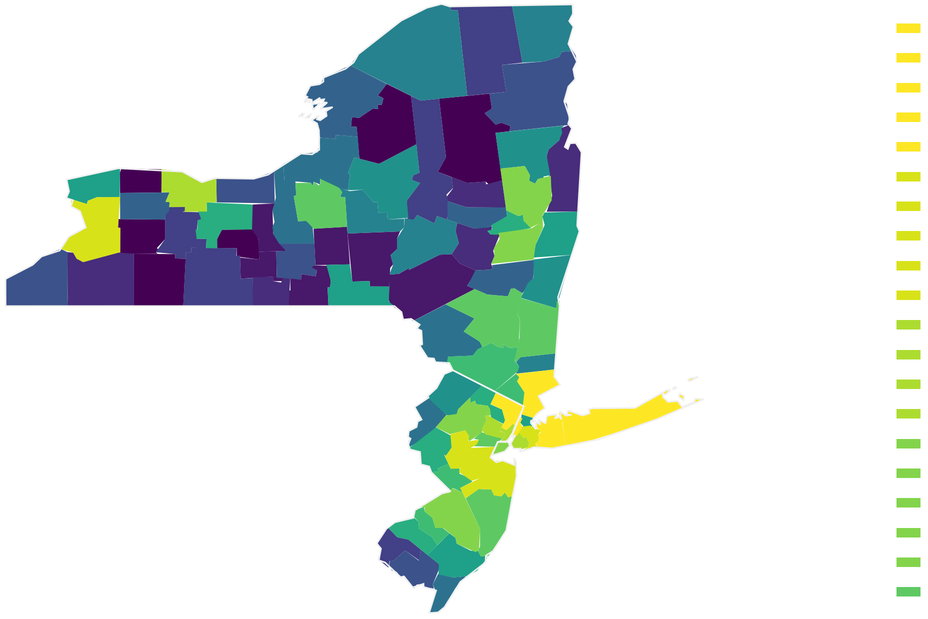
```
fig = ff.create_choropleth(fips=FIPS_EVs_Capita['FIPS'], values=FIPS_EVs_Capita['EVs Per Capita'],
                           scope=['NY', 'NJ'], round_legend_values=True,
                           title='EVs Per Capita by County')
fig.layout.template = None

fig.show()
```



In [27]:

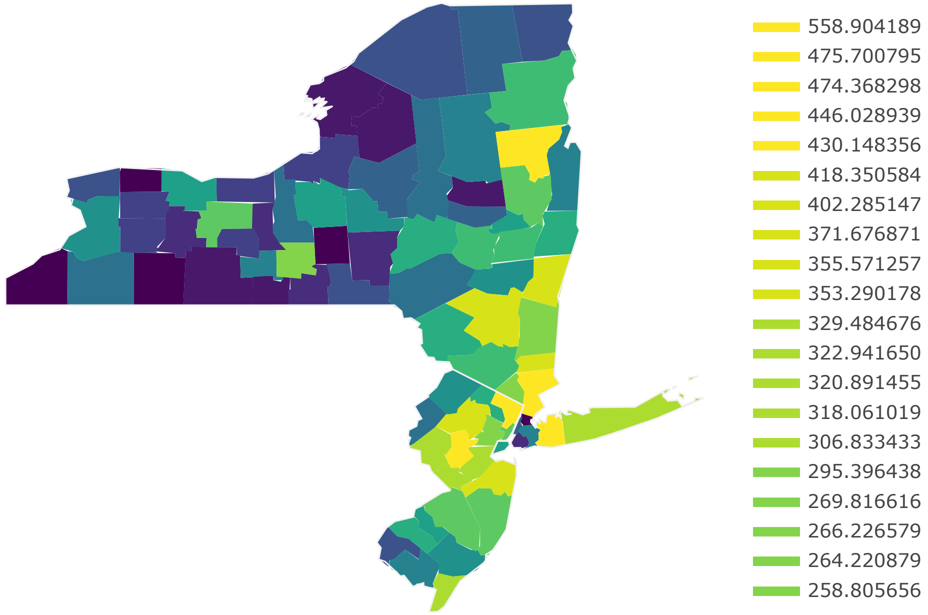
```
FIPS_EV_Growth = df_time[['FIPS', 'EVs Added in 2022']]

FIPS_EV_Growth = FIPS_EV_Growth.groupby('FIPS').sum()
FIPS_EV_Growth = FIPS_EV_Growth.reset_index()

fips = FIPS_EV_Growth['FIPS'].astype(str)

fig = ff.create_choropleth(fips=fips, values=FIPS_EV_Growth['EVs Added in 2022'],
                           scope=['NY', 'NJ'], round_legend_values=True,
                           title='EVs Added from 2022 to 2023 by County')
fig.layout.template = None

fig.show()
```

EVs Added from 2022 to 2023 by County



In [28]:
```
FIPS_EV_Growth = df_time[['FIPS', 'EV Growth per Capita']]

FIPS_EV_Growth = FIPS_EV_Growth.groupby('FIPS').mean()
FIPS_EV_Growth = FIPS_EV_Growth.reset_index()

fips = FIPS_EV_Growth['FIPS'].astype(str)

fig = ff.create_choropleth(fips=fips, values=FIPS_EV_Growth['EV Growth per Capita'],
                           scope=['NY', 'NJ'], round_legend_values=True,
                           title='Change in EVs per Capita from 2022 to 2023)')
fig.layout.template = None

fig.show()
```

Change in EVs per Capita from 2022 to 2023)



Breakdown of EV Ownership by Political Party

```
In [29]: df_pol = df[['voted for joe', 'Vehicle Count', 'Population', 'EV Charging Stations']]
```

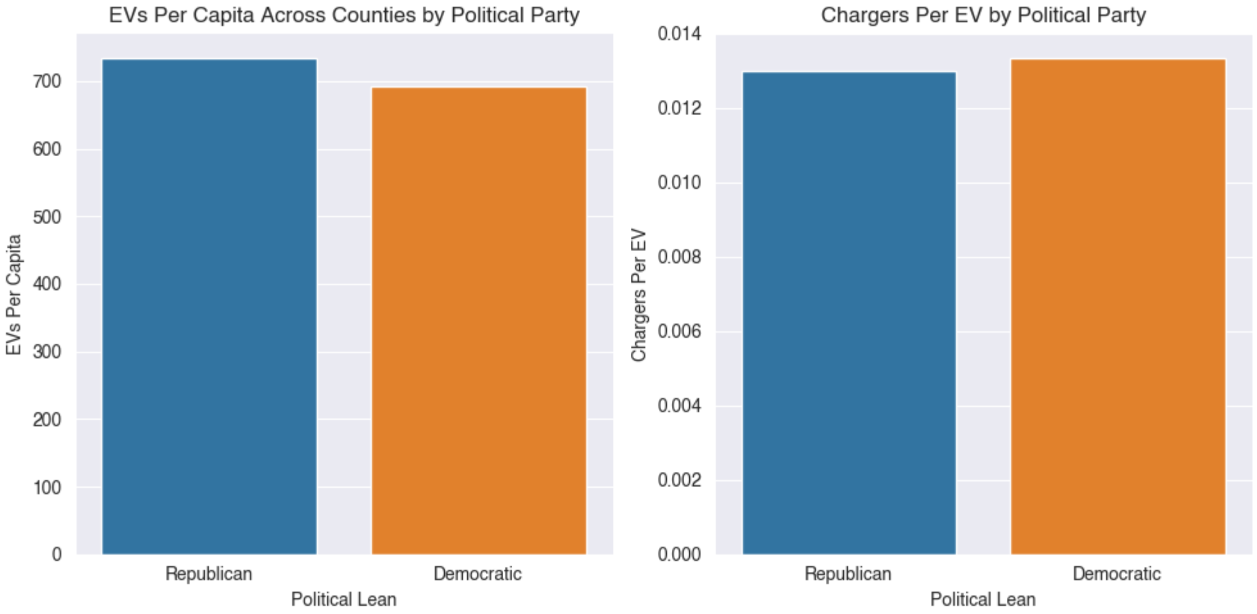
```
In [30]: df_pol = df_pol.groupby('voted for joe').sum()
```

```
In [31]: df_pol['EVs Per Capita'] = (df_pol['Vehicle Count']/df_pol['Population'])*100000
         df_pol = df_pol.reset_index()
         df_pol = df_pol.rename(columns={'voted for joe': 'Political Lean'})
```

```
In [32]: df_pol['Political Lean'] = df_pol['Political Lean'].astype(str)
         df_pol['Political Lean'] = df_pol['Political Lean'].replace({'0.0': 'Republican', '1.0': 'Democratic'}
```

```
In [33]: df_pol['Chargers Per EV'] = (df_pol['EV Charging Stations']/df_pol['Vehicle Count'])
```

```
In [34]: plt.rcParams["figure.figsize"] = [10, 5]
         plt.rcParams["figure.autolayout"] = True
         f, axes = plt.subplots(1, 2)

         sns.barplot(data=df_pol, x='Political Lean',
                     y='EVs Per Capita', ax=axes[0]).set(title=
                                                     'EVs Per Capita Across Counties by Political Party')

         # Create the second plot

         sns.barplot(x="Political Lean", y="Chargers Per EV",
                     data=df_pol, ax=axes[1]).set(title='Chargers Per EV by Political Party')

         # Show the figure
         plt.show()
```



EV Registration over Time by State

```
In [35]: df_ny = pd.read_csv('NY_EV_Registrations.csv')
```

```
In [36]: df_ny = df_ny[['DMV Snapshot (Date)', 'Vehicle Count']]
         df_ny = df_ny.groupby('DMV Snapshot (Date)').sum()
```

```
In [37]: df_ny = df_ny.reset_index()
```

```
In [38]: df_ny['DMV Snapshot (Date)'] = df_ny['DMV Snapshot (Date)'].replace({'DMV Snapshot (1/1/2022)': '2022'
                                                                               'DMV Snapshot (1/12/2021)': '202
                                                                              'DMV Snapshot (1/2/2018)': '2018'
                                                                              'DMV Snapshot (1/2/2019)': '2019'
```

```
                                                                   'DMV Snapshot (1/2/2020)': '2020'
                                                                  'DMV Snapshot (1/3/2023)': '2023'
```

In [39]:
```
df_ny = df_ny.loc[(df_ny['DMV Snapshot (Date)'] == '2018') |
                  (df_ny['DMV Snapshot (Date)'] == '2019') |
                  (df_ny['DMV Snapshot (Date)'] == '2020') |
                  (df_ny['DMV Snapshot (Date)'] == '2021') |
                  (df_ny['DMV Snapshot (Date)'] == '2022') |
                  (df_ny['DMV Snapshot (Date)'] == '2023'), :]
```

In [40]:
```
df_nj = pd.read_csv('NJ_EV_Registrations.csv')
```

In [41]:
```
df_nj = df_nj[['DMV Snapshot (Date)', 'Vehicle Count']]
df_nj = df_nj.groupby('DMV Snapshot (Date)').sum()
```

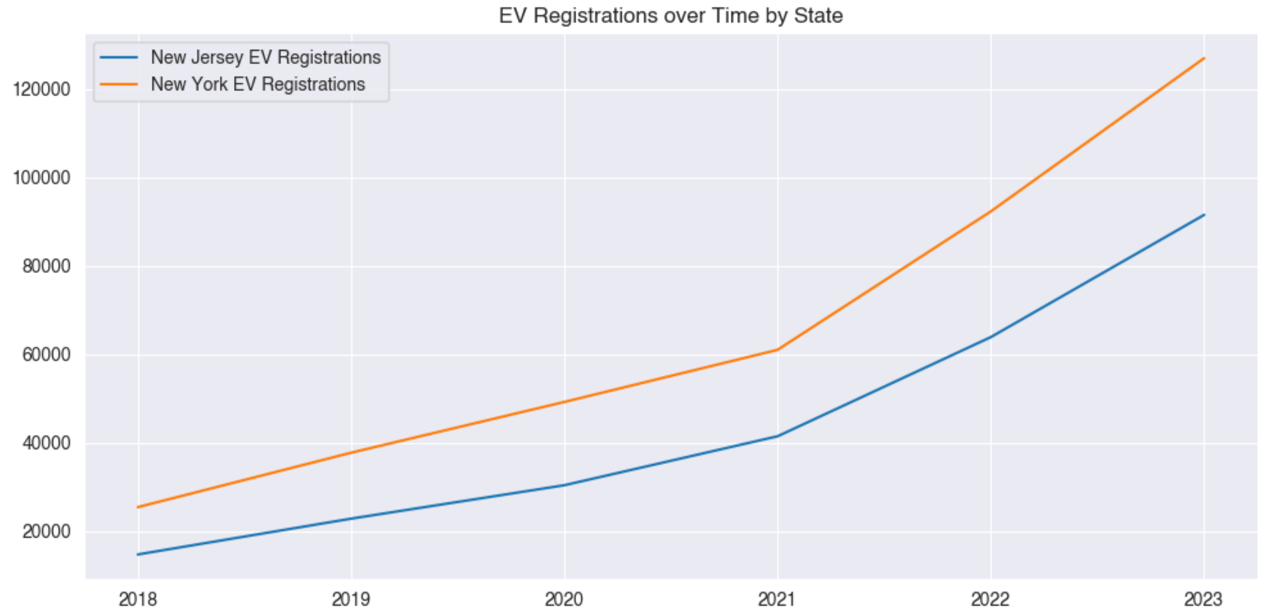
In [42]:
```
df_nj = df_nj.reset_index()
df_nj['DMV Snapshot (Date)'] = df_nj['DMV Snapshot (Date)'].replace({'DMV Snapshot (12/31/2017)': '201
                                                                    'DMV Snapshot (12/31/2021)': '20
                                                                   'DMV Snapshot (12/31/2018)': '201
                                                                   'DMV Snapshot (12/31/2019)': '202
                                                                   'DMV Snapshot (12/31/2020)': '202
                                                                  'DMV Snapshot (12/31/2022)': '2023
```

In [43]:
```
df_nj = df_nj.loc[(df_nj['DMV Snapshot (Date)'] == '2018') |
                  (df_nj['DMV Snapshot (Date)'] == '2019') |
                  (df_nj['DMV Snapshot (Date)'] == '2020') |
                  (df_nj['DMV Snapshot (Date)'] == '2021') |
                  (df_nj['DMV Snapshot (Date)'] == '2022') |
                  (df_nj['DMV Snapshot (Date)'] == '2023'), :]
```

In [44]:
```
df_ny_nj = df_ny.merge(df_nj, on='DMV Snapshot (Date)', how='left')
df_ny_nj['DMV Snapshot (Date)'] = df_ny_nj['DMV Snapshot (Date)'].astype(int)
```

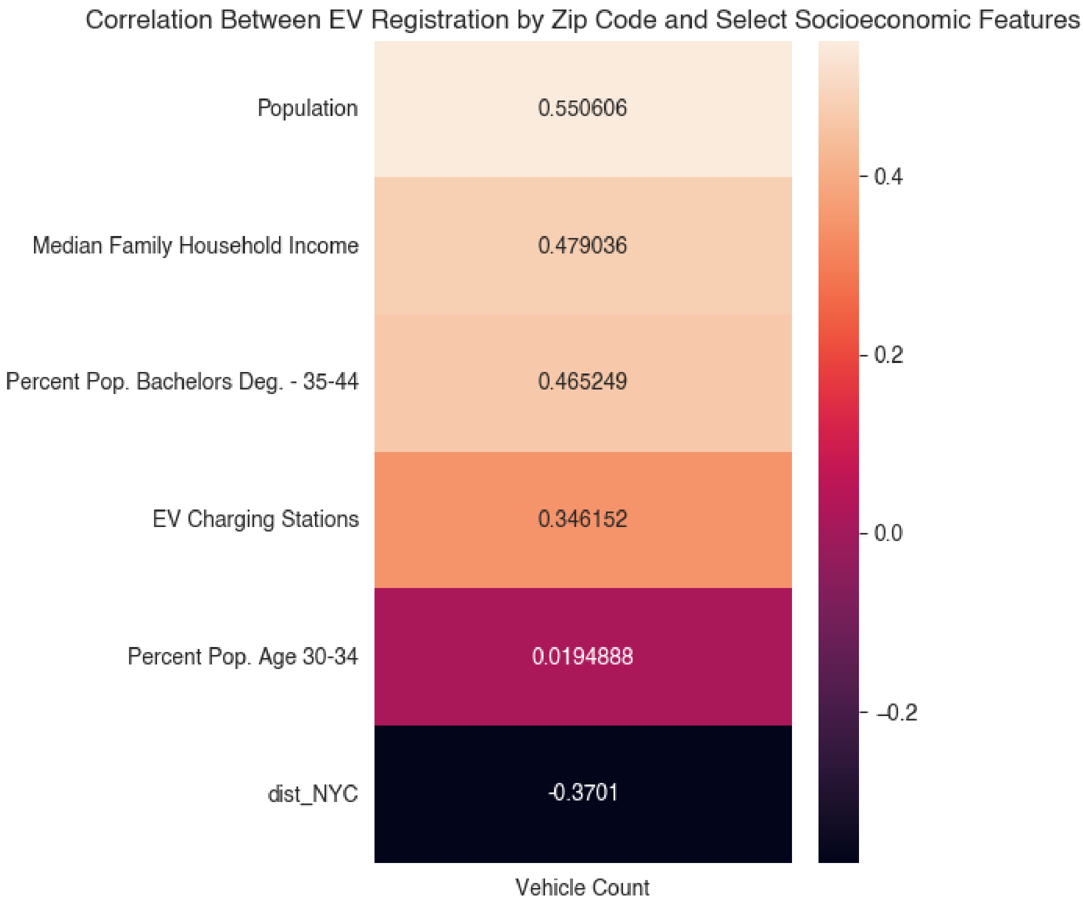
In [45]:
```
df_ny_nj = df_ny_nj.sort_values(by='DMV Snapshot (Date)')
```

In [46]:
```
x = df_ny_nj['DMV Snapshot (Date)']
y1 = df_ny_nj['Vehicle Count_y']
y2 = df_ny_nj['Vehicle Count_x']

plt.plot(x.values, y1.values, label="New Jersey EV Registrations")
plt.plot(x.values, y2.values, label="New York EV Registrations")

plt.title("EV Registrations over Time by State")

plt.legend()

plt.show()
```

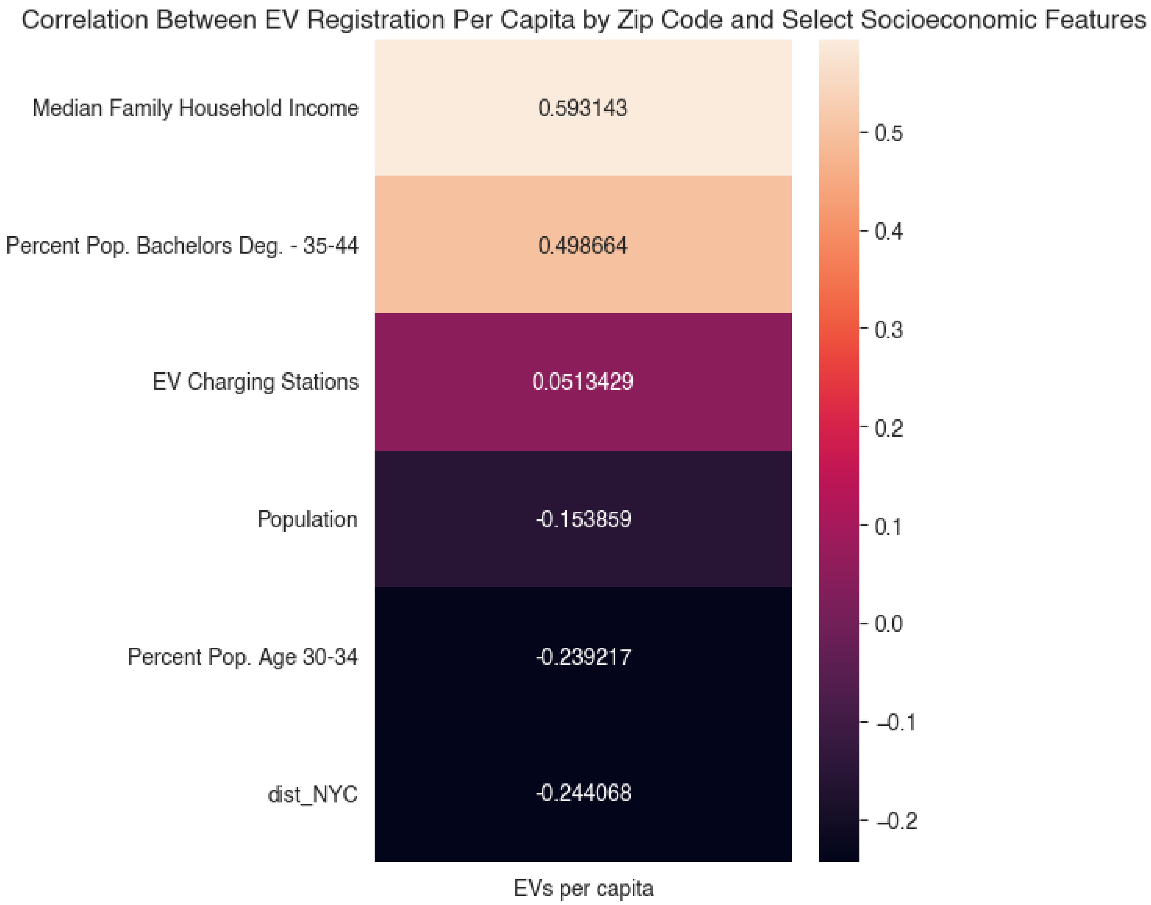


Heatmap of Correlation of Features with EV Registration

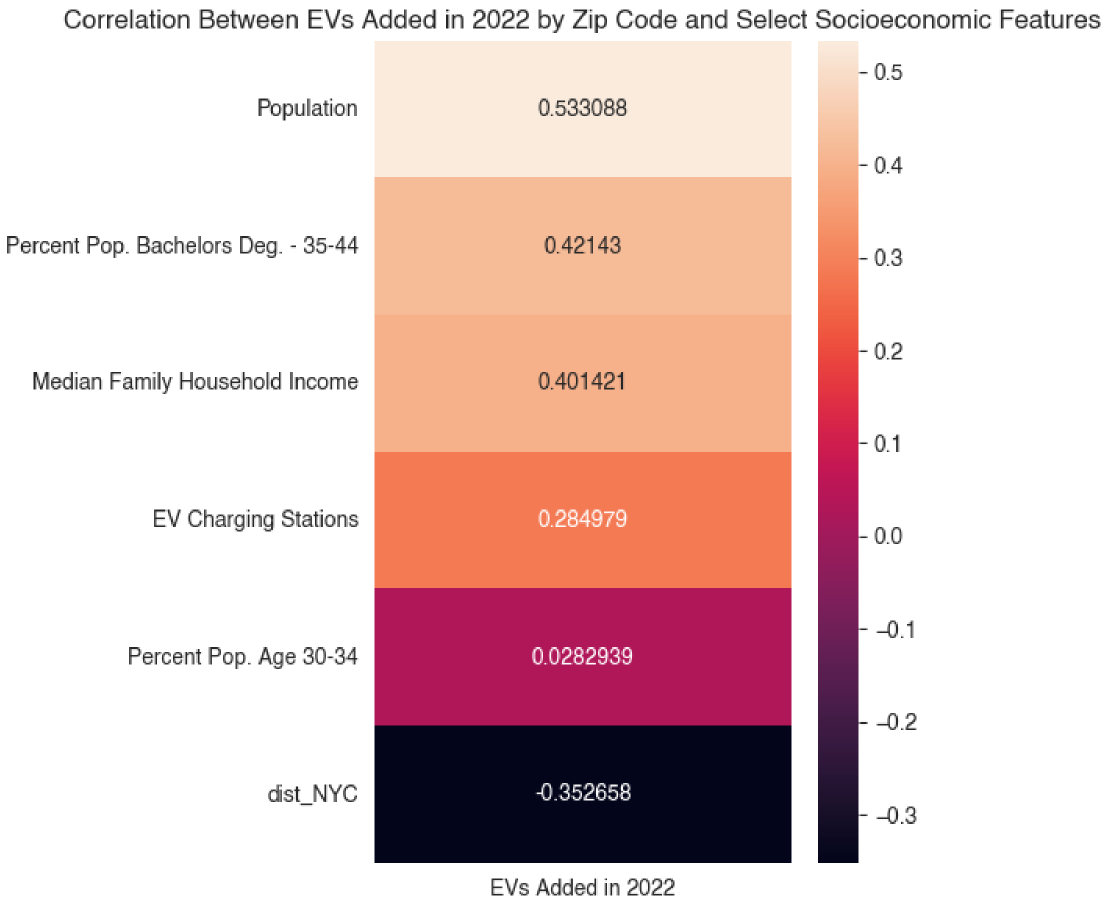
In [47]:
```
col_list = [df['Median Family Household Income'], df['Population'],
            df['EV Charging Stations'],
          df['Percent Pop. Age 30-34'], df['Percent Pop. Bachelors Deg. - 35-44'],
            df['dist_NYC']]

corr_df = df[['Median Family Household Income', 'Population', 'EV Charging Stations',
              'Percent Pop. Age 30-34', 'Percent Pop. Bachelors Deg. - 35-44', 'dist_NYC']]

corr_list = []
col_names = []

for col in col_list:
    corr_list.append(df['Vehicle Count'].corr(col))

for col in corr_df.columns:
    col_names.append(col)

df_heat = pd.DataFrame({"Vehicle Count": corr_list}, index=col_names)

df_heat = df_heat.sort_values(by=['Vehicle Count'], ascending=False)

plt.figure(figsize=(6,6))
sns.heatmap(df_heat, annot=True, fmt="g").set(Title="Correlation Between EV Registration by Zip Code a

plt.show()
```



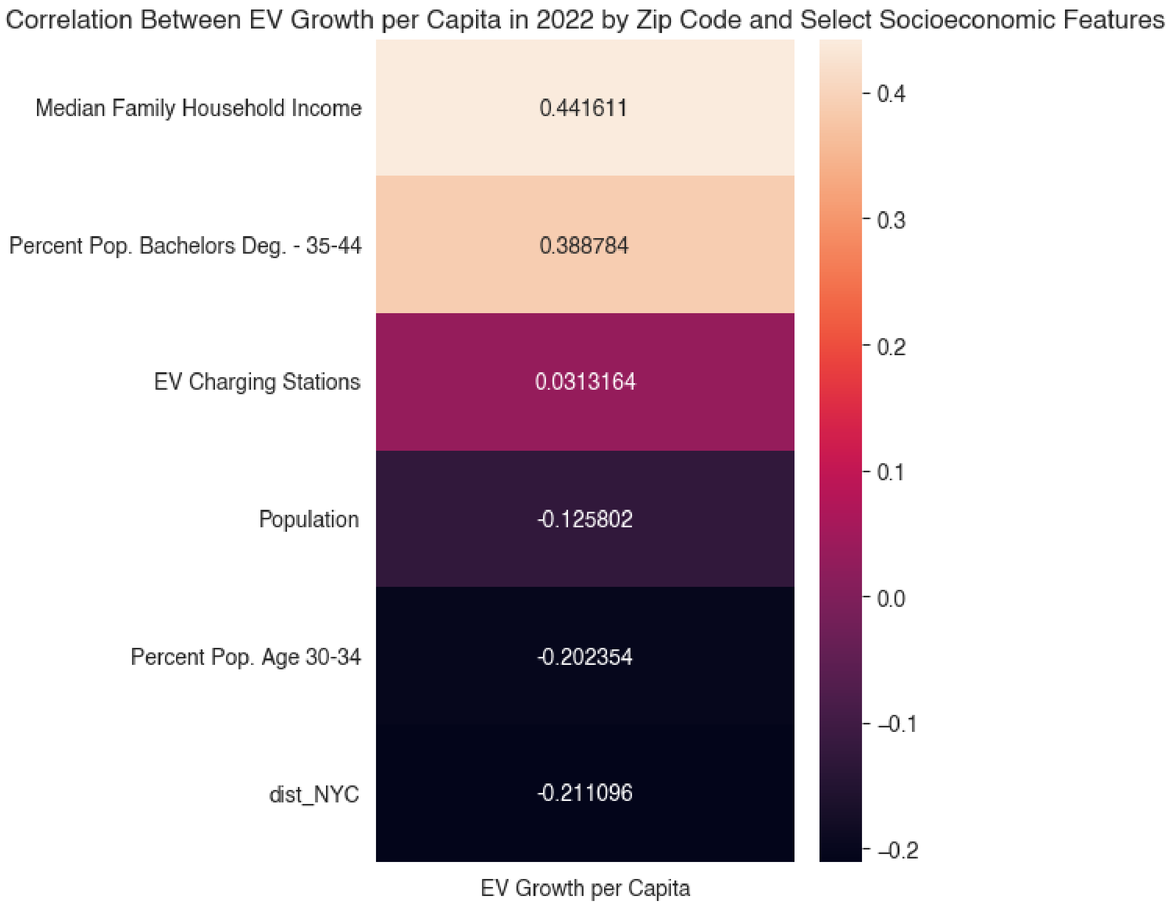
In [48]:
```
col_list = [df['Median Family Household Income'], df['Population'],
            df['EV Charging Stations'],
          df['Percent Pop. Age 30-34'], df['Percent Pop. Bachelors Deg. - 35-44'],
            df['dist_NYC']]

corr_df = df[['Median Family Household Income', 'Population', 'EV Charging Stations',
              'Percent Pop. Age 30-34', 'Percent Pop. Bachelors Deg. - 35-44', 'dist_NYC']]

corr_list = []
col_names = []

for col in col_list:
    corr_list.append(df['EVs per capita'].corr(col))

for col in corr_df.columns:
    col_names.append(col)

df_heat = pd.DataFrame({"EVs per capita": corr_list}, index=col_names)

df_heat = df_heat.sort_values(by=['EVs per capita'], ascending=False)

plt.figure(figsize=(6,6))
sns.heatmap(df_heat, annot=True, fmt="g").set(Title="Correlation Between EV Registration Per Capita by

plt.show()
```



```
In [49]: col_list = [df_time['Median Family Household Income'], df_time['Population'],
                     df_time['EV Charging Stations'],
                   df_time['Percent Pop. Age 30-34'], df_time['Percent Pop. Bachelors Deg. - 35-44'],
                     df_time['dist_NYC']]

         corr_df = df[['Median Family Household Income', 'Population', 'EV Charging Stations',
                       'Percent Pop. Age 30-34', 'Percent Pop. Bachelors Deg. - 35-44', 'dist_NYC']]

         corr_list = []
         col_names = []

         for col in col_list:
             corr_list.append(df_time['EVs Added in 2022'].corr(col))

         for col in corr_df.columns:
             col_names.append(col)

         df_heat = pd.DataFrame({'EVs Added in 2022': corr_list}, index=col_names)

         df_heat = df_heat.sort_values(by=['EVs Added in 2022'], ascending=False)

         plt.figure(figsize=(6,6))
         sns.heatmap(df_heat, annot=True, fmt="g").set(Title="Correlation Between EVs Added in 2022 by Zip Code

         plt.show()
```



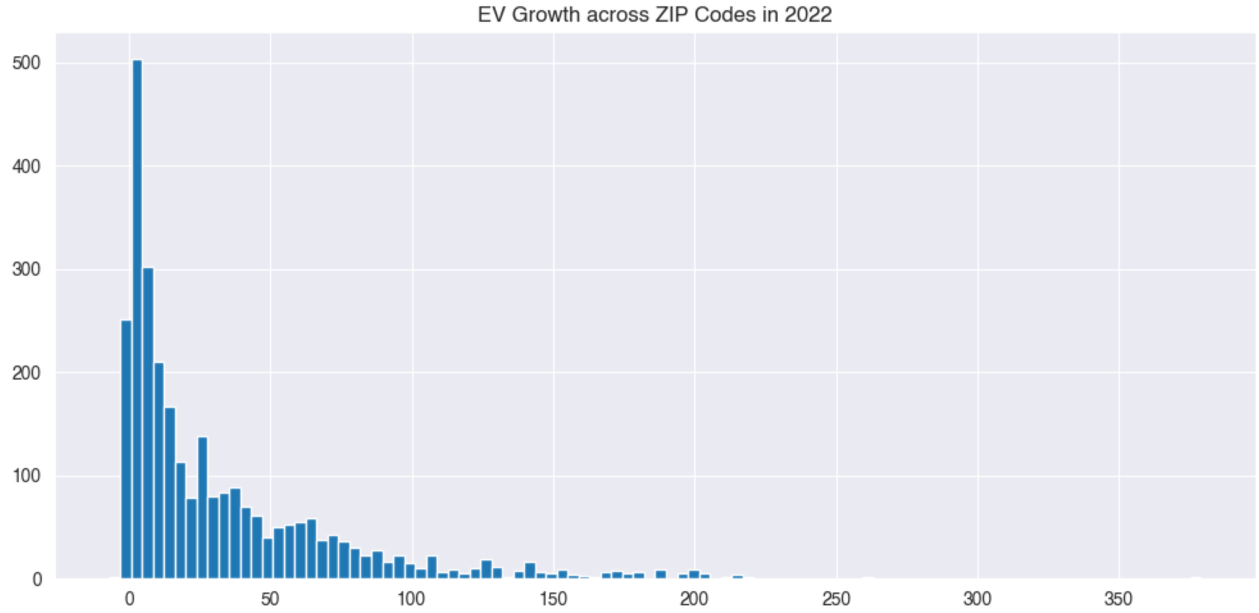
```
In [50]: col_list = [df_time['Median Family Household Income'], df_time['Population'],
                     df_time['EV Charging Stations'],
                   df_time['Percent Pop. Age 30-34'], df_time['Percent Pop. Bachelors Deg. - 35-44'],
                     df_time['dist_NYC']]

         corr_df = df[['Median Family Household Income', 'Population', 'EV Charging Stations',
                       'Percent Pop. Age 30-34', 'Percent Pop. Bachelors Deg. - 35-44', 'dist_NYC']]

         corr_list = []
         col_names = []

         for col in col_list:
             corr_list.append(df_time['EV Growth per Capita'].corr(col))

         for col in corr_df.columns:
             col_names.append(col)

         df_heat = pd.DataFrame({'EV Growth per Capita': corr_list}, index=col_names)

         df_heat = df_heat.sort_values(by=['EV Growth per Capita'], ascending=False)

         plt.figure(figsize=(6,6))
         sns.heatmap(df_heat, annot=True, fmt="g").set(Title="Correlation Between EV Growth per Capita in 2022

         plt.show()
```



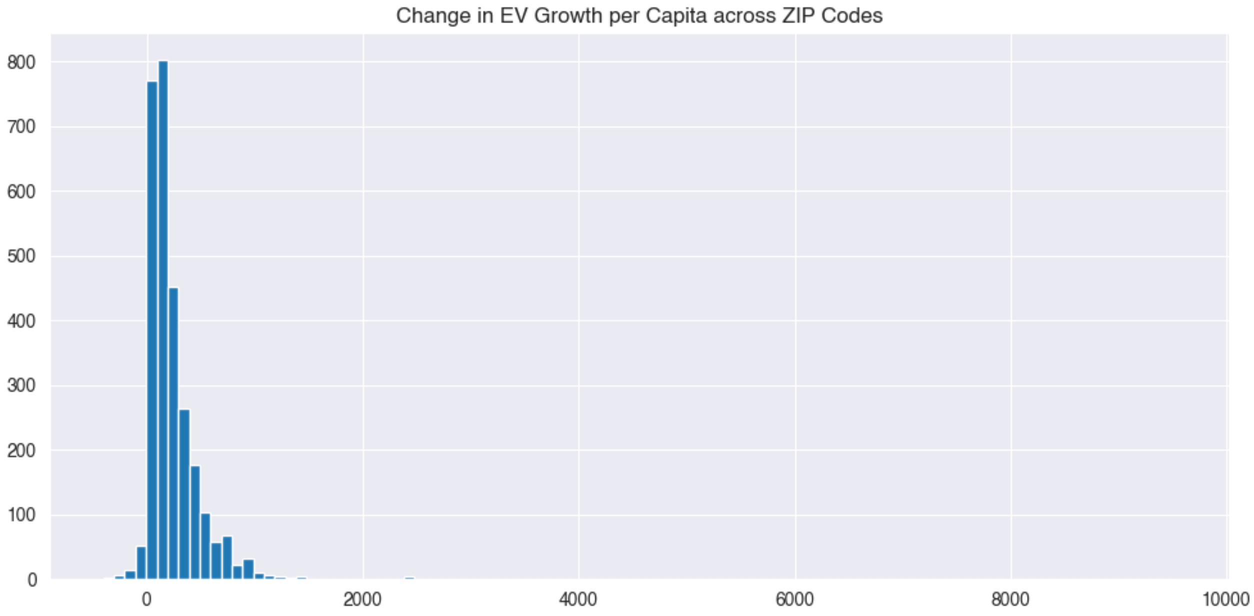
In [51]:
```
df_time.to_csv('Cleaned_EV_Reg_Data_2022-2023.csv')
```

Distribution of EV Growth in 2022 in Total and per Capita

In [52]:
```
df_time['EVs Added in 2022'].hist(bins=100).set(Title='EV Growth across ZIP Codes in 2022');
```



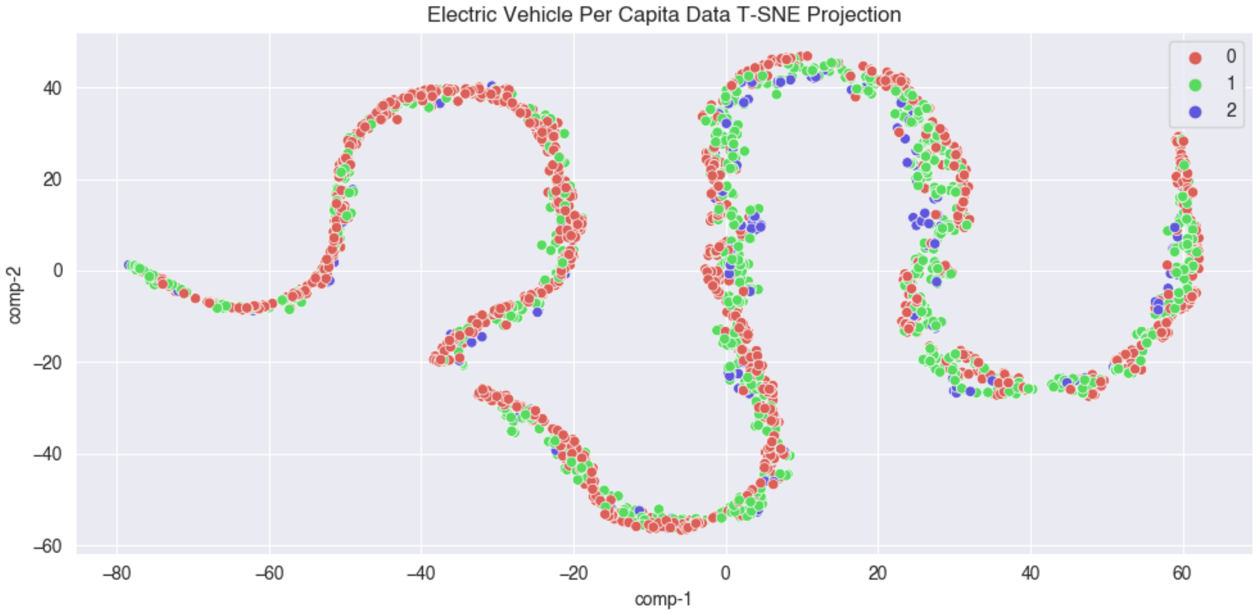
In [53]:
```
df_time['EV Growth per Capita'].hist(bins=100).set(Title='Change in EV Growth per Capita across ZIP C
```



Creating T-SNE Visualization for Breakdown of EVs Per Capita

In [54]:
```
X = df[['Median Family Household Income', 'Population', 'Percent Pop. Bachelors Deg. - 35-44',
            'dist_NYC', 'high_pop', 'voted for joe', 'in_NYC', 'EV Charging Stations', 'CT',
            'Percent Pop. Age 25-29', 'Percent Pop. Age 30-34', 'Percent Pop. Age 35-39', 'High_Income
           'Young_Liberal', 'Income_Liberal', 'Income_Pop', 'Ed_Income', 'NYC Suburb']]
y = df['High EVs Per Capita']

tsne = TSNE(n_components=2, perplexity=30, random_state=123)
z = tsne.fit_transform(X)
```

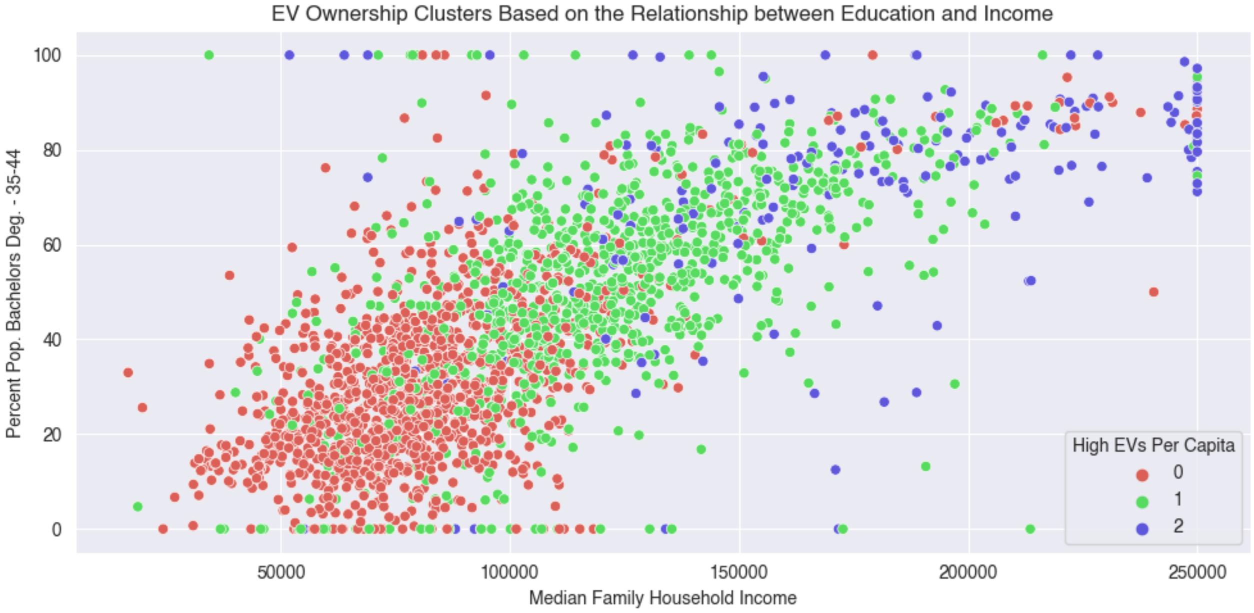
In [55]:
```
df2 = pd.DataFrame()
df2["y"] = y
df2["comp-1"] = z[:,0]
df2["comp-2"] = z[:,1]
sns.scatterplot(x="comp-1", y="comp-2", hue=df2.y.tolist(),
                palette=sns.color_palette("hls", 3),
                data=df2).set(title="Electric Vehicle Per Capita Data T-SNE Projection");
```



Creating Scatter Plot Hued to Categorical Buckets Based on Select, Important Features

Below I am creating different scatter plots hued to my categorical buckets to see if I can locate any natural clusters within my data.

```
In [56]: sns.scatterplot(x="Median Family Household Income", y="Percent Pop. Bachelors Deg. - 35-44", hue="High
                         palette=sns.color_palette("hls", 3),
                         data=df).set(title="EV Ownership Clusters Based on the Relationship between Education
```

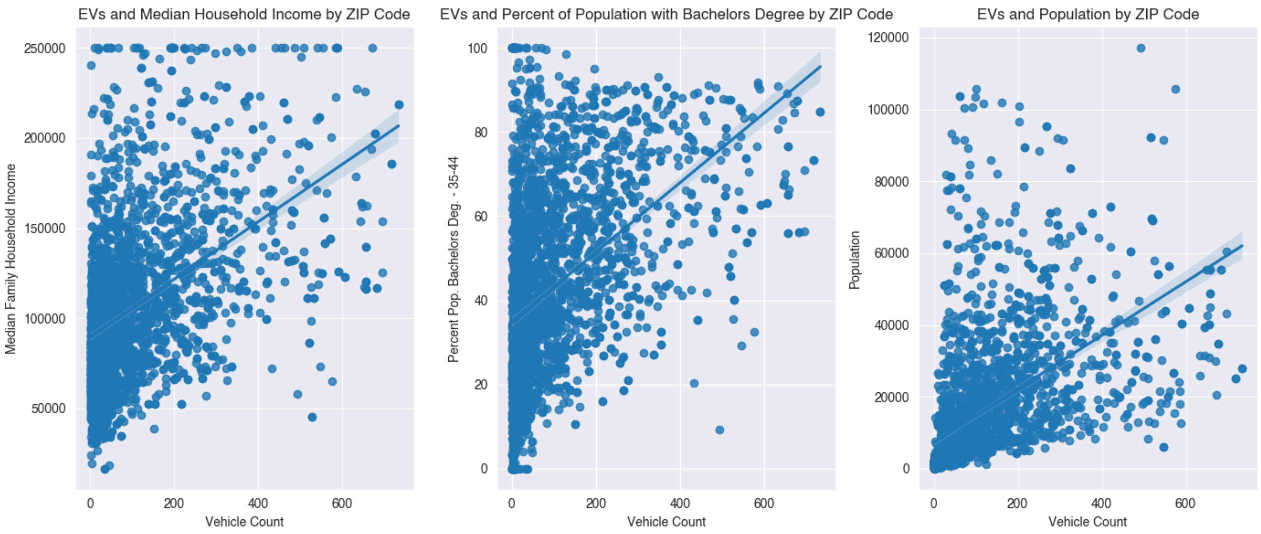


```
In [57]: df.columns
```

```
Out[57]: Index(['ZIP Code', 'Vehicle Count', 'NY', 'NJ', 'CT',
                'Median Family Household Income', 'Population',
                'Percent Pop. Bachelors Deg. - 35-44',
                'Total Pop. Bachelors Deg. - 35-44', 'EVs per capita', 'in_NYC',
                'dist_NYC', 'high_pop', 'County', 'voted for joe',
                'EV Charging Stations', 'Percent Pop. Age 25-29',
                'Percent Pop. Age 30-34', 'Percent Pop. Age 35-39', 'NYC Suburb',
                'Young_Liberal', 'Income_Liberal', 'Income_Pop', 'Ed_Income',
                'High_Income', 'High EVs Per Capita', 'State'],
               dtype='object')
```

Exploring Regression Plots Between Select Features of Interest Against EV Registrations

```
In [58]: # Creating Plots
         plt.rcParams["figure.figsize"] = [14, 6]
         plt.rcParams["figure.autolayout"] = True
         f, axes = plt.subplots(1, 3)

         sns.regplot(data = df, x='Vehicle Count', y='Median Family Household Income', ax=axes[0])
         sns.regplot(data = df, x='Vehicle Count', y='Percent Pop. Bachelors Deg. - 35-44', ax=axes[1])
         sns.regplot(data = df, x='Vehicle Count', y='Population',ax=axes[2])

         # Setting Titles
         axes[0].set_title('EVs and Median Household Income by ZIP Code')
         axes[1].set_title('EVs and Percent of Population with Bachelors Degree by ZIP Code')
         axes[2].set_title('EVs and Population by ZIP Code')

         # Displaying Plots
         plt.show()
```


EVs and Median Household Income by ZIP Code
Median Family Household Income
250000
200000
150000
100000
50000
0
200
400
600
Vehicle Count
EVs and Percent of Population with Bachelors Degree by ZIP Code
Percent Pop. Bachelors Deg. - 35-44
100
80
60
40
20
0
0
200
400
600
Vehicle Count
EVs and Population by ZIP Code
Population
120000
100000
80000
60000
40000
20000
0
0
200
400
600
Vehicle Count

```
In [ ]:
```