Data Preprocessing, ESDA and Machine Learning Model

Import All Packages

In [1]:
```
import pandas as pd
import geopandas as gpd
import matplotlib as plt
import numpy as np
import os
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn import ensemble
from sklearn.linear_model import Lasso,Ridge
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor

import osmnx as ox
import plotly
import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import geopandas as gpd
import numpy as np
import os

import plotly
import plotly.io as pio # to import to show plotly graph
pio.renderers.default='notebook' #import to show plotly graph
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#sns.set_style("white")
%matplotlib inline


from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn import ensemble
from sklearn.linear_model import Lasso,Ridge
from sklearn.cluster import KMeans
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
```

In [2]:
```
sns.set_palette("tab10", n_colors=None, desat=None, color_codes=False)
```

Area Mapping

In [3]:
```
eng_area = pd.read_excel('area_final.xlsx')
eng_area = eng_area.drop(['Unnamed: 0'],axis=1)
#{column: len(eng_area[column].unique()) for column in eng_area.columns}
#eng_area.head()
#eng_area.info(null_counts=True)
eng_area = eng_area.drop({'TCITY15CD','TCITY15NM'},axis=1)
```

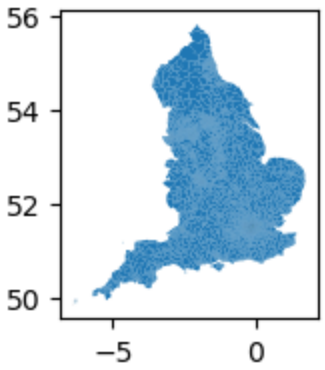
Map Data

In [4]:
```
MSOA = gpd.read_file('MSOA.geojson')
#MSOA.set_crs('EPSG:27700', allow_override=False)
#MSOA.head()
MSOA = MSOA[MSOA['MSOA11CD'].str[0] =='E']
len(MSOA)
```

Out[4]: 6791

In [5]:
```
MSOA.plot(figsize=(2, 2),linewidth=0.01)
```

Out[5]: <AxesSubplot: >



Car ownership Preprocessing

DFT 2022

In [6]:
```
cars = pd.read_csv('car by lsoa.csv',low_memory=False)#specify low memory type fals

cars = cars[cars['LicenceStatus']=='Licensed'] #filter for just licensed cars to be
cars = cars[cars['BodyType']=='Cars'] #limit to just cars by licensed ownership in

cars = cars.drop(['BodyType', 'LicenceStatus'],axis=1) #drop unnecessary columns th
cars = pd.melt(cars, id_vars=['LSOA11CD', 'LSOA11NM'], var_name='Year-Quarter', val
```

```
cars['Year'] = cars['Year-Quarter'].str[0:4] #create Year column
cars['Quarter'] = cars['Year-Quarter'].str[5:7] #create Quarter column
cars = cars[cars['Quarter'].isin(['Q2','Q4'])] #Filter Quarters required
cars = cars.loc[(cars['Year-Quarter'].str.endswith('Q4')) | (cars['Year-Quarter']==
cars['Year-Quarter'].unique() # check that year quarter chosen is accurate

cols = ['Year', 'cars']
cars[cols] = cars[cols].apply(pd.to_numeric, errors='coerce', axis=1) #change data
cars = cars[['LSOA11CD','LSOA11NM','Year','Quarter','cars']] #keep required columns
cars['Area'] = cars['LSOA11CD'].str[0:2]
cars.loc[cars['LSOA11CD'].str.startswith('E0',na=False),'Country']='England'
cars.loc[cars['LSOA11CD'].str.startswith('95',na=False),'Country'] = 'Northern Irel
cars.loc[cars['LSOA11CD'].str.startswith('S0',na=False),'Country'] = 'Scotland'
cars.loc[cars['LSOA11CD'].str.startswith('W0',na=False),'Country'] = 'Wales'
cars.loc[cars['LSOA11CD'].str.startswith('Mi',na=False),'Country'] = 'Other'
```

Exploratory Car Ownership Analysis in United Kingdom, Why we choose England as a Case Study for this research

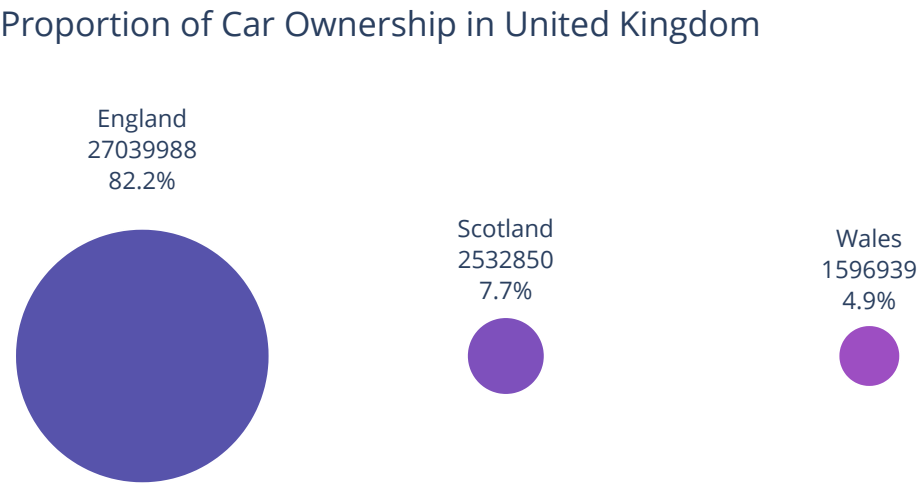
In [7]:

```
yearly = cars[cars['Year']==2022]
yearly = yearly[['Country','cars']] #Car Ownership in 2022 in UK by Country to anal
yearly = pd.DataFrame(yearly.groupby('Country').cars.sum()).reset_index() #sum car
yearly['Percent'] = [round(i*100/sum(yearly.cars),1) for i in yearly.cars] #proport
yearly = yearly.astype({'cars':'int'})
print('Cars Ownership by Country in UK By % of Total in 2022')
display(yearly)


print('Data Visualization of Car Ownership by Percentage')

yearly = yearly.sort_values(by='cars', ascending=False)
yearly['Y'] = [1]*len(yearly)
list_x = list(range(0,len(yearly)))
yearly['X'] = list_x

#create bubble chart comparison for UK car ownership as proportion of entire UK

#create a label list for each bubble
label = [i+'<br>'+str(j)+'<br>'+str(k)+'%' for i,j,k in zip(yearly.Country,
                                                            yearly.cars,
                                                            yearly.Percent)]
import plotly.express as px
fig = px.scatter(yearly, x='X', y='Y',color_discrete_sequence=px.colors.sequential.
                 color='Country',
                 size='cars', text=label, size_max=90,title="Proportion of Car Owne
                )
fig.update_layout(width=900, height=320,
                  margin = dict(t=50, l=0, r=0, b=0),
                  showlegend=False
                 )
fig.update_traces(textposition='top center')
fig.update_xaxes(showgrid=False, zeroline=False, visible=False)
fig.update_yaxes(showgrid=False, zeroline=False, visible=False)
fig.update_layout({'plot_bgcolor': 'white',
                   'paper_bgcolor': 'white'})
fig.show()
```

Cars Ownership by Country in UK By % of Total in 2022

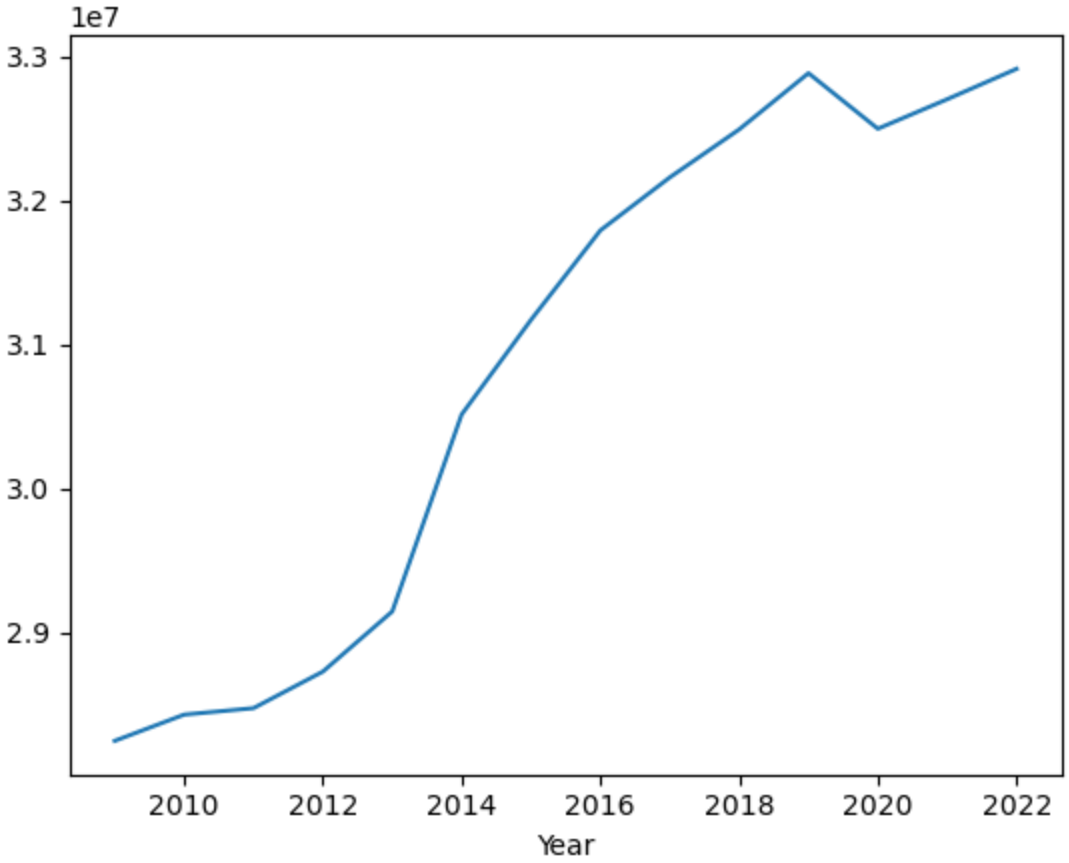
	Country	cars	Percent
0	England	27039988	82.2
1	Northern Ireland	999107	3.0
2	Other	745438	2.3
3	Scotland	2532850	7.7
4	Wales	1596939	4.9

Data Visualization of Car Ownership by Percentage



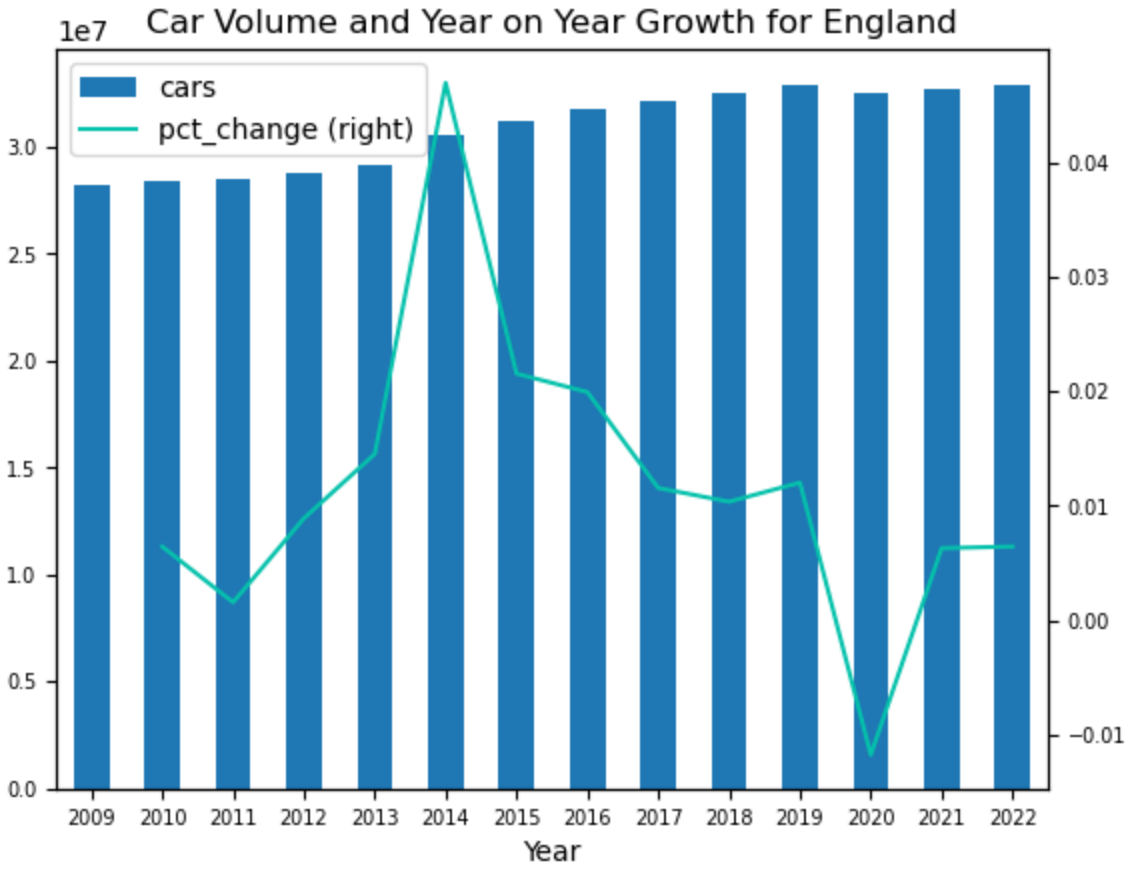
In [8]:
```
cars['date'] = pd.to_datetime(dict(year=cars.Year, month=1, day=1))
cars.groupby(['Year']).cars.sum().plot()
```

Out[8]: <AxesSubplot: xlabel='Year'>



In [9]:
```
year = cars[['LSOA11CD','LSOA11NM','Year','cars','Country']] #columns needed for ca
year = pd.merge(eng_area[['LSOA11CD','MSOA11CD','MSOA11NM']],year,how='left',on='LS
year = year[['Country','MSOA11CD','MSOA11NM','Year','cars']]#clean columns

year = year.groupby(['Country','MSOA11CD','MSOA11NM','Year'])['cars'].agg(['sum']).
year = year.rename({'sum':'cars'},axis=1) #rename sum columns to cars to indicate c
year.head(1)

year = cars.groupby(['Year'])['cars'].sum().reset_index() #group car volume by year
year = year.groupby(['Year', 'cars'])['cars'].sum().agg(['pct_change']).reset_index
year = year.astype({'Year':'int','cars':'int'}) #change data type
#add bar chart with line chart on top for yoy car volume and car volume percentage
year[['Year', 'cars']].plot(x='Year',kind='bar', width = 0.5,fontsize=8)
year['pct_change'].plot(secondary_y=True, color='#06C2AC',fontsize=7,legend=True)
plt.title('Car Volume and Year on Year Growth for England')
```

Out[9]:
```
Text(0.5, 1.0, 'Car Volume and Year on Year Growth for England')
```



Final Car data for Model

In [10]:
```
year = 2022 ##Filter 2022 as year for car ownership for this research
cars = cars[cars['Year']==year] #year
cars = cars[cars['LSOA11CD'].str[0]=='E'] #Filter for just England
cars = cars[cars['Year']==2022] #filter for latest year for car ownership study
cars.head(1)
```

Out[10]:

	LSOA11CD	LSOA11NM	Year	Quarter	cars	Area	Country	date
890	E01000001	City of London 001A	2022.0	Q2	325.0	E0	England	2022-01-01

Population

Population 2018 Density

In [11]:
```
pop1 = pd.read_excel('population2022.xlsx',sheet_name='Mid-2020 Population Density'
pop1 = pop1.iloc[3:,] # delete first three rows
pop1.columns = pop1.iloc[0] #use 4th row as header for columns
pop1 = pop1.iloc[1:,] #delete out row that already is a header column
pop1 = pop1.rename({'LSOA Code':'LSOA11CD'},axis=1) #rename columns
pop1 = pop1[pop1['LSOA11CD'].str[0]=='E'] #choose only LSOA from England
cols = ['Mid-2020 population', 'Area Sq Km','People per Sq Km'] # choose columns
```

```
pop1[cols] = pop1[cols].apply(pd.to_numeric, errors='coerce', axis=1) #change colum
#pop1.head() #check data output
```

In [12]:
```
age = pd.read_csv('MSOA-age.csv') #read median age by LSOA, we need median age by M
age = age.rename({'MSOA Code':'MSOA11CD'},axis=1) #calculation of Median Age by MSO
```

In [13]:
```
pop = pop1[['LSOA11CD', 'LSOA Name','Mid-2020 population', 'Area Sq Km']]
```

In [14]:
```
pop.head(1)
```

Out[14]:

3	LSOA11CD	LSOA Name	Mid-2020 population	Area Sq Km
4	E01011949	Hartlepool 009A	1944.0	0.5189

Household

2011 Census

In [15]:
```
household = pd.read_csv('Household Size 2011.csv') #household count
household = household[household['geography code'].str[0]=='E']
household = household.rename({'geography code':'LSOA11CD',
                'Household Size: All categories: Household size; measures: Value':'
                'Household Size: 1 person in household; measures: Value':'household
                 'Household Size: 2 people in household; measures: Value':'househol
                 'Household Size: 3 people in household; measures: Value':'househol
                 'Household Size: 4 people in household; measures: Value':'househol
                 'Household Size: 5 people in household; measures: Value':'househol
                 'Household Size: 6 people in household; measures: Value':'househol
                 'Household Size: 7 people in household; measures: Value':'househol
                 'Household Size: 8 or more people in household; measures: Value':'
household = household[['LSOA11CD','household','household1', 'household2', 'househol
       'household6', 'household7', 'household8']] # filter data from  LSOA file
household['household>=3'] = household['household3']+household['household4']+househo
household = household[['LSOA11CD','household', 'household1', 'household2','househol
```

Education

Census 2011

In [16]:
```
education = pd.read_csv('Education.csv') #add government data for Education
education = education[education['geography code'].str[0]=='E'] #choose only MSOA in
education = education.rename({'geography code':'LSOA11CD',
                'Qualification: All categories: Highest level of qualification; mea
                'Qualification: No qualifications; measures: Value':'EducationNo',
                 'Qualification: Level 1 qualifications; measures: Value':'Educatio
                 'Qualification: Level 2 qualifications; measures: Value':'Educatio
                 'Qualification: Apprenticeship; measures: Value':'EducationAppren'
                 'Qualification: Level 3 qualifications; measures: Value':'Educatio
                 'Qualification: Level 4 qualifications and above; measures: Value'
                 'Qualification: Other qualifications; measures: Value':'EducationO
```

```
education = education[['LSOA11CD','EducationAll',
       'EducationNo', 'Education1', 'Education2', 'EducationAppren',
       'Education3', 'Education4', 'EducationOther']]
```

In [17]:
```
education.head(1)
```

Out[17]:

	LSOA11CD	EducationAll	EducationNo	Education1	Education2	EducationAppren	Education3
0	E01012334	1884	241	219	306	89	230

Income

2018 Revenue

In [18]:
```
income = pd.read_excel('income.xls',sheet_name='Total annual income') #Average annu
income = income.iloc[3:,]
income.columns = income.iloc[0]
income = income.iloc[1:,]
income = income[income['MSOA code'].str[0]=='E']
income = income.rename({'MSOA code':'MSOA11CD'},axis=1)
income = income [['MSOA11CD','Total annual income (£)']]
income = income.rename({'Total annual income (£)':'income'},axis=1)
income['income'] = income['income'].apply(pd.to_numeric, errors='coerce')
```

In [19]:
```
income.head(1)
```

Out[19]:

3	MSOA11CD	income
4	E02004297	39800

Public Transport Accessibility

In [20]:
```
path = r'C:\Users\neetm\Desktop\Dissertation\Data\public transport data'
filenames = os.listdir(r'C:\Users\neetm\Desktop\Dissertation\Data\public transport
pbx = pd.read_csv(path + f'/'+'Public-transport-journey-times-to-all-FE-establishme
pbx = pbx.iloc[:,0:2]
for f in filenames:
    pb = pd.read_csv(path+f'/'+f)
    pb.rename(columns={pb.columns[2]: f }, inplace = True)
    pbx = pd.merge(pbx,pb.iloc[:, [0,2]],how='left',on='LSOA11 Code')
pbx.head(1)
```

Out[20]:

	LSOA11 Code	LSOA11 Name	Public-transport-journey-times-to-all-FE-establishments.csv	Public-transport-journey-times-to-all-GP-surgeries.csv	Public-transport-journey-times-to-all-hospitals.csv	Public-transport-journey-times-to-all-large-emp-centres.csv	Public-transport-journey-times-to-all-primary-schools.csv	t se sc
0	E01000001	City of London 001A	10.35	3.96	6.04	3.59	6.12	

In [21]:

```
pop2 = pop1[['LSOA11CD','Mid-2020 population']] #Use population data for weighted a
pop2 = pop2.rename({'Mid-2020 population':'LSOA_pop'},axis=1)#Use population data f
pb = pd.merge(pop2,pbx,how='left',left_on='LSOA11CD',right_on='LSOA11 Code') #Join
pb = pb[['LSOA11CD', 'LSOA11 Name','LSOA_pop',
       'Public-transport-journey-times-to-all-FE-establishments.csv',
       'Public-transport-journey-times-to-all-GP-surgeries.csv',
       'Public-transport-journey-times-to-all-hospitals.csv',
       'Public-transport-journey-times-to-all-large-emp-centres.csv',
       'Public-transport-journey-times-to-all-primary-schools.csv',
       'Public-transport-journey-times-to-all-secondary-schools.csv',
       'Public-transport-journey-times-to-all-town-centres.csv',
       'Public-transport-journey-times-to-good-FE-establishments.csv',
       'Public-transport-journey-times-to-good-GP-surgeries.csv',
       'Public-transport-journey-times-to-good-hospitals.csv',
       'Public-transport-journey-times-to-good-primary-schools.csv',
       'Public-transport-journey-times-to-good-secondary-schools.csv']]

####Create Public Journey Times Measure as proxy of Public Transport Accessibility
### Multiple Average Journey Times Back to Each population for Total Journey Time B
pb['PB_ALLFE'] = pb['Public-transport-journey-times-to-all-FE-establishments.csv']
pb['PB_ALLGP'] = pb['Public-transport-journey-times-to-all-GP-surgeries.csv'] * pb[
pb['PB_ALLHP'] = pb['Public-transport-journey-times-to-all-hospitals.csv'] * pb['LS
pb['PB_ALLEMP'] = pb['Public-transport-journey-times-to-all-large-emp-centres.csv']
pb['PB_ALLPS'] = pb['Public-transport-journey-times-to-all-primary-schools.csv'] *
pb['PB_ALLSS'] = pb['Public-transport-journey-times-to-all-secondary-schools.csv']
pb['PB_TC'] = pb['Public-transport-journey-times-to-all-town-centres.csv'] * pb['LS
pb['PB_GFE'] = pb['Public-transport-journey-times-to-good-FE-establishments.csv'] *
pb['PB_GGP'] = pb['Public-transport-journey-times-to-good-GP-surgeries.csv'] * pb['
pb['PB_GHP'] = pb['Public-transport-journey-times-to-good-hospitals.csv'] * pb['LSO
pb['PB_GPS'] = pb['Public-transport-journey-times-to-good-primary-schools.csv'] * p
pb['PB_GSS'] = pb['Public-transport-journey-times-to-good-secondary-schools.csv'] *


####Join with MSOA data
pb_msoa = eng_area[['LSOA11CD','MSOA11CD', 'MSOA21CD']]
pb = pd.merge(pb_msoa,pb,how='left',on = 'LSOA11CD')
pbavg = pb[['MSOA11CD', 'MSOA21CD', 'LSOA_pop',
       'PB_ALLFE', 'PB_ALLGP', 'PB_ALLHP', 'PB_ALLEMP', 'PB_ALLPS', 'PB_ALLSS',
       'PB_TC', 'PB_GFE', 'PB_GGP', 'PB_GHP', 'PB_GPS', 'PB_GSS']]
pbavg = pbavg.groupby(by=["MSOA11CD"]).sum().reset_index()

### Weighted Average by Population for Each MSOA
```

```
pbavg['PB_ALLFE'] = pbavg['PB_ALLFE']/pbavg['LSOA_pop']
pbavg['PB_ALLGP'] = pbavg['PB_ALLGP']/pbavg['LSOA_pop']
pbavg['PB_ALLHP'] = pbavg['PB_ALLHP']/pbavg['LSOA_pop']
pbavg['PB_ALLEMP'] = pbavg['PB_ALLEMP']/pbavg['LSOA_pop']
pbavg['PB_ALLPS'] = pbavg['PB_ALLPS']/pbavg['LSOA_pop']
pbavg['PB_ALLSS'] = pbavg['PB_ALLSS']/pbavg['LSOA_pop']
pbavg['PB_TC'] = pbavg['PB_TC']/pbavg['LSOA_pop']
pbavg['PB_GFE'] = pbavg['PB_GFE']/pbavg['LSOA_pop']
pbavg['PB_GGP'] = pbavg['PB_GGP']/pbavg['LSOA_pop']
pbavg['PB_GHP'] = pbavg['PB_GHP']/pbavg['LSOA_pop']
pbavg['PB_GPS'] = pbavg['PB_GPS']/pbavg['LSOA_pop']
pbavg['PB_GSS'] = pbavg['PB_GSS']/pbavg['LSOA_pop']

public = pbavg[['MSOA11CD','PB_ALLFE', 'PB_ALLGP', 'PB_ALLHP',
       'PB_ALLEMP', 'PB_ALLPS', 'PB_ALLSS', 'PB_TC', 'PB_GFE', 'PB_GGP',
       'PB_GHP', 'PB_GPS', 'PB_GSS']]
```

In [22]:
```
public.head(1)
```

Out[22]:

	MSOA11CD	PB_ALLFE	PB_ALLGP	PB_ALLHP	PB_ALLEMP	PB_ALLPS	PB_ALLSS	PB_TC	
0	E02000001	10.276263	6.606503	8.925261	4.455071	6.469298	10.530612	16.289416	1(

Urban Form:Median

In [23]:
```
urban_median = pd.read_csv('urbanmedian.csv') #Momepy Data for Urban Form in Median
urban_median = urban_median.drop({'Unnamed: 0','geometry'},axis=1)
urban_median = urban_median.rename({'msoa11cd':'MSOA11CD'},axis=1)
urban_median = urban_median[urban_median['MSOA11CD'].str[0]=='E'] #Filtering only E
urban_median = urban_median.drop(['sdbCoA','objectid', 'msoa11nm', 'msoa11nmw', 'st
#drop not needed columns
```

In [24]:
```
urban_median.describe()
```

Out[24]:

	sdbAre	sdbHei	sdbPer	sdbVol	ssbFoF	ssbVFR	ssbCCo
count	6791.000000	6791.000000	6791.000000	6791.000000	6791.000000	6791.000000	6791.000000
mean	60.422334	7.696915	34.309959	468.707069	1.059031	1.754971	0.515593
std	10.404480	1.316769	3.384536	133.393056	0.145194	0.135395	0.046554
min	31.671851	2.700000	23.377956	111.217591	0.655692	1.258766	0.251835
25%	52.717773	7.000000	32.084532	398.209330	0.953056	1.664211	0.509281
50%	58.935519	7.600000	34.137684	435.074797	1.032451	1.740335	0.525621
75%	66.284478	8.200000	36.182078	491.963050	1.145487	1.840171	0.539006
max	136.281525	23.100000	58.283511	2974.246398	1.783864	2.465677	0.625776

8 rows × 68 columns

In [25]:
```
urban_median[urban_median['MSOA11CD']=='E02000001']
```

Out[25]:

	MSOA11CD	sdbAre	sdbHei	sdbPer	sdbVol	ssbFoF	ssbVFR	ssbCCo	ssbCor
0	E02000001	136.281525	23.1	58.283511	2974.246398	0.700039	2.390425	0.449578	6.0

1 rows × 69 columns

Joining Dataset by LSOA Level

In [26]:
```
#join eng geopandas map data with car ownership
df_all = pd.merge(eng_area,cars[['LSOA11CD','cars']],how='left',left_on = 'LSOA11CD

#join eng geopandas map data with population data

df_all = pd.merge(df_all,pop[['LSOA11CD','Mid-2020 population', 'Area Sq Km']],
         how='left',left_on = 'LSOA11CD',right_on ='LSOA11CD')

#join eng geopandas map data with household data

df_all = pd.merge(df_all,household[['LSOA11CD','household','household1', 'household
         how='left',left_on = 'LSOA11CD',right_on ='LSOA11CD')

#join eng geopandas map data with education data
df_all = pd.merge(df_all,education,
         how='left',left_on = 'LSOA11CD',right_on ='LSOA11CD')
```

Final Data Join by MSOA

In [27]:
```
msoa_min = df_all.groupby('MSOA11CD').sum().reset_index() #sum all types of populat
msoa_area = eng_area[['MSOA11CD', 'MSOA11NM','MSOA21CD', 'MSOA21NM', 'LAD22CD', 'LA
msoa_area = msoa_area.drop_duplicates(subset=['MSOA11CD', 'MSOA11NM','MSOA21CD', 'M

df = pd.merge(msoa_area,msoa_min,how='left',on='MSOA11CD')
df = pd.merge(df,income,how='left',on='MSOA11CD') #join with income level by MSOA
df = pd.merge(df,urban_median,how='left',on='MSOA11CD') #join with with urban form
```

In [28]:
```
df.columns
```

Out[28]:
```
Index(['MSOA11CD', 'MSOA11NM', 'MSOA21CD', 'MSOA21NM', 'LAD22CD', 'LAD22NM',
       'CTRY21CD', 'CTRY21NM', 'cars', 'Mid-2020 population', 'Area Sq Km',
       'household', 'household1', 'household2', 'household>=3', 'EducationAll',
       'EducationNo', 'Education1', 'Education2', 'EducationAppren',
       'Education3', 'Education4', 'EducationOther', 'income', 'sdbAre',
       'sdbHei', 'sdbPer', 'sdbVol', 'ssbFoF', 'ssbVFR', 'ssbCCo', 'ssbCor',
       'ssbSqu', 'ssbERI', 'ssbElo', 'ssbCCM', 'ssbCCD', 'stbOri', 'sdcLAL',
       'sdcAre', 'sicFAR', 'sscCCo', 'sscERI', 'stcOri', 'sicCAR', 'stbCeA',
       'mtbAli', 'mtbNDi', 'mtcWNe', 'mdcAre', 'ltcWRE', 'ltbIBD', 'ltcRea',
       'ltcAre', 'sdsSPW', 'sdsSPH', 'sdsSPO', 'sdsSWD', 'sdsSHD', 'sdsSPR',
       'sdsLen', 'sssLin', 'ldsMSL', 'mtdDeg', 'lcdMes', 'linP3W', 'linP4W',
       'linPDE', 'lcnClo', 'ldsCDL', 'xcnSCl', 'mtdMDi', 'lddNDe', 'linWID',
       'ldeAre', 'ldePer', 'lseCCo', 'lseERI', 'lseCWA', 'lteOri', 'lteWNB',
       'lieWCe', 'stbSAl', 'ldbPWL', 'mtbSWR', 'sddAre', 'sdsAre', 'sisBpM',
       'misCel', 'mdsAre', 'lisCel', 'ldsAre'],
      dtype='object')
```

data map with Map MSOA

In [29]:
```
#Map with MSOA map

df_msoa = pd.merge(MSOA,df,how='left',on='MSOA11CD')
df_msoa = pd.merge(df_msoa,age,how='left',on='MSOA11CD') #Map with median age data
df_msoa = pd.merge(df_msoa,public,how='left',on="MSOA11CD") #Map with public transp

#pd.options.display.max_colwidth = 20 #show entire columns
#df_msoa.head(1)
```

In [30]:
```
df_msoa.head(1)
```

Out[30]:

	OBJECTID	MSOA11CD	MSOA11NM_x	BNG_E	BNG_N	LONG_	LAT	Shape_Leng	Shape
0	1	E02000001	City of London 001	532384	181355	-0.09349	51.5156	9651.221144	3.1514

1 rows × 116 columns

In [31]:
```
#features = pd.DataFrame(df_msoa.columns.tolist())
```

Metrics created

In [32]:
```
## Create metrics car by population and population density for use in the model
df_msoa['popdensity'] = round(df_msoa['Mid-2020 population']/df_msoa['Area Sq Km'],
df_msoa['car_by_pop'] = round(df_msoa['cars']/df_msoa['Mid-2020 population'],2)
df_msoa['car_by_area'] = round(df_msoa['cars']/df_msoa['Area Sq Km'],2)
df_msoa = df_msoa.rename({'Median':'Age-Median'},axis=1)
```

In [33]:
```
df_msoa.head(1)
```

Out[33]:

	OBJECTID	MSOA11CD	MSOA11NM_x	BNG_E	BNG_N	LONG_	LAT	Shape_Leng	Shape
0	1	E02000001	City of London 001	532384	181355	-0.09349	51.5156	9651.221144	3.1514

1 rows × 119 columns

In [34]:
```
#drop columns not needed
df_msoa = df_msoa.drop(['BNG_E',
 'BNG_N','Shape_Leng',
 'Shape__Area',
 'Shape__Length',
'MSOA11NM_y',
 'MSOA21CD',
 'MSOA21NM',
 'LAD22CD',
 'LAD22NM',
 'CTRY21CD',
 'CTRY21NM'],axis=1)
```

Final Deal with Null and Nans

In [35]:
```
df_msoa[df_msoa.isna().any(axis=1)]

#E02000001 City of London 001
#E02006781      Isles of Scilly 001
#df_msoa['sdbCoA'] check data from column with all 0 values then removed
df_msoa = df_msoa[~df_msoa.isna().any(axis=1)]
#df_msoa = df_msoa.drop(['sdbCoA'],axis=1)

df_msoa[df_msoa.isna().any(axis=1)] #removed two MSOAs that have Urban Form NA valu
#Dask crashes
```

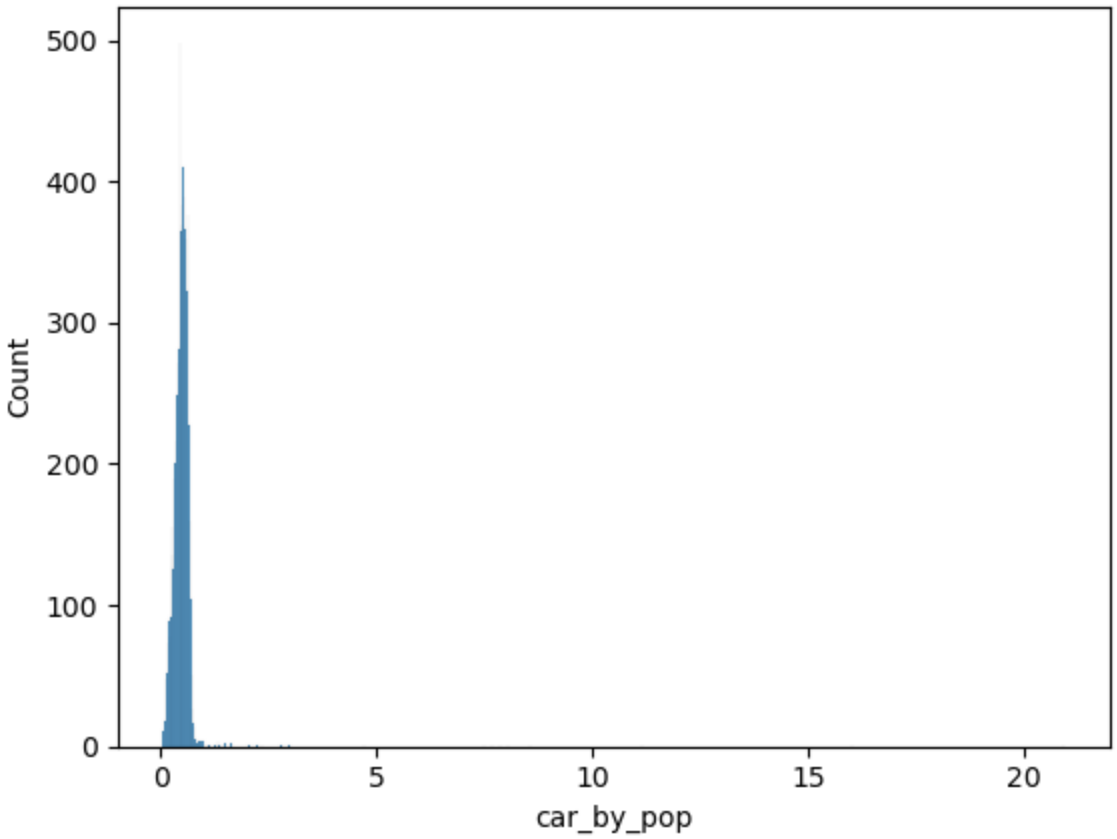
Out[35]:

	OBJECTID	MSOA11CD	MSOA11NM_x	LONG_	LAT	geometry	cars	Mid-2020 population	Area Sq Km	househ

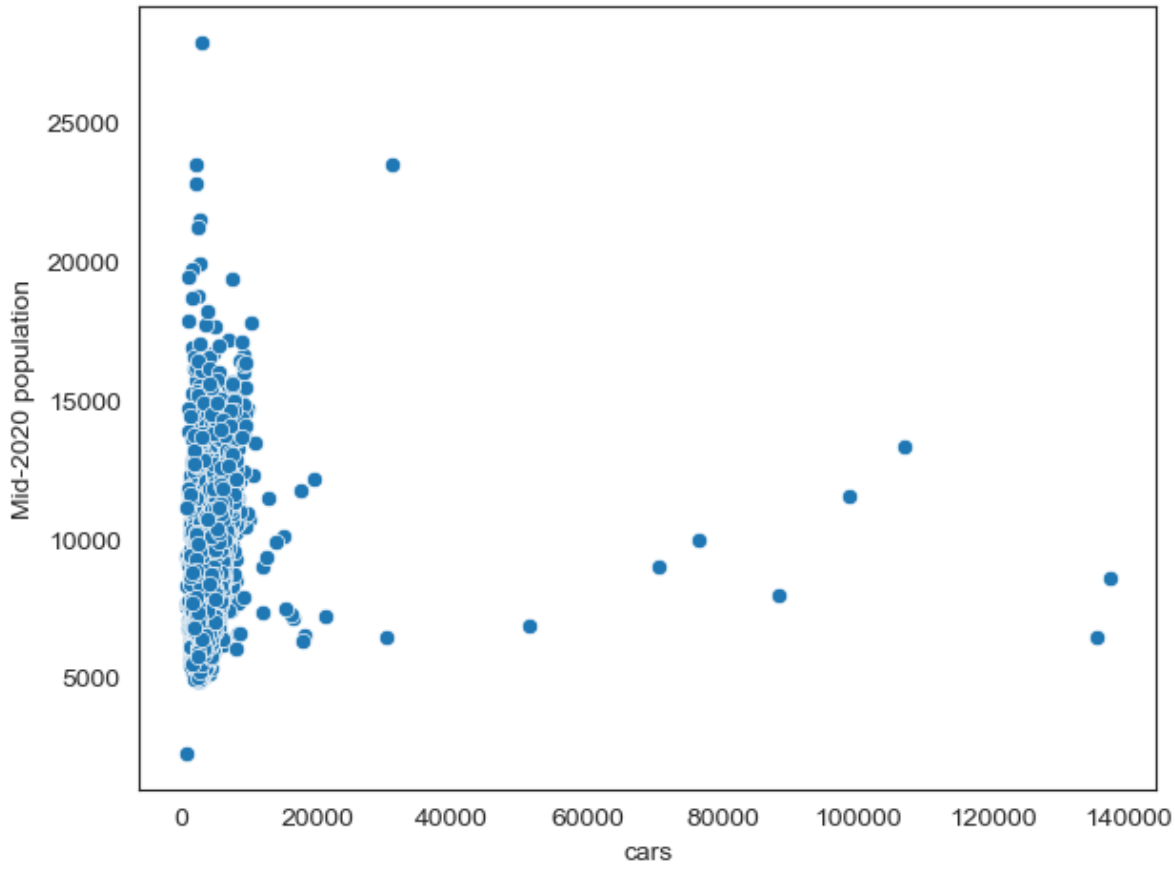
0 rows × 107 columns

Data Exploration and Outliers

In [36]:
```
import seaborn as sns
sns.histplot(df_msoa['car_by_pop'])
import matplotlib.pyplot as plt
plt.show()
```

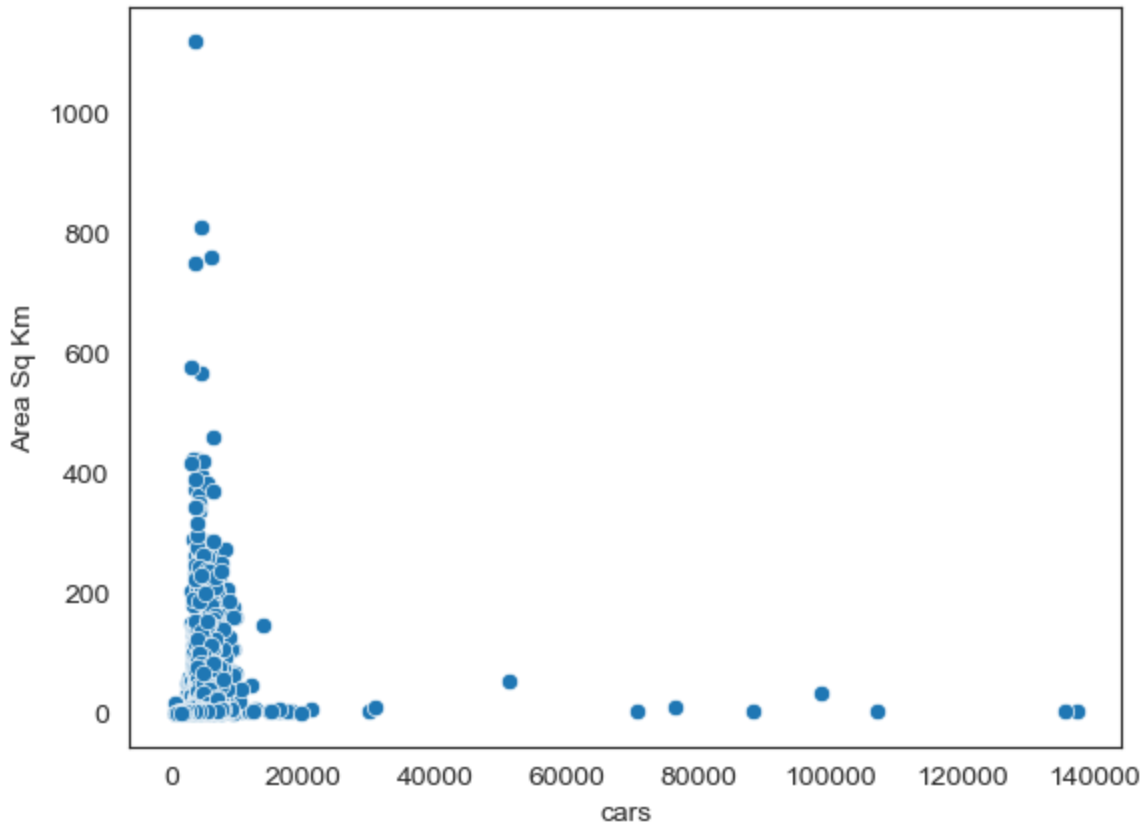


In [37]:
```
sns.set_style("white")
from scipy import stats
sns.scatterplot(data=df_msoa, x ='cars', y ='Mid-2020 population')
stats.pearsonr(df_msoa['cars'], df_msoa['Mid-2020 population'])
plt.tight_layout()
plt.show()
```



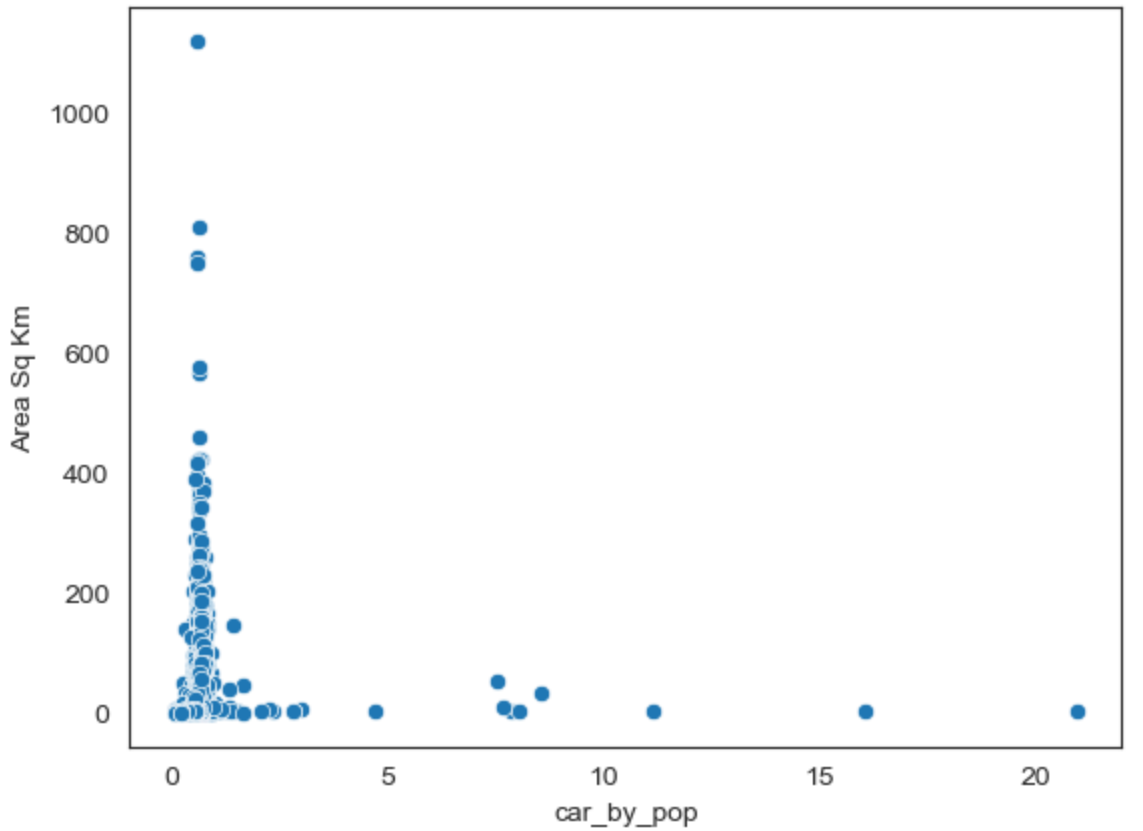
In [38]:
```
sns.scatterplot(data=df_msoa, x ='cars', y ='Area Sq Km')
stats.pearsonr(df_msoa['cars'], df_msoa['Area Sq Km'])
```

Out[38]:
```
PearsonRResult(statistic=0.11294645877365728, pvalue=1.0022144277145983e-20)
```



In [39]:
```
sns.scatterplot(data=df_msoa, x ='car_by_pop', y ='Area Sq Km') #important
```

Out[39]:
```
<AxesSubplot: xlabel='car_by_pop', ylabel='Area Sq Km'>
```



Outliers

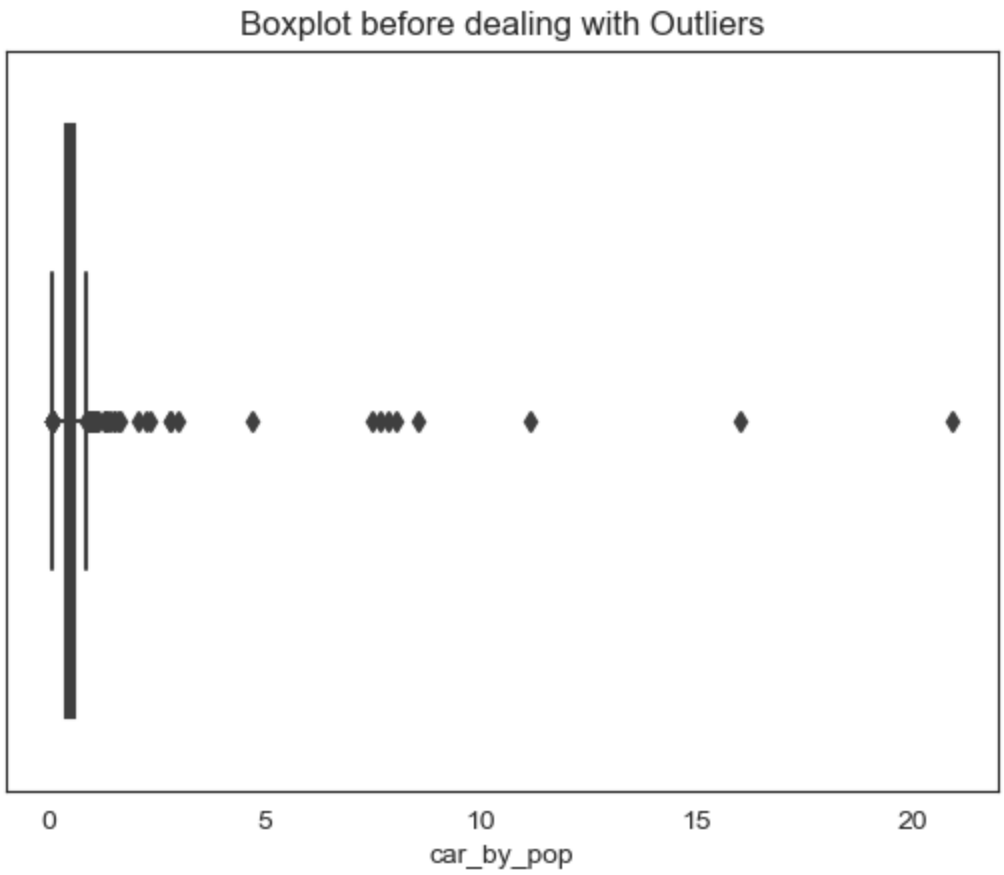
Dealing with Outliers by Imputation, replacing with median values

Like imputation of missing values, we can also impute outliers. We can use mean, median, zero value in this methods. Since we imputing there is no loss of data. Here median is appropriate because it is not affected by outliers.

In [40]:
```
df_msoa = df_msoa.rename({'MSOA11NM_x':'MSOA11NM'})
```

In [41]:
```
sns.boxplot(x=df_msoa["car_by_pop"])
plt.title('Boxplot before dealing with Outliers')
```

Out[41]:
```
Text(0.5, 1.0, 'Boxplot before dealing with Outliers')
```

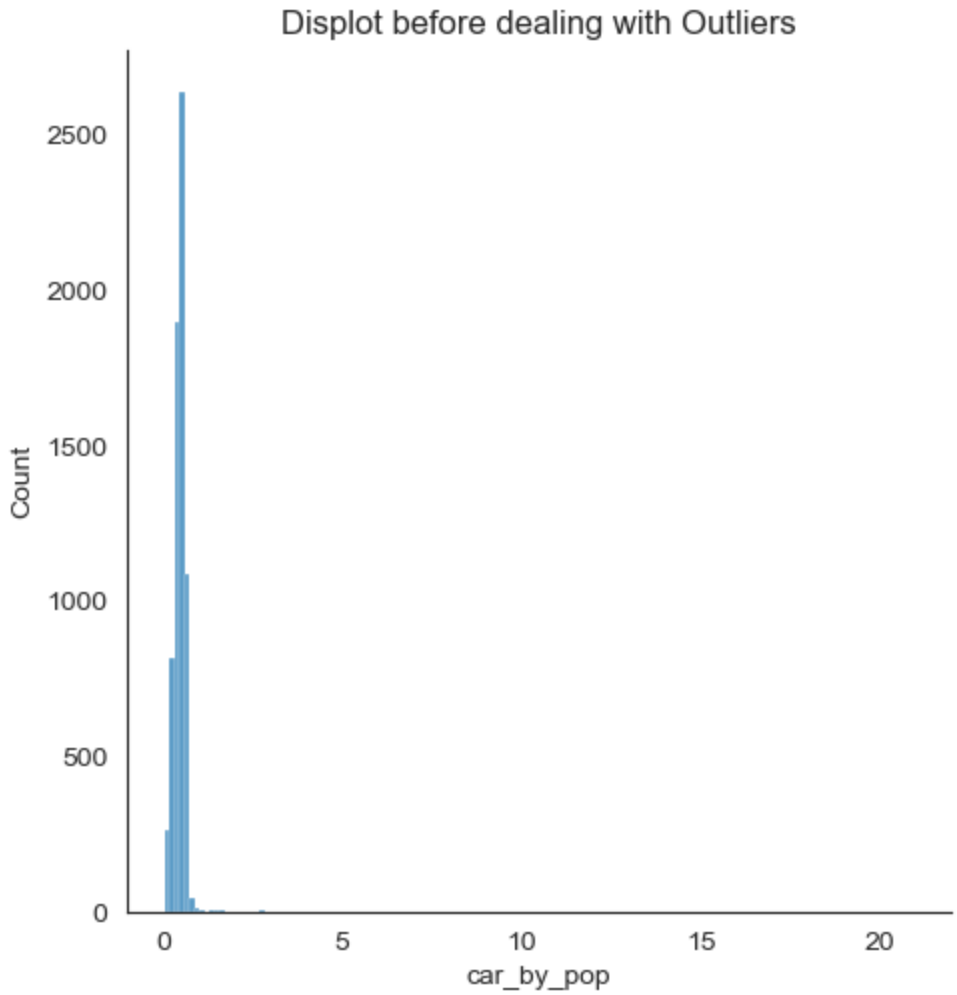


In [42]:
```
lower_limit = df_msoa["car_by_pop"].quantile(0.01)
upper_limit = df_msoa["car_by_pop"].quantile(0.99)

print(lower_limit)
print(upper_limit)
```

```
0.13
0.77
```

In [43]:
```
sns.displot(x=df_msoa["car_by_pop"],bins=150)
plt.title('Displot before dealing with Outliers')
```

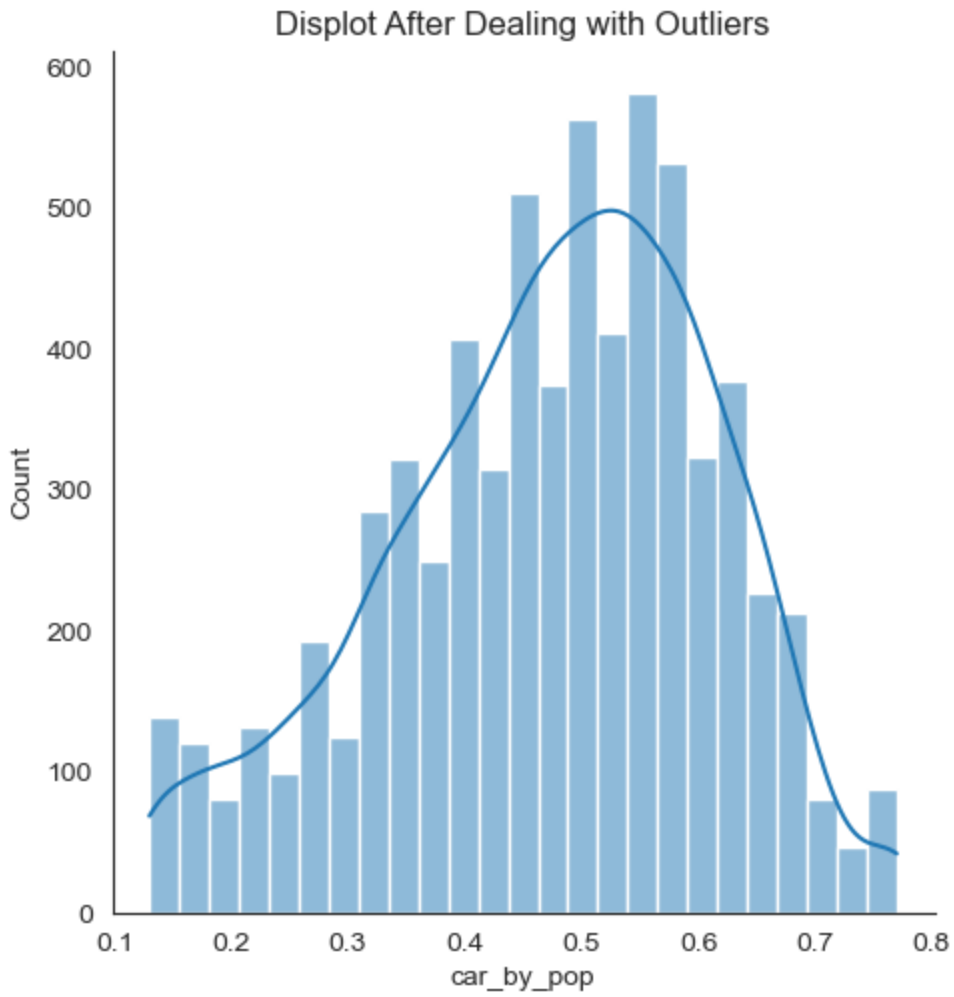
Out[43]: Text(0.5, 1.0, 'Displot before dealing with Outliers')



In [44]:
```
df_msoa["car_by_pop"] = np.where(df_msoa["car_by_pop"]> upper_limit, upper_limit,
                        np.where(df_msoa["car_by_pop"]< lower_limit, lower_limit,
                        df_msoa["car_by_pop"]))
```

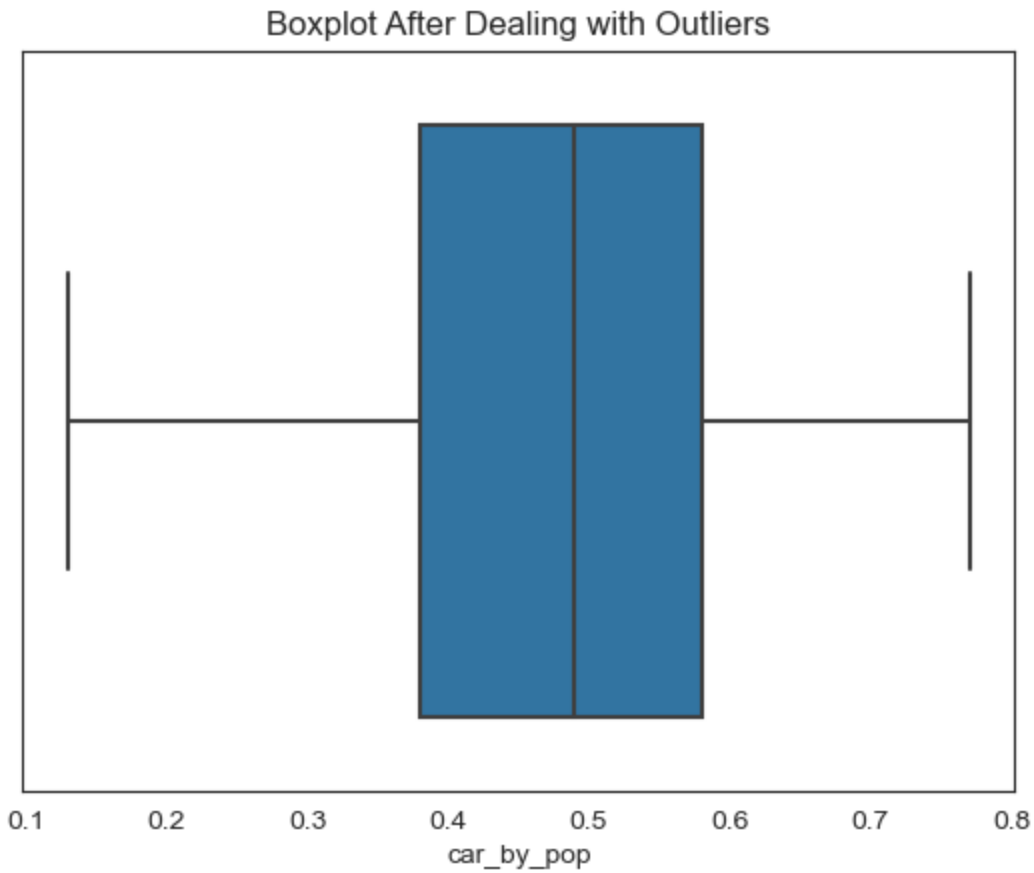
In [45]:
```
sns.displot(df_msoa['car_by_pop'], bins=25, kde=True);
plt.title('Displot After Dealing with Outliers')
```

Out[45]: Text(0.5, 1.0, 'Displot After Dealing with Outliers')

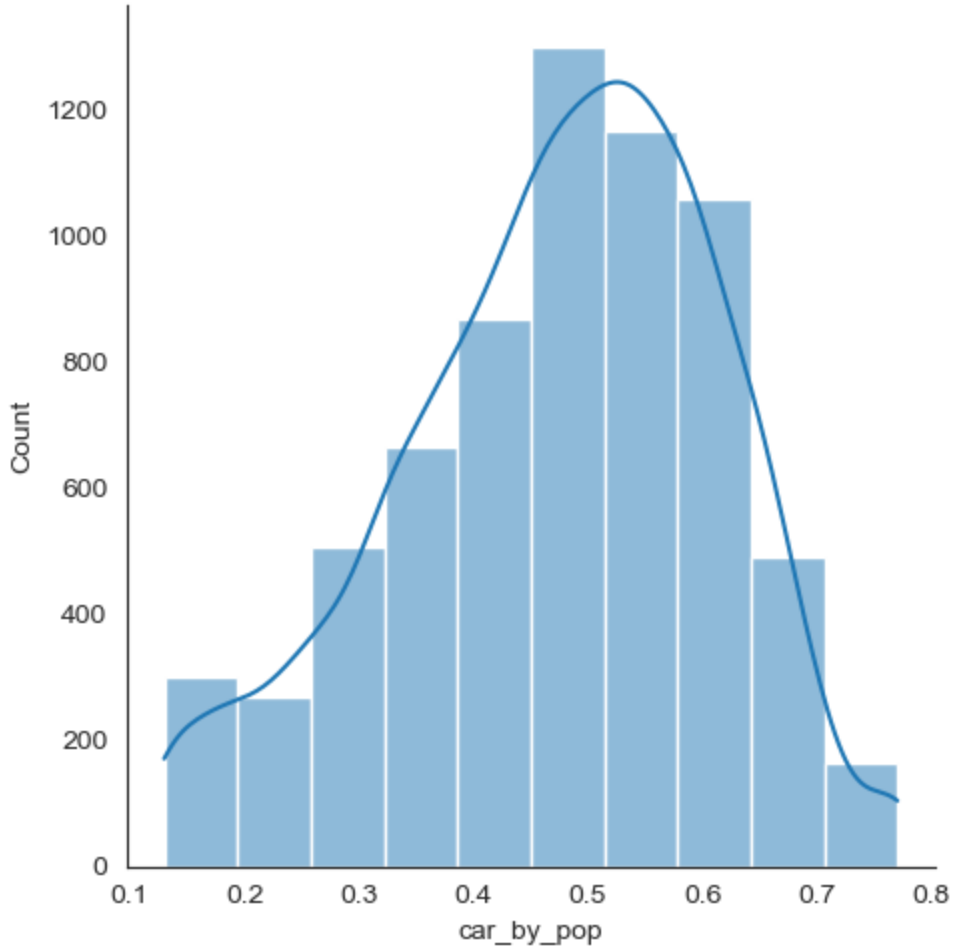


In [46]:
```
sns.boxplot(x=df_msoa["car_by_pop"])
plt.title('Boxplot After Dealing with Outliers')
```

Out[46]:
```
Text(0.5, 1.0, 'Boxplot After Dealing with Outliers')
```

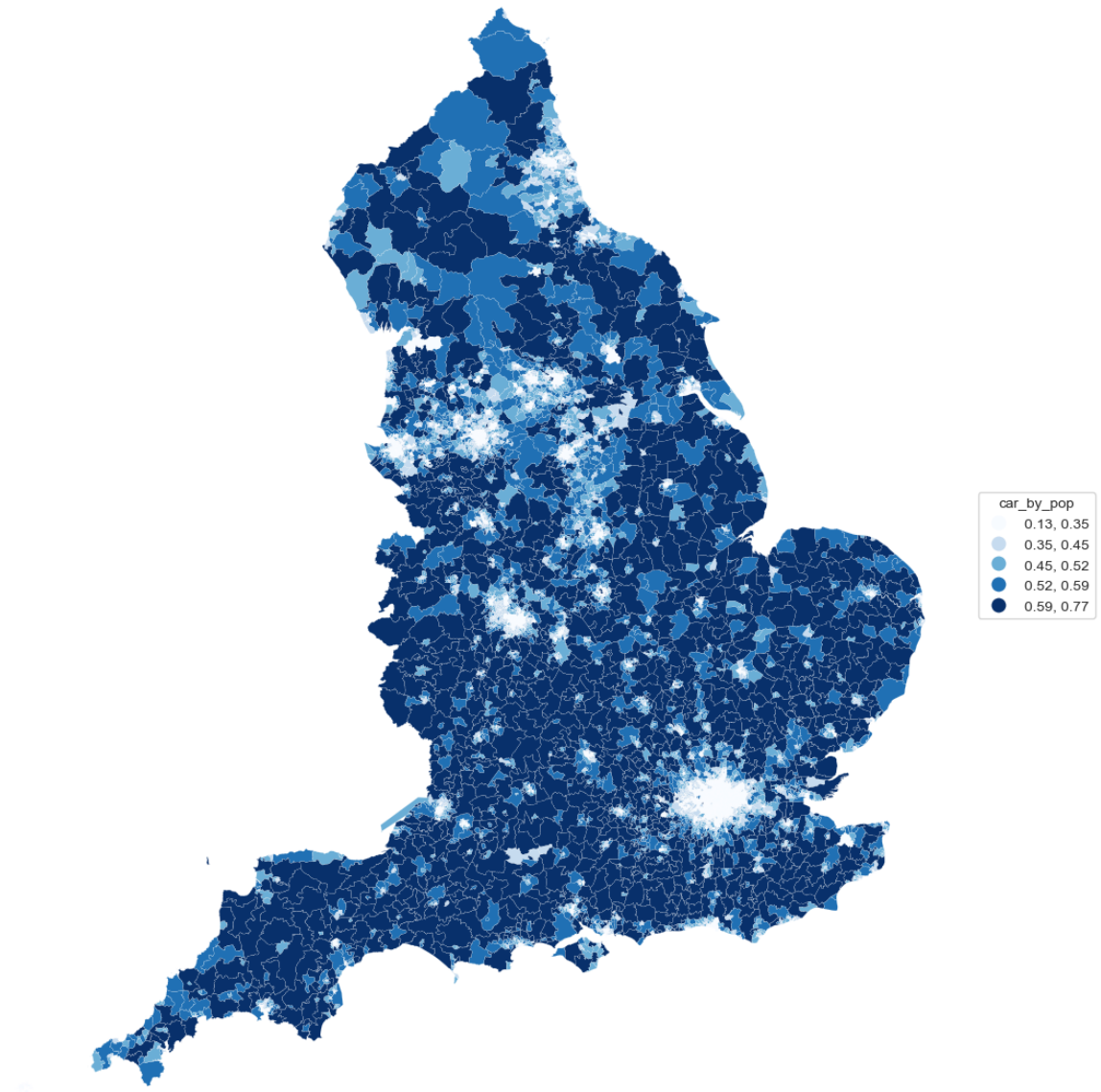


In [47]:
```
sns.displot(df_msoa['car_by_pop'], bins=10, kde=True);
```



In [48]:
```
df_msoa= gpd.GeoDataFrame(df_msoa, geometry='geometry')
car_plot = df_msoa.plot(column = 'car_by_pop', #Assign numerical data column
                        legend = True, #Decide to show legend or not
                legend_kwds={'loc': 'center left','title':'car_by_pop', 'bbox_to_ancho
                        figsize = [15,15],
                        scheme='quantiles', k=5,linewidth=0.1,cmap = 'Blues')
car_plot.axis('off')
car_plot.set_title('Car Dependency By Population', fontdict={'fontsize': 18, 'fontw
```

Out[48]:
```
Text(0.5, 1.0, 'Car Dependency By Population')
```



In [49]:
```
#display(df_msoa.describe().transpose()[['count','mean','std']])
```

In [50]:
```
df_msoa['car_by_pop'].describe()
```

Out[50]:
```
count    6791.000000
mean        0.472386
std         0.139850
min         0.130000
25%         0.380000
50%         0.490000
75%         0.580000
max         0.770000
Name: car_by_pop, dtype: float64
```

In [51]:
```
df_msoa = pd.DataFrame(df_msoa)
```

Further Data Pre-processing and Exploration

In [52]:
```
#df_msoa.columns.tolist()
```

In [53]:
```
#create dataframe to explore relationship between all variables and car by pop
# as population density has been known in literature to be inversely related to car
#also want to understand its relationship with other explanatory features in isolat
explore_by_pop = df_msoa[[
 'car_by_pop',
    'popdensity',
 'Age-Median',
    'household',
 'household1',
 'household2',
 'household>=3',
 'EducationAll',
 'EducationNo',
 'Education1',
 'Education2',
 'EducationAppren',
 'Education3',
 'Education4',
 'EducationOther',
 'income',
 'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_ALLEMP',
 'PB_ALLPS',
 'PB_ALLSS',
 'PB_TC',
 'PB_GFE',
 'PB_GGP',
 'PB_GHP',
 'PB_GPS',
 'PB_GSS',
 'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
```

```
 'stcOri',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre']]

pd.set_option('display.max_columns', None)  # or 1000
pd.set_option('display.max_rows', None)  # or 1000
matrix = pd.DataFrame(explore_by_pop.corr())
matrix.head(2)
```

Out[53]:

	car_by_pop	popdensity	Age-Median	household	household1	household2	household>=
car_by_pop	1.000000	-0.778316	0.776103	-0.166618	-0.429654	0.308891	-0.1768
popdensity	-0.778316	1.000000	-0.598154	0.136185	0.313853	-0.252039	0.19078

In [54]:
```
import dataframe_image as dfi
pd.options.display.float_format = '{:,.2f}'.format
data_described = explore_by_pop.describe().transpose()[['count','mean','std','min',
data_described  = data_described.astype({'count':'int'})
dfi.export(data_described, 'data_described.png') #export all data described into pn
display(data_described) #no null values
#Conclusion we need to standardise the data
```

	count	mean	std	min	max
car_by_pop	6791	0.47	0.14	0.13	0.77
popdensity	6791	3,567.99	3,869.24	5.64	29,452.85
Age-Median	6791	40.85	7.02	20.00	66.00
household	6791	3,248.91	683.66	989.00	6,100.00
household1	6791	981.67	352.77	249.00	3,361.00
household2	6791	1,110.94	283.31	302.00	2,455.00
household>=3	6791	1,156.31	281.96	190.00	2,730.00
EducationAll	6791	6,330.38	1,310.16	1,857.00	15,522.00
EducationNo	6791	1,422.00	537.79	82.00	3,749.00
Education1	6791	841.47	244.18	169.00	2,511.00
Education2	6791	963.72	242.01	263.00	2,764.00
EducationAppren	6791	225.73	100.32	11.00	792.00
Education3	6791	781.86	400.27	292.00	7,925.00
Education4	6791	1,733.08	864.70	254.00	7,243.00
EducationOther	6791	362.51	225.13	88.00	2,203.00
income	6791	43,888.20	9,752.65	21,300.00	86,100.00
PB_ALLFE	6791	18.40	12.25	4.16	121.00
PB_ALLGP	6791	11.17	8.83	2.67	105.04
PB_ALLHP	6791	34.98	19.44	4.96	121.00
PB_ALLEMP	6791	28.59	19.28	2.82	121.00
PB_ALLPS	6791	8.51	5.39	2.75	121.00
PB_ALLSS	6791	15.67	10.40	4.12	121.00
PB_TC	6791	17.15	11.38	3.63	121.00
PB_GFE	6791	24.99	16.45	4.22	121.00
PB_GGP	6791	11.45	8.95	2.91	105.04
PB_GHP	6791	59.23	31.84	5.98	121.00
PB_GPS	6791	9.08	5.85	2.75	121.00
PB_GSS	6791	18.20	11.79	4.51	121.00
sdbAre	6791	60.42	10.40	31.67	136.28
sdbHei	6791	7.70	1.32	2.70	23.10
sdbPer	6791	34.31	3.38	23.38	58.28
sdbVol	6791	468.71	133.39	111.22	2,974.25
ssbFoF	6791	1.06	0.15	0.66	1.78

	count	mean	std	min	max
ssbVFR	6791	1.75	0.14	1.26	2.47
ssbCCo	6791	0.52	0.05	0.25	0.63
ssbCor	6791	5.50	1.00	4.00	10.00
ssbSqu	6791	0.54	0.42	0.01	13.01
ssbERI	6791	0.97	0.02	0.86	1.00
ssbElo	6791	0.63	0.09	0.26	0.87
ssbCCM	6791	5.52	0.43	3.87	9.31
ssbCCD	6791	0.62	0.43	0.00	2.60
stbOri	6791	19.97	7.19	0.39	44.35
sdcLAL	6791	37.95	8.63	19.31	139.54
sdcAre	6791	381.06	228.55	92.90	3,897.85
sicFAR	6791	0.53	0.29	0.04	4.11
sscCCo	6791	0.32	0.04	0.18	0.48
sscERI	6791	0.95	0.01	0.89	0.99
stcOri	6791	20.68	6.04	1.08	42.78
sicCAR	6791	0.20	0.07	0.02	0.63
stbCeA	6791	2.49	1.75	0.26	8.37
mtbAli	6791	2.39	1.49	0.17	8.88
mtbNDi	6791	11.64	3.59	0.00	52.71
mtcWNe	6791	0.05	0.01	0.02	0.08
mdcAre	6791	4,467.43	7,951.32	502.07	140,384.52
ltcWRE	6791	0.00	0.00	0.00	0.00
ltbIBD	6791	21.80	21.89	4.15	250.15
ltcRea	6791	37.26	3.88	20.00	53.00
ltcAre	6791	84,683.91	226,476.09	2,989.12	4,573,089.08
sdsSPW	6791	26.86	3.83	13.21	41.21
sdsSPH	6791	7.82	1.53	2.96	28.51
sdsSPO	6791	0.53	0.16	0.12	0.97
sdsSWD	6791	2.95	0.62	0.66	5.15
sdsSHD	6791	1.11	0.45	0.21	6.33
sdsSPR	6791	0.31	0.11	0.13	1.92
sdsLen	6791	149.93	61.44	44.25	586.07
sssLin	6791	0.99	0.01	0.86	1.00

	count	mean	std	min	max
ldsMSL	6791	2,162.46	867.28	815.62	10,023.13
mtdDeg	6791	3.00	0.02	3.00	4.00
lcdMes	6791	0.07	0.03	0.00	0.21
linP3W	6791	0.73	0.04	0.44	0.89
linP4W	6791	0.06	0.05	0.00	0.42
linPDE	6791	0.20	0.06	0.00	0.35
lcnClo	6791	0.00	0.00	0.00	0.00
ldsCDL	6791	232.30	105.70	0.00	906.99
xcnSCl	6791	0.00	0.01	0.00	0.10
mtdMDi	6791	120.54	45.25	42.64	436.04
lddNDe	6791	0.01	0.00	0.00	0.02
linWID	6791	0.01	0.00	0.00	0.04
ldeAre	6791	295,066.89	506,480.44	2,095.50	12,787,759.43
ldePer	6791	2,322.56	1,815.72	232.12	18,722.01
lseCCo	6791	0.39	0.05	0.12	0.65
lseERI	6791	0.85	0.08	0.41	1.05
lseCWA	6791	531.45	417.25	52.17	4,660.63
lteOri	6791	20.63	8.04	0.62	43.85
lteWNB	6791	0.01	0.00	0.00	0.03
lieWCe	6791	0.00	0.00	0.00	0.09
stbSAl	6791	5.91	2.60	0.30	21.44
ldbPWL	6791	215,554.90	150,190.61	3,882.03	871,384.21
mtbSWR	6791	1.00	0.01	0.91	1.00
sddAre	6791	23,089.13	34,983.61	1,492.11	344,525.08
sdsAre	6791	37,110.85	65,441.37	3,023.50	735,664.64
sisBpM	6791	0.18	0.06	0.03	0.93
misCel	6791	76.11	25.63	22.52	267.00
mdsAre	6791	134,201.74	236,649.12	12,050.65	2,912,289.66
lisCel	6791	270.16	116.94	84.78	1,430.00
ldsAre	6791	577,311.85	1,012,206.06	46,341.48	12,204,228.87

In [55]:
```
## Standardise the dataset
# example of a standardization for ease in exploratory analysis
from numpy import asarray
```

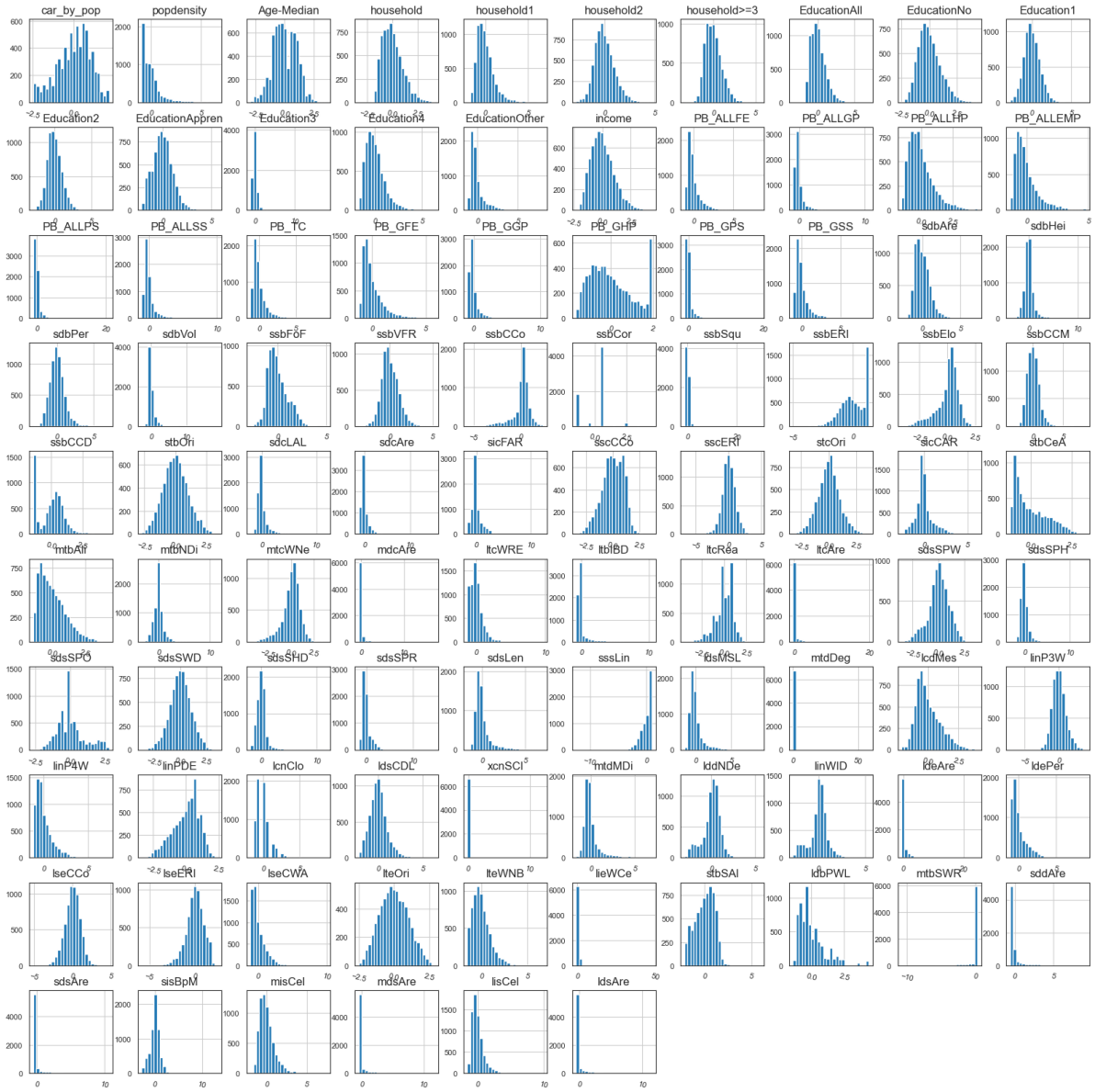
```
from sklearn.preprocessing import StandardScaler
# define data
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit_transform(explore_by_pop)
#print(scaled)
colum = explore_by_pop.columns
explore_by_scaled = pd.DataFrame(scaled)
explore_by_scaled.columns = explore_by_pop.columns
explore_by_scaled.head(1)
```

Out[55]:

	car_by_pop	popdensity	Age-Median	household	household1	household2	household>=3	Educati
0	-1.73	0.05	-0.83	1.66	4.22	0.87	-2.13	

In [56]:

```
sns.set_style("white")
explore_by_scaled.hist(bins=25,figsize=(20,20),xlabelsize='8',ylabelsize='8',xrot=-
plt.title('Histogram for every feature')
plt.show()
```



In [57]:
```
# Compute a correlation matrix and convert to long-form
corr_mat = explore_by_scaled.corr().stack().reset_index(name="correlation")

cmap=sns.diverging_palette(220,300,s=75, l=50, sep=1, n=6, as_cmap=True,center='lig
# Draw each cell as a scatter point with varying size and color
g = sns.relplot(
    data=corr_mat,
    x="level_0", y="level_1", hue="correlation", size="correlation",
    palette=cmap, hue_norm=(-1, 1), edgecolor=".7",
    height=17, sizes=(20, 150), size_norm=(-0.2, 0.8),legend=True)

# Tweak the figure to finalize
g.set(xlabel="", ylabel="", aspect="equal")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
g.legend.get_title().set_fontsize(18)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".9")
```

```
for text in g.legend.texts:
    text.set_fontsize(16)
```

In [58]:
```
explore_by_scaled.corr(method='pearson').round(2) #table with all scores to check f
```

Out[58]:

	car_by_pop	popdensity	Age-Median	household	household1	household2	household
car_by_pop	1.00	-0.78	0.78	-0.17	-0.43	0.31	
popdensity	-0.78	1.00	-0.60	0.14	0.31	-0.25	
Age-Median	0.78	-0.60	1.00	-0.10	-0.24	0.38	
household	-0.17	0.14	-0.10	1.00	0.80	0.80	
household1	-0.43	0.31	-0.24	0.80	1.00	0.50	
household2	0.31	-0.25	0.38	0.80	0.50	1.00	
household>=3	-0.18	0.19	-0.32	0.63	0.17	0.30	
EducationAll	-0.18	0.16	-0.18	0.90	0.59	0.68	
EducationNo	-0.19	-0.03	-0.14	0.45	0.36	0.25	
Education1	0.07	-0.16	-0.06	0.56	0.25	0.43	
Education2	0.31	-0.34	0.20	0.66	0.26	0.68	
EducationAppren	0.53	-0.52	0.50	0.32	0.02	0.60	
Education3	-0.18	0.10	-0.28	0.36	0.27	0.25	
Education4	-0.09	0.27	-0.02	0.45	0.32	0.40	
EducationOther	-0.58	0.59	-0.50	0.35	0.33	-0.08	
income	0.21	0.07	0.12	-0.05	-0.23	0.05	
PB_ALLFE	0.50	-0.42	0.46	-0.10	-0.22	0.20	
PB_ALLGP	0.52	-0.41	0.45	-0.12	-0.26	0.19	
PB_ALLHP	0.54	-0.46	0.48	-0.10	-0.25	0.21	
PB_ALLEMP	0.49	-0.44	0.57	-0.07	-0.20	0.25	
PB_ALLPS	0.46	-0.36	0.43	-0.10	-0.21	0.17	
PB_ALLSS	0.49	-0.39	0.44	-0.11	-0.22	0.18	
PB_TC	0.49	-0.41	0.42	-0.13	-0.28	0.17	
PB_GFE	0.45	-0.39	0.44	-0.07	-0.19	0.22	
PB_GGP	0.52	-0.41	0.45	-0.12	-0.26	0.19	
PB_GHP	0.37	-0.37	0.30	-0.11	-0.20	0.11	
PB_GPS	0.45	-0.37	0.42	-0.10	-0.20	0.17	
PB_GSS	0.47	-0.41	0.43	-0.08	-0.19	0.21	
sdbAre	0.31	-0.10	0.44	-0.01	-0.06	0.20	
sdbHei	-0.71	0.74	-0.57	0.18	0.43	-0.19	
sdbPer	0.12	0.08	0.28	0.05	0.07	0.16	
sdbVol	-0.27	0.42	-0.09	0.12	0.27	0.03	

	car_by_pop	popdensity	Age-Median	household	household1	household2	househo
ssbFoF	0.78	-0.66	0.75	-0.14	-0.37	0.32	
ssbVFR	0.48	-0.30	0.55	-0.07	-0.18	0.23	
ssbCCo	0.41	-0.51	0.26	-0.15	-0.29	0.05	
ssbCor	-0.13	0.19	-0.04	0.07	0.12	-0.01	
ssbSqu	-0.29	0.31	-0.17	0.05	0.17	-0.09	
ssbERI	0.22	-0.29	0.09	-0.10	-0.20	0.01	
ssbElo	0.51	-0.57	0.38	-0.17	-0.33	0.10	
ssbCCM	0.03	0.16	0.21	0.08	0.14	0.16	
ssbCCD	-0.23	0.31	-0.11	0.11	0.20	-0.01	
stbOri	-0.02	0.01	-0.03	-0.02	0.01	-0.03	
sdcLAL	0.56	-0.44	0.52	-0.17	-0.29	0.16	
sdcAre	0.60	-0.47	0.56	-0.16	-0.29	0.19	
sicFAR	-0.75	0.80	-0.57	0.21	0.47	-0.17	
sscCCo	0.68	-0.59	0.62	-0.05	-0.19	0.35	
sscERI	-0.13	0.06	-0.08	0.11	0.17	0.08	
stcOri	0.03	-0.03	0.01	-0.03	-0.01	-0.01	
sicCAR	-0.75	0.77	-0.59	0.21	0.44	-0.20	
stbCeA	0.79	-0.59	0.71	-0.16	-0.38	0.29	
mtbAli	0.64	-0.54	0.59	-0.12	-0.27	0.26	
mtbNDi	0.70	-0.62	0.60	-0.20	-0.39	0.20	
mtcWNe	-0.43	0.38	-0.44	0.15	0.23	-0.12	
mdcAre	0.37	-0.27	0.37	-0.11	-0.18	0.12	
ltcWRE	-0.65	0.65	-0.52	0.11	0.30	-0.24	
ltbIBD	0.47	-0.37	0.45	-0.13	-0.23	0.15	
ltcRea	0.44	-0.43	0.29	-0.05	-0.14	0.18	
ltcAre	0.36	-0.27	0.37	-0.11	-0.18	0.13	
sdsSPW	0.66	-0.62	0.51	-0.19	-0.38	0.14	
sdsSPH	-0.69	0.74	-0.52	0.19	0.45	-0.15	
sdsSPO	0.66	-0.63	0.53	-0.12	-0.26	0.25	
sdsSWD	0.62	-0.58	0.54	-0.07	-0.21	0.29	
sdsSHD	-0.18	0.32	-0.02	0.15	0.30	0.14	
sdsSPR	-0.72	0.76	-0.54	0.21	0.47	-0.14	

	car_by_pop	popdensity	Age-Median	household	household1	household2	househo
sdsLen	0.42	-0.31	0.43	-0.14	-0.27	0.12	
sssLin	-0.50	0.45	-0.42	0.12	0.26	-0.18	
ldsMSL	0.30	-0.18	0.34	-0.12	-0.21	0.09	
mtdDeg	-0.02	0.04	-0.02	-0.01	-0.01	-0.01	
lcdMes	-0.62	0.56	-0.46	0.09	0.30	-0.23	
linP3W	-0.28	0.25	-0.20	0.04	0.12	-0.08	
linP4W	-0.55	0.53	-0.40	0.08	0.28	-0.21	
linPDE	0.65	-0.59	0.48	-0.11	-0.32	0.22	
lcnClo	-0.37	0.34	-0.33	0.14	0.32	-0.08	
ldsCDL	0.68	-0.60	0.58	-0.14	-0.33	0.24	
xcnSCl	-0.19	0.25	-0.15	-0.01	0.01	-0.11	
mtdMDi	0.48	-0.36	0.47	-0.15	-0.30	0.14	
lddNDe	-0.46	0.35	-0.47	0.14	0.31	-0.15	
linWID	-0.63	0.51	-0.57	0.16	0.39	-0.20	
ldeAre	0.48	-0.37	0.41	-0.12	-0.26	0.16	
ldePer	0.60	-0.50	0.49	-0.14	-0.33	0.21	
lseCCo	0.18	-0.16	0.13	-0.02	-0.05	0.08	
lseERI	-0.29	0.32	-0.19	0.08	0.18	-0.08	
lseCWA	0.57	-0.48	0.46	-0.14	-0.32	0.19	
lteOri	0.00	-0.01	-0.01	-0.03	-0.01	-0.03	
lteWNB	-0.71	0.66	-0.58	0.16	0.43	-0.23	
lieWCe	0.02	-0.04	0.00	-0.05	-0.06	-0.04	
stbSAl	0.65	-0.59	0.51	-0.12	-0.32	0.24	
ldbPWL	0.40	-0.41	0.40	-0.01	-0.09	0.26	
mtbSWR	0.34	-0.33	0.26	-0.04	-0.18	0.12	
sddAre	0.46	-0.34	0.43	-0.13	-0.25	0.15	
sdsAre	0.45	-0.33	0.43	-0.12	-0.23	0.15	
sisBpM	-0.64	0.59	-0.59	0.09	0.20	-0.30	
misCel	-0.37	0.41	-0.32	-0.02	-0.04	-0.25	
mdsAre	0.44	-0.33	0.43	-0.12	-0.22	0.16	
lisCel	-0.45	0.50	-0.36	0.01	0.02	-0.26	
ldsAre	0.43	-0.33	0.42	-0.11	-0.21	0.16	

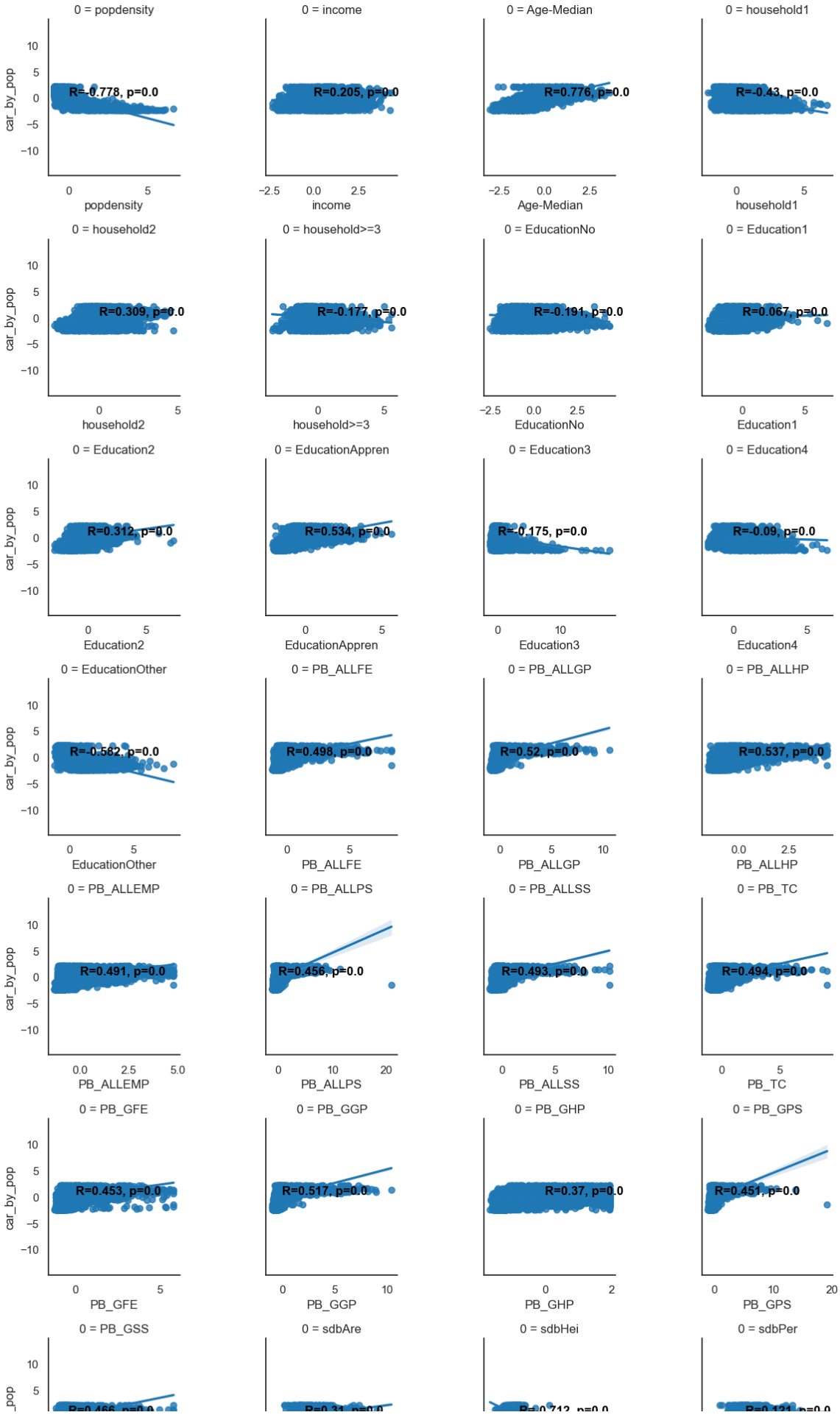
See all Scatterplots of all 96 Features

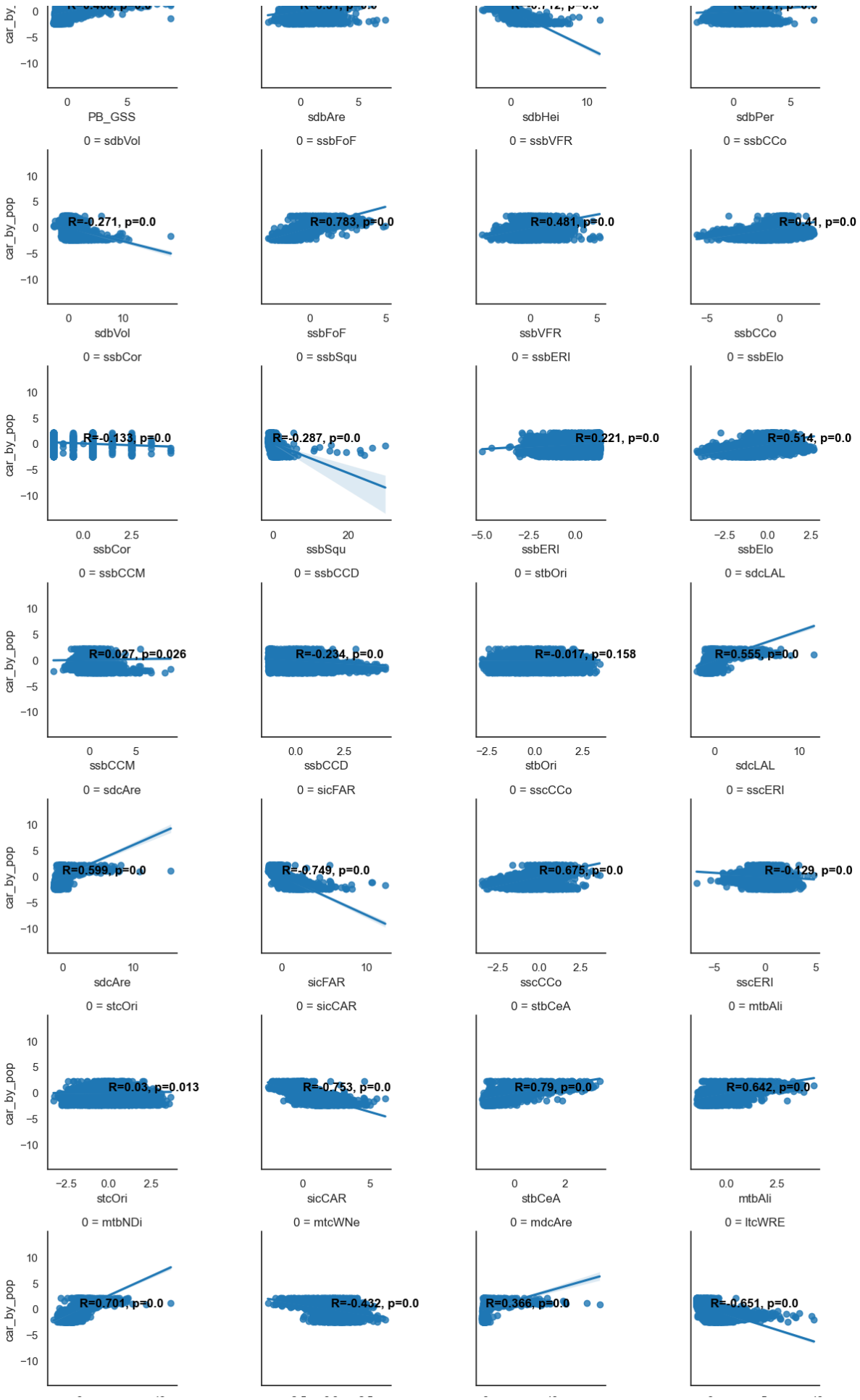
In [59]:
```
sns.set_theme(style="white", palette=None)
#MissingAccess to Public Transport
from scipy import stats
x_vars= ['popdensity',
'income',
'Age-Median',
 'household1',
 'household2',
 'household>=3',
 'EducationNo',
 'Education1',
 'Education2',
 'EducationAppren',
 'Education3',
 'Education4',
 'EducationOther',
'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_ALLEMP',
 'PB_ALLPS',
 'PB_ALLSS',
 'PB_TC',
 'PB_GFE',
 'PB_GGP',
 'PB_GHP',
 'PB_GPS',
 'PB_GSS',
'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
 'sicCAR',
 'stbCeA',
 'mtbAli',
```

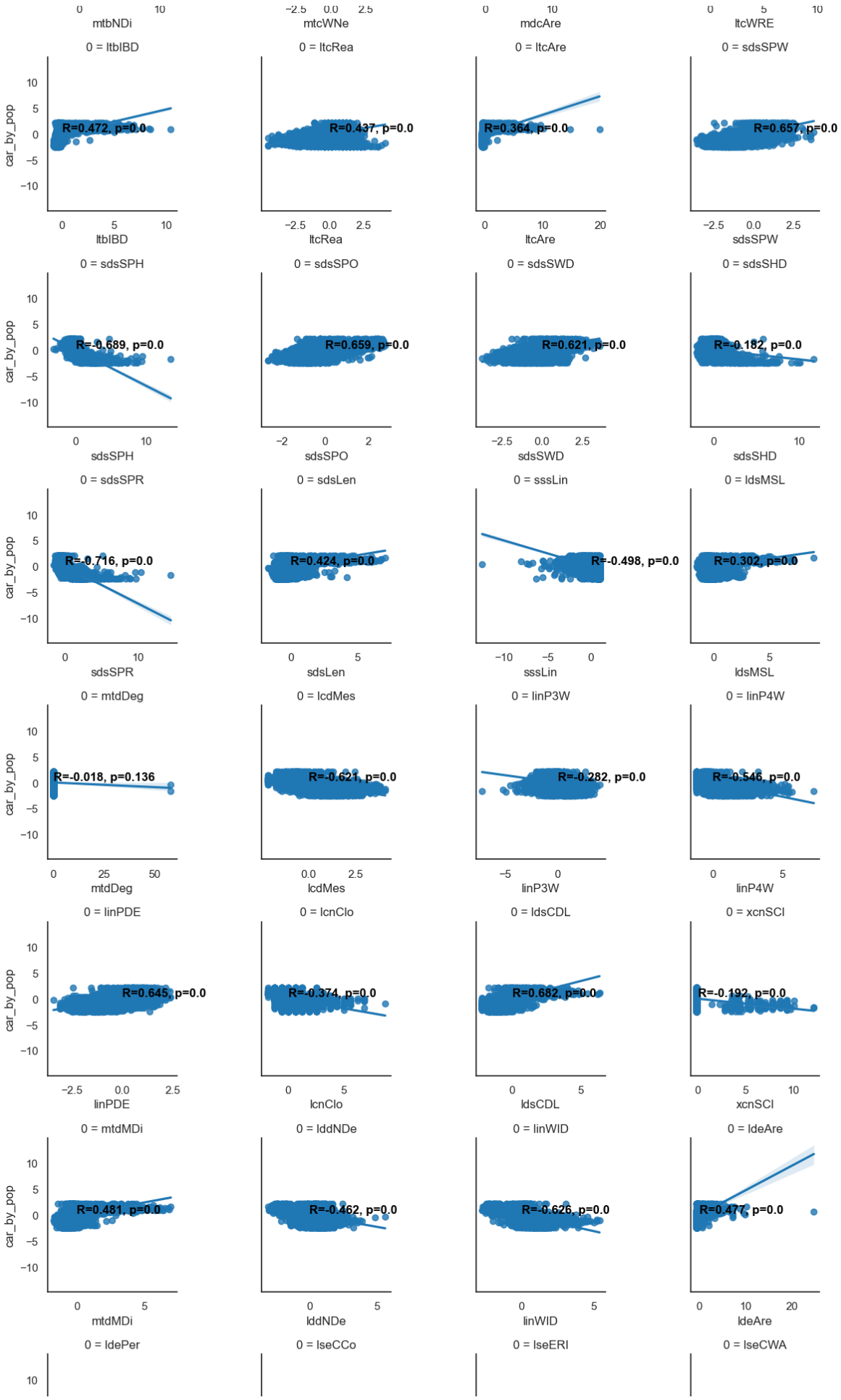
```
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre'
]

y_vars='car_by_pop'


g = sns.FacetGrid(pd.DataFrame(x_vars), col=0, col_wrap=4, sharex=False)

for ax, x_var in zip(g.axes, x_vars):
        sns.regplot(data=explore_by_scaled, x=x_var, y=y_vars,ax=ax)
        r, p = round(stats.pearsonr(explore_by_scaled[y_vars], explore_by_scaled[x_
        ax.text(0, 0, "R={}, p={}".format(r,p), horizontalalignment='left',
```

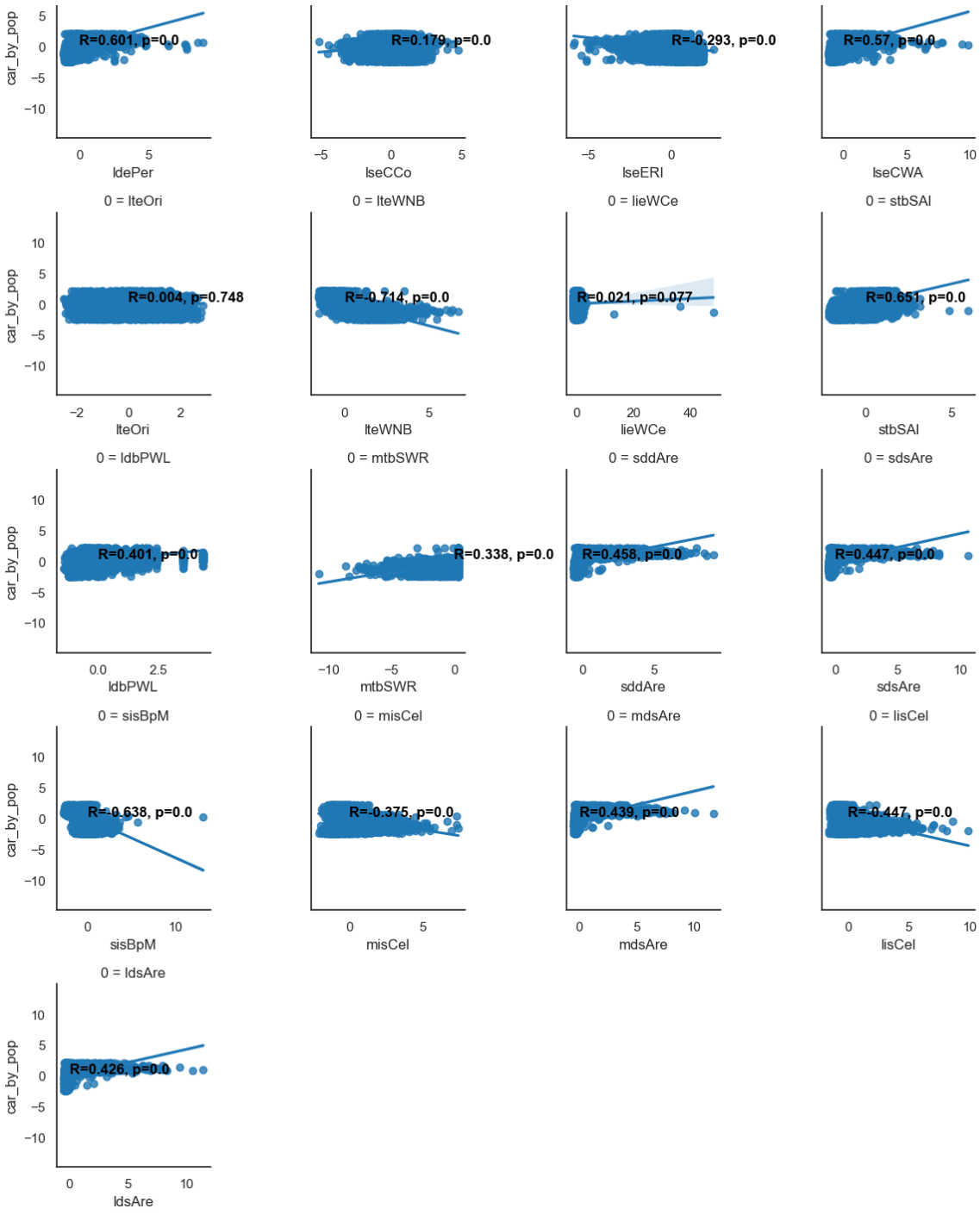
```
            verticalalignment='bottom',size='medium', color='black', weight='semibo

g.tight_layout()
#for i in range(len(x_vars)):
 #   print('car_by_pop with ' + x_vars[i])
  #  print(stats.pearsonr(explore_by_scaled[y_vars], explore_by_scaled[x_vars[i]]))
```

Out[59]: <seaborn.axisgrid.FacetGrid at 0x21963d800d0>


0 = popdensity
R=-0.778, p=0.0
popdensity
0 = income
R=0.205, p=0.0
income
0 = Age-Median
R=0.776, p=0.0
Age-Median
0 = household1
R=-0.43, p=0.0
household1
0 = household2
R=0.309, p=0.0
household2
0 = household>=3
R=-0.177, p=0.0
household>=3
0 = EducationNo
R=-0.191, p=0.0
EducationNo
0 = Education1
R=0.067, p=0.0
Education1
0 = Education2
R=0.312, p=0.0
Education2
0 = EducationAppren
R=0.534, p=0.0
EducationAppren
0 = Education3
R=-0.175, p=0.0
Education3
0 = Education4
R=-0.09, p=0.0
Education4
0 = EducationOther
R=-0.562, p=0.0
EducationOther
0 = PB_ALLFE
R=0.498, p=0.0
PB_ALLFE
0 = PB_ALLGP
R=0.52, p=0.0
PB_ALLGP
0 = PB_ALLHP
R=0.537, p=0.0
PB_ALLHP
0 = PB_ALLEMP
R=0.491, p=0.0
PB_ALLEMP
0 = PB_ALLPS
R=0.456, p=0.0
PB_ALLPS
0 = PB_ALLSS
R=0.493, p=0.0
PB_ALLSS
0 = PB_TC
R=0.494, p=0.0
PB_TC
0 = PB_GFE
R=0.453, p=0.0
PB_GFE
0 = PB_GGP
R=0.517, p=0.0
PB_GGP
0 = PB_GHP
R=0.37, p=0.0
PB_GHP
0 = PB_GPS
R=0.451, p=0.0
PB_GPS
0 = PB_GSS
0 = sdbAre
0 = sdbHei
0 = sdbPer
car_by_pop


car_by_pop
PB_GSS
sdbAre
sdbHei
sdbPer
0 = sdbVol
R=-0.271, p=0.0
0 = ssbFoF
R=0.783, p=0.0
0 = ssbVFR
R=0.481, p=0.0
0 = ssbCCo
R=0.41, p=0.0
sdbVol
ssbFoF
ssbVFR
ssbCCo
0 = ssbCor
R=-0.133, p=0.0
0 = ssbSqu
R=-0.287, p=0.0
0 = ssbERI
R=0.221, p=0.0
0 = ssbElo
R=0.514, p=0.0
ssbCor
ssbSqu
ssbERI
ssbElo
0 = ssbCCM
R=0.027, p=0.026
0 = ssbCCD
R=-0.234, p=0.0
0 = stbOri
R=-0.017, p=0.158
0 = sdcLAL
R=0.555, p=0.0
ssbCCM
ssbCCD
stbOri
sdcLAL
0 = sdcAre
R=0.599, p=0.0
0 = sicFAR
R=-0.749, p=0.0
0 = sscCCo
R=0.675, p=0.0
0 = sscERI
R=-0.129, p=0.0
sdcAre
sicFAR
sscCCo
sscERI
0 = stcOri
R=0.03, p=0.013
0 = sicCAR
R=-0.753, p=0.0
0 = stbCeA
R=0.79, p=0.0
0 = mtbAli
R=0.642, p=0.0
stcOri
sicCAR
stbCeA
mtbAli
0 = mtbNDi
R=0.701, p=0.0
0 = mtcWNe
R=-0.432, p=0.0
0 = mdcAre
R=0.366, p=0.0
0 = ltcWRE
R=-0.651, p=0.0


mtbNDi
0 = ltbIBD
R=0.472, p=0.0
ltbIBD
mtcWNe
0 = ltcRea
R=0.437, p=0.0
ltcRea
mdcAre
0 = ltcAre
R=0.364, p=0.0
ltcAre
ltcWRE
0 = sdsSPW
R=0.657, p=0.0
sdsSPW
0 = sdsSPH
R=-0.689, p=0.0
sdsSPH
0 = sdsSPO
R=0.659, p=0.0
sdsSPO
0 = sdsSWD
R=0.621, p=0.0
sdsSWD
0 = sdsSHD
R=-0.182, p=0.0
sdsSHD
0 = sdsSPR
R=-0.716, p=0.0
sdsSPR
0 = sdsLen
R=0.424, p=0.0
sdsLen
0 = sssLin
R=-0.498, p=0.0
sssLin
0 = ldsMSL
R=0.302, p=0.0
ldsMSL
0 = mtdDeg
R=-0.018, p=0.136
mtdDeg
0 = lcdMes
R=-0.621, p=0.0
lcdMes
0 = linP3W
R=-0.282, p=0.0
linP3W
0 = linP4W
R=-0.546, p=0.0
linP4W
0 = linPDE
R=0.645, p=0.0
linPDE
0 = lcnClo
R=-0.374, p=0.0
lcnClo
0 = ldsCDL
R=0.682, p=0.0
ldsCDL
0 = xcnSCl
R=-0.192, p=0.0
xcnSCl
0 = mtdMDi
R=0.481, p=0.0
mtdMDi
0 = lddNDe
R=-0.462, p=0.0
lddNDe
0 = linWID
R=-0.626, p=0.0
linWID
0 = ldeAre
R=0.477, p=0.0
ldeAre
0 = ldePer
0 = lseCCo
0 = lseERI
0 = lseCWA
car_by_pop



Filter only Features with High Absolute Value (Pos, Neg) Correlation >0.5

In [60]:
```
#filter only features with higher correlation towards car by pop than 0.5
filter_corr = explore_by_scaled.corr(method='pearson').round(2)
filter_corr = filter_corr[abs(filter_corr['car_by_pop'])>=0.5]
filter_corr[abs(filter_corr['car_by_pop'])>=0.5].index.tolist()
selected_feature_esda = filter_corr.loc[:,filter_corr.columns.isin(filter_corr[abs(
selected_feature_list = selected_feature_esda.index.tolist()
display(filter_corr[abs(filter_corr['car_by_pop'])>=0.5])
```

	car_by_pop	popdensity	Age-Median	household	household1	household2	househo
car_by_pop	1.00	-0.78	0.78	-0.17	-0.43	0.31	
popdensity	-0.78	1.00	-0.60	0.14	0.31	-0.25	
Age-Median	0.78	-0.60	1.00	-0.10	-0.24	0.38	
EducationAppren	0.53	-0.52	0.50	0.32	0.02	0.60	
EducationOther	-0.58	0.59	-0.50	0.35	0.33	-0.08	
PB_ALLFE	0.50	-0.42	0.46	-0.10	-0.22	0.20	
PB_ALLGP	0.52	-0.41	0.45	-0.12	-0.26	0.19	
PB_ALLHP	0.54	-0.46	0.48	-0.10	-0.25	0.21	
PB_GGP	0.52	-0.41	0.45	-0.12	-0.26	0.19	
sdbHei	-0.71	0.74	-0.57	0.18	0.43	-0.19	
ssbFoF	0.78	-0.66	0.75	-0.14	-0.37	0.32	
ssbElo	0.51	-0.57	0.38	-0.17	-0.33	0.10	
sdcLAL	0.56	-0.44	0.52	-0.17	-0.29	0.16	
sdcAre	0.60	-0.47	0.56	-0.16	-0.29	0.19	
sicFAR	-0.75	0.80	-0.57	0.21	0.47	-0.17	
sscCCo	0.68	-0.59	0.62	-0.05	-0.19	0.35	
sicCAR	-0.75	0.77	-0.59	0.21	0.44	-0.20	
stbCeA	0.79	-0.59	0.71	-0.16	-0.38	0.29	
mtbAli	0.64	-0.54	0.59	-0.12	-0.27	0.26	
mtbNDi	0.70	-0.62	0.60	-0.20	-0.39	0.20	
ltcWRE	-0.65	0.65	-0.52	0.11	0.30	-0.24	
sdsSPW	0.66	-0.62	0.51	-0.19	-0.38	0.14	
sdsSPH	-0.69	0.74	-0.52	0.19	0.45	-0.15	
sdsSPO	0.66	-0.63	0.53	-0.12	-0.26	0.25	
sdsSWD	0.62	-0.58	0.54	-0.07	-0.21	0.29	
sdsSPR	-0.72	0.76	-0.54	0.21	0.47	-0.14	
sssLin	-0.50	0.45	-0.42	0.12	0.26	-0.18	
lcdMes	-0.62	0.56	-0.46	0.09	0.30	-0.23	
linP4W	-0.55	0.53	-0.40	0.08	0.28	-0.21	
linPDE	0.65	-0.59	0.48	-0.11	-0.32	0.22	
ldsCDL	0.68	-0.60	0.58	-0.14	-0.33	0.24	
linWID	-0.63	0.51	-0.57	0.16	0.39	-0.20	

	car_by_pop	popdensity	Age-Median	household	household1	household2	househo
ldePer	0.60	-0.50	0.49	-0.14	-0.33	0.21	
lseCWA	0.57	-0.48	0.46	-0.14	-0.32	0.19	
lteWNB	-0.71	0.66	-0.58	0.16	0.43	-0.23	
stbSAl	0.65	-0.59	0.51	-0.12	-0.32	0.24	
sisBpM	-0.64	0.59	-0.59	0.09	0.20	-0.30	

In [61]:
```
pd.options.display.float_format = "{:,.6f}".format
#display pvalue for all correlation matrix
#check p-value of features with correlation with car ownership higher than 0.5 abso
#select features in a list for further observation on ESDA (Exploratory Data Analys
from scipy.stats import pearsonr

def corr_pval(df):
    corr_pval_df = pd.DataFrame(index=df.columns, columns=df.columns)
    for i in range(len(corr_pval_df.index)):
        for c in range(len(corr_pval_df.columns)):
            corr_pval_df.iloc[i, c] = round(pearsonr(df[corr_pval_df.index[i]], df[
    return corr_pval_df

pvalue_corr = corr_pval(explore_by_scaled)
display(pvalue_corr)
selected_feature_esda = pvalue_corr.loc[pvalue_corr.index.isin(selected_feature_lis
selected_feature_esda
esda_selectedfeatures = selected_feature_esda.index.tolist()
```

	car_by_pop	popdensity	Age-Median	household	household1	household2	household
car_by_pop	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
popdensity	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
Age-Median	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
household	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
household1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
household2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
household>=3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
EducationAll	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
EducationNo	0.000000	0.005631	0.000000	0.000000	0.000000	0.000000	0.
Education1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
Education2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
EducationAppren	0.000000	0.000000	0.000000	0.000000	0.196075	0.000000	0.
Education3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
Education4	0.000000	0.000000	0.176318	0.000000	0.000000	0.000000	0.
EducationOther	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
income	0.000000	0.000000	0.000000	0.000102	0.000000	0.000009	0.
PB_ALLFE	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_ALLGP	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_ALLHP	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_ALLEMP	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_ALLPS	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_ALLSS	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_TC	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_GFE	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_GGP	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_GHP	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_GPS	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
PB_GSS	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdbAre	0.000000	0.000000	0.000000	0.291491	0.000001	0.000000	0.
sdbHei	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdbPer	0.000000	0.000000	0.000000	0.000014	0.000000	0.000000	0.
sdbVol	0.000000	0.000000	0.000000	0.000000	0.000000	0.033021	0.

	car_by_pop	popdensity	Age-Median	household	household1	household2	househo
ssbFoF	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ssbVFR	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ssbCCo	0.000000	0.000000	0.000000	0.000000	0.000000	0.000047	0.
ssbCor	0.000000	0.000000	0.000290	0.000000	0.000000	0.447008	0.
ssbSqu	0.000000	0.000000	0.000000	0.000008	0.000000	0.000000	0.
ssbERI	0.000000	0.000000	0.000000	0.000000	0.000000	0.340727	0.
ssbElo	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ssbCCM	0.026357	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ssbCCD	0.000000	0.000000	0.000000	0.000000	0.000000	0.222999	0.
stbOri	0.157901	0.468222	0.019090	0.171985	0.418423	0.007399	0.
sdcLAL	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdcAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sicFAR	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sscCCo	0.000000	0.000000	0.000000	0.000112	0.000000	0.000000	0.
sscERI	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
stcOri	0.012846	0.004267	0.278604	0.024971	0.263327	0.247967	0.
sicCAR	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
stbCeA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
mtbAli	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
mtbNDi	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
mtcWNe	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
mdcAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ltcWRE	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ltbIBD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ltcRea	0.000000	0.000000	0.000000	0.000052	0.000000	0.000000	0.
ltcAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdsSPW	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdsSPH	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdsSPO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdsSWD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdsSHD	0.000000	0.000000	0.199656	0.000000	0.000000	0.000000	0.
sdsSPR	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.

	car_by_pop	popdensity	Age-Median	household	household1	household2	househo
sdsLen	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sssLin	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ldsMSL	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
mtdDeg	0.136069	0.002078	0.092319	0.234299	0.495110	0.334212	0.
lcdMes	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
linP3W	0.000000	0.000000	0.000000	0.000372	0.000000	0.000000	0.
linP4W	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
linPDE	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lcnClo	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ldsCDL	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
xcnSCl	0.000000	0.000000	0.000000	0.464701	0.336964	0.000000	0.
mtdMDi	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lddNDe	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
linWID	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ldeAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ldePer	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lseCCo	0.000000	0.000000	0.000000	0.080410	0.000021	0.000000	0.
lseERI	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lseCWA	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lteOri	0.748248	0.271130	0.391347	0.007989	0.234455	0.008169	0.
lteWNB	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lieWCe	0.077214	0.001690	0.751439	0.000043	0.000002	0.003290	0.
stbSAl	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
ldbPWL	0.000000	0.000000	0.000000	0.623547	0.000000	0.000000	0.
mtbSWR	0.000000	0.000000	0.000000	0.003727	0.000000	0.000000	0.
sddAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sdsAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
sisBpM	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
misCel	0.000000	0.000000	0.000000	0.153251	0.000264	0.000000	0.
mdsAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
lisCel	0.000000	0.000000	0.000000	0.667446	0.152009	0.000000	0.
ldsAre	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.

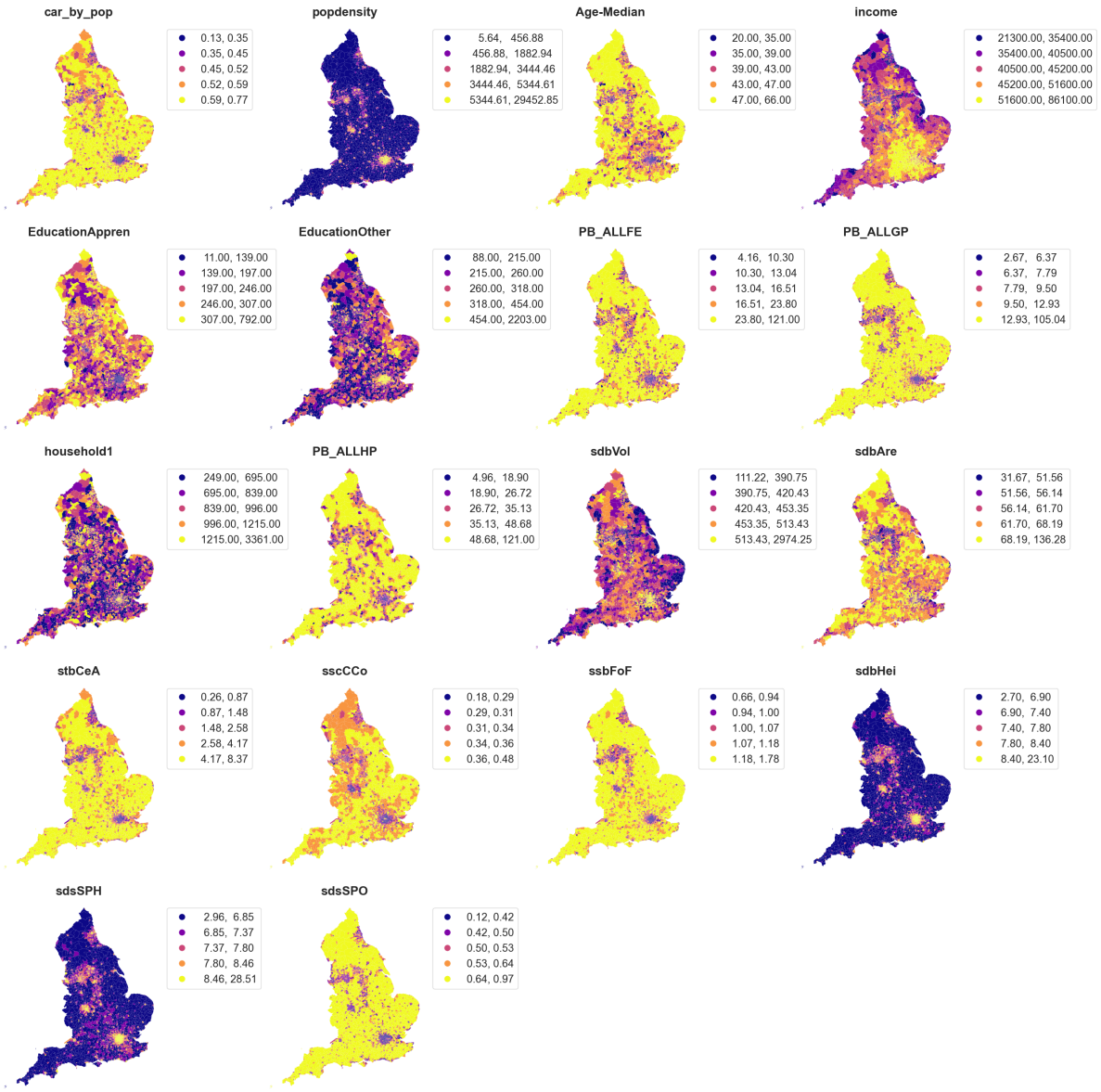
In [62]:
```
esda_selectedfeatures #all features with absolute value correlation higher than 0.5
```

Out[62]:
```
['car_by_pop',
 'popdensity',
 'Age-Median',
 'EducationAppren',
 'EducationOther',
 'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_GGP',
 'sdbHei',
 'ssbFoF',
 'ssbElo',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'ltcWRE',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSPR',
 'sssLin',
 'lcdMes',
 'linP4W',
 'linPDE',
 'ldsCDL',
 'linWID',
 'ldePer',
 'lseCWA',
 'lteWNB',
 'stbSAl',
 'sisBpM']
```

Exploratory Spatial Data Analysis (Choropleth Map) ESDA of highly Correlated Features with Car Ownership

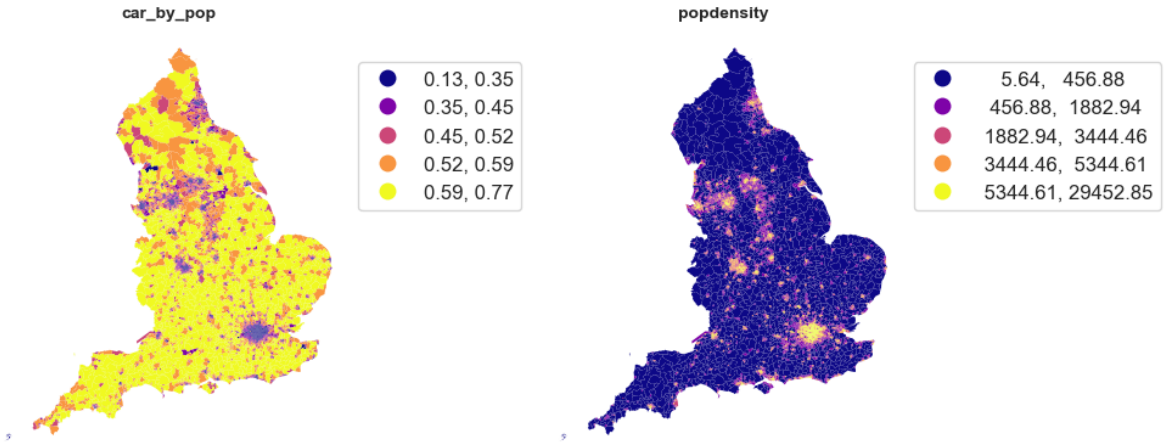
In [63]:
```
df_msoa= gpd.GeoDataFrame(df_msoa, geometry='geometry')
p_vars= ['car_by_pop',
 'popdensity',
 'Age-Median',
  'income',
 'EducationAppren',
 'EducationOther',
 'PB_ALLFE',
```

```
 'PB_ALLGP',
'household1',
 'PB_ALLHP','sdbVol','sdbAre',
'stbCeA','sscCCo',
'ssbFoF',
'sdbHei','sdsSPH','sdsSPO']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    df_msoa.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 20, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```



In [64]:
```
df_msoa= gpd.GeoDataFrame(df_msoa, geometry='geometry')
p_vars= ['car_by_pop',
 'popdensity']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  2 #nt(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
```

```
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    df_msoa.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 10, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```



In [ ]:
```
#same features as esda_selectedfeatures with features that have higher correlation

df_msoa= gpd.GeoDataFrame(df_msoa, geometry='geometry')
p_vars= [
'car_by_pop',
 'popdensity',
 'Age-Median',
 'EducationAppren',
 'EducationOther',
'household1',
 'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_GGP',
 'sdbHei',
 'ssbFoF',
 'ssbElo',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'ltcWRE',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSPR',
 'sssLin',
 'lcdMes',
 'linP4W',
```

```
 'linPDE',
 'ldsCDL',
 'linWID',
 'ldePer',
 'lseCWA',
 'lteWNB',
 'stbSAl',
 'sisBpM']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    df_msoa.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 20, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```

In [ ]:

```
#same features as esda_selectedfeatures with features that have higher correlation

df_msoa= gpd.GeoDataFrame(df_msoa, geometry='geometry')
p_vars= [
'popdensity',
'income',
'Age-Median',
 'household1',
 'household2',
 'household>=3',
 'EducationNo',
 'Education1',
 'Education2',
 'EducationAppren',
 'Education3',
 'Education4',
 'EducationOther',
'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_ALLEMP',
 'PB_ALLPS',
 'PB_ALLSS',
 'PB_TC',
```

```
 'PB_GFE',
 'PB_GGP',
 'PB_GHP',
 'PB_GPS',
 'PB_GSS',]


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    df_msoa.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 20, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```

Preparation of Data for Machine Learning

Split test and train by K-Means Cluster of Spatial Area Close to Each Other

In [65]:
```
import geopandas
df_msoa = geopandas.GeoDataFrame(df_msoa)
```

In [134…
```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
plt.scatter(x=df_msoa['LONG_'], y=df_msoa['LAT'],marker="o",s=4,c='grey')
plt.show()
```

In [67]:
```
df_msoa.head(1)
```

Out[67]:

	OBJECTID	MSOA11CD	MSOA11NM_x	LONG_	LAT	geometry	cars	Mid-2 popula
0	1	E02000001	City of London 001	-0.093490	51.515600	POLYGON ((-0.09650 51.52295, -0.09644 51.52282...	2,471.000000	10,938.00

In [68]:
```
# Create dataframe with the Longitude and Latitude to use K-Means to create a clust
clust=df_msoa.loc[:,['MSOA11CD','LONG_','LAT']]
clust.head(1)
```

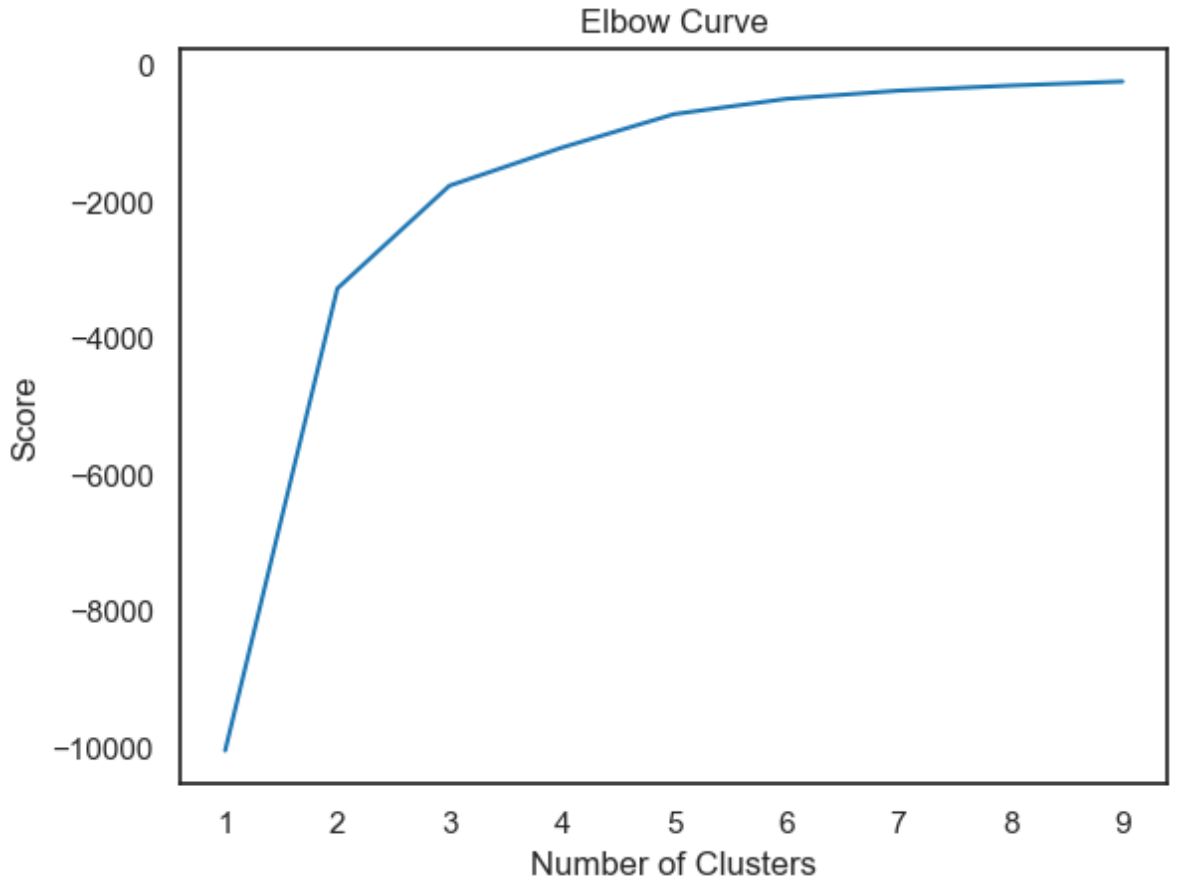
Out[68]:

	MSOA11CD	LONG_	LAT
0	E02000001	-0.093490	51.515600

K-Means to Group create 4 Clusters by Spatial Area for K-Fold Cross Validation for Linear Regression and RainForest

In [69]:
```
K_clusters = range(1,10)
kmeans = [KMeans(n_clusters=i) for i in K_clusters]
Y_axis = clust[['LONG_']]
X_axis = clust[['LAT']]
score = [kmeans[i].fit(Y_axis).score(Y_axis) for i in range(len(kmeans))]
```

```
# Visualize
plt.plot(K_clusters, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



In [70]:
```
kmeans = KMeans(n_clusters = 4, init ='k-means++',random_state=1)
kmeans.fit(clust[clust.columns[1:3]]) # Compute k-means clustering.
clust['cluster_label'] = kmeans.fit_predict(clust[clust.columns[1:3]])
centers = kmeans.cluster_centers_ # Coordinates of cluster centers.
labels = kmeans.predict(clust[clust.columns[1:3]]) # Labels of each point
clust.head(2)
```

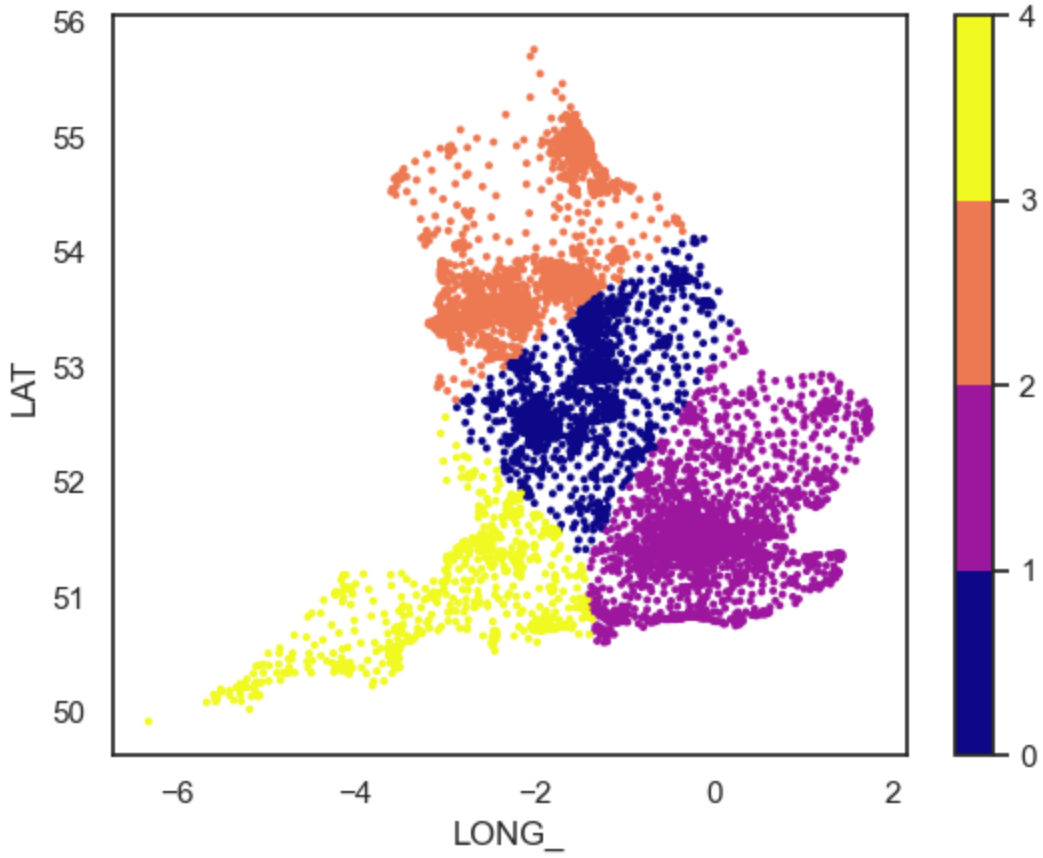
Out[70]:

	MSOA11CD	LONG_	LAT	cluster_label
0	E02000001	-0.093490	51.515600	1
1	E02000002	0.138756	51.586500	1

In [71]:
```
from matplotlib import pyplot as plt
import numpy as np
from matplotlib import colors
cmap = plt.cm.plasma
norm = colors.BoundaryNorm(np.arange(0, 5, 1), cmap.N)
clust.plot.scatter(x = 'LONG_', y = 'LAT', c=labels, cmap=cmap, norm=norm, edgecolo
plt.figure(figsize=(5,5))
```

```
print('How Many MSOA in Each Cluster')
print(clust.groupby('cluster_label').size()) #find how many MSOA in each cluster
```

```
How Many MSOA in Each Cluster
cluster_label
0    1597
1    2718
2    1713
3     763
dtype: int64
```




```
<Figure size 500x500 with 0 Axes>
```

In [72]:
```
clust = clust.sort_values(by=['cluster_label']).reset_index(drop=True) #Sort by Clu
```

In [73]:
```
clust.head(1)
```

Out[73]:

	MSOA11CD	LONG_	LAT	cluster_label
0	E02003212	-1.700090	51.633400	0

In [74]:
```
df = clust.merge(df_msoa, left_on='MSOA11CD', right_on='MSOA11CD') #Merge with Orig
```

In [75]:
```
df = df.drop(['car_by_area','Area Sq Km'],axis=1) #drop unrelated factors to the st
```

Create Three Data Frames for Machine Learning

1st Model with Car By Pop, Explanatory Variables: Original Features Discussed in Literature df1

2nd Model with Car By Pop, Urban Form Features Only df3

3rd Model with Car By Pop, All Features Including Urban Features df2

In [76]:
```
df1 = pd.DataFrame(df[['MSOA11CD',
           'car_by_pop',
 'cluster_label',
 'Age-Median',
 'popdensity',
 'household',
 'household1',
 'household2',
 'household>=3',
 'EducationAll',
 'EducationNo',
 'Education1',
 'Education2',
 'EducationAppren',
 'Education3',
 'Education4',
 'EducationOther',
 'income',
'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_ALLEMP',
 'PB_ALLPS',
 'PB_ALLSS',
 'PB_TC',
 'PB_GFE',
 'PB_GGP',
 'PB_GHP',
 'PB_GPS',
 'PB_GSS']])

#Data Frame 2 All Features
df2 = pd.DataFrame(df[['MSOA11CD',
           'car_by_pop',
 'cluster_label',
 'Age-Median',
 'popdensity',
 'household',
 'household1',
 'household2',
 'household>=3',
 'EducationAll',
 'EducationNo',
 'Education1',
 'Education2',
 'EducationAppren',
```

```
 'Education3',
 'Education4',
 'EducationOther',
 'income',
'PB_ALLFE',
 'PB_ALLGP',
 'PB_ALLHP',
 'PB_ALLEMP',
 'PB_ALLPS',
 'PB_ALLSS',
 'PB_TC',
 'PB_GFE',
 'PB_GGP',
 'PB_GHP',
 'PB_GPS',
 'PB_GSS',
'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
 'sicCAR',
 'stbCeA',
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
```

```
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre']])

#Data Frame 3 Only Urban Features

df3 = pd.DataFrame(df[['MSOA11CD',
           'car_by_pop',
 'cluster_label',
          'sdbAre',
 'sdbHei',
 'sdbPer',
 'sdbVol',
 'ssbFoF',
 'ssbVFR',
 'ssbCCo',
 'ssbCor',
 'ssbSqu',
 'ssbERI',
 'ssbElo',
 'ssbCCM',
 'ssbCCD',
 'stbOri',
 'sdcLAL',
 'sdcAre',
 'sicFAR',
 'sscCCo',
 'sscERI',
 'stcOri',
 'sicCAR',
 'stbCeA',
```

```
 'mtbAli',
 'mtbNDi',
 'mtcWNe',
 'mdcAre',
 'ltcWRE',
 'ltbIBD',
 'ltcRea',
 'ltcAre',
 'sdsSPW',
 'sdsSPH',
 'sdsSPO',
 'sdsSWD',
 'sdsSHD',
 'sdsSPR',
 'sdsLen',
 'sssLin',
 'ldsMSL',
 'mtdDeg',
 'lcdMes',
 'linP3W',
 'linP4W',
 'linPDE',
 'lcnClo',
 'ldsCDL',
 'xcnSCl',
 'mtdMDi',
 'lddNDe',
 'linWID',
 'ldeAre',
 'ldePer',
 'lseCCo',
 'lseERI',
 'lseCWA',
 'lteOri',
 'lteWNB',
 'lieWCe',
 'stbSAl',
 'ldbPWL',
 'mtbSWR',
 'sddAre',
 'sdsAre',
 'sisBpM',
 'misCel',
 'mdsAre',
 'lisCel',
 'ldsAre']])
```

In [77]:
```
type(df3)
```

Out[77]:
```
pandas.core.frame.DataFrame
```

Results: Machine Learning Models

Lasso Regression

Lasso Regression for Original Features

In [78]:
```
i = [3,2,1,0]
results_train = list()
results_test = list()
for i in df1['cluster_label'].unique():
    test = df1[df1['cluster_label'] == i]
    X_test = test.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    features = X_test.columns
    y_test = test['car_by_pop']
    train= df1[df1['cluster_label'] != i]
    X_train = train.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_train = train['car_by_pop']
    y_train= y_train.values.reshape(-1,1)
    y_test= y_test.values.reshape(-1,1)
    sc_X = StandardScaler()
    sc_y = StandardScaler()
    X_train = sc_X.fit_transform(X_train)
    X_test = sc_X.fit_transform(X_test)
    y_train = sc_X.fit_transform(y_train)
    y_test = sc_y.fit_transform(y_test)
    lasso_model1 = Lasso(alpha = 0.001)
    lasso_model1.fit(X_train,y_train)
    train_score=lasso_model1.score(X_train,y_train)
    test_score=lasso_model1.score(X_test,y_test)
    coeff_used = np.sum(lasso_model1.coef_!=0)
    print('only Original Factors')
    print('')
    print('training score Fold',i+1,':',train_score)
    print('test score Fold',i+1,':', test_score)
    print('number of features Fold',i+1,':', coeff_used)
    results_train.append(train_score)
    results_test.append(test_score)
print('')
print("training score mean:", np.mean(results_train))
print("testing score mean:", np.mean(results_test))
lasso_orig_acc_train = results_train
lasso_orig_acc_test = results_test
```

```
only Original Factors

training score Fold 1 : 0.859520922288507
test score Fold 1 : 0.8189522052242113
number of features Fold 1 : 25
only Original Factors

training score Fold 2 : 0.8558156523814595
test score Fold 2 : 0.7675625058705998
number of features Fold 2 : 23
only Original Factors

training score Fold 3 : 0.8626095329993035
test score Fold 3 : 0.8078298314155758
number of features Fold 3 : 22
only Original Factors

training score Fold 4 : 0.8545491369828699
test score Fold 4 : 0.8116393144383313
number of features Fold 4 : 24

training score mean: 0.8581238111630349
testing score mean: 0.8014959642371796
```

Lasso Regression for Urban Form Features

In [79]:

```
from sklearn import metrics
i = [3,2,1,0]
results_train = list()
results_test = list()
results_mse = list()
results_mae = list()
y_pred_list = list()
y_test_list = list()
results_r2_list = list()

for i in df3['cluster_label'].unique():
    test = df3[df3['cluster_label'] == i]

    X_test = test.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    features = X_test.columns
    y_test = test['car_by_pop']
    train= df3[df3['cluster_label'] != i]
    X_train = train.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_train = train['car_by_pop']
    y_train= y_train.values.reshape(-1,1)
    y_test= y_test.values.reshape(-1,1)
    sc_X = StandardScaler()
    sc_y = StandardScaler()
    X_train = sc_X.fit_transform(X_train)
    X_test = sc_X.fit_transform(X_test)
    y_train = sc_X.fit_transform(y_train)
    y_test = sc_y.fit_transform(y_test)
    lasso_all = Lasso(alpha = 0.01)
    lasso_all.fit(X_train,y_train)
```

```
    train_score=lasso_all.score(X_train,y_train)
    test_score=lasso_all.score(X_test,y_test)
    coeff_used = np.sum(lasso_all.coef_!=0)
    y_pred = lasso_all.predict(X_test)
    mse_score = metrics.mean_squared_error(y_test, y_pred)
    mae_score = metrics.mean_absolute_error(y_test, y_pred)
    r2score = metrics.r2_score(y_test, y_pred)
    globals()["p" + str(i)] = lasso_all.coef_
    print('Only Urban Form Features')
    print('')
    print('training score Fold',i+1,':',train_score)
    print('test score Fold',i+1,':', test_score)
    print('number of features Fold',i+1,':', coeff_used)
    print('MSE score',i+1,':',mse_score)
    print('MAE score',i+1,':',mae_score)
    print('r2 score',i+1,':', r2score)


    results_train.append(train_score)
    results_test.append(test_score)
    results_mse.append(mse_score)
    results_mae.append(mae_score)
    results_r2_list.append(r2score)
    y_pred_list.append(y_pred)
    y_test_list.append(y_test)

    y_test_list.append(y_test)
print('')
print("training score mean:", np.mean(results_train))
print("testing score mean:", np.mean(results_test))
print("MSE All Fold:", np.mean(results_mse))
print("MAE All Fold:", np.mean(results_mae))

lasso_urb_train = results_train
lasso_urb_test = results_test
```

```
Only Urban Form Features

training score Fold 1 : 0.7916139804028521
test score Fold 1 : 0.7138282458368772
number of features Fold 1 : 25
MSE score 1 : 0.2861717541631229
MAE score 1 : 0.38594970311334154
r2 score 1 : 0.7138282458368772
Only Urban Form Features

training score Fold 2 : 0.7425898063625805
test score Fold 2 : 0.7800256150722832
number of features Fold 2 : 26
MSE score 2 : 0.2199743849277168
MAE score 2 : 0.35085012089086565
r2 score 2 : 0.7800256150722832
Only Urban Form Features

training score Fold 3 : 0.8004003384465515
test score Fold 3 : 0.6778032679620605
number of features Fold 3 : 24
MSE score 3 : 0.32219673203793947
MAE score 3 : 0.41942890863873206
r2 score 3 : 0.6778032679620605
Only Urban Form Features

training score Fold 4 : 0.7828236355225839
test score Fold 4 : 0.7061185730946361
number of features Fold 4 : 26
MSE score 4 : 0.2938814269053639
MAE score 4 : 0.3932352329672884
r2 score 4 : 0.7061185730946361

training score mean: 0.779356940183642
testing score mean: 0.7194439254914643
MSE All Fold: 0.2805560745085358
MAE All Fold: 0.3873659914025569
```

Lasso Regression for All Features and Feature Importance Analysis

In [80]:

```
from sklearn import metrics
i = [3,2,1,0] #cluster number list for kfold to take as test data set
results_train = list()
results_test = list()
results_mse = list()
results_mae = list()
results_r2_list = list()

msoa11cd_list = list()
cluster_nolist= list()
y_pred_list = list()
y_test_list = list()
for i in df2['cluster_label'].unique():
    test = df2[df2['cluster_label'] == i] #separate test data
    cluster_no = test['cluster_label'].tolist()
```

```
    msoa11cd = test['MSOA11CD'].tolist()
    X_test = test.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)#drop unnec
    features = X_test.columns #save feature names
    y_test = test['car_by_pop']
    train= df2[df2['cluster_label'] != i]
    X_train = train.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_train = train['car_by_pop']
    y_train= y_train.values.reshape(-1,1)
    y_test= y_test.values.reshape(-1,1)
    sc_X = StandardScaler()
    sc_y = StandardScaler()
    X_train = sc_X.fit_transform(X_train)
    X_test = sc_X.fit_transform(X_test)
    y_train = sc_X.fit_transform(y_train)
    y_test = sc_y.fit_transform(y_test)
    lasso_all = Lasso(alpha = 0.01)
    lasso_all.fit(X_train,y_train)
    train_score=lasso_all.score(X_train,y_train)
    test_score=lasso_all.score(X_test,y_test)
    coeff_used = np.sum(lasso_all.coef_!=0)
    y_pred = lasso_all.predict(X_test)

    mse_score = metrics.mean_squared_error(y_test, y_pred)
    mae_score = metrics.mean_absolute_error(y_test, y_pred)
    r2score = metrics.r2_score(y_test, y_pred)
    globals()["p" + str(i)] = lasso_all.coef_
    print('All Factors: Original Baseline Variables and Urban Form')
    print('')
    print('training score Fold',i+1,':',train_score)
    print('test score Fold',i+1,':', test_score)
    print('number of features Fold',i+1,':', coeff_used)
    print('MSE score',i+1,':',mse_score)
    print('MAE score',i+1,':',mae_score)
    print('r2 score',i+1,':', r2score)
    results_train.append(train_score)
    results_test.append(test_score)
    results_mse.append(mse_score)
    results_mae.append(mae_score)
    results_r2_list.append(r2score)
    msoa11cd_list.extend(msoa11cd)
    cluster_nolist.extend(cluster_no)
    y_pred_list.extend(y_pred.tolist())
    y_test_list.extend(y_test.tolist())

print('')
print("training score mean:", np.mean(results_train))
print("testing score mean:", np.mean(results_test))
print("MSE All Fold:", np.mean(results_mse))
print("MAE All Fold:", np.mean(results_mae))
lasso_all_train = results_train
lasso_all_test = results_test
```

```
All Factors: Original Baseline Variables and Urban Form

training score Fold 1 : 0.8809304042361873
test score Fold 1 : 0.8104050702616453
number of features Fold 1 : 28
MSE score 1 : 0.1895949297383547
MAE score 1 : 0.30991794644778836
r2 score 1 : 0.8104050702616453
All Factors: Original Baseline Variables and Urban Form

training score Fold 2 : 0.8625745167950488
test score Fold 2 : 0.7920031812854147
number of features Fold 2 : 30
MSE score 2 : 0.20799681871458534
MAE score 2 : 0.3481978162774864
r2 score 2 : 0.7920031812854147
All Factors: Original Baseline Variables and Urban Form

training score Fold 3 : 0.8792309079793021
test score Fold 3 : 0.8238728248003409
number of features Fold 3 : 32
MSE score 3 : 0.17612717519965918
MAE score 3 : 0.29816163302994503
r2 score 3 : 0.8238728248003409
All Factors: Original Baseline Variables and Urban Form

training score Fold 4 : 0.8711082954879269
test score Fold 4 : 0.8316559310502796
number of features Fold 4 : 30
MSE score 4 : 0.16834406894972037
MAE score 4 : 0.29399137464227454
r2 score 4 : 0.8316559310502796

training score mean: 0.8734610311246163
testing score mean: 0.8144842518494201
MSE All Fold: 0.18551574815057992
MAE All Fold: 0.3125671925993736
```

In [81]:
```
#combine list of all ypredicted and ytest by MSOA from Lasso test with all features
import numpy as np
result_lasso = pd.DataFrame(np.column_stack([msoa11cd_list, cluster_nolist, y_pred_
                                columns=['MSOA11CD', 'ClusterNo', 'y_pred','y_test']
result_lasso = result_lasso.astype({'y_pred':'float','y_test':'float'})
```

In [82]:
```
df_msoa= gpd.GeoDataFrame(df_msoa, geometry='geometry')
result_lasso = pd.merge(df_msoa[['MSOA11CD','MSOA11NM_x','LONG_','LAT','geometry']]
```

In [83]:
```
features = pd.DataFrame(features)
```

In [84]:
```
lasso3 = pd.DataFrame(list(p3))
feature_model = pd.merge(features,lasso3,how='left',left_index=True, right_index=Tr
feature_model.columns = ['feature','lasso_coef']
feature_model['feature_importance_lasso'] = feature_model['lasso_coef'].abs()
feature_model = feature_model.sort_values('feature_importance_lasso',ascending=Fals
```

In [85]:
```
feature_model
```

Out[85]:

	feature	lasso_coef	feature_importance_lasso
1	popdensity	-0.232598	0.232598
0	Age-Median	0.228623	0.228623
48	stbCeA	0.187241	0.187241
14	income	0.096507	0.096507
9	Education2	0.090376	0.090376
58	sdsSPH	-0.089960	0.089960
10	EducationAppren	0.088663	0.088663
3	household1	-0.079698	0.079698
7	EducationNo	-0.077059	0.077059
44	sscCCo	0.064073	0.064073
13	EducationOther	-0.062224	0.062224
8	Education1	0.040605	0.040605
11	Education3	-0.034495	0.034495
18	PB_ALLEMP	-0.033345	0.033345
24	PB_GHP	0.032851	0.032851
71	lcnClo	0.029826	0.029826
6	EducationAll	-0.028097	0.028097
45	sscERI	-0.027128	0.027128
70	linPDE	0.026179	0.026179
23	PB_GGP	0.020031	0.020031
20	PB_ALLSS	0.017533	0.017533
57	sdsSPW	0.013086	0.013086
59	sdsSPO	0.010080	0.010080
15	PB_ALLFE	0.007380	0.007380
35	ssbSqu	-0.003567	0.003567
17	PB_ALLHP	0.002946	0.002946
68	linP3W	-0.002398	0.002398
19	PB_ALLPS	0.002140	0.002140
84	lieWCe	-0.000647	0.000647
25	PB_GPS	0.000014	0.000014
69	linP4W	-0.000000	0.000000
90	sisBpM	-0.000000	0.000000
91	misCel	-0.000000	0.000000

	feature	lasso_coef	feature_importance_lasso
67	lcdMes	-0.000000	0.000000
87	mtbSWR	-0.000000	0.000000
66	mtdDeg	0.000000	0.000000
89	sdsAre	-0.000000	0.000000
64	sssLin	0.000000	0.000000
63	sdsLen	-0.000000	0.000000
62	sdsSPR	-0.000000	0.000000
61	sdsSHD	-0.000000	0.000000
92	mdsAre	-0.000000	0.000000
93	lisCel	-0.000000	0.000000
65	ldsMSL	-0.000000	0.000000
72	ldsCDL	0.000000	0.000000
88	sddAre	0.000000	0.000000
73	xcnSCl	0.000000	0.000000
74	mtdMDi	-0.000000	0.000000
75	lddNDe	0.000000	0.000000
76	linWID	-0.000000	0.000000
77	ldeAre	0.000000	0.000000
78	ldePer	0.000000	0.000000
79	lseCCo	0.000000	0.000000
80	lseERI	0.000000	0.000000
81	lseCWA	0.000000	0.000000
60	sdsSWD	0.000000	0.000000
83	lteWNB	-0.000000	0.000000
85	stbSAl	-0.000000	0.000000
86	ldbPWL	-0.000000	0.000000
82	lteOri	-0.000000	0.000000
47	sicCAR	-0.000000	0.000000
56	ltcAre	-0.000000	0.000000
26	PB_GSS	0.000000	0.000000
32	ssbVFR	-0.000000	0.000000
31	ssbFoF	0.000000	0.000000
30	sdbVol	-0.000000	0.000000

	feature	lasso_coef	feature_importance_lasso
29	sdbPer	-0.000000	0.000000
28	sdbHei	-0.000000	0.000000
27	sdbAre	-0.000000	0.000000
22	PB_GFE	-0.000000	0.000000
55	ltcRea	-0.000000	0.000000
21	PB_TC	0.000000	0.000000
16	PB_ALLGP	0.000000	0.000000
12	Education4	-0.000000	0.000000
5	household>=3	0.000000	0.000000
4	household2	0.000000	0.000000
2	household	-0.000000	0.000000
33	ssbCCo	0.000000	0.000000
34	ssbCor	-0.000000	0.000000
36	ssbERI	0.000000	0.000000
37	ssbElo	0.000000	0.000000
38	ssbCCM	-0.000000	0.000000
39	ssbCCD	-0.000000	0.000000
40	stbOri	0.000000	0.000000
41	sdcLAL	0.000000	0.000000
42	sdcAre	0.000000	0.000000
43	sicFAR	-0.000000	0.000000
46	stcOri	0.000000	0.000000
49	mtbAli	0.000000	0.000000
50	mtbNDi	0.000000	0.000000
51	mtcWNe	-0.000000	0.000000
52	mdcAre	-0.000000	0.000000
53	ltcWRE	-0.000000	0.000000
54	ltbIBD	-0.000000	0.000000
94	ldsAre	-0.000000	0.000000

Summarize Results for Lasso

In [86]:
```
lasso_orig_acc_train,lasso_urb_train,lasso_all_train
lasso_orig_acc_test,lasso_urb_test,lasso_all_test
```

```
#combine list of all ypredicted and ytest by MSOA from Lasso test with all features
import numpy as np
model_accuracy_test = pd.DataFrame(np.column_stack([lasso_orig_acc_test,lasso_urb_t
                                columns=['lasso_original_test', 'lasso_urban_test',
model_accuracy_train = pd.DataFrame(np.column_stack([lasso_orig_acc_train,lasso_urb
                                columns=['lasso_original_train', 'lasso_urban_train'
```

In [87]:
```
lasso_model_results = pd.merge(model_accuracy_train,model_accuracy_test,how='left',
```

In [88]:
```
lasso_model_results
lasso_model_results['Fold'] = lasso_model_results.index+1
lasso_model_results
```

Out[88]:

	lasso_original_train	lasso_urban_train	lasso_all_train	lasso_original_test	lasso_urban_test	lasso_
0	0.859521	0.791614	0.880930	0.818952	0.713828	0.
1	0.855816	0.742590	0.862575	0.767563	0.780026	0.
2	0.862610	0.800400	0.879231	0.807830	0.677803	0.
3	0.854549	0.782824	0.871108	0.811639	0.706119	0.

Rainforest

Rainforest Regression for Original Features

In [89]:
```
i = [3,2,1,0]
results_train = list()
results_test = list()
results_mse = list()
results_mae = list()
results_r2_list = list()

for i in df1['cluster_label'].unique():
    test = df1[df1['cluster_label'] == i]
    X_test = test.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_test = test['car_by_pop']
    train= df1[df1['cluster_label'] != i]
    X_train = train.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_train = train['car_by_pop']
    y_train= np.ravel(y_train.values.reshape(-1,1))
    y_test= np.ravel(y_test.values.reshape(-1,1))
    regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    mse_score = metrics.mean_squared_error(y_test, y_pred)
    mae_score = metrics.mean_absolute_error(y_test, y_pred)
    r2score = metrics.r2_score(y_test, y_pred)
    train_score = regressor.score(X_train, y_train)
    print("R-squared:", score)
    test_score=regressor.score(X_test,y_test)
```

```
    print('Only Original Factors')
    print('')
    print('training score Fold',i+1,':',train_score)
    print('test score Fold',i+1,':', test_score)
    print('MSE score',i+1,':',mse_score)
    print('MAE score',i+1,':',mae_score)
    print('r2 score',i+1,':', r2score)
    print('')
    results_train.append(train_score)
    results_test.append(test_score)
    results_mse.append(mse_score)
    results_mae.append(mae_score)
    results_r2_list.append(r2score)
print('')
print("training score mean:", np.mean(results_train))
print("testing score mean:", np.mean(results_test))
print("MSE All Fold:", np.mean(results_mse))
print("MAE All Fold:", np.mean(results_mae))
rf_orig_train = results_train
rf_orig_test = results_test
```

```
R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only Original Factors

training score Fold 1 : 0.9847060896171183
test score Fold 1 : 0.8292317135447822
MSE score 1 : 0.002622554647603144
MAE score 1 : 0.0350073262366938
r2 score 1 : 0.8292317135447822

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only Original Factors

training score Fold 2 : 0.980557095167241
test score Fold 2 : 0.6762287005685335
MSE score 2 : 0.008313668508707387
MAE score 2 : 0.06793348050036793
r2 score 2 : 0.6762287005685335

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only Original Factors

training score Fold 3 : 0.9844092640778531
test score Fold 3 : 0.7953131349470902
MSE score 3 : 0.0028371504825841667
MAE score 3 : 0.03979278069663367
r2 score 3 : 0.7953131349470902

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only Original Factors

training score Fold 4 : 0.9833875442925688
test score Fold 4 : 0.8312983108596598
MSE score 4 : 0.002113994625018201
MAE score 4 : 0.03363097422455219
r2 score 4 : 0.8312983108596598


training score mean: 0.9832649982886953
testing score mean: 0.7830179649800164
MSE All Fold: 0.003971842065978224
MAE All Fold: 0.0440911404145619
```

Rainforest Regression for Urban features

In [90]:
```
i = [3,2,1,0]
results_train = list()
results_test = list()
```

```
results_mse = list()
results_mae = list()
results_r2_list = list()

for i in df3['cluster_label'].unique():
    test = df3[df3['cluster_label'] == i]
    X_test = test.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_test = test['car_by_pop']
    train= df3[df3['cluster_label'] != i]
    X_train = train.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_train = train['car_by_pop']
    y_train= np.ravel(y_train.values.reshape(-1,1))
    y_test= np.ravel(y_test.values.reshape(-1,1))
    regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    mse_score = metrics.mean_squared_error(y_test, y_pred)
    mae_score = metrics.mean_absolute_error(y_test, y_pred)
    r2score = metrics.r2_score(y_test, y_pred)
    train_score = regressor.score(X_train, y_train)
    print("R-squared:", score)
    test_score=regressor.score(X_test,y_test)
    print('Only urban Factors')
    print('')
    print('training score Fold',i+1,':',train_score)
    print('test score Fold',i+1,':', test_score)
    print('MSE score',i+1,':',mse_score)
    print('MAE score',i+1,':',mae_score)
    print('r2 score',i+1,':', r2score)
    print('')
    results_train.append(train_score)
    results_test.append(test_score)
    results_mse.append(mse_score)
    results_mae.append(mae_score)
    results_r2_list.append(r2score)
print('')
print("training score mean:", np.mean(results_train))
print("testing score mean:", np.mean(results_test))
print("MSE All Fold:", np.mean(results_mse))
print("MAE All Fold:", np.mean(results_mae))
rf_urb_train = results_train
rf_urb_test = results_test
```

```
R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only urban Factors

training score Fold 1 : 0.9782400311273804
test score Fold 1 : 0.7533226991072941
MSE score 1 : 0.0037883187525220878
MAE score 1 : 0.04350171154247546
r2 score 1 : 0.7533226991072941

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only urban Factors

training score Fold 2 : 0.9702845376436546
test score Fold 2 : 0.8101419774000996
MSE score 2 : 0.004875097534543373
MAE score 2 : 0.0523294211429973
r2 score 2 : 0.8101419774000996

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only urban Factors

training score Fold 3 : 0.9784178370921296
test score Fold 3 : 0.7145885338033537
MSE score 3 : 0.003956068597003305
MAE score 3 : 0.044944424985405705
r2 score 3 : 0.7145885338033537

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
Only urban Factors

training score Fold 4 : 0.9760821961331071
test score Fold 4 : 0.7640485465064026
MSE score 4 : 0.002956698934032326
MAE score 4 : 0.039529226736566184
r2 score 4 : 0.7640485465064026


training score mean: 0.9757561504990679
testing score mean: 0.7605254392042875
MSE All Fold: 0.003894045954525273
MAE All Fold: 0.04507619610186116
```

Rainforest Regression for All Features

In [91]:
```
#rainforest doesn't need standardisation
i = [3,2,1,0]
results_train = list()
```

```
results_test = list()
results_mse = list()
results_mae = list()
results_r2_list = list()

msoa11cd_list = list()
cluster_nolist= list()
y_pred_list = list()
y_test_list = list()

for i in df2['cluster_label'].unique():
    test = df2[df2['cluster_label'] == i]
    msoa11cd = test['MSOA11CD'].tolist()
    cluster_nono = test['cluster_label'].tolist()
    X_test = test.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_test = test['car_by_pop']
    train= df2[df2['cluster_label'] != i]
    X_train = train.drop(['MSOA11CD','car_by_pop','cluster_label'],axis=1)
    y_train = train['car_by_pop']
    y_train= np.ravel(y_train.values.reshape(-1,1))
    y_test= np.ravel(y_test.values.reshape(-1,1))
    regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    mse_score = metrics.mean_squared_error(y_test, y_pred)
    mae_score = metrics.mean_absolute_error(y_test, y_pred)
    r2score = metrics.r2_score(y_test, y_pred)
    train_score = regressor.score(X_train, y_train)
    print("R-squared:", score)
    test_score=regressor.score(X_test,y_test)
    print('All factors including Urban Form')
    print('')
    print('training score Fold',i+1,':',train_score)
    print('test score Fold',i+1,':', test_score)
    print('MSE score',i+1,':',mse_score)
    print('MAE score',i+1,':',mae_score)
    print('r2 score',i+1,':', r2score)
    print('')
    results_train.append(train_score)
    results_test.append(test_score)
    results_mse.append(mse_score)
    results_mae.append(mae_score)
    results_r2_list.append(r2score)
    msoa11cd_list.extend(msoa11cd)
    cluster_nolist.extend(cluster_no)
    y_pred_list.extend(y_pred.tolist())
    y_test_list.extend(y_test.tolist())
print('')
print("training score mean:", np.mean(results_train))
print("testing score mean:", np.mean(results_test))
print("MSE All Fold:", np.mean(results_mse))
print("MAE All Fold:", np.mean(results_mae))
rf_all_train = results_train
rf_all_test = results_test
```

```
R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
All factors including Urban Form

training score Fold 1 : 0.9856244500342116
test score Fold 1 : 0.8329912443548089
MSE score 1 : 0.0025648180783413325
MAE score 1 : 0.034379670214986426
r2 score 1 : 0.8329912443548089

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
All factors including Urban Form

training score Fold 2 : 0.9811697257695231
test score Fold 2 : 0.7051192210085422
MSE score 2 : 0.007571829406835081
MAE score 2 : 0.06427285994603872
r2 score 2 : 0.7051192210085422

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
All factors including Urban Form

training score Fold 3 : 0.984624507012852
test score Fold 3 : 0.8167798624362875
MSE score 3 : 0.002539601657261469
MAE score 3 : 0.03573498735162488
r2 score 3 : 0.8167798624362875

R-squared: [-10046.998594713712, -3277.5313268955542, -1770.9118881856, -1213.7817
728409302, -723.3975940698532, -499.47924799648354, -379.18433328419513, -304.2397
1019750945, -244.5296620921503]
All factors including Urban Form

training score Fold 4 : 0.9838223182990107
test score Fold 4 : 0.8480337269272781
MSE score 4 : 0.0019042837454492508
MAE score 4 : 0.031133245958934058
r2 score 4 : 0.8480337269272781


training score mean: 0.9838102502788993
testing score mean: 0.8007310136817292
MSE All Fold: 0.003645133221971783
MAE All Fold: 0.04138019086789602
```

In [92]:
```
rf_accuracy_train = pd.DataFrame(np.column_stack([rf_orig_train,rf_urb_train,rf_all
                                 columns=['rf_original_train', 'rf_urban_train', 'rf_
rf_accuracy_test = pd.DataFrame(np.column_stack([rf_orig_test,rf_urb_test,rf_all_te
                                 columns=['rf_original_test', 'rf_urban_test', 'rf_al
```

In [93]:
```
rf_model_results = pd.merge(rf_accuracy_train,rf_accuracy_test,how='left',left_inde
```

```
rf_model_results
mylist = ['KFold 1','KFold 2','KFold 3','KFold 4']
rf_model_results['KFold Cluster'] = np.array(mylist)
rf_model_results
```

Out[93]:

	rf_original_train	rf_urban_train	rf_all_train	rf_original_test	rf_urban_test	rf_all_test	KFold Cluster
0	0.984706	0.978240	0.985624	0.829232	0.753323	0.832991	KFold 1
1	0.980557	0.970285	0.981170	0.676229	0.810142	0.705119	KFold 2
2	0.984409	0.978418	0.984625	0.795313	0.714589	0.816780	KFold 3
3	0.983388	0.976082	0.983822	0.831298	0.764049	0.848034	KFold 4

In [94]:
```
rf_pred_list = sc_y.fit_transform(pd.DataFrame(y_pred_list)) #save predicted by rai
```

In [95]:
```
rf_feature_imp = pd.merge(pd.DataFrame(features),pd.DataFrame(regressor.feature_imp
rf_feature_imp.columns = ['feature','rf_feature_importance'] #save feature importan
```

In [96]:
```
combined_feature = pd.merge(feature_model,rf_feature_imp,how='left',on='feature') #
combined_feature
```

Out[96]:

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance
0	popdensity	-0.232598	0.232598	0.033814
1	Age-Median	0.228623	0.228623	0.051243
2	stbCeA	0.187241	0.187241	0.610355
3	income	0.096507	0.096507	0.019245
4	Education2	0.090376	0.090376	0.000995
5	sdsSPH	-0.089960	0.089960	0.005175
6	EducationAppren	0.088663	0.088663	0.120016
7	household1	-0.079698	0.079698	0.002672
8	EducationNo	-0.077059	0.077059	0.003015
9	sscCCo	0.064073	0.064073	0.002951
10	EducationOther	-0.062224	0.062224	0.006781
11	Education1	0.040605	0.040605	0.001502
12	Education3	-0.034495	0.034495	0.003644
13	PB_ALLEMP	-0.033345	0.033345	0.002161
14	PB_GHP	0.032851	0.032851	0.002609
15	lcnClo	0.029826	0.029826	0.000613
16	EducationAll	-0.028097	0.028097	0.000972
17	sscERI	-0.027128	0.027128	0.001750
18	linPDE	0.026179	0.026179	0.001033
19	PB_GGP	0.020031	0.020031	0.001521
20	PB_ALLSS	0.017533	0.017533	0.002062
21	sdsSPW	0.013086	0.013086	0.001173
22	sdsSPO	0.010080	0.010080	0.001102
23	PB_ALLFE	0.007380	0.007380	0.001646
24	ssbSqu	-0.003567	0.003567	0.004613
25	PB_ALLHP	0.002946	0.002946	0.002130
26	linP3W	-0.002398	0.002398	0.000874
27	PB_ALLPS	0.002140	0.002140	0.002459
28	lieWCe	-0.000647	0.000647	0.001103
29	PB_GPS	0.000014	0.000014	0.002700
30	linP4W	-0.000000	0.000000	0.000954
31	sisBpM	-0.000000	0.000000	0.001642
32	misCel	-0.000000	0.000000	0.000866

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance
33	lcdMes	-0.000000	0.000000	0.000984
34	mtbSWR	-0.000000	0.000000	0.000161
35	mtdDeg	0.000000	0.000000	0.000000
36	sdsAre	-0.000000	0.000000	0.000740
37	sssLin	0.000000	0.000000	0.001213
38	sdsLen	-0.000000	0.000000	0.001213
39	sdsSPR	-0.000000	0.000000	0.009403
40	sdsSHD	-0.000000	0.000000	0.003015
41	mdsAre	-0.000000	0.000000	0.001052
42	lisCel	-0.000000	0.000000	0.001023
43	ldsMSL	-0.000000	0.000000	0.001047
44	ldsCDL	0.000000	0.000000	0.000982
45	sddAre	0.000000	0.000000	0.000706
46	xcnSCl	0.000000	0.000000	0.000018
47	mtdMDi	-0.000000	0.000000	0.000700
48	lddNDe	0.000000	0.000000	0.000765
49	linWID	-0.000000	0.000000	0.000834
50	ldeAre	0.000000	0.000000	0.000855
51	ldePer	0.000000	0.000000	0.000898
52	lseCCo	0.000000	0.000000	0.001247
53	lseERI	0.000000	0.000000	0.001133
54	lseCWA	0.000000	0.000000	0.001141
55	sdsSWD	0.000000	0.000000	0.001369
56	lteWNB	-0.000000	0.000000	0.001625
57	stbSAl	-0.000000	0.000000	0.001239
58	ldbPWL	-0.000000	0.000000	0.001359
59	lteOri	-0.000000	0.000000	0.001570
60	sicCAR	-0.000000	0.000000	0.001718
61	ltcAre	-0.000000	0.000000	0.002530
62	PB_GSS	0.000000	0.000000	0.001343
63	ssbVFR	-0.000000	0.000000	0.001390
64	ssbFoF	0.000000	0.000000	0.012497
65	sdbVol	-0.000000	0.000000	0.001059

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance
66	sdbPer	-0.000000	0.000000	0.000731
67	sdbHei	-0.000000	0.000000	0.001135
68	sdbAre	-0.000000	0.000000	0.000941
69	PB_GFE	-0.000000	0.000000	0.001312
70	ltcRea	-0.000000	0.000000	0.000707
71	PB_TC	0.000000	0.000000	0.001457
72	PB_ALLGP	0.000000	0.000000	0.001441
73	Education4	-0.000000	0.000000	0.001564
74	household>=3	0.000000	0.000000	0.001119
75	household2	0.000000	0.000000	0.000952
76	household	-0.000000	0.000000	0.000849
77	ssbCCo	0.000000	0.000000	0.001292
78	ssbCor	-0.000000	0.000000	0.000056
79	ssbERI	0.000000	0.000000	0.001163
80	ssbElo	0.000000	0.000000	0.001137
81	ssbCCM	-0.000000	0.000000	0.001218
82	ssbCCD	-0.000000	0.000000	0.001153
83	stbOri	0.000000	0.000000	0.001155
84	sdcLAL	0.000000	0.000000	0.000905
85	sdcAre	0.000000	0.000000	0.001271
86	sicFAR	-0.000000	0.000000	0.015312
87	stcOri	0.000000	0.000000	0.001016
88	mtbAli	0.000000	0.000000	0.001672
89	mtbNDi	0.000000	0.000000	0.001474
90	mtcWNe	-0.000000	0.000000	0.000956
91	mdcAre	-0.000000	0.000000	0.001260
92	ltcWRE	-0.000000	0.000000	0.002494
93	ltbIBD	-0.000000	0.000000	0.001134
94	ldsAre	-0.000000	0.000000	0.000906

In [99]:
```
#add pvalue with car by pop for the above table to show, final feature added and in
pvalue_corr['feature']= pvalue_corr[['car_by_pop']].index #from pvalue dataframe cr
pvalue_corr = pvalue_corr.rename({'car_by_pop':'pvalue'},axis=1) #rename car by pop
final_feature = pd.merge(combined_feature,pvalue_corr[['feature','pvalue']],how='le
final_feature #combine with lasso results
```

Out[99]:

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance	pvalue
0	popdensity	-0.232598	0.232598	0.033814	0.000000
1	Age-Median	0.228623	0.228623	0.051243	0.000000
2	stbCeA	0.187241	0.187241	0.610355	0.000000
3	income	0.096507	0.096507	0.019245	0.000000
4	Education2	0.090376	0.090376	0.000995	0.000000
5	sdsSPH	-0.089960	0.089960	0.005175	0.000000
6	EducationAppren	0.088663	0.088663	0.120016	0.000000
7	household1	-0.079698	0.079698	0.002672	0.000000
8	EducationNo	-0.077059	0.077059	0.003015	0.000000
9	sscCCo	0.064073	0.064073	0.002951	0.000000
10	EducationOther	-0.062224	0.062224	0.006781	0.000000
11	Education1	0.040605	0.040605	0.001502	0.000000
12	Education3	-0.034495	0.034495	0.003644	0.000000
13	PB_ALLEMP	-0.033345	0.033345	0.002161	0.000000
14	PB_GHP	0.032851	0.032851	0.002609	0.000000
15	lcnClo	0.029826	0.029826	0.000613	0.000000
16	EducationAll	-0.028097	0.028097	0.000972	0.000000
17	sscERI	-0.027128	0.027128	0.001750	0.000000
18	linPDE	0.026179	0.026179	0.001033	0.000000
19	PB_GGP	0.020031	0.020031	0.001521	0.000000
20	PB_ALLSS	0.017533	0.017533	0.002062	0.000000
21	sdsSPW	0.013086	0.013086	0.001173	0.000000
22	sdsSPO	0.010080	0.010080	0.001102	0.000000
23	PB_ALLFE	0.007380	0.007380	0.001646	0.000000
24	ssbSqu	-0.003567	0.003567	0.004613	0.000000
25	PB_ALLHP	0.002946	0.002946	0.002130	0.000000
26	linP3W	-0.002398	0.002398	0.000874	0.000000
27	PB_ALLPS	0.002140	0.002140	0.002459	0.000000
28	lieWCe	-0.000647	0.000647	0.001103	0.077214
29	PB_GPS	0.000014	0.000014	0.002700	0.000000
30	linP4W	-0.000000	0.000000	0.000954	0.000000
31	sisBpM	-0.000000	0.000000	0.001642	0.000000
32	misCel	-0.000000	0.000000	0.000866	0.000000

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance	pvalue
33	lcdMes	-0.000000	0.000000	0.000984	0.000000
34	mtbSWR	-0.000000	0.000000	0.000161	0.000000
35	mtdDeg	0.000000	0.000000	0.000000	0.136069
36	sdsAre	-0.000000	0.000000	0.000740	0.000000
37	sssLin	0.000000	0.000000	0.001213	0.000000
38	sdsLen	-0.000000	0.000000	0.001213	0.000000
39	sdsSPR	-0.000000	0.000000	0.009403	0.000000
40	sdsSHD	-0.000000	0.000000	0.003015	0.000000
41	mdsAre	-0.000000	0.000000	0.001052	0.000000
42	lisCel	-0.000000	0.000000	0.001023	0.000000
43	ldsMSL	-0.000000	0.000000	0.001047	0.000000
44	ldsCDL	0.000000	0.000000	0.000982	0.000000
45	sddAre	0.000000	0.000000	0.000706	0.000000
46	xcnSCl	0.000000	0.000000	0.000018	0.000000
47	mtdMDi	-0.000000	0.000000	0.000700	0.000000
48	lddNDe	0.000000	0.000000	0.000765	0.000000
49	linWID	-0.000000	0.000000	0.000834	0.000000
50	ldeAre	0.000000	0.000000	0.000855	0.000000
51	ldePer	0.000000	0.000000	0.000898	0.000000
52	lseCCo	0.000000	0.000000	0.001247	0.000000
53	lseERI	0.000000	0.000000	0.001133	0.000000
54	lseCWA	0.000000	0.000000	0.001141	0.000000
55	sdsSWD	0.000000	0.000000	0.001369	0.000000
56	lteWNB	-0.000000	0.000000	0.001625	0.000000
57	stbSAl	-0.000000	0.000000	0.001239	0.000000
58	ldbPWL	-0.000000	0.000000	0.001359	0.000000
59	lteOri	-0.000000	0.000000	0.001570	0.748248
60	sicCAR	-0.000000	0.000000	0.001718	0.000000
61	ltcAre	-0.000000	0.000000	0.002530	0.000000
62	PB_GSS	0.000000	0.000000	0.001343	0.000000
63	ssbVFR	-0.000000	0.000000	0.001390	0.000000
64	ssbFoF	0.000000	0.000000	0.012497	0.000000
65	sdbVol	-0.000000	0.000000	0.001059	0.000000

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance	pvalue
66	sdbPer	-0.000000	0.000000	0.000731	0.000000
67	sdbHei	-0.000000	0.000000	0.001135	0.000000
68	sdbAre	-0.000000	0.000000	0.000941	0.000000
69	PB_GFE	-0.000000	0.000000	0.001312	0.000000
70	ltcRea	-0.000000	0.000000	0.000707	0.000000
71	PB_TC	0.000000	0.000000	0.001457	0.000000
72	PB_ALLGP	0.000000	0.000000	0.001441	0.000000
73	Education4	-0.000000	0.000000	0.001564	0.000000
74	household>=3	0.000000	0.000000	0.001119	0.000000
75	household2	0.000000	0.000000	0.000952	0.000000
76	household	-0.000000	0.000000	0.000849	0.000000
77	ssbCCo	0.000000	0.000000	0.001292	0.000000
78	ssbCor	-0.000000	0.000000	0.000056	0.000000
79	ssbERI	0.000000	0.000000	0.001163	0.000000
80	ssbElo	0.000000	0.000000	0.001137	0.000000
81	ssbCCM	-0.000000	0.000000	0.001218	0.026357
82	ssbCCD	-0.000000	0.000000	0.001153	0.000000
83	stbOri	0.000000	0.000000	0.001155	0.157901
84	sdcLAL	0.000000	0.000000	0.000905	0.000000
85	sdcAre	0.000000	0.000000	0.001271	0.000000
86	sicFAR	-0.000000	0.000000	0.015312	0.000000
87	stcOri	0.000000	0.000000	0.001016	0.012846
88	mtbAli	0.000000	0.000000	0.001672	0.000000
89	mtbNDi	0.000000	0.000000	0.001474	0.000000
90	mtcWNe	-0.000000	0.000000	0.000956	0.000000
91	mdcAre	-0.000000	0.000000	0.001260	0.000000
92	ltcWRE	-0.000000	0.000000	0.002494	0.000000
93	ltbIBD	-0.000000	0.000000	0.001134	0.000000
94	ldsAre	-0.000000	0.000000	0.000906	0.000000

In [100…

```
#assign asterisk instead of pvalue to show the strenght of significance in academic
def assign_asterisk(row):
    if row >= 0.05:
        result = 'ns'
    elif row <= 0.0001:
```

```
        result = '****'
    elif row <= 0.001:
        result = '***'
    elif row <=0.01:
        result = '**'
    elif row <=0.05:
        result = '*'
    else:
        pass
    return result
final_feature['pval'] = final_feature['pvalue'].apply(assign_asterisk)
```

In [101…
```
final_feature.head(1)
```

Out[101]:

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance	pvalue	pval
0	popdensity	-0.232598	0.232598	0.033814	0.000000	****

In [102…
```
final_feature = final_feature.fillna(0) #fill na for any feature not related to las
final_feature = final_feature[['feature','lasso_coef','feature_importance_lasso','r
final_feature = final_feature.iloc[:30,:]
```

In [103…
```
final_feature
```

Out[103]:

	feature	lasso_coef	feature_importance_lasso	rf_feature_importance	pval
0	popdensity	-0.232598	0.232598	0.033814	****
1	Age-Median	0.228623	0.228623	0.051243	****
2	stbCeA	0.187241	0.187241	0.610355	****
3	income	0.096507	0.096507	0.019245	****
4	Education2	0.090376	0.090376	0.000995	****
5	sdsSPH	-0.089960	0.089960	0.005175	****
6	EducationAppren	0.088663	0.088663	0.120016	****
7	household1	-0.079698	0.079698	0.002672	****
8	EducationNo	-0.077059	0.077059	0.003015	****
9	sscCCo	0.064073	0.064073	0.002951	****
10	EducationOther	-0.062224	0.062224	0.006781	****
11	Education1	0.040605	0.040605	0.001502	****
12	Education3	-0.034495	0.034495	0.003644	****
13	PB_ALLEMP	-0.033345	0.033345	0.002161	****
14	PB_GHP	0.032851	0.032851	0.002609	****
15	lcnClo	0.029826	0.029826	0.000613	****
16	EducationAll	-0.028097	0.028097	0.000972	****
17	sscERI	-0.027128	0.027128	0.001750	****
18	linPDE	0.026179	0.026179	0.001033	****
19	PB_GGP	0.020031	0.020031	0.001521	****
20	PB_ALLSS	0.017533	0.017533	0.002062	****
21	sdsSPW	0.013086	0.013086	0.001173	****
22	sdsSPO	0.010080	0.010080	0.001102	****
23	PB_ALLFE	0.007380	0.007380	0.001646	****
24	ssbSqu	-0.003567	0.003567	0.004613	****
25	PB_ALLHP	0.002946	0.002946	0.002130	****
26	linP3W	-0.002398	0.002398	0.000874	****
27	PB_ALLPS	0.002140	0.002140	0.002459	****
28	lieWCe	-0.000647	0.000647	0.001103	ns
29	PB_GPS	0.000014	0.000014	0.002700	****

In [104…

```
#style final feature and pvalue table for feature results with html style bar
final_feature = final_feature.rename({'feature_importance_lasso':'lasso_feature_imp
final_feature.style.bar(subset=['lasso_coef','lasso_feature_importance','rf_feature
```

Out[104]:

feature	lasso_coef	lasso_feature_importance	rf_feature_importance	pval
popdensity	-0.232598	0.232598	0.033814	****
Age-Median	0.228623	0.228623	0.051243	****
stbCeA	0.187241	0.187241	0.610355	****
income	0.096507	0.096507	0.019245	****
Education2	0.090376	0.090376	0.000995	****
sdsSPH	-0.089960	0.089960	0.005175	****
EducationAppren	0.088663	0.088663	0.120016	****
household1	-0.079698	0.079698	0.002672	****
EducationNo	-0.077059	0.077059	0.003015	****
sscCCo	0.064073	0.064073	0.002951	****
EducationOther	-0.062224	0.062224	0.006781	****
Education1	0.040605	0.040605	0.001502	****
Education3	-0.034495	0.034495	0.003644	****
PB_ALLEMP	-0.033345	0.033345	0.002161	****
PB_GHP	0.032851	0.032851	0.002609	****
lcnClo	0.029826	0.029826	0.000613	****
EducationAll	-0.028097	0.028097	0.000972	****
sscERI	-0.027128	0.027128	0.001750	****
linPDE	0.026179	0.026179	0.001033	****
PB_GGP	0.020031	0.020031	0.001521	****
PB_ALLSS	0.017533	0.017533	0.002062	****
sdsSPW	0.013086	0.013086	0.001173	****
sdsSPO	0.010080	0.010080	0.001102	****
PB_ALLFE	0.007380	0.007380	0.001646	****
ssbSqu	-0.003567	0.003567	0.004613	****
PB_ALLHP	0.002946	0.002946	0.002130	****
linP3W	-0.002398	0.002398	0.000874	****
PB_ALLPS	0.002140	0.002140	0.002459	****
lieWCe	-0.000647	0.000647	0.001103	ns
PB_GPS	0.000014	0.000014	0.002700	****

In [105…
```
top_features = final_feature['feature'].tolist()
```

In [106…
```
top_features
```

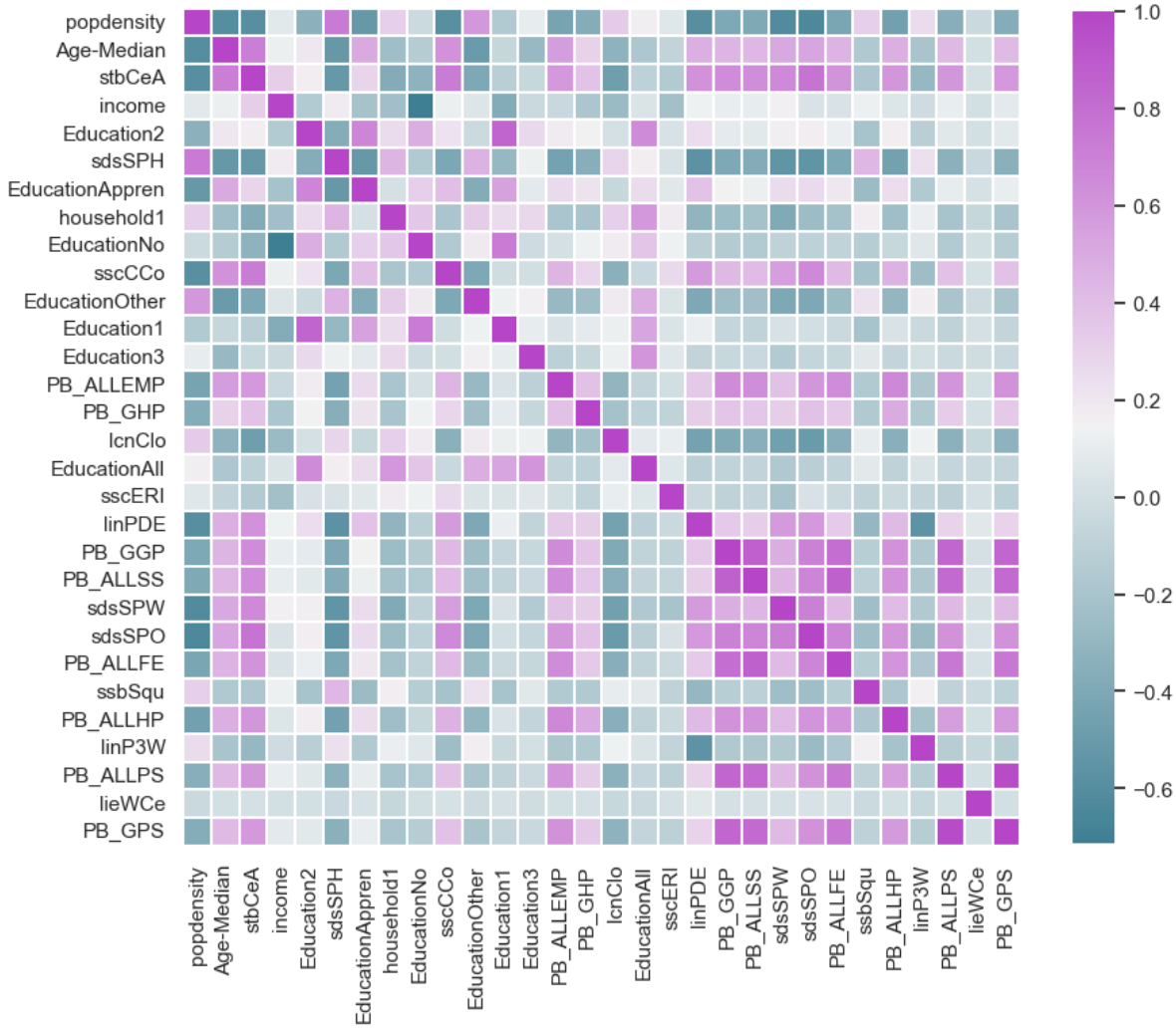
Out[106]:
```
['popdensity',
 'Age-Median',
 'stbCeA',
 'income',
 'Education2',
 'sdsSPH',
 'EducationAppren',
 'household1',
 'EducationNo',
 'sscCCo',
 'EducationOther',
 'Education1',
 'Education3',
 'PB_ALLEMP',
 'PB_GHP',
 'lcnClo',
 'EducationAll',
 'sscERI',
 'linPDE',
 'PB_GGP',
 'PB_ALLSS',
 'sdsSPW',
 'sdsSPO',
 'PB_ALLFE',
 'ssbSqu',
 'PB_ALLHP',
 'linP3W',
 'PB_ALLPS',
 'lieWCe',
 'PB_GPS']
```

In [107…
```
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
top_features_corr= df2[top_features]
scaler = MinMaxScaler()
# transform data
top_features_corr = scaler.fit_transform(top_features_corr)
```

In [108…
```
top_features_corr = pd.DataFrame(top_features_corr,columns = top_features)
```

In [109…
```
corr = top_features_corr.corr()
#study correlation of features only isolating and coloring features with negative o
plt.figure(figsize=(10,10))
corr = top_features_corr.corr()
cmap=sns.diverging_palette(220, 300, s=75, l=50, sep=1, n=6, as_cmap=True,center='l
#cmap = sns.diverging_palette(145, 300, s=60, as_cmap=True)
#cmap = sns.diverging_palette(500,350, s=75, l=50, sep=1, n=6, center='light', as_c
sns.heatmap(corr,cmap=cmap,cbar=True,square=True,linewidths=0.05, cbar_kws={"shrink
```

Out[109]:
```
<AxesSubplot: >
```



Compile Results of For RF Predicted

In [110…
```
import numpy as np
result_rf = pd.DataFrame(np.column_stack([msoa11cd_list, rf_pred_list]),
                                columns=['MSOA11CD', 'rf_pred'])
result_rf = result_rf.astype({'rf_pred':'float'})
```

In [111…
```
model_results = pd.merge(result_lasso,result_rf[['MSOA11CD','rf_pred']],how='left',
```

In [112…
```
model_results = model_results.rename({'y_pred':'lasso_pred','y_test':'actual'},axis
```

Compile Results of All Folds and Accuracy for Mean/Train Test, MSE and MAE

In [113…
```
rf_model_results
```

Out[113]:

	rf_original_train	rf_urban_train	rf_all_train	rf_original_test	rf_urban_test	rf_all_test	KFold Cluster
0	0.984706	0.978240	0.985624	0.829232	0.753323	0.832991	KFold 1
1	0.980557	0.970285	0.981170	0.676229	0.810142	0.705119	KFold 2
2	0.984409	0.978418	0.984625	0.795313	0.714589	0.816780	KFold 3
3	0.983388	0.976082	0.983822	0.831298	0.764049	0.848034	KFold 4

In [114…
```
lasso_model_results
```

Out[114]:

	lasso_original_train	lasso_urban_train	lasso_all_train	lasso_original_test	lasso_urban_test	lasso_
0	0.859521	0.791614	0.880930	0.818952	0.713828	0.
1	0.855816	0.742590	0.862575	0.767563	0.780026	0.
2	0.862610	0.800400	0.879231	0.807830	0.677803	0.
3	0.854549	0.782824	0.871108	0.811639	0.706119	0.

In [115…
```
model_accuracy = pd.merge(lasso_model_results,rf_model_results,how='left',left_inde
model_accuracy
```

Out[115]:

	lasso_original_train	lasso_urban_train	lasso_all_train	lasso_original_test	lasso_urban_test	lasso_
0	0.859521	0.791614	0.880930	0.818952	0.713828	0.
1	0.855816	0.742590	0.862575	0.767563	0.780026	0.
2	0.862610	0.800400	0.879231	0.807830	0.677803	0.
3	0.854549	0.782824	0.871108	0.811639	0.706119	0.

In [116…
```
m = model_accuracy.melt(id_vars='KFold Cluster',value_vars=['lasso_original_train',
                                    'rf_original_train','rf_urban_train','rf_all_trai
```

In [117…
```
m = m.loc[m['variable'].str.contains('all',case=False)]
```

In [118…
```
m.loc[m['variable'].str.contains('train', case=False, na=None), 'Train or Test'] =
m.loc[m['variable'].str.contains('test', case=False, na=None), 'Train or Test'] = '
```

In [119…
```
m.loc[m['variable'].str.contains('lasso', case=False, na=None), 'ML Model'] = 'Lass
m.loc[m['variable'].str.contains('rf', case=False, na=None), 'ML Model'] = 'Rain Fo
```

In [120…
```
m = m.rename({'Fold_x':'K-Fold Cluster No'},axis=1)
m.head()
```

Out[120]:

	KFold Cluster	variable	Model_Results	Train or Test	ML Model
8	KFold 1	lasso_all_train	0.880930	Train	Lasso Regression
9	KFold 2	lasso_all_train	0.862575	Train	Lasso Regression
10	KFold 3	lasso_all_train	0.879231	Train	Lasso Regression
11	KFold 4	lasso_all_train	0.871108	Train	Lasso Regression
20	KFold 1	lasso_all_test	0.810405	Test	Lasso Regression

In [121…

```
accuracy = pd.pivot_table(m,index=['ML Model','Train or Test'],columns=['KFold Clus
accuracy = accuracy.iloc[1:,:]
#accuracy.droplevel('K-Fold Cluster No')
accuracy
```

Out[121]:

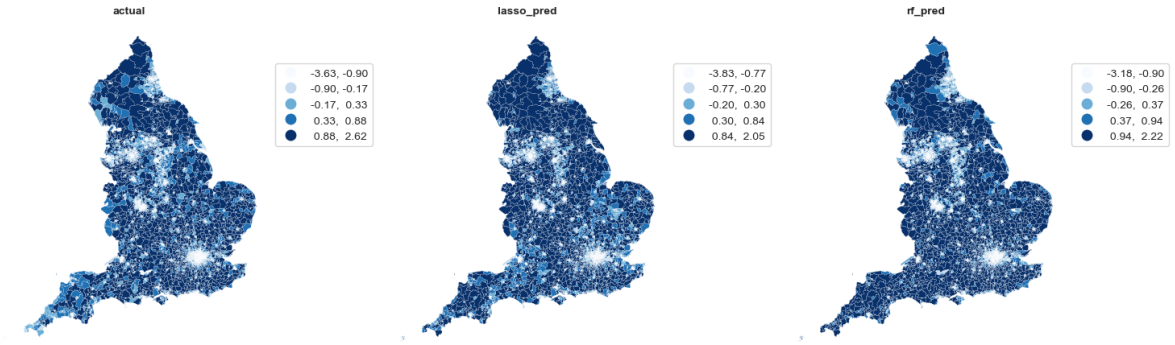
	ML Model	Train or Test	Model_Results				
KFold Cluster			KFold 1	KFold 2	KFold 3	KFold 4	Final_Result
1	Lasso Regression	Train	0.880930	0.862575	0.879231	0.871108	0.873461
2	Lasso Regression	Test	0.810405	0.792003	0.823873	0.831656	0.814484
3	Rain Forest	Train	0.985624	0.981170	0.984625	0.983822	0.983810
4	Rain Forest	Test	0.832991	0.705119	0.816780	0.848034	0.800731

Exploratory Spatial Data Analysis for Predicted Vs Test for Lasso, RF

In [131…

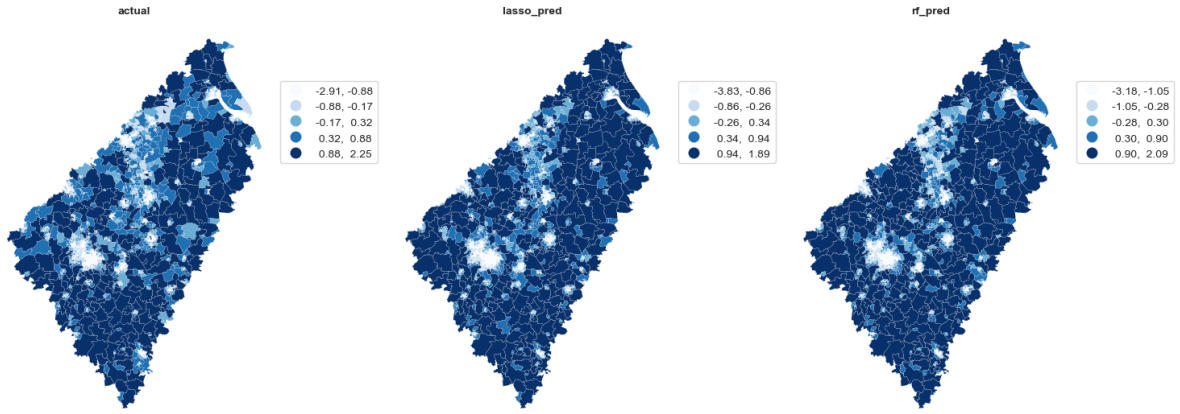
```
model_results= gpd.GeoDataFrame(model_results, geometry='geometry')
p_vars= ['actual','lasso_pred','rf_pred']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  3#int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
```

```
    ax = fig.add_subplot(gs[i])
    model_results.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'ce
    ax.set_title(p_vars[i], fontdict={'fontsize': 10, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```

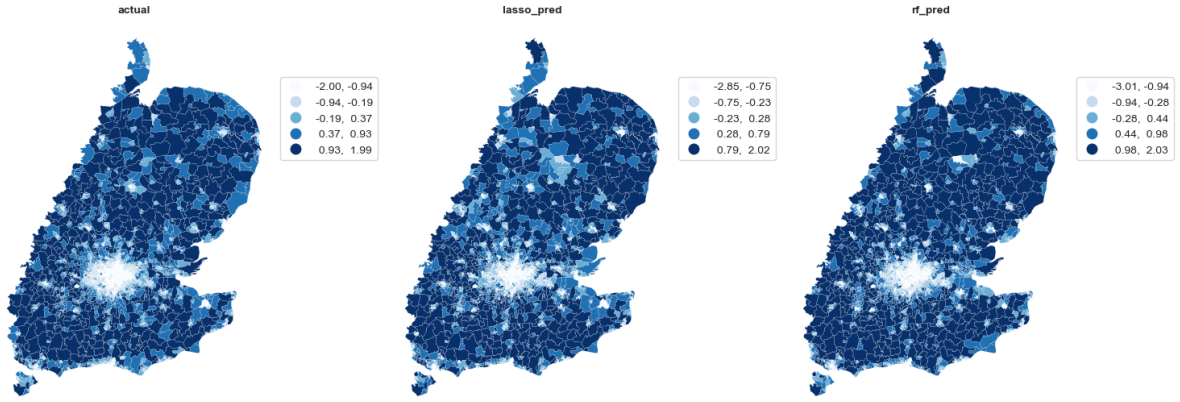


In [123…

```
model_results= gpd.GeoDataFrame(model_results, geometry='geometry')
clust_0 = model_results[model_results['ClusterNo']=='0']
p_vars= ['actual','lasso_pred','rf_pred']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  3#int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    clust_0.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 10, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```

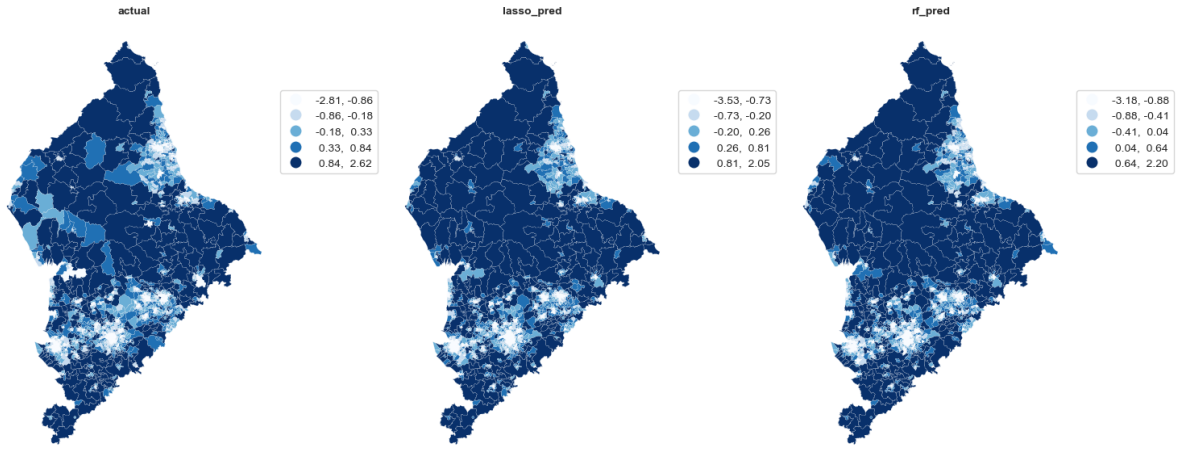


In [124…
```
model_results = gpd.GeoDataFrame(model_results, geometry='geometry')
clust_1 = model_results[model_results['ClusterNo']=='1']
p_vars= ['actual','lasso_pred','rf_pred']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  3#int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    clust_1.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 10, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```



In [128…
```
model_results = gpd.GeoDataFrame(model_results, geometry='geometry')
clust_2 = model_results[model_results['ClusterNo']=='2']
```

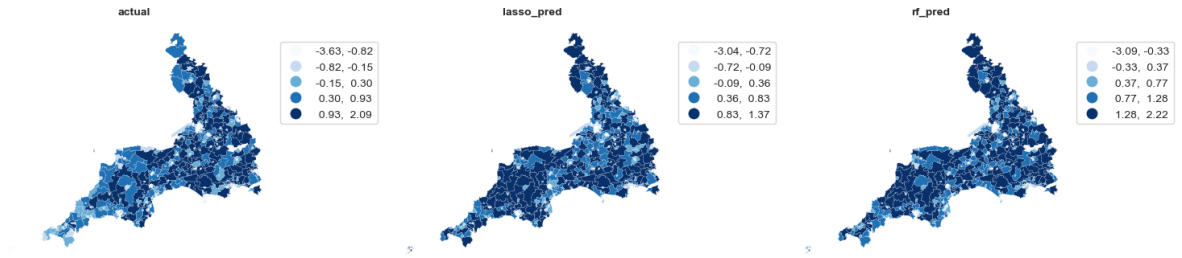
```
p_vars= ['actual','lasso_pred','rf_pred']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  3#int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    clust_2.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 10, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```



In [126…

```
model_results = gpd.GeoDataFrame(model_results, geometry='geometry')
clust_3 = model_results[model_results['ClusterNo']=='3']
p_vars= ['actual','lasso_pred','rf_pred']


import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

n_plots = len(p_vars)
# compute the number of rows and columns
n_cols =  3#int(np.sqrt(n_plots))
n_rows = int(np.ceil(n_plots / n_cols))

# setup the plot
gs = gridspec.GridSpec(n_rows, n_cols)
```

```
scale = max(n_cols, n_rows)
fig = plt.figure(figsize=(5 * scale, 5 * scale))

# loop through each subplot and plot values there
for i in range(n_plots):
    ax = fig.add_subplot(gs[i])
    clust_3.plot(column = p_vars[i],ax=ax,legend=True,legend_kwds={'loc': 'center l
    ax.set_title(p_vars[i], fontdict={'fontsize': 10, 'fontweight': 'bold'})
    ax.axis('off')
gs.tight_layout(fig, rect=[0, 0, 1.0, 1.0])
```



Appendix: Own Notes