Model_III_MOREparameters

April 10, 2021

[1]:
```
import numpy as np
import pandas as pd
import os
from functions.poll_data import party_in_region, region_in_party
import pickle
import matplotlib.pyplot as plt
import seaborn as sn
from scipy.optimize import curve_fit
from itertools import starmap

from sklearn.metrics import r2_score
from sklearn.linear_model import Lasso
#https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
from sklearn.ensemble import RandomForestRegressor
from sklearn.decomposition import PCA
```

[2]:
```
pd.options.mode.chained_assignment = None  # default='warn'
```

1 Prepare data

1.1 colors

[3]:
```
colors = pd.read_csv('dane_pdf/woj_col.csv', index_col=0)
colors
```

[3]:

	wojewodztwo	color
0	ŚLĄSKIE	black
1	OPOLSKIE	slategray
2	WIELKOPOLSKIE	darkred
3	ZACHODNIOPOMORSKIE	deeppink
4	ŚWIĘTOKRZYSKIE	darkorange
5	KUJAWSKO-POMORSKIE	bisque
6	PODLASKIE	gold
7	DOLNOŚLĄSKIE	olive
8	PODKARPACKIE	green
9	MAŁOPOLSKIE	lime
10	POMORSKIE	cyan

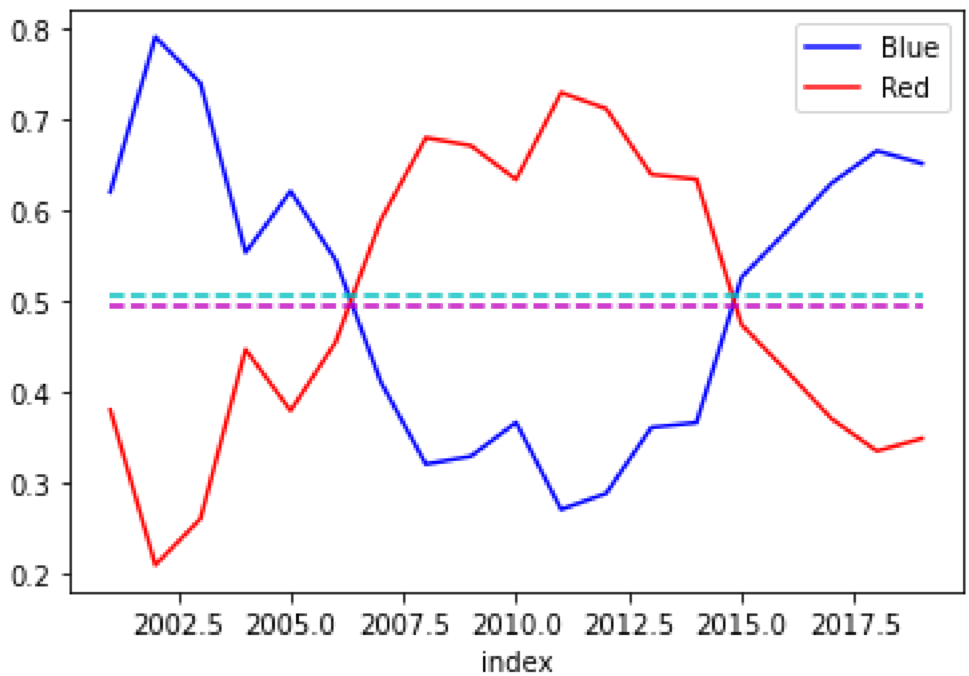
```
11  WARMIŃSKO-MAZURSKIE        blue
12               ŁÓDZKIE      violet
13           MAZOWIECKIE      indigo
14             LUBELSKIE   tab:brown
15             LUBUSKIE   tab:purple
```

1.2 Percent voting people

[4]:
```
voter_w = pd.read_csv('dane_years/voters/percent_voters.csv',header=None)
```

1.3 Poll data

[5]:
```
#pool_data = pd.read_csv('dane_years/pools_edited.csv', index_col=0)
pool_data_middle = pd.read_csv('dane_years/pools_data/no_votes.csv',␣
 ↪index_col=0).iloc[:,:-1]
pool_data_middle = pool_data_middle.divide(pool_data_middle.sum(1),0)
pool_data_middle.plot(color=['b','r'])
means = pool_data_middle.mean(0)
plt.plot([2001,2019],[means[0],means[0]],'c--')
plt.plot([2001,2019],[means[1],means[1]],'m--')
plt.show()
```



1.4 Voting data

[6]:
```
path = 'wyniki_wyborow/Simple/'
files = list(filter(lambda x: os.path.isfile(path+x), os.listdir(path)))
files.sort()
files
```

[6]:
```
['2001_WS.csv',
 '2005_WS.csv',
 '2007_WS.csv',
 '2011_WS.csv',
 '2015_WS.csv',
 '2019_WS.csv']
```

[7]:
```
vote_list = [(lambda x: pd.read_csv(path+x,index_col=0, header=0))(f) for f in␣
↪files[:]]
vote_list[0] = vote_list[0].iloc[1:,:]
```

[8]:
```
vote_list[0]['jednostka'] = [j.upper() for j in vote_list[0]['jednostka']]
vote_list[0] = vote_list[0].sort_values(['jednostka'])
vote_list[0].columns = ['województwo'] + vote_list[0].columns.values.tolist()[1:
↪]
```

[9]:
```
vote_list[0] = vote_list[0].set_index('województwo')
```

1.5 Neighbours

[10]:
```
with open('wojew_neighbours.pkl', 'rb') as f:
    neighbours = pickle.load(f)
```

1.6 Stat data

[11]:
```
path = 'dane_years/'
files = list(filter(lambda x: os.path.isfile(path+x), os.listdir(path)))
files.sort()
```

[12]:
```
stat_list_org = [(lambda x: pd.read_csv(path+x,index_col=0, header=0))(f) for f␣
↪in files]
```

[13]:
```
for s in stat_list_org:
    s['wyksztalcenie_wyzsze'] = s['wyksztalcenie_wyzsze'].fillna(0)
```

[14]:
```
stat_list_org[0].isna().sum(0)
```

[14]:
```
emeryci_i_rencisci                  0
bezrobocie_zarejsestrowane          0
bezrobocie_zarejsestrowane_gminy   16
```

```
malzenstwa_zawarte                  0
ludnosc_na_1km2                    16
dochody_gminy                       0
dochody_na_mieszkanca              16
wyksztalcenie_wyzsze                0
wyksztalcenie_gim_pod_nizsze        0
wyksztalcenie_srednie               0
rozwody_powiat                      0
udzial_wiek_przedprodukcyjny       16
udzial_wiek_produkcyjny            16
udzial_wiek_poprodukcyjny          16
praca_najemna                       0
praca_wlasny_rachunek               0
socjal_500plus                     16
socjal                              0
dochody_brutto_na_mieszkanca        0
PKB_na_1_mieszkanca                 0
przestepstwa_ogolem                16
dtype: int64
```

[15]:
```
stat_list_org[0].columns[stat_list_org[0].isna().sum(0) > 0]
```

[15]:
```
Index(['bezrobocie_zarejsestrowane_gminy', 'ludnosc_na_1km2',
       'dochody_na_mieszkanca', 'udzial_wiek_przedprodukcyjny',
       'udzial_wiek_produkcyjny', 'udzial_wiek_poprodukcyjny',
       'socjal_500plus', 'przestepstwa_ogolem'],
      dtype='object')
```

[16]:
```
data = [['mazowieckie',35558],
['wielkopolskie',29826],
['lubelskie',25122],
['warmińsko-mazurskie',24173],
['zachodniopomorskie',22892],
['podlaskie',20187],
['dolnośląskie',19947],
['pomorskie',18310],
['łódzkie',18219],
['kujawsko-pomorskie',17972],
['podkarpackie',17846],
['małopolskie',15183],
['lubuskie',13988],
['śląskie',12333],
['świętokrzyskie',11711],
['opolskie',9412]]

woj_pow = pd.DataFrame(data, columns=['jednostka','powierzchnnia_km2'])
```

[17]:
```
woj_pow['jednostka'] = woj_pow['jednostka'] .str.upper()
```

[18]:
```
for s in range(len(stat_list_org)):
    stat_list_org[s] = pd.merge(stat_list_org[s], woj_pow, left_index=True,␣
 ↪right_on='jednostka')
    stat_list_org[s] = stat_list_org[s].set_index('jednostka')
```

1.7 Use 2 approaches to estimate date from years without elections

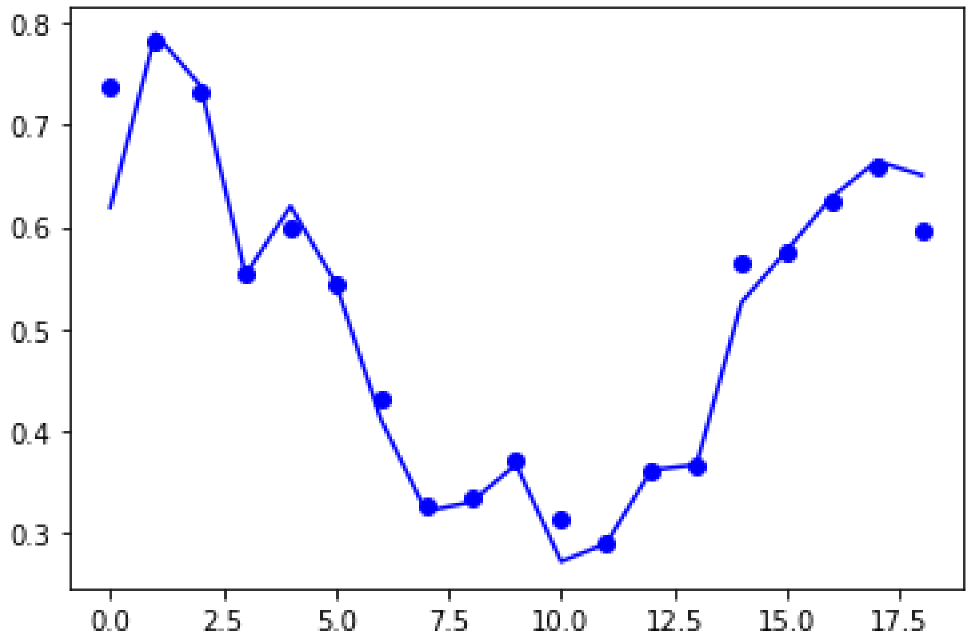
[19]:
```
#party_in_region(df_vote, df_poll)
par_in_reg_list = [vote_list[0].iloc[:,:-1]]
#region_in_party(df_vote, df_poll)
reg_in_par_list = [vote_list[0].iloc[:,:-1]]
for pool in pool_data_middle[1:].iterrows():
    if int(pool[0]) < 2005: df_vote = vote_list[0]
    elif int(pool[0]) < 2007: df_vote = vote_list[1]
    elif int(pool[0]) < 2011: df_vote = vote_list[2]
    elif int(pool[0]) < 2015: df_vote = vote_list[3]
    elif int(pool[0]) < 2019: df_vote = vote_list[4]
    else: df_vote = vote_list[5]

    par_in_reg_list.append(party_in_region(df_vote.iloc[:,:-1], pool[1]))
    reg_in_par_list.append(region_in_party(df_vote.iloc[:,:-1], pool[1]))
```

[20]:
```
for vl, i in zip(vote_list.copy(),[0,4,6,10,14,18]):
    par_in_reg_list[i] = vl.iloc[:,:-1].div(vl.iloc[:,:-1].sum(1),0) #vl.iloc[:
 ↪,:-1]
    reg_in_par_list[i] = vl.iloc[:,:-1].div(vl.iloc[:,:-1].sum(1),0) #vl.iloc[:
 ↪,:-1]
```

[21]:
```
pool_d = par_in_reg_list if (False) else reg_in_par_list
```

[22]:
```
pool_d_plot = []
voter_percent = voter_w.iloc[:,1].values
for p in pool_d:
    pool_d_plot.append(np.average(np.average(p['Blue'].values,weights =␣
 ↪voter_percent)))
```

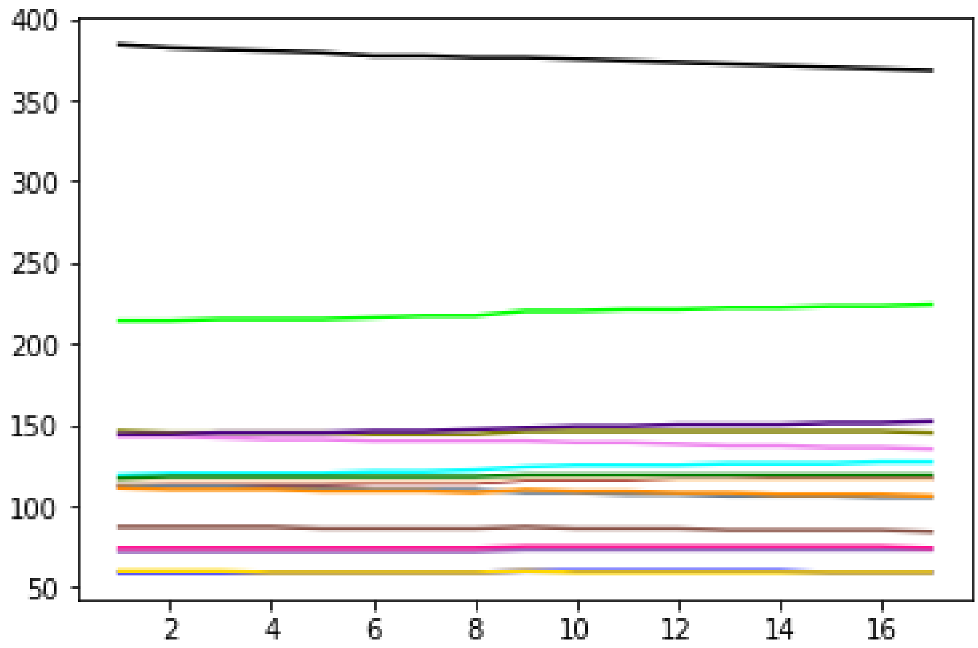
[23]:
```
plt.plot(pool_d_plot,'bo')
plt.plot(pool_data_middle['Blue'].values ,'b-')
plt.show()
```



1.8 Approximating ludnosc_na_1km2

[24]:
```
pool_stat_org = [pd.merge(s, p, left_index=True, right_index=True).
↪reset_index() for s,p in zip(stat_list_org, pool_d[1:])]
pool_stat_df_org = pd.concat(pool_stat_org).reset_index(drop=True)
```

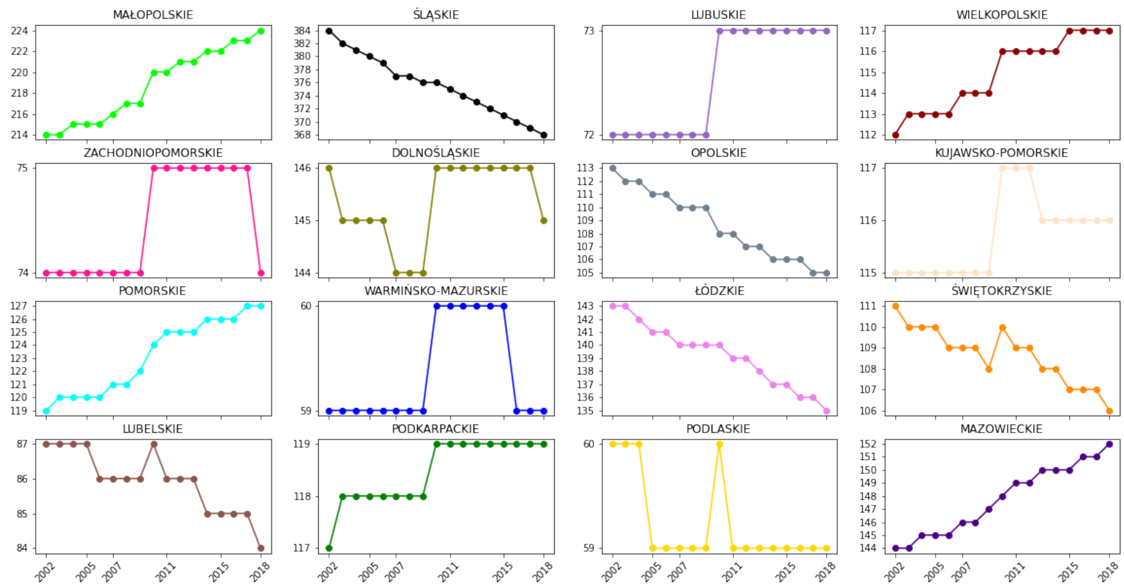
[25]:
```
data_reshaped = pool_stat_df_org['ludnosc_na_1km2'].values.reshape(18,-1)
for x in range(data_reshaped.shape[1]):
    plt.plot(data_reshaped[:,x], c=colors.loc[colors.
↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1])
plt.savefig('dane_pdf/dane_stat/ludnosc_na_1km2_all.pdf',  bbox_inches='tight')
```



[26]:
```
def lq_lin(x, a, b):
    return x*a + b
```

[27]:
```
from matplotlib.ticker import MaxNLocator
import matplotlib.ticker as ticker

x = np.arange(17)

fig = plt.figure(1, figsize=(20,10))

for w in range(data_reshaped.shape[1]):
    (a,b), pcov = curve_fit(lq_lin, x, data_reshaped[1:,w], p0=[1,1])
    xmodel = np.arange(18)
    ymodel = lq_lin(xmodel, a, b)

    ax1 = fig.add_subplot(4,4,w+1)
    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))

    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, data_reshaped[1:,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])
    #ax1.plot(xmodel,ymodel, "r--", label="Model")
    ax1.set_xticks([0,3,5,9,13,16])
```

```
    #
    if w > 11:
        ax1.set_xticklabels(['2002','2005','2007','2011','2015','2018'], ␣
↪rotation=45)
    else:
        ax1.set_xticklabels([])
        #plt.xticks([str(i) for i in range(2002,2018)], rotation=45)
    #ax.set_xticklabels(xlabels, rotation=40, ha=ha[n])
    #stat_list_org[-1]['dochody_brutto_na_mieszkanca'][w] = ymodel[-1]
plt.savefig('dane_pdf/dane_stat/ludnosc_na_1km2.pdf',  bbox_inches='tight')
```



Setup value as last year

[28]:
```
for w in range(data_reshaped.shape[1]):
    stat_list_org[0]['ludnosc_na_1km2'][w] = data_reshaped[1,w]
```

1.9 Creating new stat data

[29]:
```
stat_list_org[1].columns[stat_list_org[1].max(0) > 100]
```

[29]:
```
Index(['emeryci_i_rencisci', 'bezrobocie_zarejsestrowane',
       'malzenstwa_zawarte', 'ludnosc_na_1km2', 'dochody_gminy',
       'dochody_na_mieszkanca', 'rozwody_powiat', 'praca_najemna', 'socjal',
       'dochody_brutto_na_mieszkanca', 'PKB_na_1_mieszkanca',
       'przestepstwa_ogolem', 'powierzchnnia_km2'],
      dtype='object')
```

```
[30]: for s in stat_list_org:
    s['ludnosc'] = s['ludnosc_na_1km2']*s['powierzchnnia_km2']
    s['emeryci_i_rencisci_ludnosc'] = s['emeryci_i_rencisci']/s['ludnosc']
    s['bezrobocie_zarejsestrowane_ludnosc'] = s['bezrobocie_zarejsestrowane']/
 ↪s['ludnosc']
    s['malzenstwa_zawarte_ludnosc'] = s['malzenstwa_zawarte']/s['ludnosc']
    s['dochody_gminy_ludnosc'] = s['dochody_gminy']/s['ludnosc']
    s['rozwody_powiat_ludnosc'] = s['rozwody_powiat']/s['ludnosc']
    s['przestepstwa_ludnosc'] = s['przestepstwa_ogolem']/s['ludnosc']
    s['rozwody_malzenstwa'] = s['rozwody_powiat']/s['malzenstwa_zawarte']
```

1.9.1 Correlation

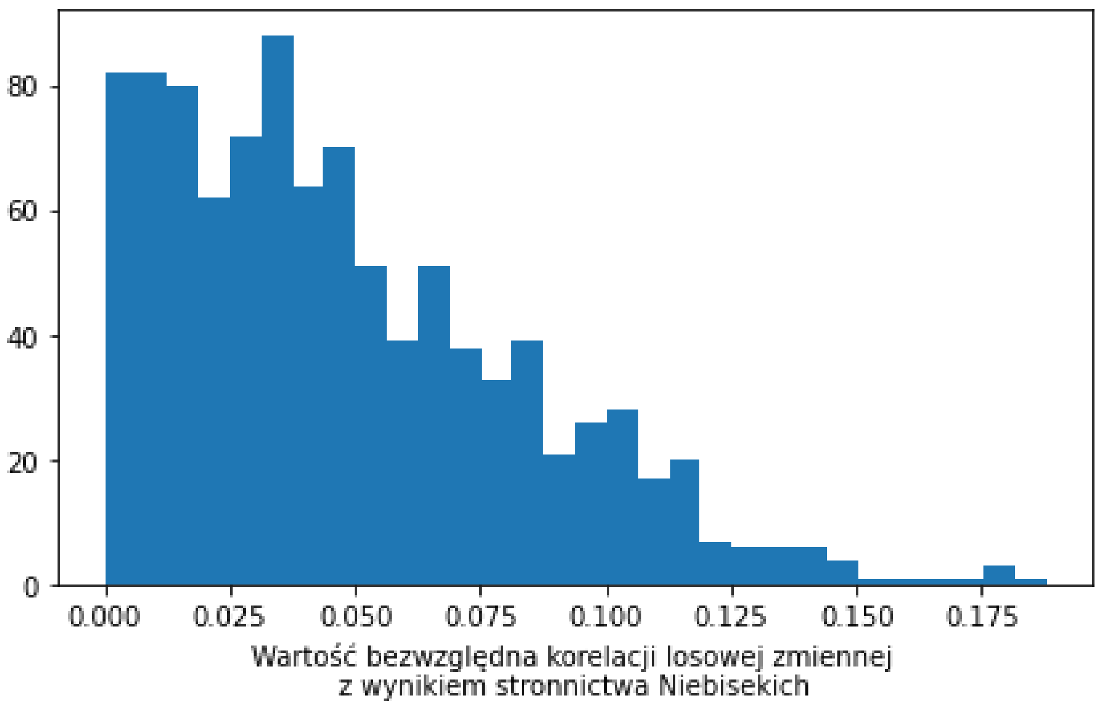
Corr between real data before scaling and outpots for each party based on rescaled output with poll and real data

```
[31]: pool_stat_org = [pd.merge(s, p, left_index=True, right_index=True).
 ↪reset_index() for s,p in zip(stat_list_org, pool_d[1:])]
pool_stat_df_org = pd.concat(pool_stat_org).reset_index(drop=True)

pool_stat_df_org['random'] = np.random.randn(pool_stat_df_org.shape[0])

cols = pool_stat_df_org.columns.tolist()
cols = cols[:-3] + ['random'] + cols[-3:-1]

pool_stat_df_org = pool_stat_df_org[cols]
```

```
[32]: rand_corr = []
for i in range(1000):
    pool_stat_df_org['random'] = np.random.randn(pool_stat_df_org.shape[0])
    rd_corr = pool_stat_df_org.corr().iloc[-1:,-3].values[0]
    rand_corr.append(np.abs(rd_corr))
```

```
[33]: #plt.axhline(np.mean(rand_corr)+np.std(rand_corr))
print(np.mean(rand_corr)+np.std(rand_corr))
#sn.displot(rand_corr,bins=20)
plt.figure(figsize=(7,4))
plt.hist(rand_corr,bins=30)
print((np.array(rand_corr)>0.1).sum()/1000)
#plt.ylabel('Liczba wystapień wartości w dnaym przedziale')
plt.xlabel(f'Wartość bezwzględna korelacji losowej zmiennej \nz wynikiem␣
 ↪stronnictwa Niebisekich')
plt.savefig('dane_pdf/dane_stat/random_vvariable_hist.pdf', ␣
 ↪bbox_inches='tight')
```

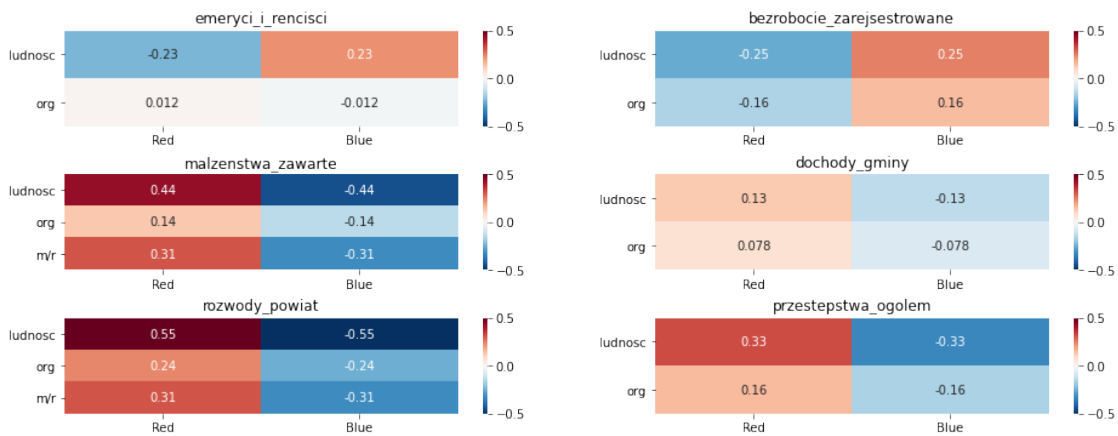
```
0.08379088895868103
0.102
```



1.9.2 Compare Old and New variables

[34]:
```
fig = plt.figure(1, figsize=(16,8))
plt.subplots_adjust(hspace=0.5)
for w, c in enumerate([
    'emeryci_i_rencisci',
    'bezrobocie_zarejsestrowane',
    'malzenstwa_zawarte',
    'dochody_gminy',
    'rozwody_powiat',
    'przestepstwa_ogolem']):
    #fig, ax = plt.subplots(figsize=(5,2))
    ax = fig.add_subplot(4,2,w+1)
    ax.set_title(c)
    if(c in ['malzenstwa_zawarte', 'rozwody_powiat']):
        tmp = pool_stat_df_org.corr().
 ↪loc[[c+'_ludnosc',c,'rozwody_malzenstwa'],['Red','Blue']].copy()
        tmp.index = ['ludnosc','org','m/r']
        g = sn.heatmap(tmp,annot=True, cmap='RdBu_r', ax=ax, vmin=-1/2, vmax=1/
 ↪2)
    elif c == 'przestepstwa_ogolem':
        tmp = pool_stat_df_org.corr().
 ↪loc[['przestepstwa_ludnosc',c],['Red','Blue']].copy()
```

```python
        tmp.index = ['ludnosc','org']
        g = sn.heatmap(tmp,annot=True, cmap='RdBu_r', ax=ax, vmin=-1/2, vmax=1/2)
    else:
        tmp = pool_stat_df_org.corr().loc[[c+'_ludnosc',c],['Red','Blue']].copy()
        tmp.index = ['ludnosc','org']
        g = sn.heatmap(tmp,annot=True, cmap='RdBu_r', ax=ax, vmin=-1/2, vmax=1/2)
    g.set_yticklabels(g.get_yticklabels(), rotation = 0)

plt.savefig('dane_pdf/dane_stat/corrs_new_old.pdf',  bbox_inches='tight')
```



[35]:
```python
(pool_stat_df_org
                        .corr()
                        .iloc[-2:,:-2][pool_stat_df_org
                                       .corr()
                                       .iloc[-2:,:-2]
                                       .abs() > 0.1])
```

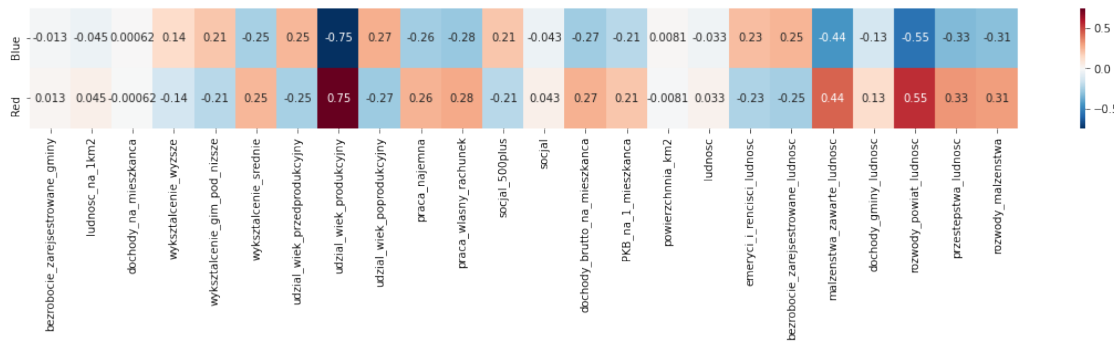
[35]:
```
      emeryci_i_rencisci  bezrobocie_zarejsestrowane  \
Blue                 NaN                    0.155267
Red                  NaN                   -0.155267

      bezrobocie_zarejsestrowane_gminy  malzenstwa_zawarte  ludnosc_na_1km2  \
Blue                               NaN           -0.135066              NaN
Red                                NaN            0.135066              NaN

      dochody_gminy  dochody_na_mieszkanca  wyksztalcenie_wyzsze  \
Blue            NaN                    NaN               0.13546
Red             NaN                    NaN              -0.13546
```

```
      wyksztalcenie_gim_pod_nizsze  wyksztalcenie_srednie  …  \
Blue                      0.205325              -0.247968  …
Red                      -0.205325               0.247968  …

      powierzchnnia_km2  ludnosc  emeryci_i_rencisci_ludnosc  \
Blue                NaN      NaN                    0.228949
Red                 NaN      NaN                   -0.228949

      bezrobocie_zarejsestrowane_ludnosc  malzenstwa_zawarte_ludnosc  \
Blue                             0.25258                   -0.438434
Red                             -0.25258                    0.438434

      dochody_gminy_ludnosc  rozwody_powiat_ludnosc  przestepstwa_ludnosc  \
Blue               -0.12987               -0.550899             -0.328121
Red                 0.12987                0.550899              0.328121

      rozwody_malzenstwa  random
Blue           -0.313341     NaN
Red             0.313341     NaN

[2 rows x 31 columns]
```

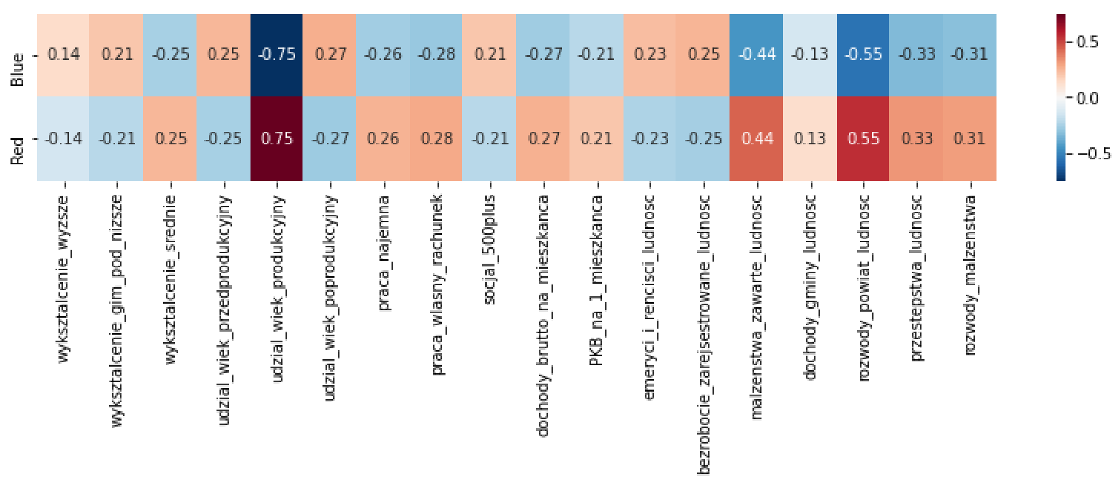
1.9.3 Delete old variables

[36]:
```python
for w, c in enumerate([
    'emeryci_i_rencisci',
    'bezrobocie_zarejsestrowane',
    'malzenstwa_zawarte',
    'dochody_gminy',
    'rozwody_powiat',
    'przestepstwa_ogolem']):
    pool_stat_df_org = pool_stat_df_org.drop(c, axis=1)
```

[37]:
```python
fig, ax = plt.subplots(figsize=(20,2))
sn.heatmap(pool_stat_df_org.corr().iloc[-2:,:-3], annot=True, cmap='RdBu_r',␣
 ↪ax=ax)

plt.savefig('dane_pdf/dane_stat/all_correlations.pdf',  bbox_inches='tight')
```




```
[38]: fig, ax = plt.subplots(figsize=(14,2))
      pool_names_highcorr = (pool_stat_df_org
                             .corr()
                             .iloc[-2:,:-2][pool_stat_df_org
                                            .corr()
                                            .iloc[-2:,:-2]
                                            .abs() > 0.1]
                             .dropna(1))
      sn.heatmap(pool_names_highcorr.dropna(1), annot=True, cmap='RdBu_r', ax=ax)
      plt.savefig('dane_pdf/dane_stat/all_correlations_choosed.pdf', ␣
       ↪bbox_inches='tight')
```




```
[39]: pool_names_highcorr
```

```
[39]:       wyksztalcenie_wyzsze  wyksztalcenie_gim_pod_nizsze  \
      Blue               0.13546                      0.205325
      Red               -0.13546                     -0.205325
```

```
      wyksztalcenie_srednie  udzial_wiek_przedprodukcyjny  \
Blue              -0.247968                       0.24773
Red                0.247968                      -0.24773

      udzial_wiek_produkcyjny  udzial_wiek_poprodukcyjny  praca_najemna  \
Blue                -0.747516                   0.270503      -0.261207
Red                  0.747516                  -0.270503       0.261207

      praca_wlasny_rachunek  socjal_500plus  dochody_brutto_na_mieszkanca  \
Blue              -0.280812         0.20763                      -0.26749
Red                0.280812        -0.20763                       0.26749

      PKB_na_1_mieszkanca  emeryci_i_rencisci_ludnosc  \
Blue            -0.210211                    0.228949
Red              0.210211                   -0.228949

      bezrobocie_zarejsestrowane_ludnosc  malzenstwa_zawarte_ludnosc  \
Blue                             0.25258                   -0.438434
Red                             -0.25258                    0.438434

      dochody_gminy_ludnosc  rozwody_powiat_ludnosc  przestepstwa_ludnosc  \
Blue               -0.12987               -0.550899             -0.328121
Red                 0.12987                0.550899              0.328121

      rozwody_malzenstwa
Blue           -0.313341
Red             0.313341
```

[40]:

```
col_highcorr = pool_names_highcorr.columns.values.tolist()
```

[41]:

```
for ch in col_highcorr:
    print('\item',ch )
```

```
\item wyksztalcenie_wyzsze
\item wyksztalcenie_gim_pod_nizsze
\item wyksztalcenie_srednie
\item udzial_wiek_przedprodukcyjny
\item udzial_wiek_produkcyjny
\item udzial_wiek_poprodukcyjny
\item praca_najemna
\item praca_wlasny_rachunek
\item socjal_500plus
\item dochody_brutto_na_mieszkanca
\item PKB_na_1_mieszkanca
\item emeryci_i_rencisci_ludnosc
\item bezrobocie_zarejsestrowane_ludnosc
\item malzenstwa_zawarte_ludnosc
```

```
\item dochody_gminy_ludnosc
\item rozwody_powiat_ludnosc
\item przestepstwa_ludnosc
\item rozwody_malzenstwa
```

[42]:
```
col_not_na0 = stat_list_org[0].columns[stat_list_org[0].isna().sum(0) == 0]
col_not_na17 = stat_list_org[0].columns[stat_list_org[17].isna().sum(0) == 0]
```

[43]:
```
df1_null = stat_list_org[0].count()
col_df1_null = df1_null[df1_null==0].index.values.tolist()
```

[44]:
```
col_df1_null
```

[44]:
```
['bezrobocie_zarejsestrowane_gminy',
 'dochody_na_mieszkanca',
 'udzial_wiek_przedprodukcyjny',
 'udzial_wiek_produkcyjny',
 'udzial_wiek_poprodukcyjny',
 'socjal_500plus',
 'przestepstwa_ogolem',
 'przestepstwa_ludnosc']
```

[45]:
```
cols_to_fill = list(set(col_df1_null).intersection(col_highcorr))
cols_to_fill = cols_to_fill
cols_to_fill
```

[45]:
```
['udzial_wiek_produkcyjny',
 'przestepstwa_ludnosc',
 'udzial_wiek_poprodukcyjny',
 'udzial_wiek_przedprodukcyjny',
 'socjal_500plus']
```

[46]:
```
dflast_null = stat_list_org[-1].count()
col_dflast_null = dflast_null[dflast_null==0].index.values.tolist()
cols_to_fill_last = list(set(col_dflast_null).intersection(col_highcorr))
cols_to_fill_last
```

[46]:
```
['dochody_brutto_na_mieszkanca']
```

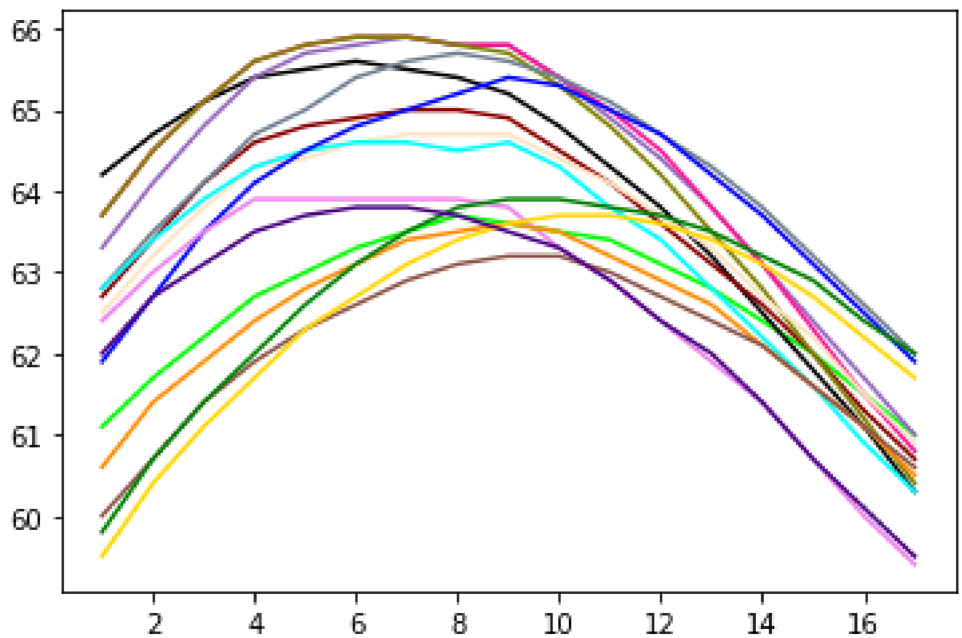
Selecting columns: - udzial_wiek_produkcyjny - udzial_wiek_przedprodukcyjny - udzial_wiek_poprodukcyjny - przestepstwa_ogolem - dochody_brutto_na_mieszkanca (for last)

[47]:
```
df_sel_null_col = pd.
 ↪DataFrame(columns=cols_to_fill+cols_to_fill_last+['ludnosc_na_1km2'])
for s in stat_list_org:
    cols_list = df_sel_null_col.columns.values.tolist()
    df_sel_null_col = df_sel_null_col.append(s[cols_list])
```

1.9.4 udzial_wiek_produkcyjny

[48]:
```
def lq(x, a, b, c):
    return x*x*a + x*b + c
```

[49]:
```
data_reshaped = df_sel_null_col['udzial_wiek_produkcyjny'].values.reshape(18,-1)
for x in range(data_reshaped.shape[1]):
    plt.plot(data_reshaped[:,x], c=colors.loc[colors.
 ↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1])
plt.savefig('dane_pdf/dane_stat/udzial_wiek_produkcyjny_all.pdf', ␣
 ↪bbox_inches='tight')
```



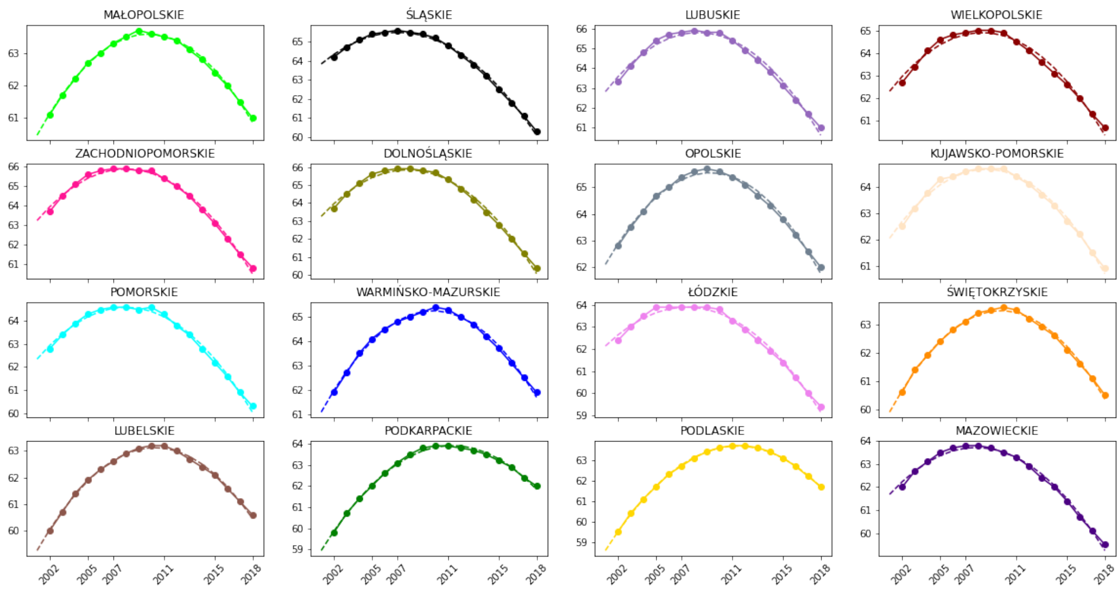
[50]:
```
data_reshaped.shape
```

[50]: (18, 16)

[51]:
```
fig = plt.figure(1, figsize=(20,10))
x = np.arange(17)
for w in range(data_reshaped.shape[1]):
    (a,b,c), pcov = curve_fit(lq, x, data_reshaped[1:,w], p0=[0.05,0.05,0.05])
    xmodel = np.arange(-1,17)
    ymodel = lq(xmodel, a, b, c)
    perr = np.sqrt(np.diag(pcov))

    print(df_sel_null_col.index[w].ljust(20)," RMSE",np.sum((data_reshaped[1:
 ↪,w]-ymodel[1:])**2)/ymodel[1:].shape[0])
```

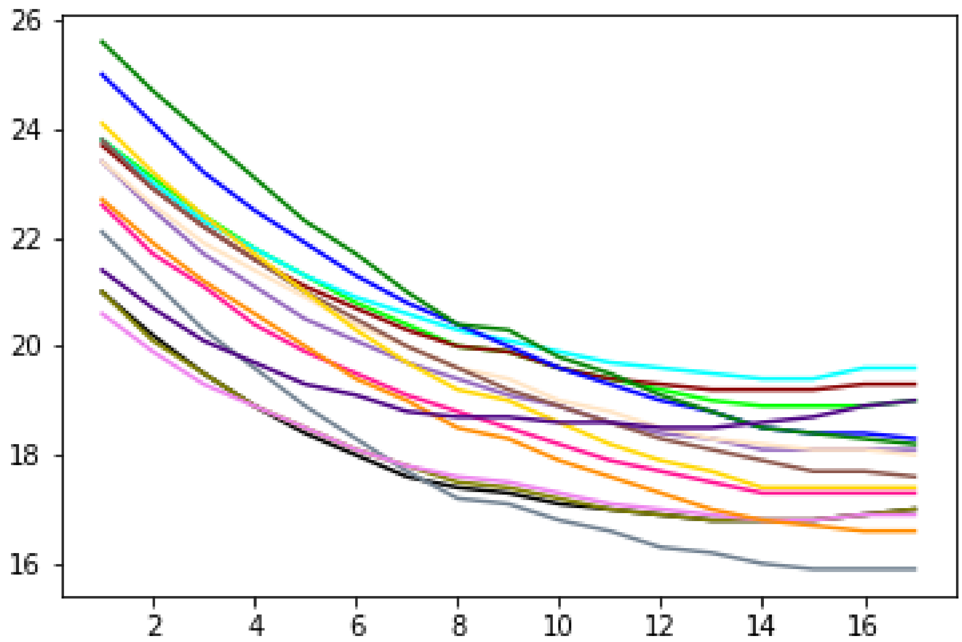
```
    stat_list_org[0]['udzial_wiek_produkcyjny'][w] = ymodel[0]
    ax1 = fig.add_subplot(4,4,w+1)

    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))

    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, data_reshaped[1:,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
 ↪values[0][1])
    ax1.plot(xmodel,ymodel, linestyle="--", label="Model",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
 ↪values[0][1])
    ax1.set_xticks([0,3,5,9,13,16])
    #
    if w > 11:
        ax1.set_xticklabels(['2002','2005','2007','2011','2015','2018'], ␣
 ↪rotation=45)
    else:
        ax1.set_xticklabels([])

    stat_list_org[0]['udzial_wiek_produkcyjny'][w] = ymodel[0]
plt.savefig('dane_pdf/dane_stat/udzial_wiek_produkcyjny.pdf', ␣
 ↪bbox_inches='tight')
```

```
MAŁOPOLSKIE          RMSE 0.0038150306562253766
ŚLĄSKIE              RMSE 0.010225520548776748
LUBUSKIE             RMSE 0.032914769623019605
WIELKOPOLSKIE        RMSE 0.029697383597401848
ZACHODNIOPOMORSKIE   RMSE 0.017715959448794717
DOLNOŚLĄSKIE         RMSE 0.022857706550112062
OPOLSKIE             RMSE 0.012787895343895975
KUJAWSKO-POMORSKIE   RMSE 0.011256905238875656
POMORSKIE            RMSE 0.01374127359922268
WARMIŃSKO-MAZURSKIE  RMSE 0.00923298731257192
ŁÓDZKIE              RMSE 0.01986037758756752
ŚWIĘTOKRZYSKIE       RMSE 0.0059309172585441305
LUBELSKIE            RMSE 0.004243003703029074
PODKARPACKIE         RMSE 0.005847447338068247
PODLASKIE            RMSE 0.0008501790809202414
MAZOWIECKIE          RMSE 0.014280033995022024
```



1.9.5 udzial_wiek_przedprodukcyjny

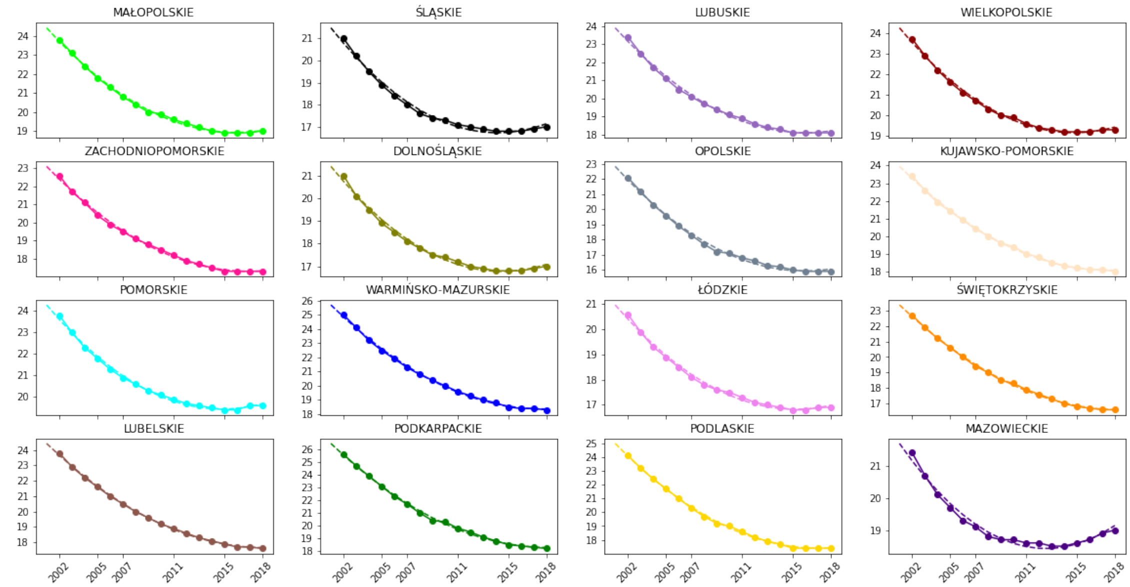
[52]:
```
data_reshaped = df_sel_null_col['udzial_wiek_przedprodukcyjny'].values.
 ↪reshape(18,-1)
for x in range(data_reshaped.shape[1]):
    plt.plot(data_reshaped[:,x], c=colors.loc[colors.
 ↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1])
plt.savefig('dane_pdf/dane_stat/udzial_wiek_przedprodukcyjny_all.pdf', ␣
 ↪bbox_inches='tight')
```



[53]:
```
fig = plt.figure(1, figsize=(20,10))
x = np.arange(17)
for w in range(data_reshaped.shape[1]):
    (a,b,c), pcov = curve_fit(lq, x, data_reshaped[1:,w], p0=[0.05,0.05,0.05])
    xmodel = np.arange(-1,17)
    ymodel = lq(xmodel, a, b, c)
    perr = np.sqrt(np.diag(pcov))

    print(df_sel_null_col.index[w].ljust(20)," RMSE",np.sum((data_reshaped[1:
↪,w]-ymodel[1:])**2)/ymodel[1:].shape[0])
    stat_list_org[0]['udzial_wiek_przedprodukcyjny'][w] = ymodel[0]
    ax1 = fig.add_subplot(4,4,w+1)

    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))

    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, data_reshaped[1:,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])
    ax1.plot(xmodel,ymodel, linestyle="--", label="Model",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])
    ax1.set_xticks([0,3,5,9,13,16])
    #
```

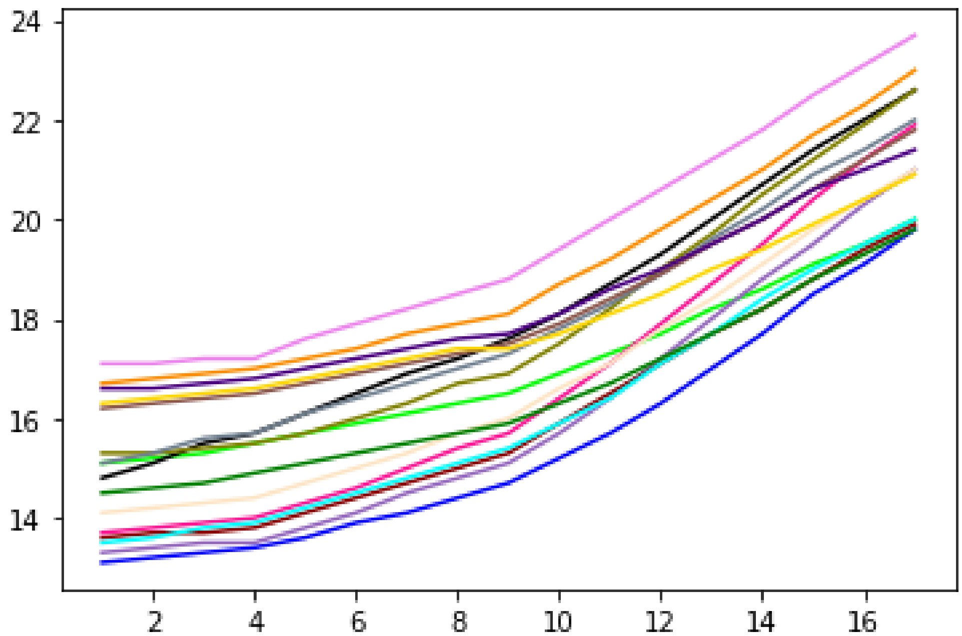
```
    if w > 11:
        ax1.set_xticklabels(['2002','2005','2007','2011','2015','2018'], ␣
 ↪rotation=45)
    else:
        ax1.set_xticklabels([])

    stat_list_org[0]['udzial_wiek_przedprodukcyjny'][w] = ymodel[0]
plt.savefig('dane_pdf/dane_stat/udzial_wiek_przedprodukcyjny.pdf', ␣
 ↪bbox_inches='tight')
```

```
MAŁOPOLSKIE          RMSE 0.004825775511442974
ŚLĄSKIE              RMSE 0.01353487525040982
LUBUSKIE             RMSE 0.011480301098767532
WIELKOPOLSKIE        RMSE 0.0073535482304376365
ZACHODNIOPOMORSKIE   RMSE 0.007275238268682167
DOLNOŚLĄSKIE         RMSE 0.010096825107752026
OPOLSKIE             RMSE 0.015084987555393876
KUJAWSKO-POMORSKIE   RMSE 0.0037977296181630417
POMORSKIE            RMSE 0.00702998846597465
WARMIŃSKO-MAZURSKIE  RMSE 0.0055484732592727785
ŁÓDZKIE              RMSE 0.007325016693984078
ŚWIĘTOKRZYSKIE       RMSE 0.003784070903903387
LUBELSKIE            RMSE 0.0030070418260183503
PODKARPACKIE         RMSE 0.009793298124203317
PODLASKIE            RMSE 0.00494111576519157
MAZOWIECKIE          RMSE 0.013645359072421447
```



1.9.6 udzial_wiek_poprodukcyjny

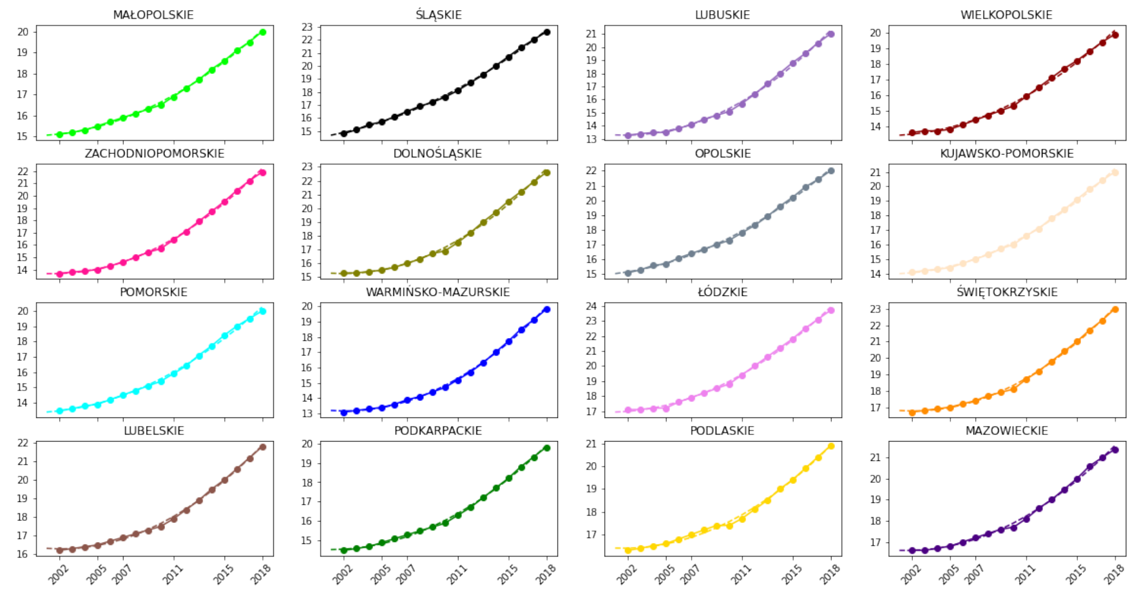
[54]:
```
data_reshaped = df_sel_null_col['udzial_wiek_poprodukcyjny'].values.
 ↪reshape(18,-1)
for x in range(data_reshaped.shape[1]):
    plt.plot(data_reshaped[:,x], c=colors.loc[colors.
 ↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1])
plt.savefig('dane_pdf/dane_stat/udzial_wiek_poprodukcyjny_all.pdf', ␣
 ↪bbox_inches='tight')
```



[55]:
```
fig = plt.figure(1, figsize=(20,10))

x = np.arange(17)
for w in range(data_reshaped.shape[1]):
    (a,b,c), pcov = curve_fit(lq, x, data_reshaped[1:,w], p0=[0.05,0.05,0.05])
    xmodel = np.arange(-1,17)
    ymodel = lq(xmodel, a, b, c)
    perr = np.sqrt(np.diag(pcov))

    print(df_sel_null_col.index[w].ljust(20)," RMSE",np.sum((data_reshaped[1:
 ↪,w]-ymodel[1:])**2)/ymodel[1:].shape[0])
    stat_list_org[0]['udzial_wiek_poprodukcyjny'][w] = ymodel[0]
    ax1 = fig.add_subplot(4,4,w+1)

    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))
```

```
    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, data_reshaped[1:,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
 ↪values[0][1])
    ax1.plot(xmodel,ymodel, linestyle="--", label="Model",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
 ↪values[0][1])
    ax1.set_xticks([0,3,5,9,13,16])
    #
    if w > 11:
        ax1.set_xticklabels(['2002','2005','2007','2011','2015','2018'], ␣
 ↪rotation=45)
    else:
        ax1.set_xticklabels([])

    stat_list_org[0]['udzial_wiek_poprodukcyjny'][w] = ymodel[0]
plt.savefig('dane_pdf/dane_stat/udzial_wiek_poprodukcyjny.pdf', ␣
 ↪bbox_inches='tight')
```

```
MAŁOPOLSKIE           RMSE 0.00316396527651308
ŚLĄSKIE               RMSE 0.007505311722211999
LUBUSKIE              RMSE 0.016178898804403423
WIELKOPOLSKIE         RMSE 0.015075881745887513
ZACHODNIOPOMORSKIE    RMSE 0.0130458932799202
DOLNOŚLĄSKIE          RMSE 0.016595641352516143
OPOLSKIE              RMSE 0.00845231591088442
KUJAWSKO-POMORSKIE    RMSE 0.008168518181266456
POMORSKIE             RMSE 0.01249559885873848
WARMIŃSKO-MAZURSKIE   RMSE 0.006106962909082264
ŁÓDZKIE               RMSE 0.010684453347902865
ŚWIĘTOKRZYSKIE        RMSE 0.006365264372081888
LUBELSKIE             RMSE 0.005254659139197894
PODKARPACKIE          RMSE 0.004853092939962439
PODLASKIE             RMSE 0.007789716505800923
MAZOWIECKIE           RMSE 0.007950889334073284
```

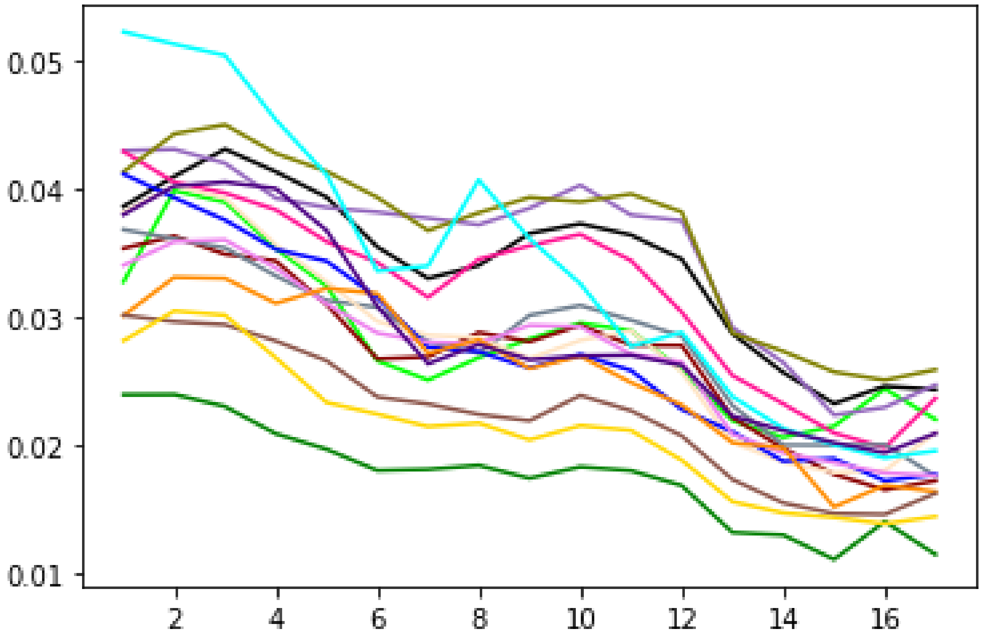


1.9.7 przestepstwa_ogolem

[56]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn import preprocessing
```

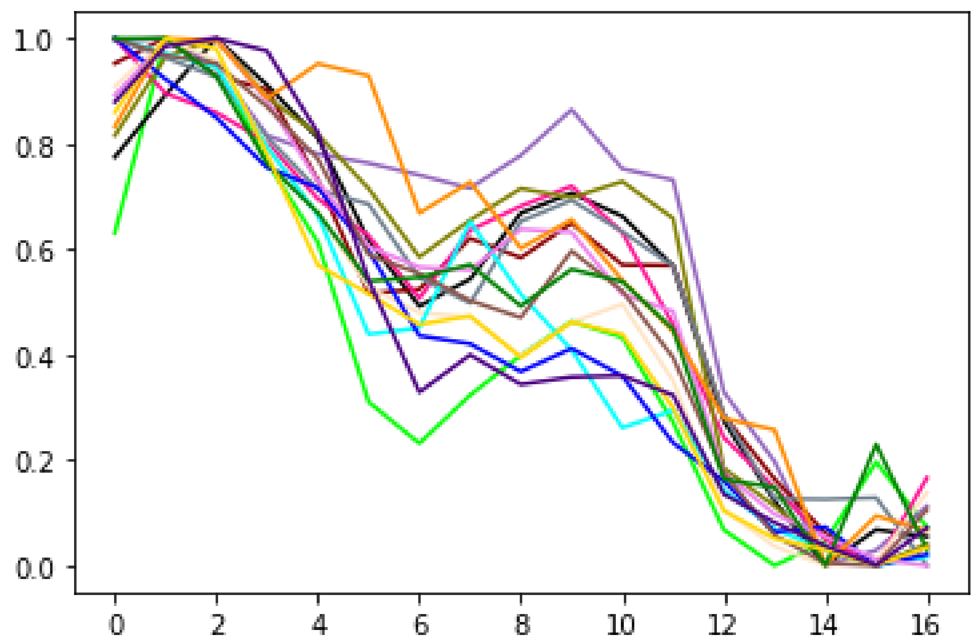
[57]:
```
def lq_sin(x, a, b, c, d):
    return np.sin(x/a*np.pi)*b + x*c + d
```

[58]:
```
data_reshaped = df_sel_null_col['przestepstwa_ludnosc'].values.reshape(18,-1)
for x in range(data_reshaped.shape[1]):
    plt.plot(data_reshaped[:,x], c=colors.loc[colors.
 ↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1])
plt.savefig('dane_pdf/dane_stat/przestepstwa_ludnosc_all.pdf', ␣
 ↪bbox_inches='tight')
```



MINMAX rescaling

[59]:
```
minmax_data_reshaped = preprocessing.minmax_scale(data_reshaped[~np.
 ↪isnan(data_reshaped)].reshape(17,-1))
for x in range(minmax_data_reshaped.shape[1]):
    plt.plot(minmax_data_reshaped[:,x], c=colors.loc[colors.
 ↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1] )
plt.savefig('dane_pdf/dane_stat/przestepstwa_ludnosc_all.pdf', ␣
 ↪bbox_inches='tight')
```



Comparing each year with best line

[60]:
```
notnull_data_reshaped = data_reshaped[~np.isnan(data_reshaped)].reshape(17,-1)
```

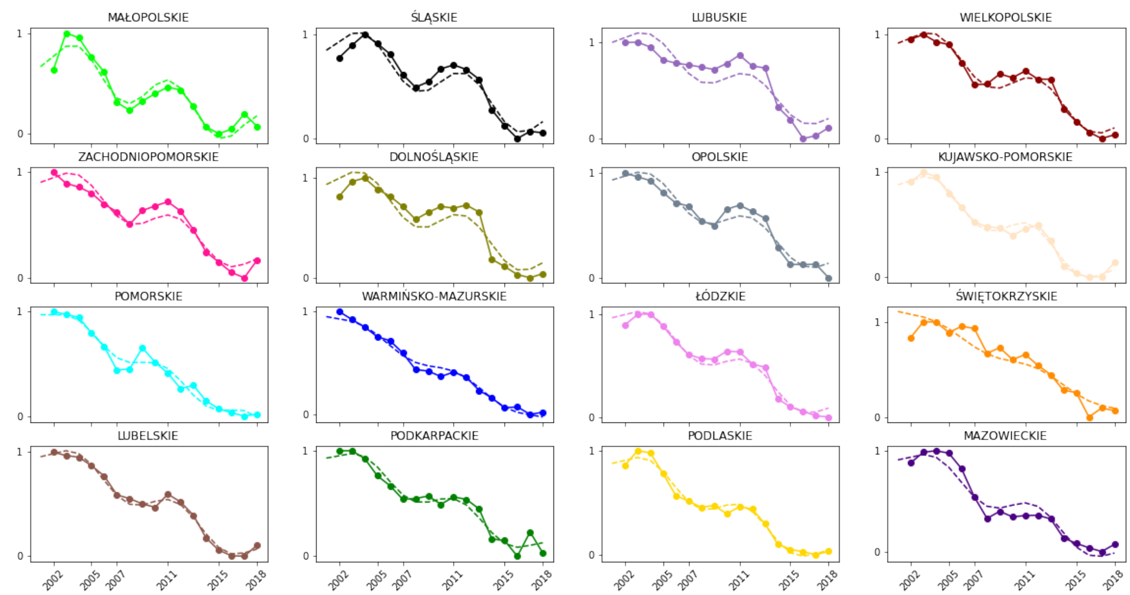
[61]:
```
mx = notnull_data_reshaped.max(0)
mn = notnull_data_reshaped.min(0)
```

[62]:
```
x = np.arange(17)

fig = plt.figure(1, figsize=(20,10))

for w in range(data_reshaped.shape[1]):
    (a,b,c,d), pcov = curve_fit(lq_sin,
                                x,
                                minmax_data_reshaped[:,w], # data_reshaped[1:
 ↪,w],
                                p0=[4,6,-1,1], maxfev = 100000)
    xmodel = np.arange(-1,17)
    ymodel = lq_sin(xmodel, a,b,c,d)
    perr = np.sqrt(np.diag(pcov))
    ax1 = fig.add_subplot(4,4,w+1)

    plt.xticks(range(-1,17), rotation=45)
    print(df_sel_null_col.index[w].ljust(20),
          " RMSE minmax",
          np.sum((minmax_data_reshaped[:,w]-ymodel[1:])**2),
```

```
        " RMSE",
        np.sum((data_reshaped[1:,w]-(ymodel[1:]*(mx[w]-mn[w])+mn[w]))**2),
       )

    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))

    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, minmax_data_reshaped[:,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])
    ax1.plot(xmodel,ymodel, linestyle="--", label="Model",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])

    ax1.set_xticks([0,3,5,9,13,16])
    #
    if w > 11:
        ax1.set_xticklabels(['2002','2005','2007','2011','2015','2018'], ␣
↪rotation=45)
    else:
        ax1.set_xticklabels([])

    #stat_list_org[0]['przestepstwa_ludnosc'][w] =␣
↪stat_list_org[-1]['przestepstwa_ludnosc'][w]
    stat_list_org[0]['przestepstwa_ludnosc'][w] =␣
↪stat_list_org[1]['przestepstwa_ludnosc'][w]
    stat_list_org[0]['przestepstwa_ludnosc'][w] = ymodel[0]

plt.savefig('dane_pdf/dane_stat/przestepstwa_ogolem.pdf',  bbox_inches='tight')
```

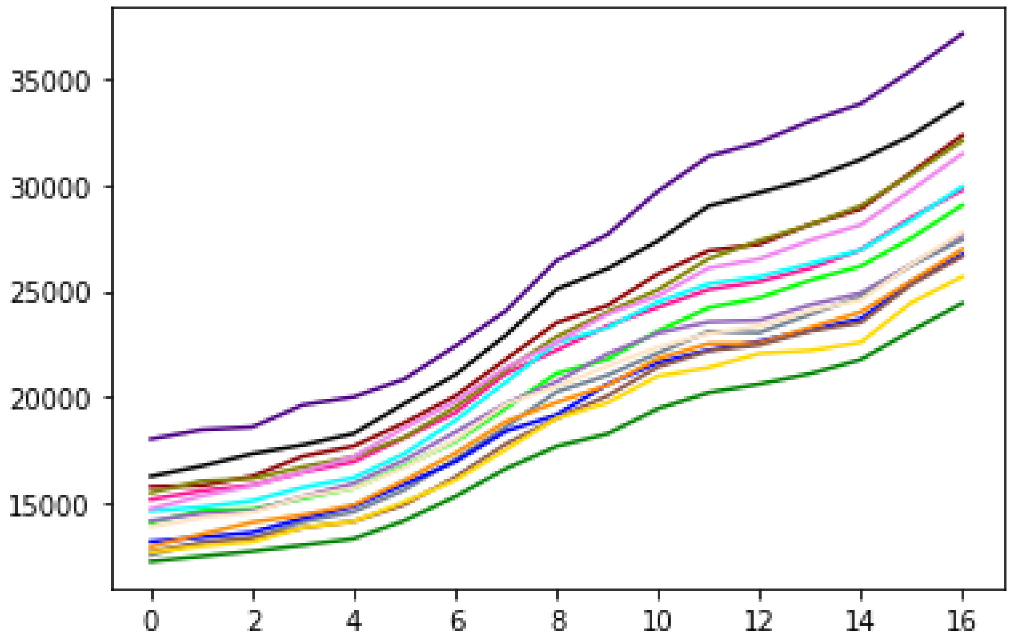
```
MAŁOPOLSKIE           RMSE minmax 0.10159950018252682  RMSE
3.760434896534175e-05
ŚLĄSKIE               RMSE minmax 0.10284964205116577  RMSE
4.0500741763200743e-05
LUBUSKIE              RMSE minmax 0.26965514620904707  RMSE
0.0001156695032832995
WIELKOPOLSKIE         RMSE minmax 0.05532254975537902  RMSE
2.160596817793382e-05
ZACHODNIOPOMORSKIE    RMSE minmax 0.10444152567119525  RMSE
5.567933303552253e-05
DOLNOŚLĄSKIE          RMSE minmax 0.19400485932482214  RMSE 7.69586217744842e-05
OPOLSKIE              RMSE minmax 0.08140556446554996  RMSE
3.005830035374502e-05
KUJAWSKO-POMORSKIE    RMSE minmax 0.028122580548659022  RMSE
1.4025168027181782e-05
POMORSKIE             RMSE minmax 0.06255714701293735  RMSE 6.89557247919933e-05
```

```
WARMIŃSKO-MAZURSKIE   RMSE minmax 0.02916912460607001  RMSE
1.6688740380454326e-05
ŁÓDZKIE               RMSE minmax 0.052388322760763835  RMSE
1.7714073897048288e-05
ŚWIĘTOKRZYSKIE        RMSE minmax 0.17343759930856817  RMSE
5.559125094388598e-05
LUBELSKIE             RMSE minmax 0.02060948764510956  RMSE
4.972728036140719e-06
PODKARPACKIE          RMSE minmax 0.06777053034095752  RMSE
1.1210347770179588e-05
PODLASKIE             RMSE minmax 0.031613499874988016  RMSE
8.708612956476871e-06
MAZOWIECKIE           RMSE minmax 0.11750808845447258  RMSE
5.2295810872992436e-05
```



1.9.8 dochody_brutto_na_mieszkanca

[63]:
```
data_reshaped = df_sel_null_col['dochody_brutto_na_mieszkanca'].values.
 ↪reshape(18,-1)
for x in range(data_reshaped.shape[1]):
    plt.plot(data_reshaped[:,x], c=colors.loc[colors.
 ↪wojewodztwo==stat_list_org[0].index[x],:].values[0][1])
plt.savefig('dane_pdf/dane_stat/dochody_brutto_na_mieszkanca_all.pdf', ␣
 ↪bbox_inches='tight')
```



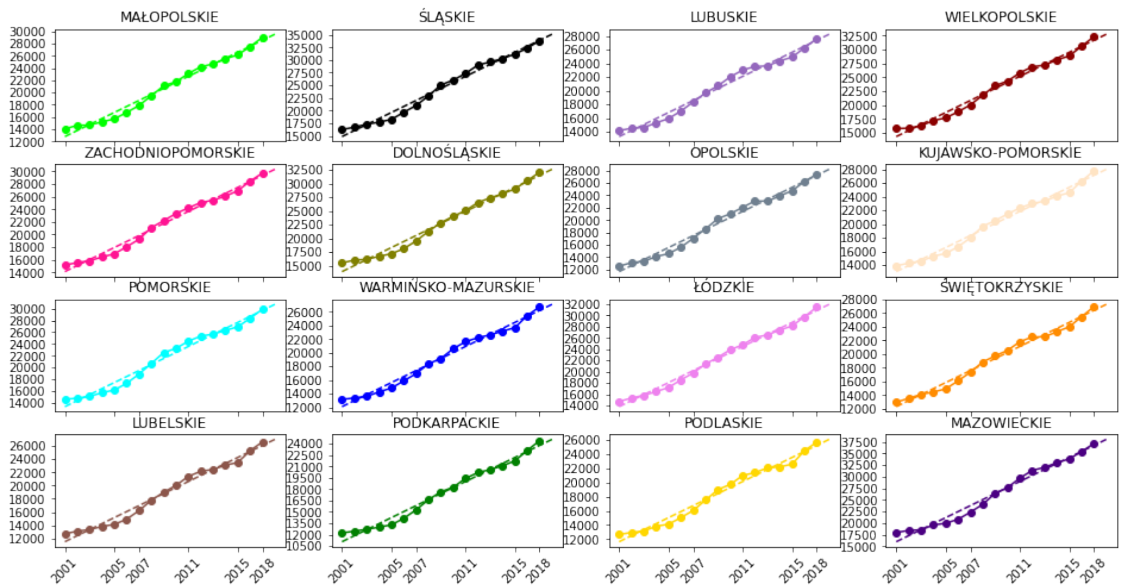
[64]:
```
def lq3(x, a, b, c, d):
    return x*x*x*a + x*x*b + x*c + d

def lq1(x, a, b):
    return x*a + b

x = np.arange(17)
```

[65]:
```
x = np.arange(17)

fig = plt.figure(1, figsize=(16,8))

mse = []
rmse = []

for w in range(data_reshaped.shape[1]):
    (a,b), pcov = curve_fit(lq1, x, data_reshaped[:-1,w], p0=[1,1])
    xmodel = np.arange(18)
    ymodel = lq1(xmodel, a, b)
    perr = np.sqrt(np.diag(pcov))

    mse.append(np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:-1].
↪shape[0])
    rmse.append(np.sqrt(np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:
↪-1].shape[0]))
```

```
    print(df_sel_null_col.index[w].ljust(20),
          " MSE",np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:-1].
↪shape[0],
          " RMSE",np.sqrt(np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:
↪-1].shape[0]))
    ax1 = fig.add_subplot(4,4,w+1)

    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))

    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, data_reshaped[:-1,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])
    ax1.plot(xmodel,ymodel, linestyle="--", label="Model",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
↪values[0][1])
    ax1.set_xticks([0,4,6,10,14,16])
    #
    if w > 11:
        ax1.set_xticklabels(['2001','2005','2007','2011','2015','2018'], ␣
↪rotation=45)
    else:
        ax1.set_xticklabels([])
    stat_list_org[-1]['dochody_brutto_na_mieszkanca'][w] = ymodel[-1]
plt.savefig('dane_pdf/dane_stat/dochody_brutto_na_mieszkanca_straight.pdf', ␣
↪bbox_inches='tight')

print("Avg MSE:", np.mean(mse), "+-", np.std(mse),
      "Avg RMSE:", np.mean(rmse), "+-", np.std(rmse),)
```

```
MAŁOPOLSKIE          MSE 1353130.8878275864  RMSE 1163.2415432005455
ŚLĄSKIE              MSE 1983029.2907284459  RMSE 1408.2007281380186
LUBUSKIE             MSE 1102659.114827824  RMSE 1050.0757662320486
WIELKOPOLSKIE        MSE 1593938.7789102101  RMSE 1262.5128826709888
ZACHODNIOPOMORSKIE   MSE 1234955.1230084854  RMSE 1111.2853472481697
DOLNOŚLĄSKIE         MSE 1721935.4852209485  RMSE 1312.225394214328
OPOLSKIE             MSE 1291217.3776066771  RMSE 1136.317463390701
KUJAWSKO-POMORSKIE   MSE 1051546.4923072509  RMSE 1025.4494099209628
POMORSKIE            MSE 1591253.8691449473  RMSE 1261.4491147664053
WARMIŃSKO-MAZURSKIE  MSE 1060632.5905710773  RMSE 1029.8701814166081
ŁÓDZKIE              MSE 1387655.7189286721  RMSE 1177.9879960885307
ŚWIĘTOKRZYSKIE       MSE 1033690.0239022665  RMSE 1016.70547549537
LUBELSKIE            MSE 1253862.4052856152  RMSE 1119.759976640358
PODKARPACKIE         MSE 924161.7456697712  RMSE 961.3333166336072
PODLASKIE            MSE 1083520.5237824072  RMSE 1040.9229192319704
```

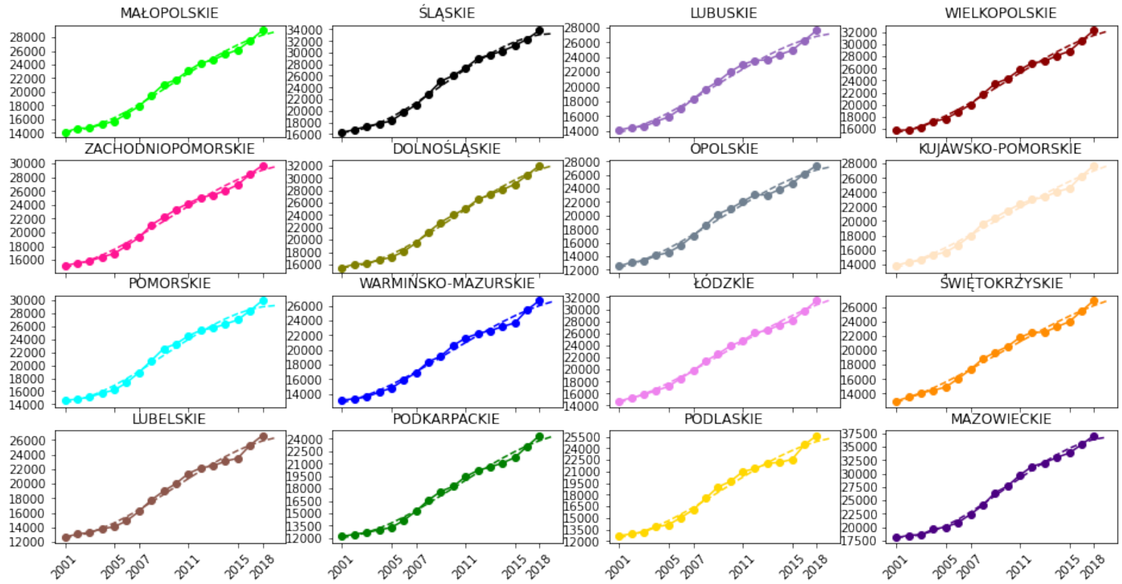
```
MAZOWIECKIE           MSE 2543464.1602963726  RMSE 1594.8241784900217
Avg MSE: 1388165.84925116 +- 408623.4655871618 Avg RMSE: 1167.0101058611644 +-
162.02858411117808
```



[66]:
```
x = np.arange(17)

fig = plt.figure(1, figsize=(16,8))

mse = []
rmse = []
for w in range(data_reshaped.shape[1]):
    (a,b,c,d), pcov = curve_fit(lq3, x, data_reshaped[:-1,w], p0=[1,1,1,1])
    xmodel = np.arange(18)
    ymodel = lq3(xmodel, a, b, c,d)
    perr = np.sqrt(np.diag(pcov))

    mse.append(np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:-1].
 ↪shape[0])
    rmse.append(np.sqrt(np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:
 ↪-1].shape[0]))

    print(df_sel_null_col.index[w].ljust(20),
          " MSE",np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:-1].
 ↪shape[0],
          " RMSE",np.sqrt(np.sum((data_reshaped[:-1,w]-ymodel[1:])**2)/ymodel[:
 ↪-1].shape[0]))
    ax1 = fig.add_subplot(4,4,w+1)
```

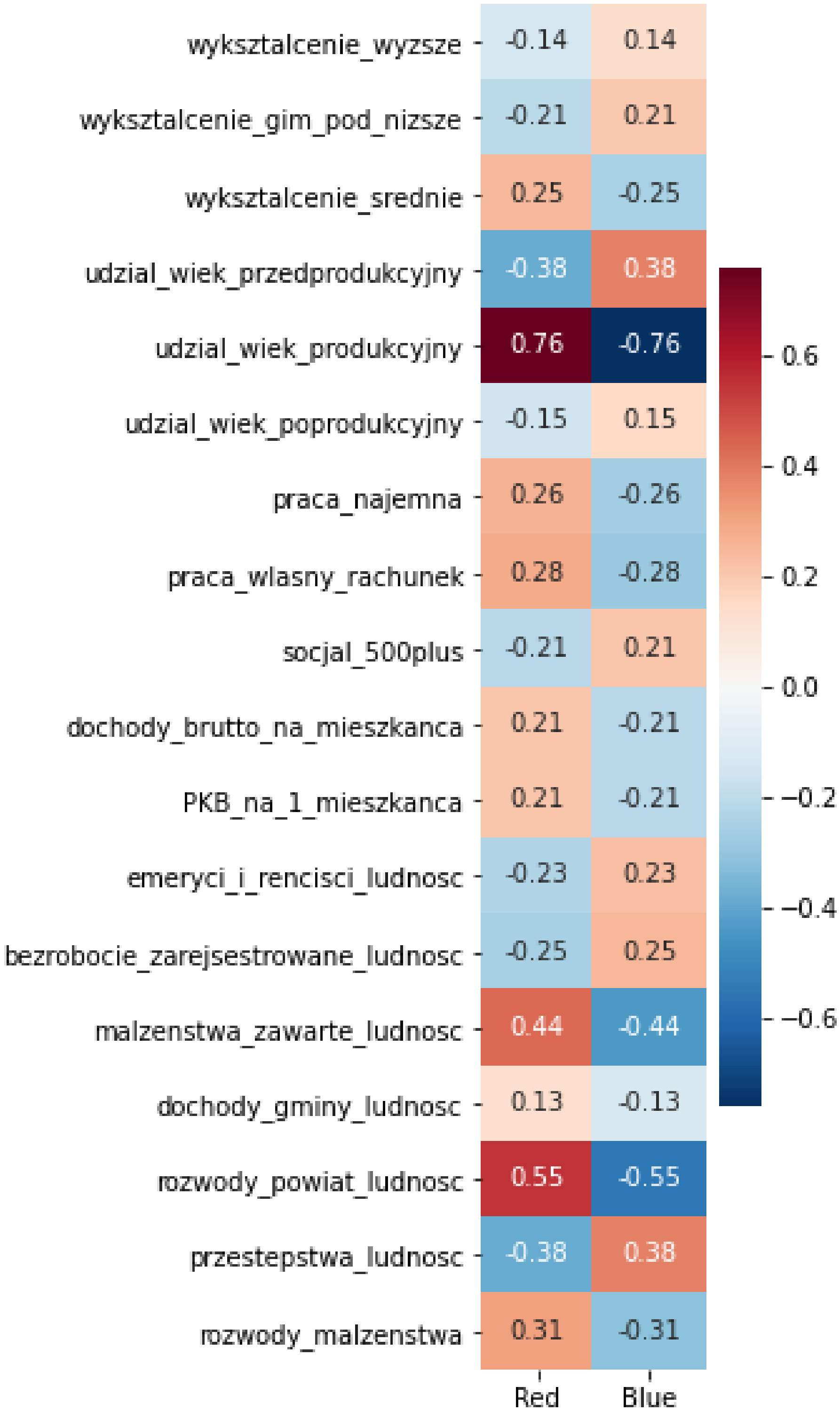
```
    for axis in [ax1.yaxis]:
        axis.set_major_locator(ticker.MaxNLocator(integer=True))

    ax1.set_title(stat_list_org[0].index[w])
    ax1.plot(x, data_reshaped[:-1,w], marker="o", label="Experiment",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
 ↪values[0][1])
    ax1.plot(xmodel,ymodel, linestyle="--", label="Model",
             color=colors.loc[colors.wojewodztwo==stat_list_org[0].index[w],:].
 ↪values[0][1])
    ax1.set_xticks([0,4,6,10,14,16])
    #
    if w > 11:
        ax1.set_xticklabels(['2001','2005','2007','2011','2015','2018'], ␣
 ↪rotation=45)
    else:
        ax1.set_xticklabels([])
    stat_list_org[-1]['dochody_brutto_na_mieszkanca'][w] = ymodel[-1]
plt.savefig('dane_pdf/dane_stat/dochody_brutto_na_mieszkanca.pdf', ␣
 ↪bbox_inches='tight')
print("Avg MSE:", np.mean(mse), "+-", np.std(mse),
      "Avg RMSE:", np.mean(rmse), "+-", np.std(rmse),)
```

```
MAŁOPOLSKIE           MSE 1001566.2174107108  RMSE 1000.7828023156227
ŚLĄSKIE               MSE 1380131.3024248013  RMSE 1174.789897141102
LUBUSKIE              MSE 901479.4779143524  RMSE 949.4627311876715
WIELKOPOLSKIE         MSE 1274430.1843914797  RMSE 1128.9066322736703
ZACHODNIOPOMORSKIE    MSE 1010010.6990843877  RMSE 1004.9928850914258
DOLNOŚLĄSKIE          MSE 1210706.095420665  RMSE 1100.3209056546482
OPOLSKIE              MSE 1055740.3209630873  RMSE 1027.4922486146002
KUJAWSKO-POMORSKIE    MSE 939531.6255735898  RMSE 969.2943957196852
POMORSKIE             MSE 1212321.9707982892  RMSE 1101.0549354134375
WARMIŃSKO-MAZURSKIE   MSE 882571.2444670415  RMSE 939.4526302411641
ŁÓDZKIE               MSE 1200029.3466260068  RMSE 1095.4585097693143
ŚWIĘTOKRZYSKIE        MSE 927161.7310035249  RMSE 962.8923776848194
LUBELSKIE             MSE 953127.8858876805  RMSE 976.2826874874308
PODKARPACKIE          MSE 710405.9622500475  RMSE 842.8558371691137
PODLASKIE             MSE 876238.26588332  RMSE 936.0759936475885
MAZOWIECKIE           MSE 1650991.9794324937  RMSE 1284.9093273194392
Avg MSE: 1074152.7693457175 +- 226682.53398915633 Avg RMSE: 1030.9390497956708
+- 106.38348063546673
```



1.10 Compare correlations after rescaling

[67]:
```
pool_stat_org = [pd.merge(s, p, left_index=True, right_index=True).
 ↪reset_index() for s,p in zip(stat_list_org, pool_d[1:])]
pool_stat_df_org = pd.concat(pool_stat_org).reset_index(drop=True)

for w, c in enumerate([
    'emeryci_i_rencisci',
    'bezrobocie_zarejsestrowane',
    'malzenstwa_zawarte',
    'dochody_gminy',
    'rozwody_powiat',
    'przestepstwa_ogolem']):
    pool_stat_df_org = pool_stat_df_org.drop(c, axis=1)

fig, ax = plt.subplots(figsize=(2,10))
sn.heatmap(pool_stat_df_org[pool_names_highcorr.columns.values.
 ↪tolist()+['Red','Blue']].corr().iloc[:-2,-2:], annot=True, cmap='RdBu_r',␣
 ↪ax=ax)
plt.show()
```


wyksztalcenie_wyzsze
wyksztalcenie_gim_pod_nizsze
wyksztalcenie_srednie
udzial_wiek_przedprodukcyjny
udzial_wiek_produkcyjny
udzial_wiek_poprodukcyjny
praca_najemna
praca_wlasny_rachunek
socjal_500plus
dochody_brutto_na_mieszkanca
PKB_na_1_mieszkanca
emeryci_i_rencisci_ludnosc
bezrobocie_zarejsestrowane_ludnosc
malzenstwa_zawarte_ludnosc
dochody_gminy_ludnosc
rozwody_powiat_ludnosc
przestepstwa_ludnosc
rozwody_malzenstwa
-0.14 0.14
-0.21 0.21
0.25 -0.25
-0.38 0.38
0.76 -0.76
-0.15 0.15
0.26 -0.26
0.28 -0.28
-0.21 0.21
0.21 -0.21
0.21 -0.21
-0.23 0.23
-0.25 0.25
0.44 -0.44
0.13 -0.13
0.55 -0.55
-0.38 0.38
0.31 -0.31
Red
Blue
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

1.11 Rescale the data

[74]:
```
col_not_na0 = stat_list_org[0].columns[stat_list_org[0].isna().sum(0) == 0]
col_not_na17 = stat_list_org[0].columns[stat_list_org[17].isna().sum(0) == 0]
```

[75]:
```
stat_list_minmax = stat_list_org.copy()
stat_list_mean = stat_list_org.copy()
```

[76]:
```
#MINMAX scaling
for st in range(len(stat_list_org)):
    stat_list_minmax[st] = stat_list_org[st][col_not_na0 & col_not_na17].copy()
    # min max normalization
    stat_list_minmax[st] = (stat_list_minmax[st]-stat_list_minmax[st].min(0)) /␣
↪(stat_list_minmax[st].max(0)-stat_list_minmax[st].min(0))
```

[77]:
```
#NORLAMISATION
for st in range(len(stat_list_org)):
    stat_list_mean[st] = stat_list_org[st][col_not_na0 & col_not_na17].copy()
    # nrmalization mean (around 0)
    stat_list_mean[st] = (stat_list_mean[st]-stat_list_mean[st].mean(0)) /␣
↪(stat_list_mean[st].std(0))
```

[78]:
```
stat_list = stat_list_mean if (True) else stat_list_minmax
```

1.12 Examining stat data

[91]:
```
high_corr_columns_final = pool_names_highcorr.columns & col_not_na0 &␣
↪col_not_na17
print(high_corr_columns_final.shape, high_corr_columns_final)
```

```
(17,) Index(['wyksztalcenie_wyzsze', 'wyksztalcenie_gim_pod_nizsze',
       'wyksztalcenie_srednie', 'udzial_wiek_przedprodukcyjny',
       'udzial_wiek_produkcyjny', 'udzial_wiek_poprodukcyjny', 'praca_najemna',
       'praca_wlasny_rachunek', 'dochody_brutto_na_mieszkanca',
       'PKB_na_1_mieszkanca', 'emeryci_i_rencisci_ludnosc',
       'bezrobocie_zarejsestrowane_ludnosc', 'malzenstwa_zawarte_ludnosc',
       'dochody_gminy_ludnosc', 'rozwody_powiat_ludnosc',
       'przestepstwa_ludnosc', 'rozwody_malzenstwa'],
      dtype='object')
```

[108]:
```
pool_names_highcorr
```

[108]:
```
      wyksztalcenie_wyzsze  wyksztalcenie_gim_pod_nizsze  \
Blue               0.13546                      0.205325
Red               -0.13546                     -0.205325

      wyksztalcenie_srednie  udzial_wiek_przedprodukcyjny  \
```

```
Blue              -0.247968                        0.24773
Red                0.247968                       -0.24773

      udzial_wiek_produkcyjny  udzial_wiek_poprodukcyjny  praca_najemna  \
Blue                -0.747516                   0.270503      -0.261207
Red                  0.747516                  -0.270503       0.261207

      praca_wlasny_rachunek  socjal_500plus  dochody_brutto_na_mieszkanca  \
Blue              -0.280812         0.20763                      -0.26749
Red                0.280812        -0.20763                       0.26749

      PKB_na_1_mieszkanca  emeryci_i_rencisci_ludnosc  \
Blue            -0.210211                    0.228949
Red              0.210211                   -0.228949

      bezrobocie_zarejsestrowane_ludnosc  malzenstwa_zawarte_ludnosc  \
Blue                             0.25258                   -0.438434
Red                             -0.25258                    0.438434

      dochody_gminy_ludnosc  rozwody_powiat_ludnosc  przestepstwa_ludnosc  \
Blue               -0.12987               -0.550899             -0.328121
Red                 0.12987                0.550899              0.328121

      rozwody_malzenstwa
Blue           -0.313341
Red             0.313341
```

[92]:
```
len(stat_list)
```

[92]: 18

[93]:
```
len(stat_list_org)
```

[93]: 18

[94]:
```
len(pool_d[:-1])
```

[94]: 18

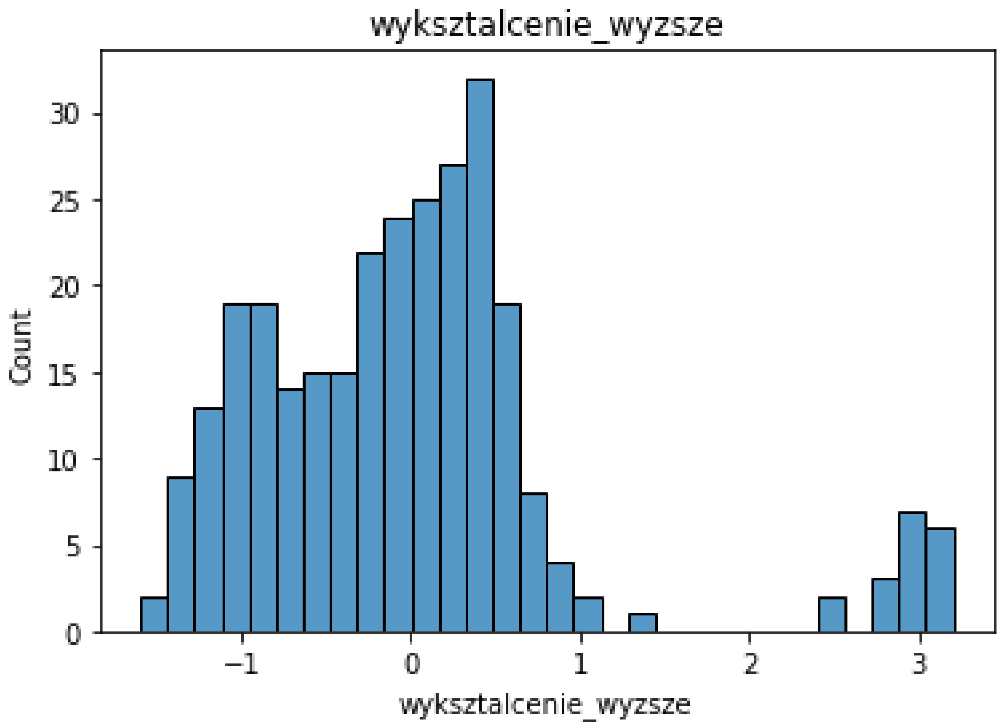
[95]:
```
pool_stat_org = [pd.merge(s, p, left_index=True, right_index=True).
 ↪reset_index() for s,p in zip(stat_list_org, pool_d[1:])]
pool_stat_m = [pd.merge(s, p, left_index=True, right_index=True).reset_index()␣
 ↪for s,p in zip(stat_list_mean, pool_d[1:])]
pool_stat_mm = [pd.merge(s, p, left_index=True, right_index=True).reset_index()␣
 ↪for s,p in zip(stat_list_minmax, pool_d[1:])]
```

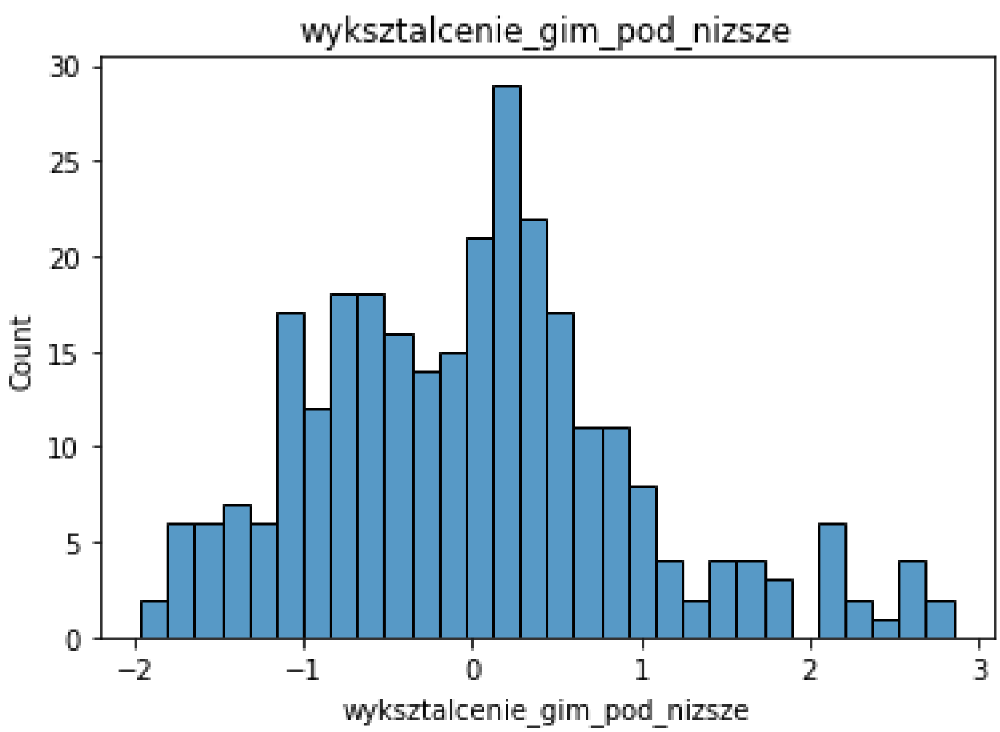
[96]:
```
pool_stat_df_org =  pd.concat(pool_stat_org).reset_index(drop=True)
pool_stat_df_m = pd.concat(pool_stat_m).reset_index(drop=True)
pool_stat_df_mm = pd.concat(pool_stat_mm).reset_index(drop=True)
```

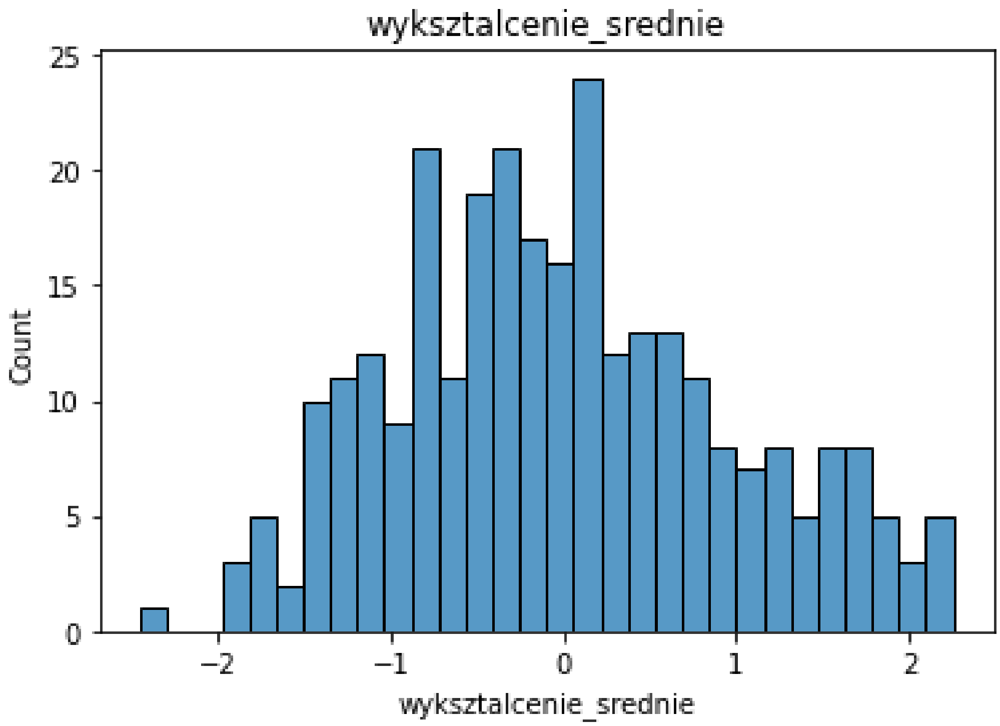
[104]:
```
try:
    for w, c in enumerate([
        'emeryci_i_rencisci',
        'bezrobocie_zarejsestrowane',
        'malzenstwa_zawarte',
        'dochody_gminy',
        'rozwody_powiat']):
        pool_stat_df_org = pool_stat_df_org.drop(c, axis=1)
        pool_stat_df_m = pool_stat_df_m.drop(c, axis=1)
        pool_stat_df_mm = pool_stat_df_mm.drop(c, axis=1)
except:
    print('already deleted')

pool_stat_df_org = pool_stat_df_org[high_corr_columns_final.values.
 ↪tolist()+['Red','Blue']]
pool_stat_df_m = pool_stat_df_m[high_corr_columns_final.values.
 ↪tolist()+['Red','Blue']]
pool_stat_df_mm = pool_stat_df_mm[high_corr_columns_final.values.
 ↪tolist()+['Red','Blue']]
```

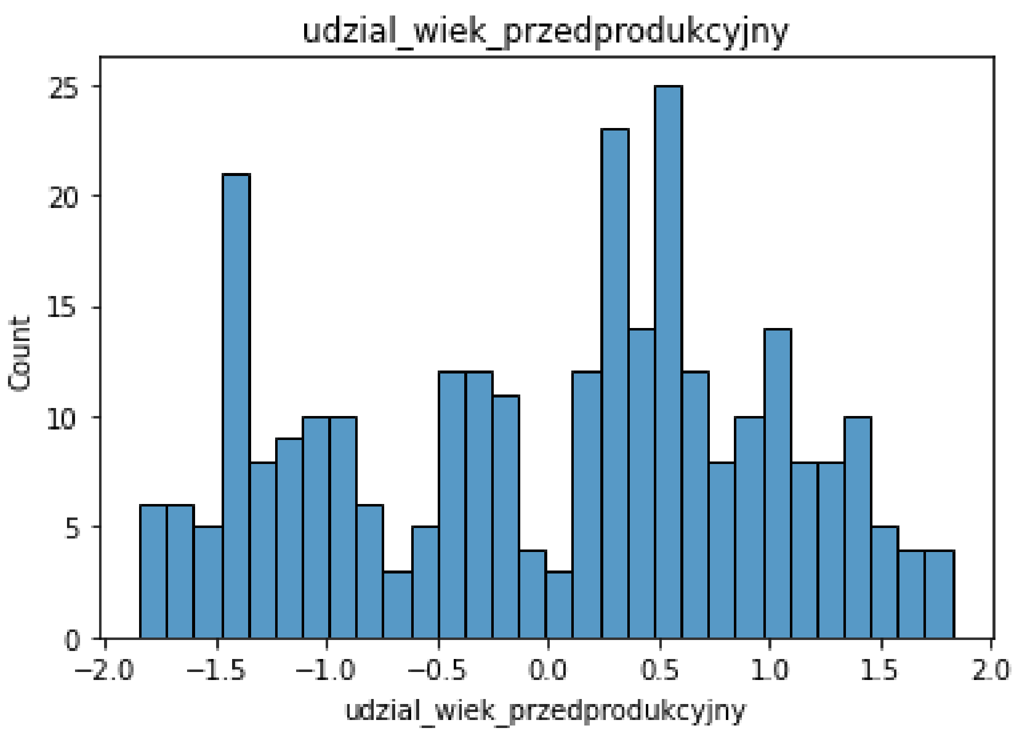
```
already deleted
```

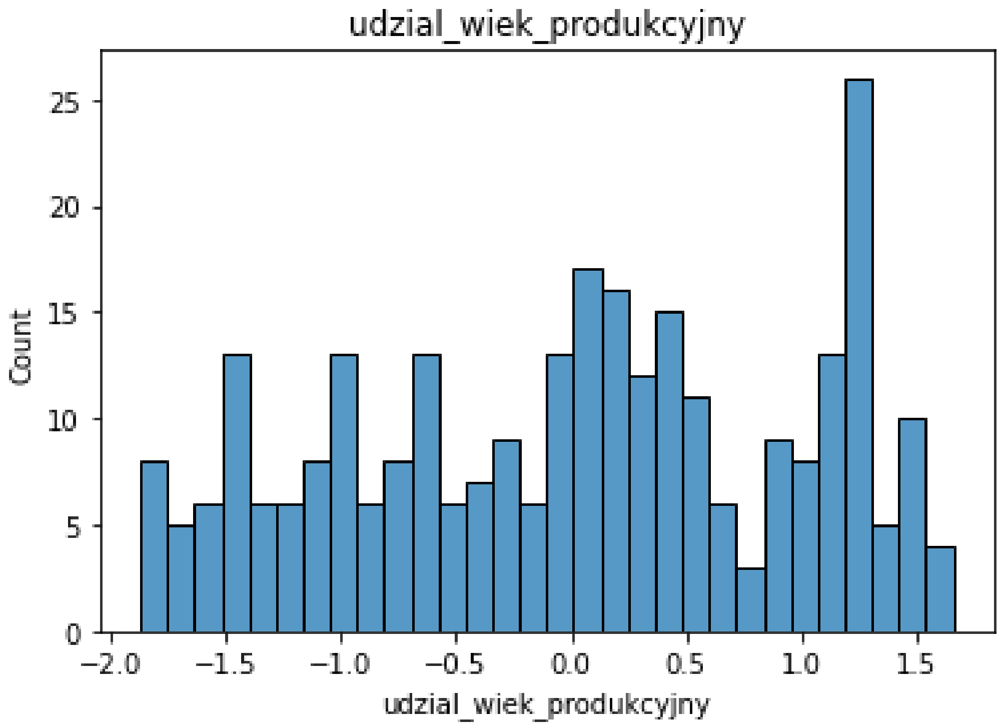
[105]:
```
for c in high_corr_columns_final:
    sn.histplot(pool_stat_df_m[c], bins=30)
    plt.title(c)
    plt.show()
```


wyksztalcenie_wyzsze
Count
wyksztalcenie_wyzsze
0
5
10
15
20
25
30
−1
0
1
2
3

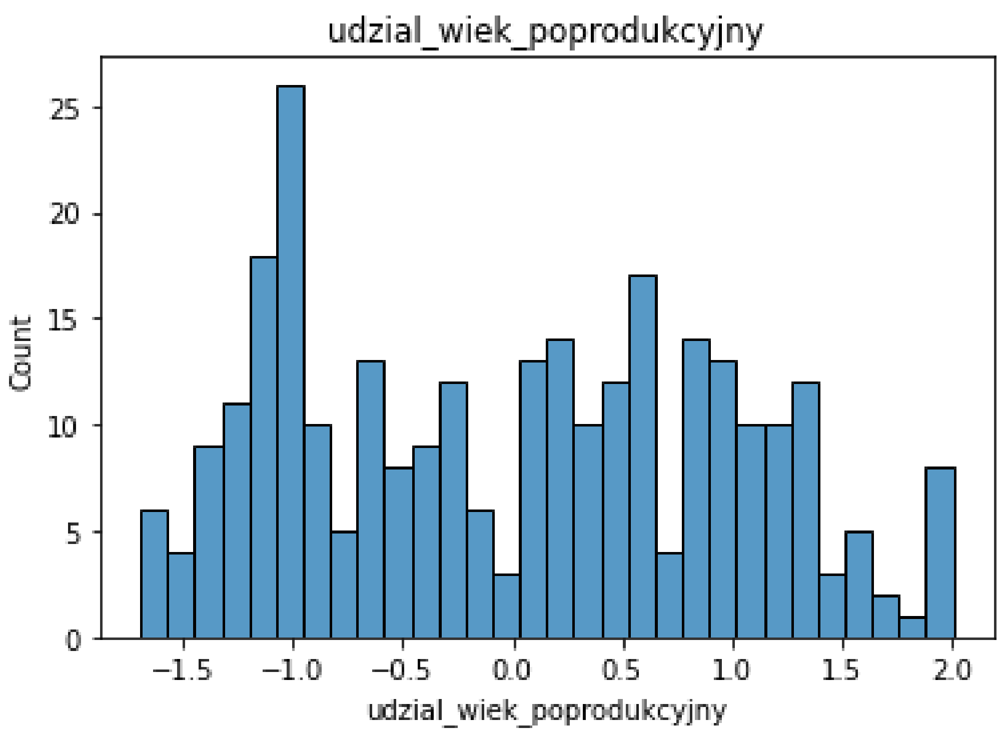

wyksztalcenie_gim_pod_nizsze
Count
wyksztalcenie_gim_pod_nizsze
0
5
10
15
20
25
30
−2
−1
0
1
2
3


wyksztalcenie_srednie
Count
25
20
15
10
5
0
−2
−1
0
1
2
wyksztalcenie_srednie

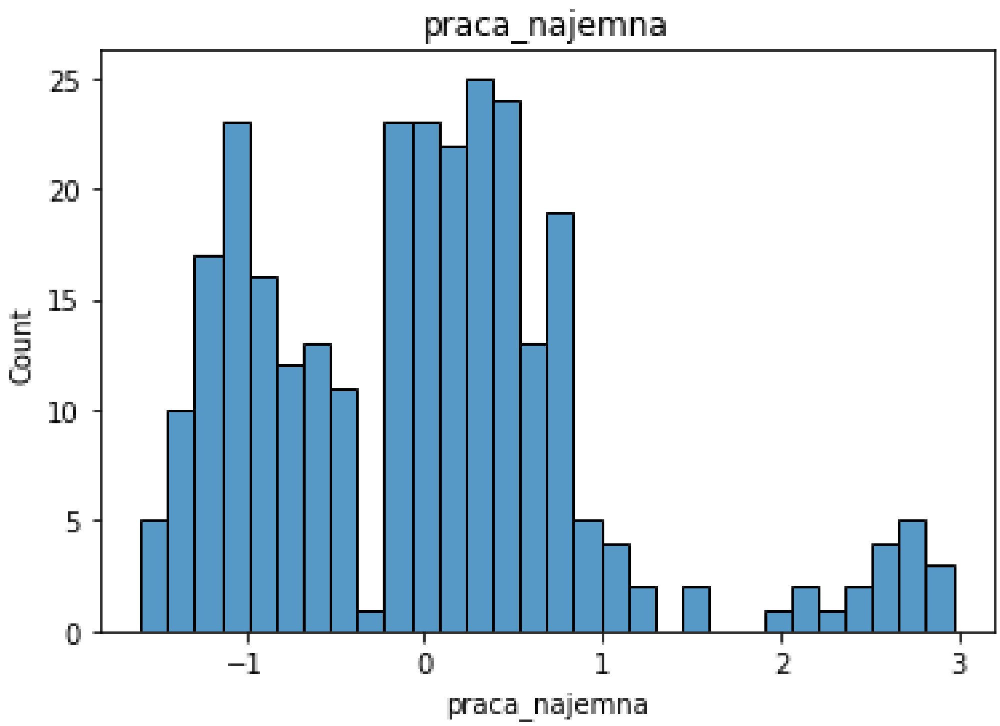

udzial_wiek_przedprodukcyjny
Count
25
20
15
10
5
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
udzial_wiek_przedprodukcyjny

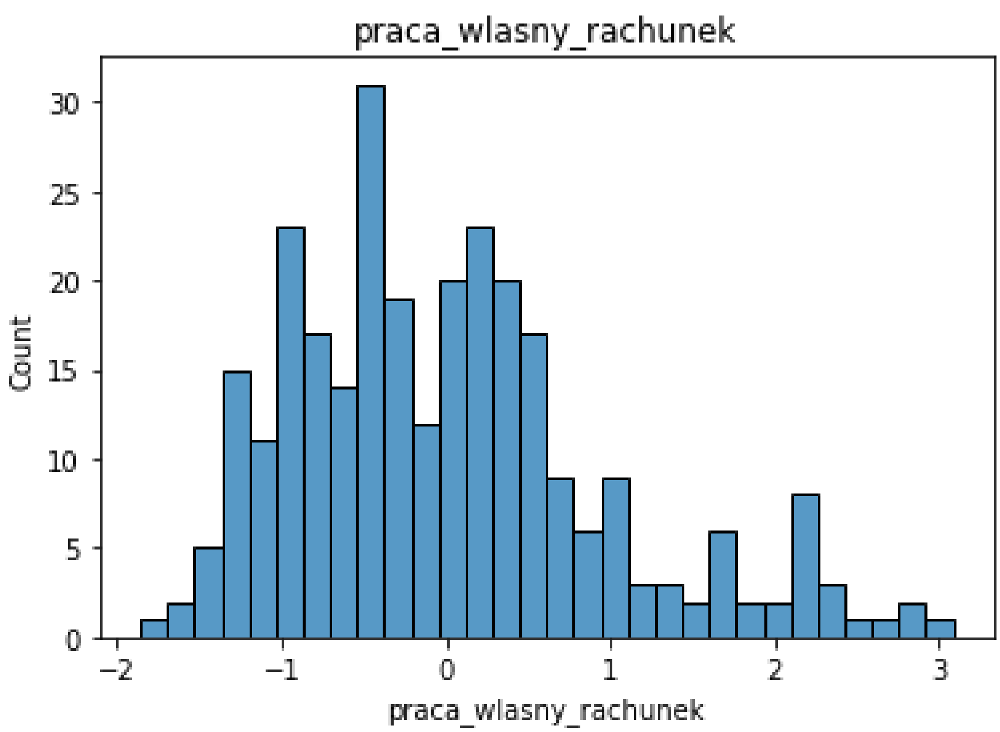


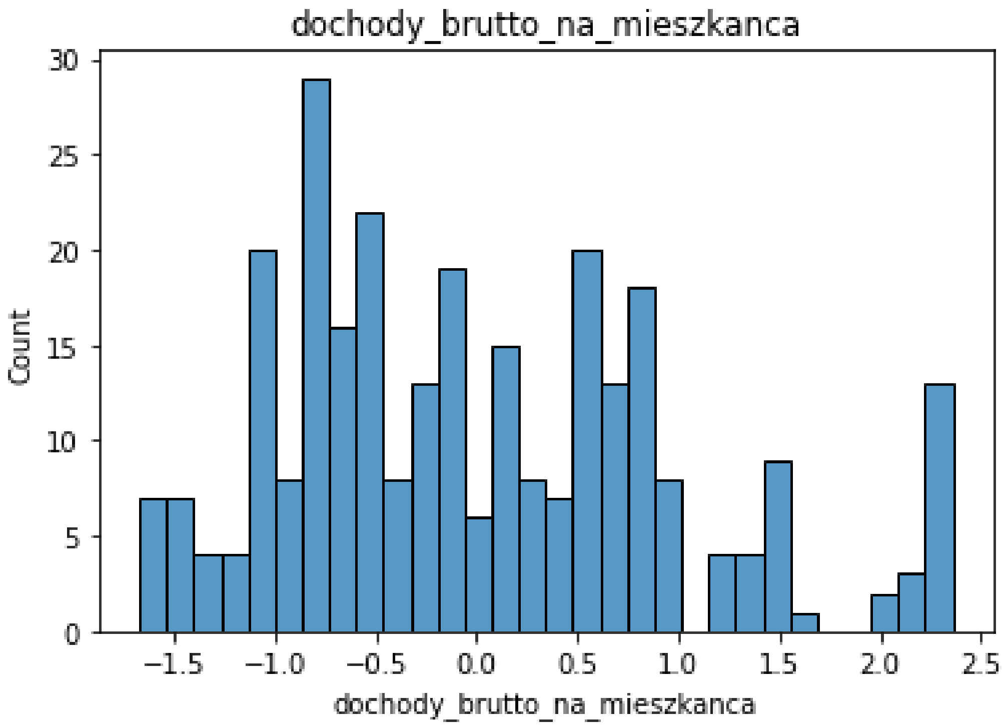
udzial_wiek_produkcyjny
Count
udzial_wiek_produkcyjny

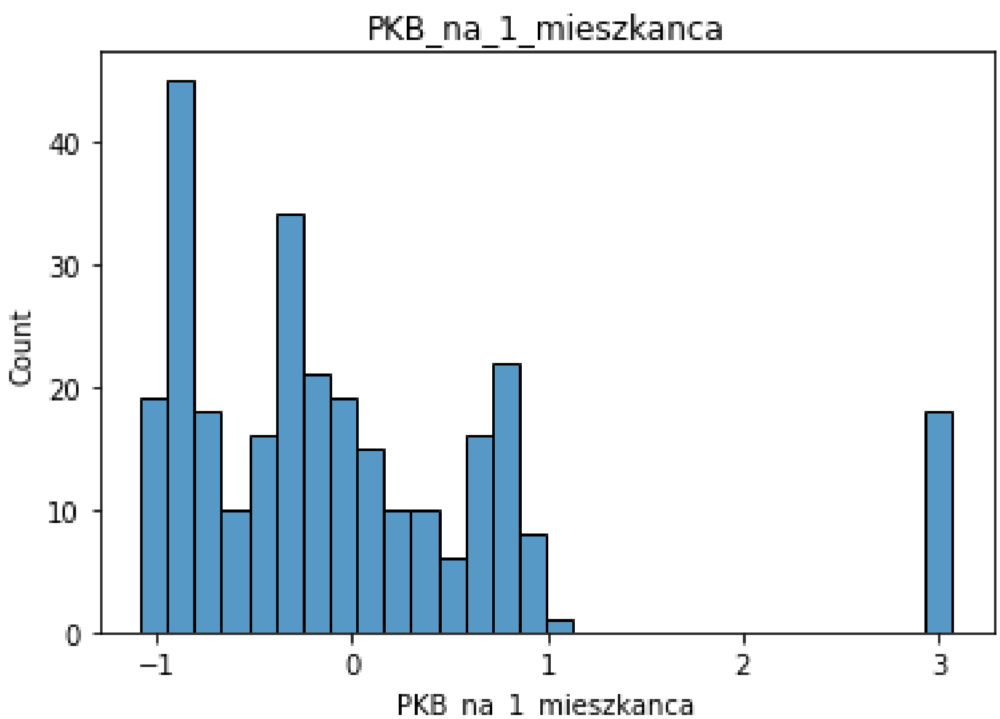


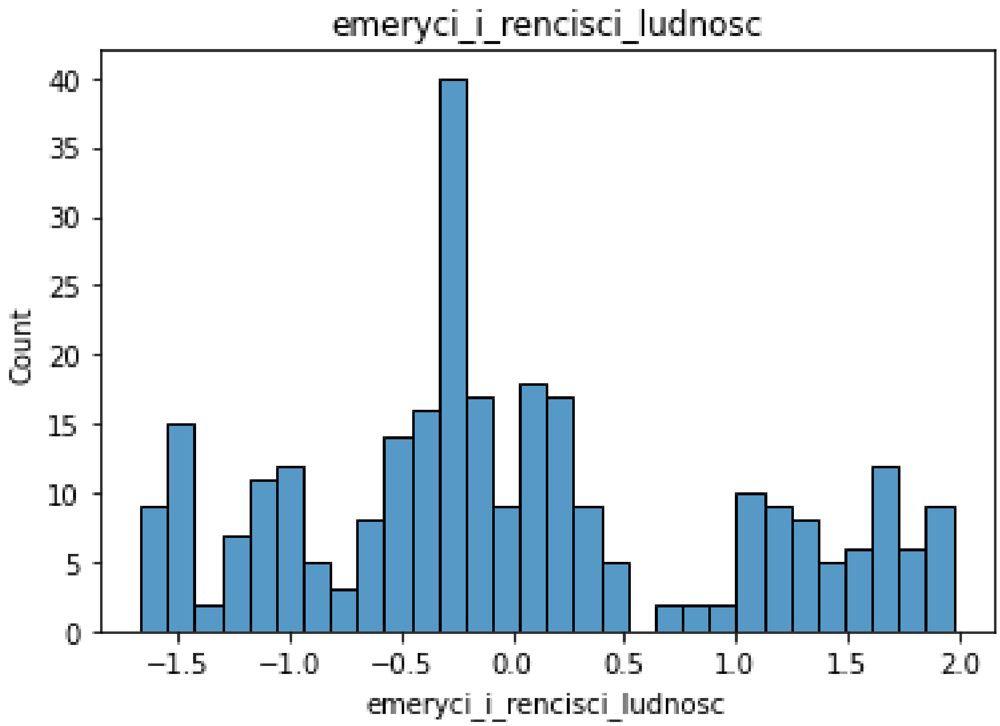
udzial_wiek_poprodukcyjny
Count
udzial_wiek_poprodukcyjny

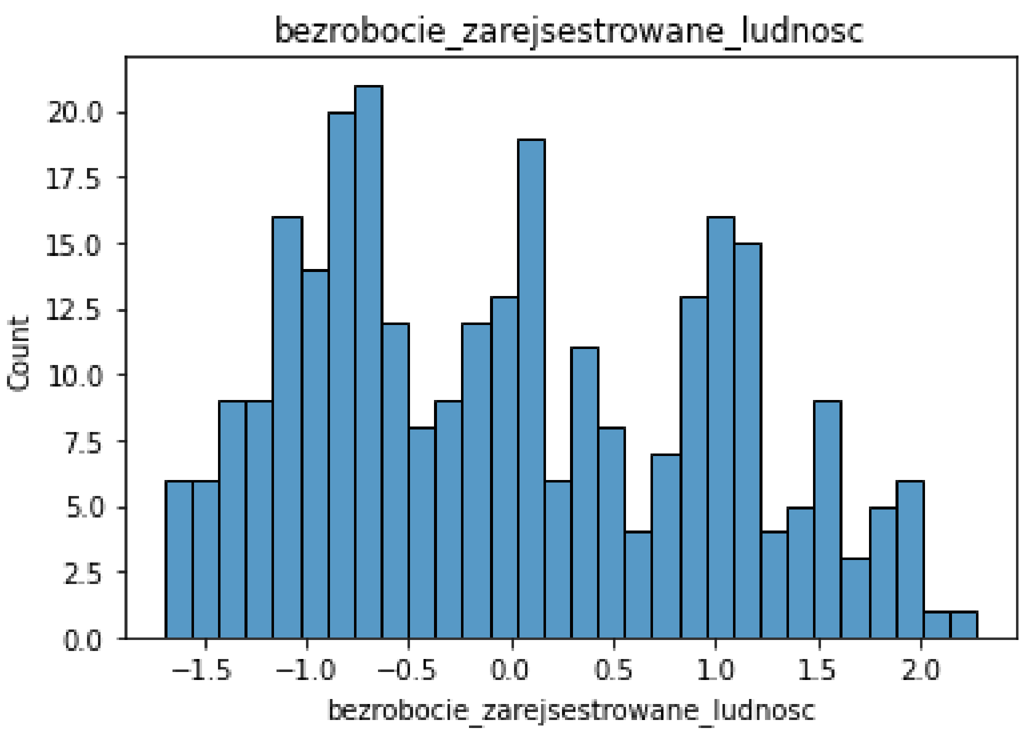

praca_najemna
Count
praca_najemna

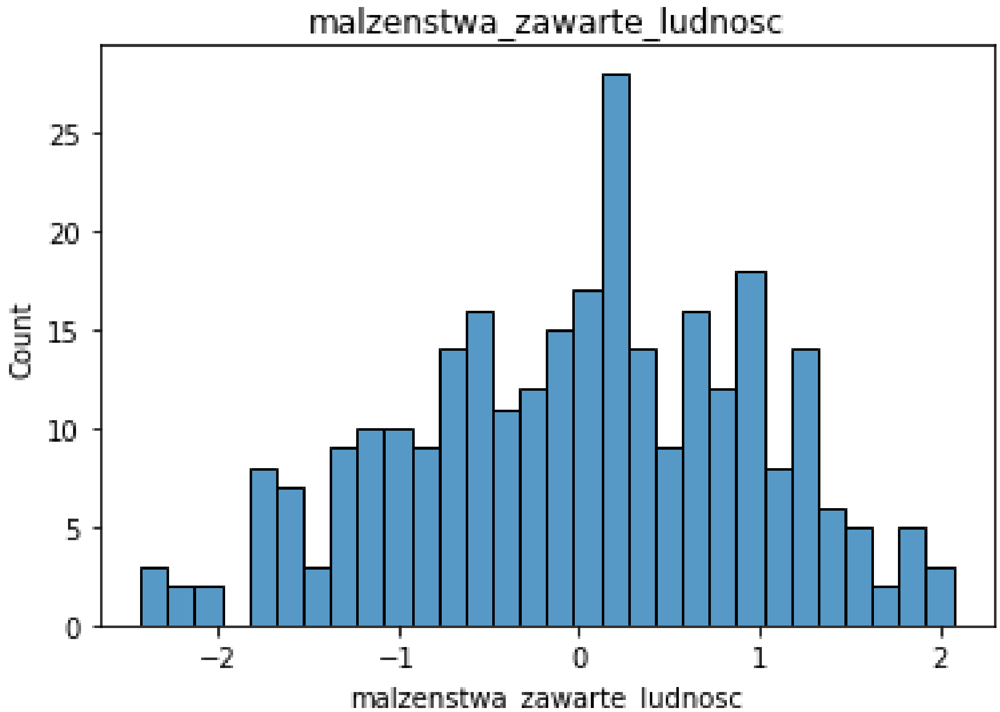

praca_wlasny_rachunek
Count
praca_wlasny_rachunek

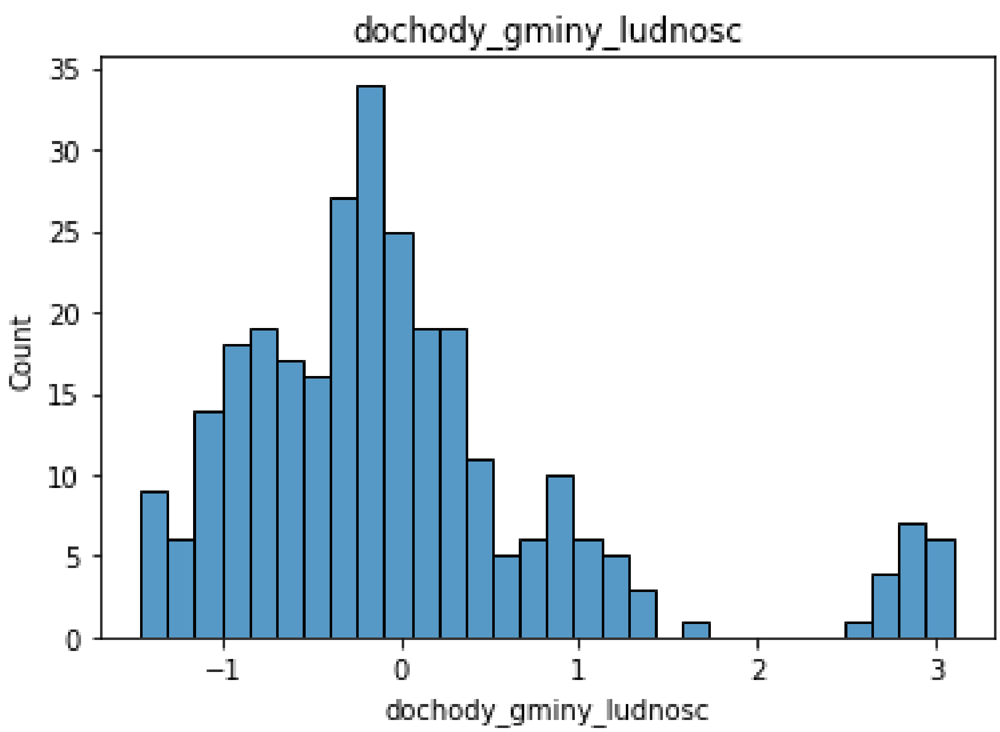

dochody_brutto_na_mieszkanca
Count
dochody_brutto_na_mieszkanca

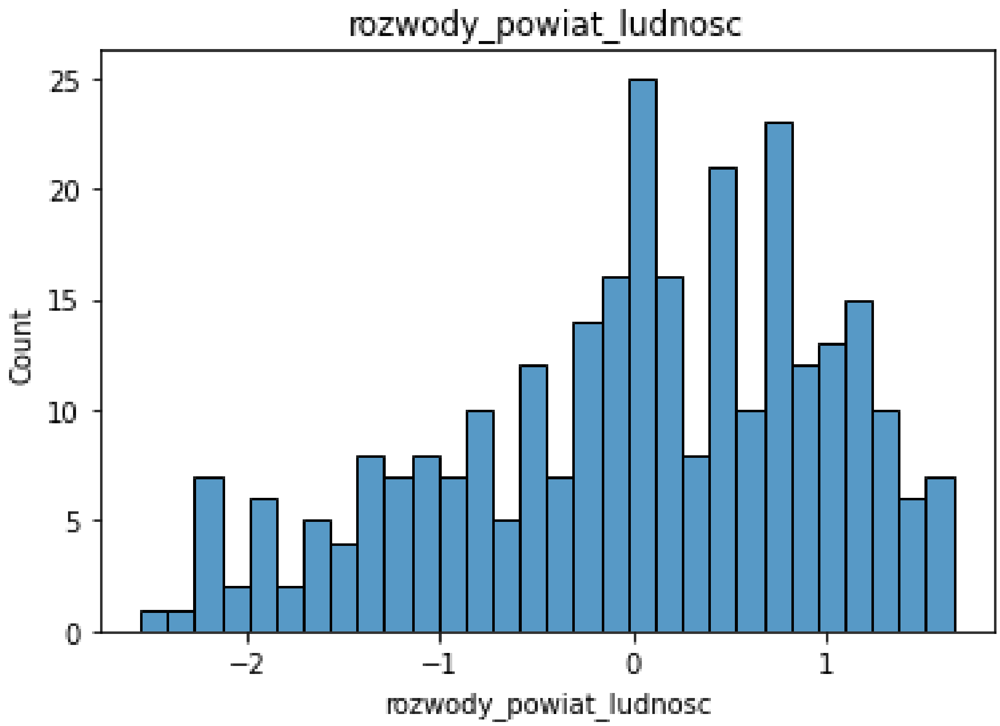

PKB_na_1_mieszkanca
Count
PKB_na_1_mieszkanca

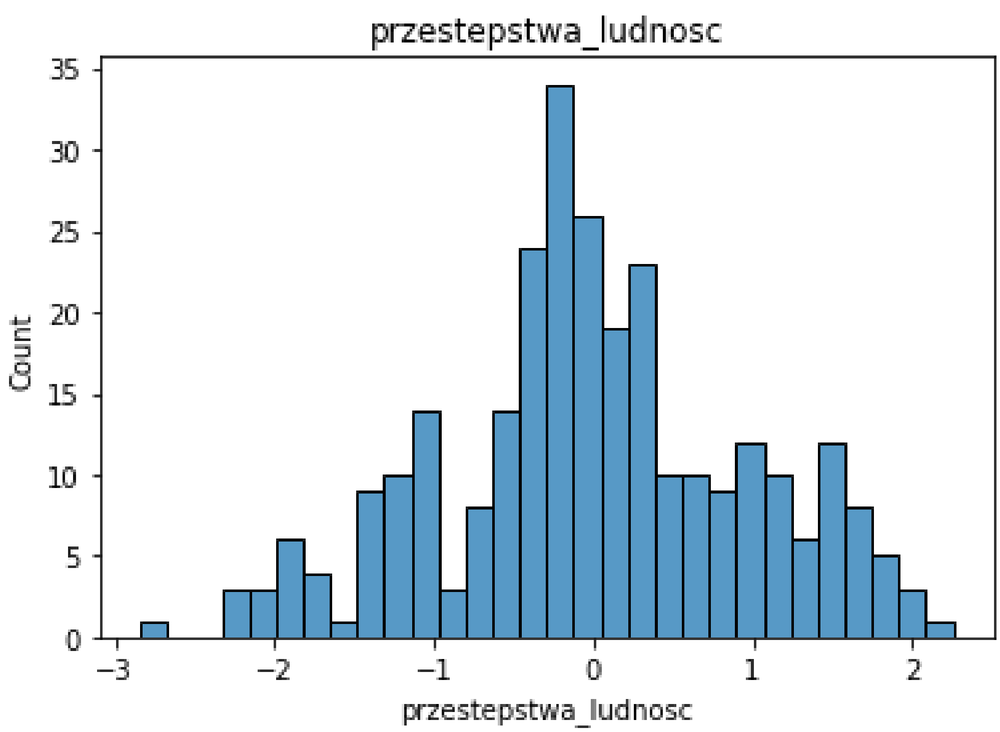

emeryci_i_rencisci_ludnosc
Count
40
35
30
25
20
15
10
5
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
emeryci_i_rencisci_ludnosc

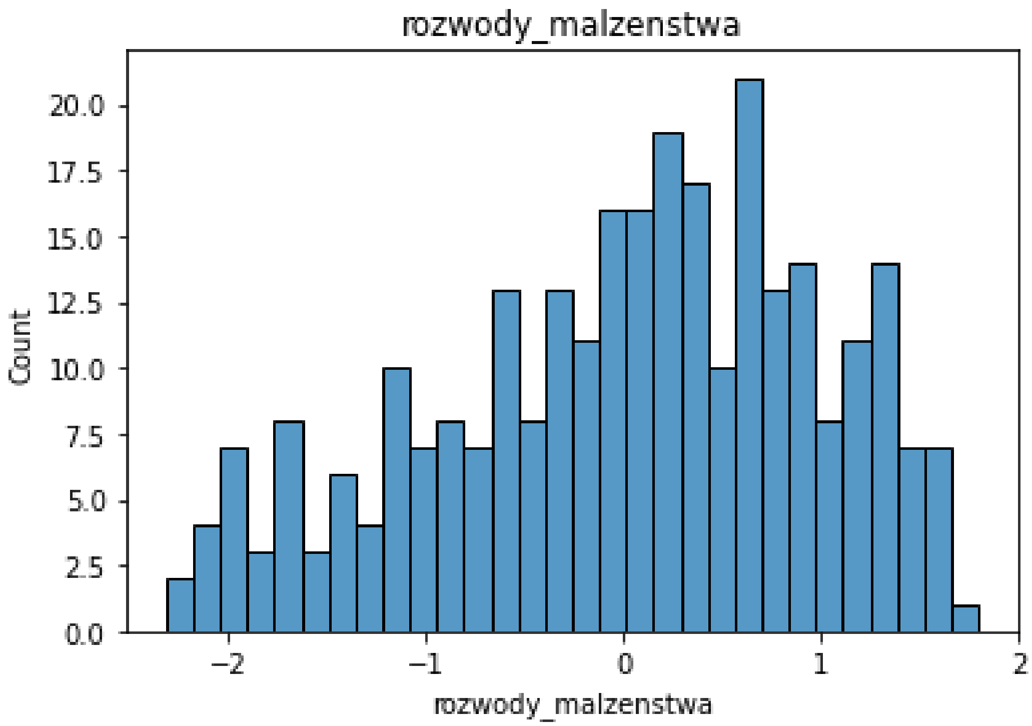

bezrobocie_zarejsestrowane_ludnosc
Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
bezrobocie_zarejsestrowane_ludnosc


malzenstwa_zawarte_ludnosc
Count
25
20
15
10
5
0
−2
−1
0
1
2
malzenstwa_zawarte_ludnosc

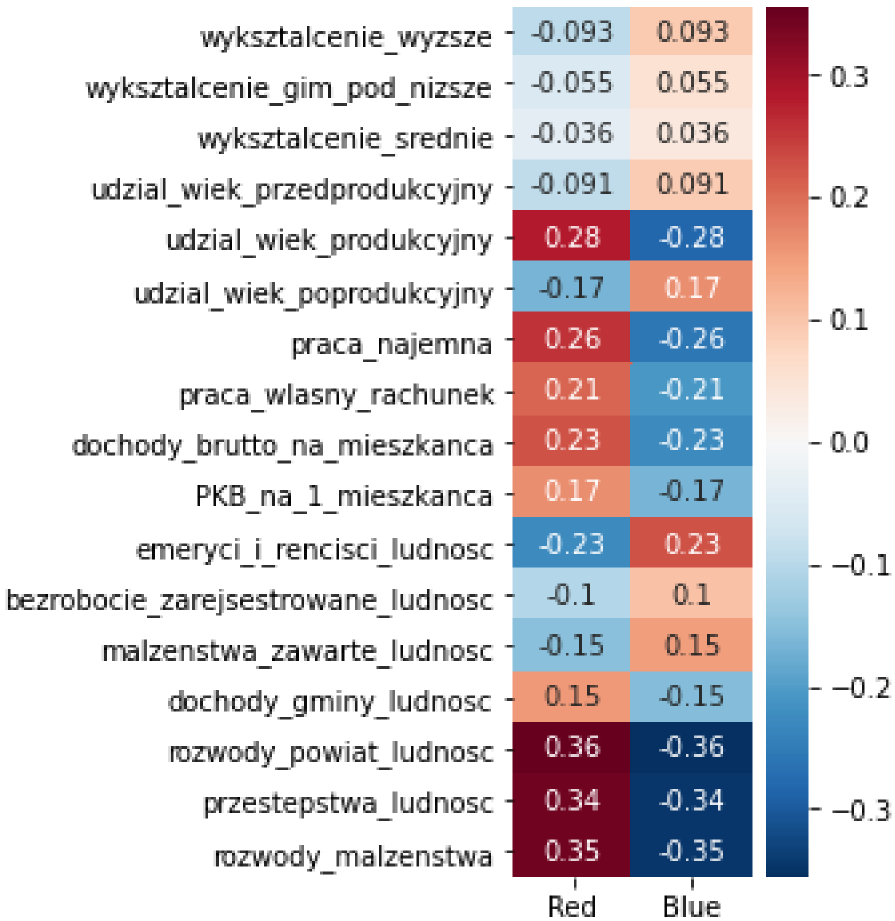

dochody_gminy_ludnosc
Count
35
30
25
20
15
10
5
0
−1
0
1
2
3
dochody_gminy_ludnosc


rozwody_powiat_ludnosc
Count
0
5
10
15
20
25
−2
−1
0
1
rozwody_powiat_ludnosc

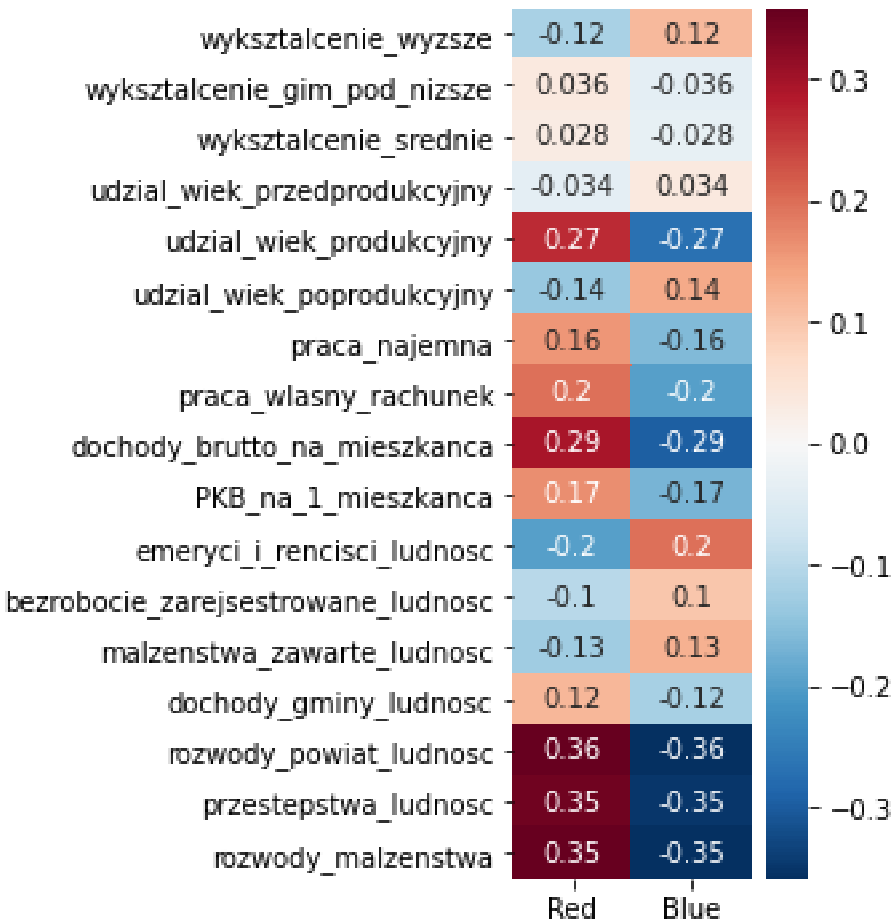

przestepstwa_ludnosc
Count
0
5
10
15
20
25
30
35
−3
−2
−1
0
1
2
przestepstwa_ludnosc



[106]:
```
fig, ax = plt.subplots(figsize=(2,6))
sn.heatmap(pool_stat_df_m.corr().iloc[:-2,-2:], annot=True, cmap='RdBu_r',␣
↪ax=ax)
plt.show()
```

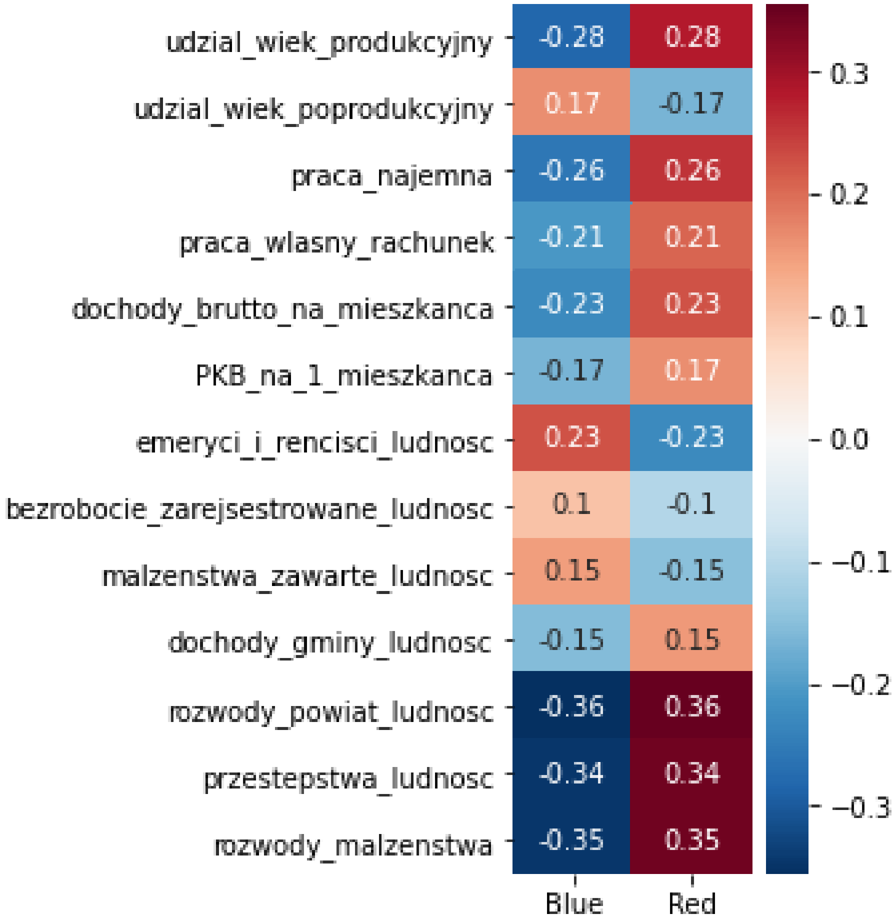


[107]:
```
fig, ax = plt.subplots(figsize=(2,6))
sn.heatmap(pool_stat_df_mm.corr().iloc[:-2,-2:], annot=True, cmap='RdBu_r',␣
↪ax=ax)
plt.show()
```

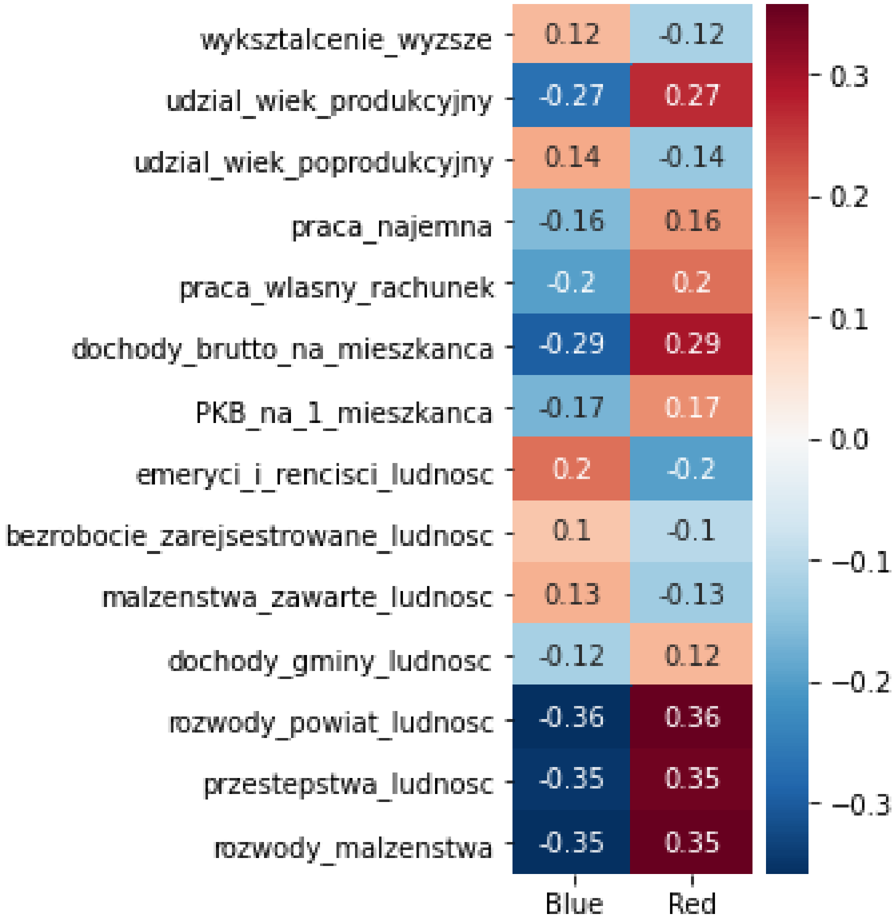


[109]:
```
potential_cols = pool_stat_df_mm.corr().iloc[:-2,-1].abs()
potential_cols = potential_cols[potential_cols>0.1]
high_cols = potential_cols.index.values.tolist()

potential_cols = pool_stat_df_m.corr().iloc[:-2,-1].abs()
potential_cols = potential_cols[potential_cols>0.1]
high_cols_m = potential_cols.index.values.tolist()
```

[110]:
```
fig, ax = plt.subplots(figsize=(2,6))
sn.heatmap(pool_stat_df_m.corr().loc[high_cols_m,['Blue','Red']], annot=True,␣
↪cmap='RdBu_r', ax=ax)
plt.show()
```



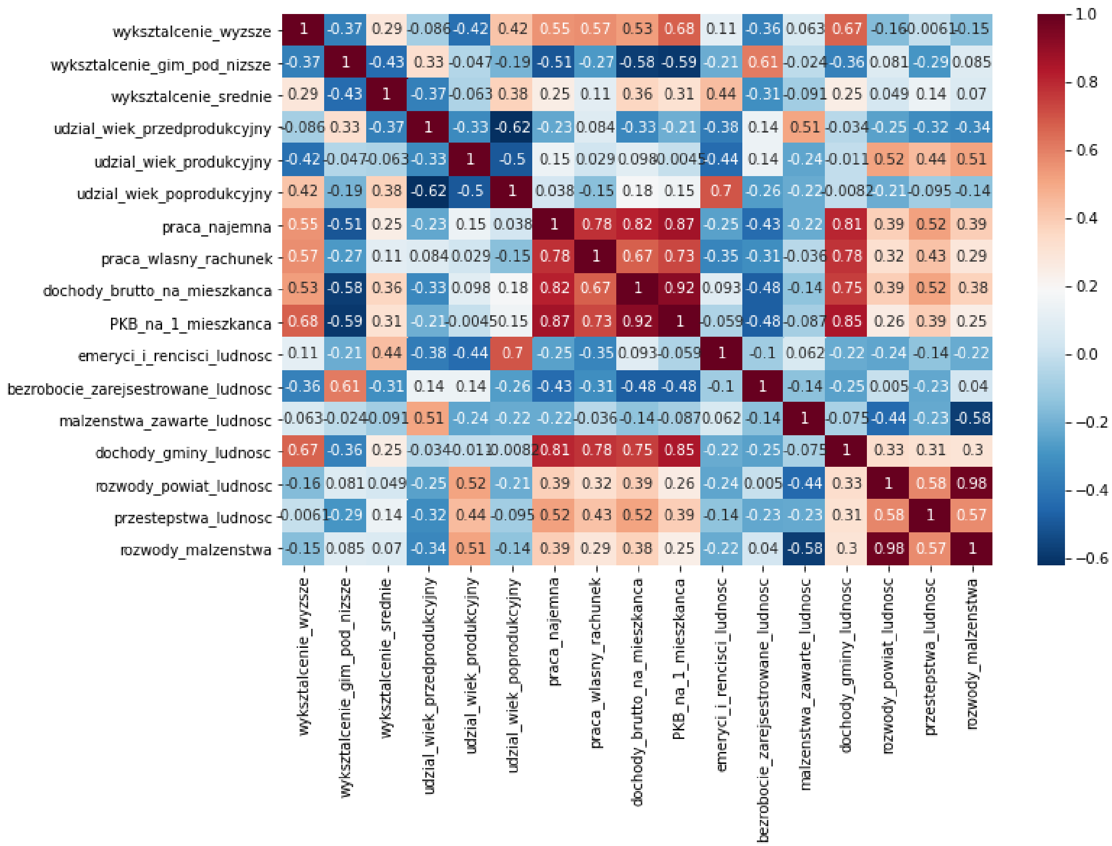
[111]:
```
fig, ax = plt.subplots(figsize=(2,6))
sn.heatmap(pool_stat_df_mm.corr().loc[high_cols,['Blue','Red']], annot=True,␣
↪cmap='RdBu_r', ax=ax)
plt.show()
```



[85]:

```
high_cols = [#'rozwody_powiat',
 'udzial_wiek_produkcyjny',
 'udzial_wiek_poprodukcyjny',
 'praca_najemna',
 'praca_wlasny_rachunek',
 'socjal',
 'dochody_brutto_na_mieszkanca',
 'PKB_na_1_mieszkanca',
 #'przestepstwa_ogolem',
 'przestepstwa_ludnosc',
 'emeryci_i_rencisci_ludnosc',
 'bezrobocie_zarejsestrowane_ludnosc',
 'malzenstwa_zawarte_ludnosc',
 'dochody_gminy_ludnosc',
 'rozwody_powiat_ludnosc',
 'rozwody_malzenstwa'
]
```

```
[112]: fig, ax = plt.subplots(figsize=(11,7))
sn.heatmap(pool_stat_df_mm.corr().
 ↪loc[high_corr_columns_final,high_corr_columns_final], annot=True,␣
 ↪cmap='RdBu_r')
plt.savefig('dane_pdf/dane_stat/corr_between.pdf',  bbox_inches='tight')
```



1.12.1 Lasso

```
[115]: columns_to_use = high_corr_columns_final
#columns_to_use = columns_to_use[~columns_to_use.
 ↪isin(['powierzchnnia_km2','ludnosc'])]
columns_to_use
```

```
[115]: Index(['wyksztalcenie_wyzsze', 'wyksztalcenie_gim_pod_nizsze',
       'wyksztalcenie_srednie', 'udzial_wiek_przedprodukcyjny',
       'udzial_wiek_produkcyjny', 'udzial_wiek_poprodukcyjny', 'praca_najemna',
       'praca_wlasny_rachunek', 'dochody_brutto_na_mieszkanca',
       'PKB_na_1_mieszkanca', 'emeryci_i_rencisci_ludnosc',
       'bezrobocie_zarejsestrowane_ludnosc', 'malzenstwa_zawarte_ludnosc',
       'dochody_gminy_ludnosc', 'rozwody_powiat_ludnosc',
```

```
       'przestepstwa_ludnosc', 'rozwody_malzenstwa'],
      dtype='object')
```

[116]:
```
X = pool_stat_df_m.loc[:,columns_to_use].values
Y = pool_stat_df_m.iloc[:,-1].values
```

[134]:
```
clf = Lasso(alpha=0.001)
clf.fit(X, Y)
```

[134]:
```
Lasso(alpha=0.001)
```

[135]:
```
df_lasso = pd.DataFrame({"weights abs":np.abs(clf.coef_),
                         "weights":clf.coef_,
                         "names":columns_to_use})
```

[136]:
```
df_lasso.sort_values("weights abs", ascending = False)[:10]
```

[136]:

	weights abs	weights	names
8	0.032323	-0.032323	dochody_brutto_na_mieszkanca
10	0.030519	0.030519	emeryci_i_rencisci_ludnosc
0	0.028746	0.028746	wyksztalcenie_wyzsze
14	0.023856	-0.023856	rozwody_powiat_ludnosc
15	0.021899	-0.021899	przestepstwa_ludnosc
11	0.014485	0.014485	bezrobocie_zarejsestrowane_ludnosc
3	0.007307	0.007307	udzial_wiek_przedprodukcyjny
2	0.006946	0.006946	wyksztalcenie_srednie
9	0.003961	0.003961	PKB_na_1_mieszkanca
6	0.003332	-0.003332	praca_najemna

1.12.2 Random Forest

[137]:
```
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X, Y)

df_rand = pd.DataFrame({"importance":regr.feature_importances_, "names":
 ↪columns_to_use})

df_rand.sort_values("importance", ascending = False)[:10]
```

[137]:

	importance	names
6	0.305737	praca_najemna
4	0.116922	udzial_wiek_produkcyjny
16	0.107075	rozwody_malzenstwa
15	0.068461	przestepstwa_ludnosc
8	0.059886	dochody_brutto_na_mieszkanca
14	0.059379	rozwody_powiat_ludnosc
13	0.057082	dochody_gminy_ludnosc

```
9     0.051681           PKB_na_1_mieszkanca
7     0.044387         praca_wlasny_rachunek
0     0.028586          wyksztalcenie_wyzsze
```

[154]:
```
df_rf_results = None

for i in range(100):
    regr = RandomForestRegressor(max_depth=2)
    regr.fit(X, Y)

    df_rand = pd.DataFrame({"names":columns_to_use, "importance":regr.
 ↪feature_importances_ })

    if df_rf_results is None:
        df_rf_results = df_rand
    else:
        df_rf_results = df_rf_results.merge(df_rand, on='names', how='inner',␣
 ↪suffixes=(str(i), str(i+1)))

    #df_rand.sort_values("importance", ascending = False)[:10]
```

[155]:
```
df_rf_results['mean'] = df_rf_results.iloc[:,1:].mean(1)
df_rf_results['std'] = df_rf_results.iloc[:,1:-1].std(1)
df_rf_results[['names','mean','std']].sort_values("mean", ascending = False)[:
 ↪10]
```

[155]:

	names	mean	std
6	praca_najemna	0.360071	0.030328
4	udzial_wiek_produkcyjny	0.105540	0.011912
16	rozwody_malzenstwa	0.087683	0.019860
14	rozwody_powiat_ludnosc	0.071629	0.017418
15	przestepstwa_ludnosc	0.071122	0.016830
13	dochody_gminy_ludnosc	0.059916	0.010032
8	dochody_brutto_na_mieszkanca	0.044917	0.016700
9	PKB_na_1_mieszkanca	0.041225	0.013288
0	wyksztalcenie_wyzsze	0.033006	0.008068
5	udzial_wiek_poprodukcyjny	0.028303	0.007095

1.12.3 PCA

[211]:
```
pca = PCA(n_components=10)
x_new = pca.fit_transform(X)
```

[212]:
```
print(pca.explained_variance_ratio_)
print(np.sum(pca.explained_variance_ratio_[:3]))
```

```
[0.34584517 0.18489667 0.16964044 0.08466176 0.04469152 0.0331832
```

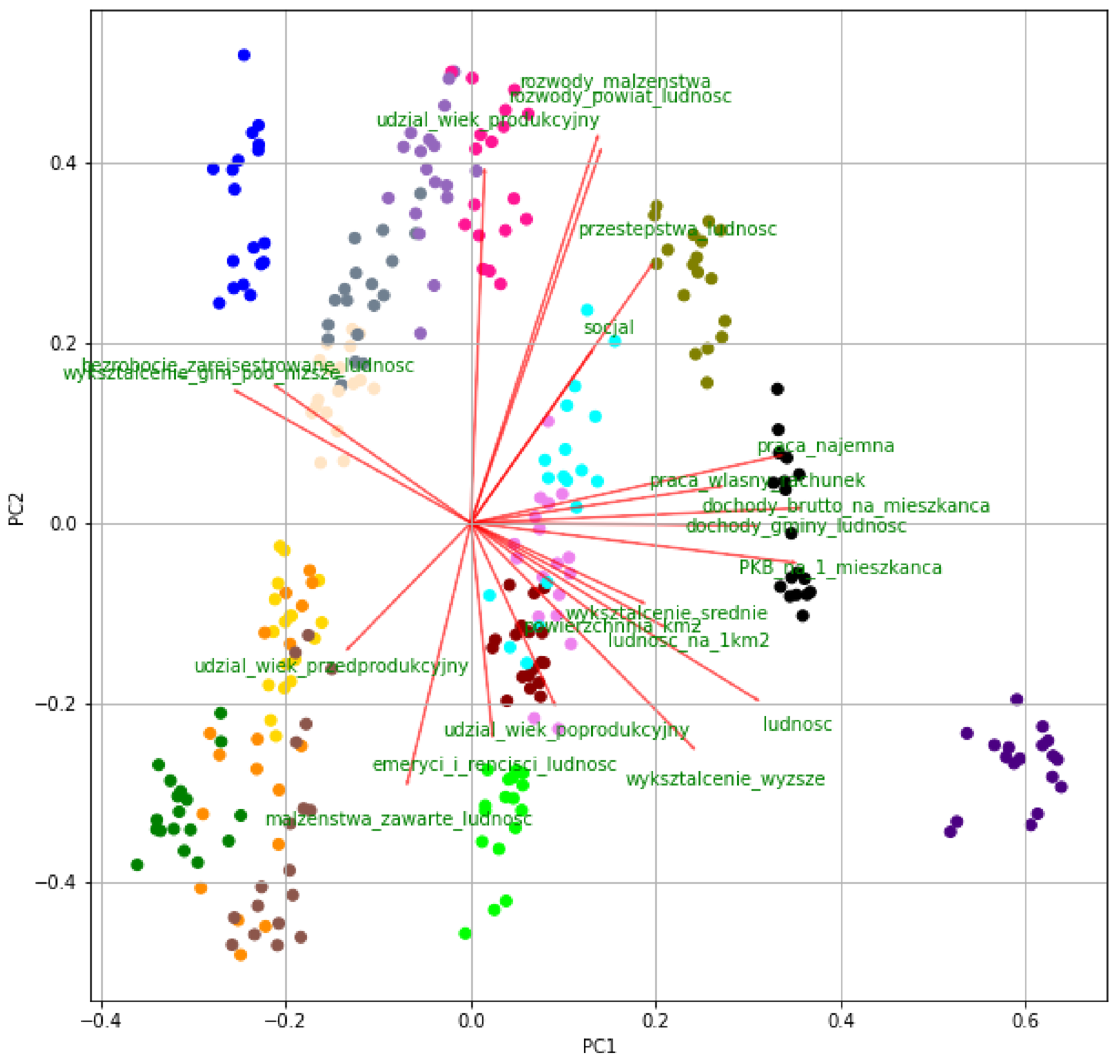
```
 0.02812613 0.02617308 0.02167534 0.01641299]
0.7003822734084834
```

[213]:
```
for x in range(data_reshaped.shape[1]):
    colors.loc[colors.wojewodztwo==stat_list_org[0].index[x],:].values[0][1]
```

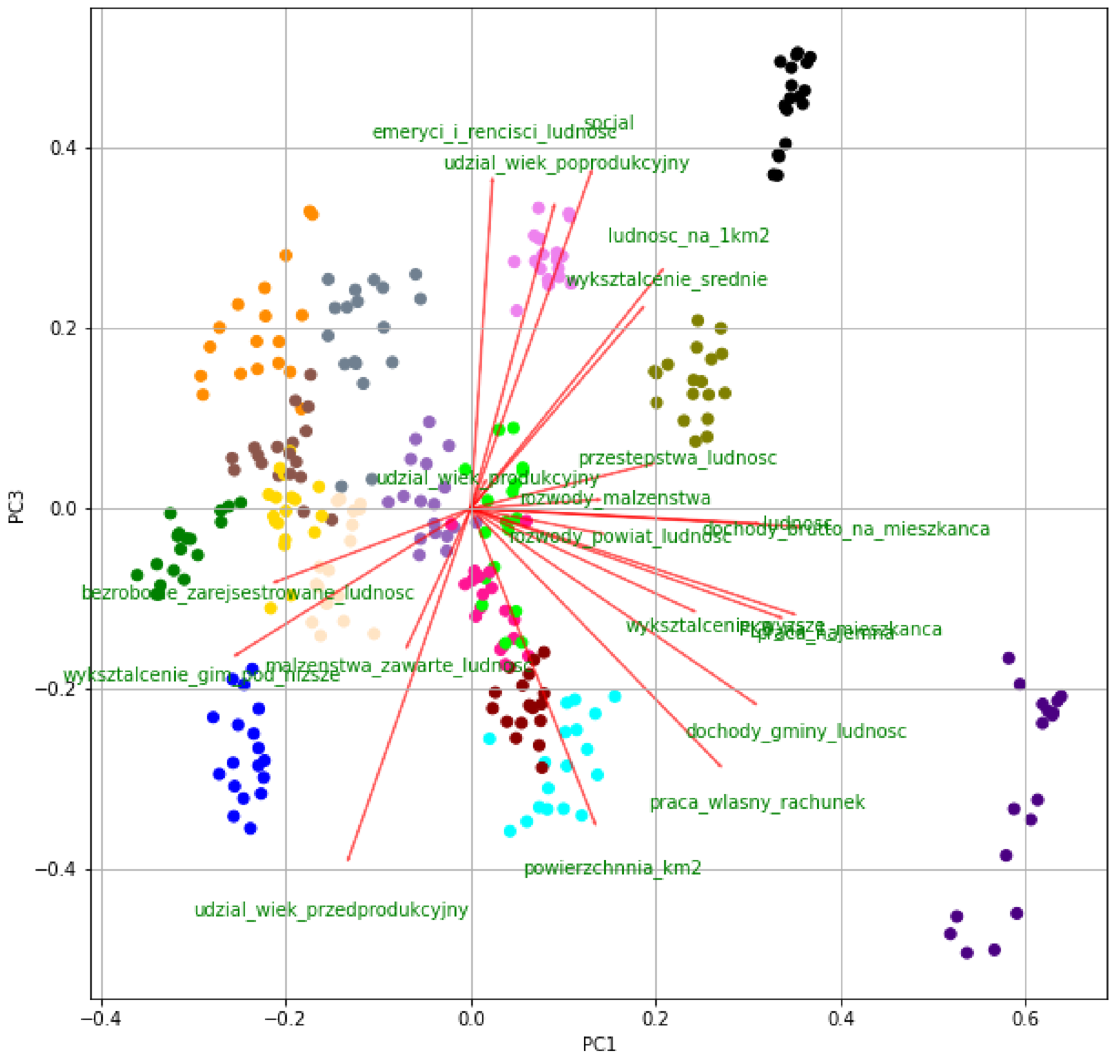
[214]:
```
color_list = []
for w in pool_stat_df_m['index']:
    if w == 'MAŁOPOLSKIE': color_list.append('C0')
    elif w == 'ŚLĄSKIE': color_list.append('C1')
    elif w == 'LUBUSKIE': color_list.append('C2')
    elif w == 'WIELKOPOLSKIE': color_list.append('C3')
    elif w == 'ZACHODNIOPOMORSKIE': color_list.append('C4')
    elif w == 'DOLNOŚLĄSKIE': color_list.append('C5')
    elif w == 'OPOLSKIE': color_list.append('C6')
    elif w == 'KUJAWSKO-POMORSKIE': color_list.append('C7')
    elif w == 'POMORSKIE': color_list.append('C8')
    elif w == 'WARMIŃSKO-MAZURSKIE': color_list.append('C9')
    elif w == 'ŁÓDZKIE': color_list.append('C10')
    elif w == 'ŚWIĘTOKRZYSKIE': color_list.append('C11')
    elif w == 'LUBELSKIE': color_list.append('C12')
    elif w == 'PODKARPACKIE': color_list.append('C13')
    elif w == 'PODLASKIE': color_list.append('C14')
    elif w == 'MAZOWIECKIE': color_list.append('C15')

color_list = []
for w in pool_stat_df_m['index']:
    color_list.append(colors.loc[colors.wojewodztwo==w,:].values[0][1])
```

[215]:
```
cols_plotting = pool_stat_df_m.columns[1:-2]
plt.figure(figsize=(10,10))

def myplot(score,coeff,width=0.001,c_arr=['r'],scale=1.15):
    xs = score[:,0]
    ys = score[:,1]
    n = coeff.shape[0]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
    plt.scatter(xs * scalex,ys * scaley, c = color_list)

    c_txt = ['g']*n
    if len(c_arr) == 1:
        c_arr = c_arr*n
    else:
        c_txt = c_arr

    for i in range(n):
```

```
        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = c_arr[i],alpha = 0.
↪5,width=width)

        plt.text(coeff[i,0]* scale, coeff[i,1] * scale, cols_plotting[i], color␣
↪= c_txt[i], ha = 'center', va = 'center')

#plt.xlim(-1,1)
#plt.ylim(-1,1)
plt.xlabel("PC{}".format(1))
plt.ylabel("PC{}".format(2))
plt.grid()

#Call the function. Use only the 2 PCs.
myplot(x_new[:,0:2],np.transpose(pca.components_[0:2, :]))
plt.savefig('dane_pdf/dane_stat/PCA12.pdf',  bbox_inches='tight')
```

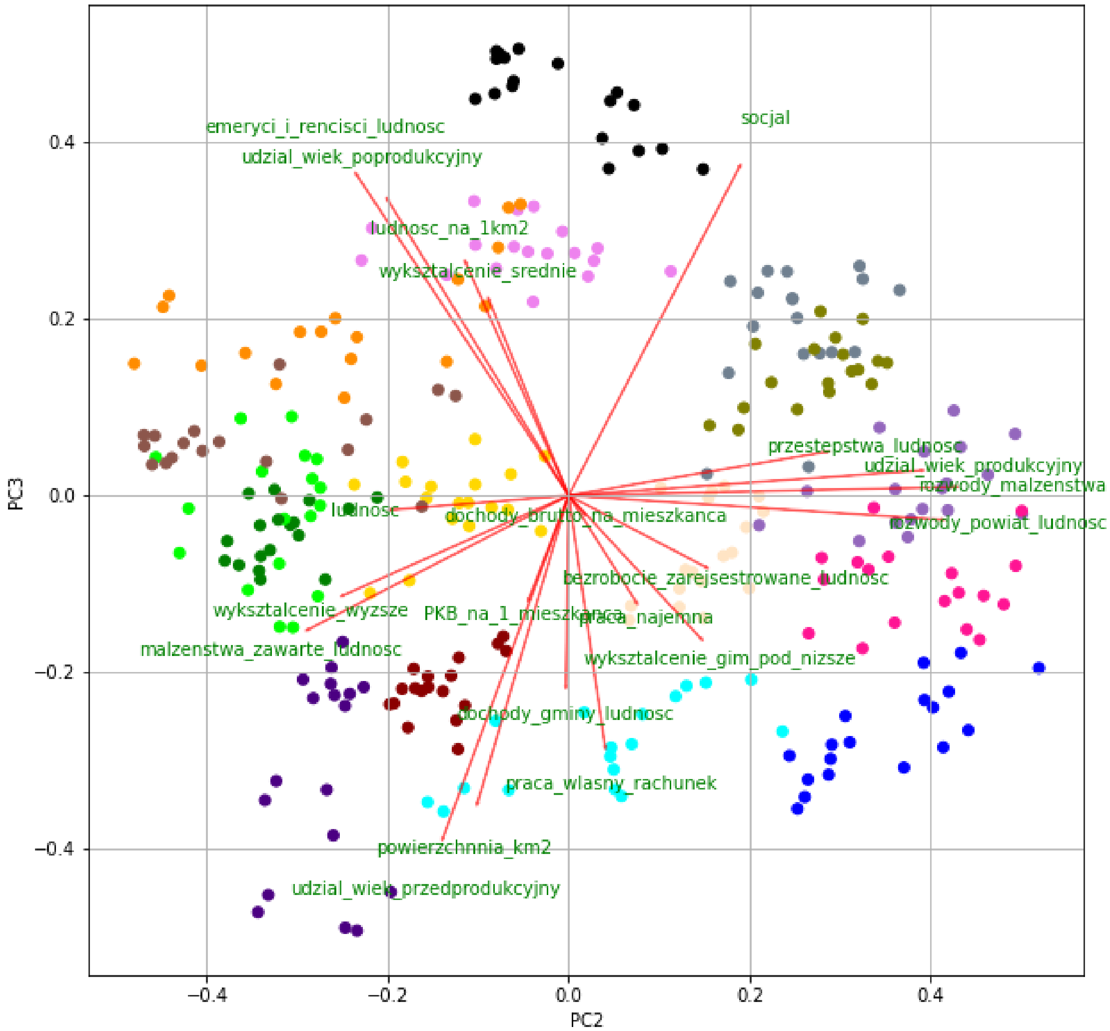


[216]:
```
plt.figure(figsize=(10,10))
plt.xlabel("PC{}".format(1))
plt.ylabel("PC{}".format(3))
plt.grid()

#Call the function. Use only the 2 PCs.
myplot(x_new[:,[0,2]],np.transpose(pca.components_[[0,2], :]))
plt.savefig('dane_pdf/dane_stat/PCA13.pdf',  bbox_inches='tight')
```



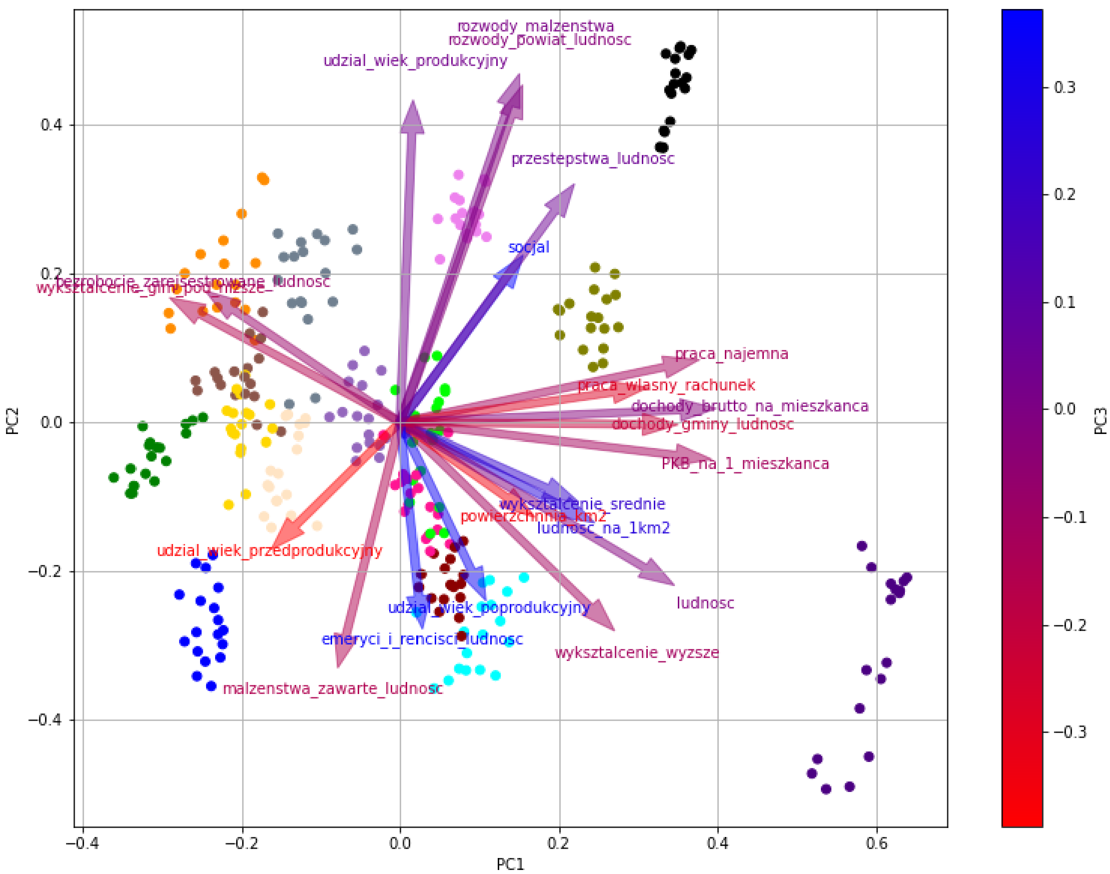
[217]:
```
plt.figure(figsize=(10,10))
plt.xlabel("PC{}".format(2))
plt.ylabel("PC{}".format(3))
plt.grid()
```

```
#Call the function. Use only the 2 PCs.
myplot(x_new[:,[1,2]],np.transpose(pca.components_[[1,2], :]))
plt.savefig('dane_pdf/dane_stat/PCA23.pdf',  bbox_inches='tight')
```



[218]:
```
import matplotlib.colors as mcol
import matplotlib.cm as cm

z_axis = pca.components_[2, :]

cm1 = mcol.LinearSegmentedColormap.from_list("MyCmapName",["r","b"])
cnorm = mcol.Normalize(vmin=np.min(z_axis),vmax=np.max(z_axis))

cpick = cm.ScalarMappable(norm=cnorm,cmap=cm1)
cpick.set_array([])
```

```
plt.figure(figsize=(13,10))
plt.xlabel("PC{}".format(1))
plt.ylabel("PC{}".format(2))
plt.grid()

#Call the function. Use only the 2 PCs.
myplot(x_new[:,[0,2]],
       np.transpose(pca.components_[[0,1], :]),
       width=0.01,
       c_arr=cpick.to_rgba(pca.components_[2, :]),
       scale = 1.25)
plt.colorbar(cpick,label="PC3")
plt.savefig('dane_pdf/dane_stat/PCA12_colors_3nd.pdf',  bbox_inches='tight')
```



[202]: `pca.components_.shape`

[202]: `(10, 19)`

[250]:
```
%matplotlib notebook
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d

class Arrow3D(FancyArrowPatch):
    def __init__(self, xs, ys, zs, *args, **kwargs):
        FancyArrowPatch.__init__(self, (0,0), (0,0), *args, **kwargs)
        self._verts3d = xs, ys, zs

    def draw(self, renderer):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
        self.set_positions((xs[0],ys[0]),(xs[1],ys[1]))
        FancyArrowPatch.draw(self, renderer)

Xax = x_new[:,0]
Yax = x_new[:,1]
Zax = x_new[:,2]

fig = plt.figure(figsize=(9,9))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(Xax, Yax, Zax)

#coeff = pca.components_[[0,1,2], :]
coeff = np.transpose(pca.components_[0:3, :])
n = coeff.shape[0]

for i in range(n):
    #ax.plot([mean_x,v[0]], [mean_y,v[1]], [mean_z,v[2]], color='red', alpha=0.
 ↪8, lw=3)
    #I will replace this line with:
    a = Arrow3D([0, coeff[i,0]*15],
                [0, coeff[i,1]*15],
                [0, coeff[i,2]*15], mutation_scale=20,
                lw=1, arrowstyle="-|>", color="r")
    ax.add_artist(a)

for i in range(n):
    ax.text(coeff[i,0]* 15,
            coeff[i,1] * 15,
            coeff[i,2] * 15, cols_plotting[i], (1,1,0), fontsize=7)


# for loop ends
ax.set_xlabel("PC1", fontsize=14)
```

```
ax.set_ylabel("PC2", fontsize=14)
ax.set_zlabel("PC3", fontsize=14)

plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[194]:
```
high_cols = [
 'udzial_wiek_produkcyjny',
 #'udzial_wiek_poprodukcyjny',
 'praca_najemna',
 #'praca_wlasny_rachunek',
 'socjal',
 #'dochody_brutto_na_mieszkanca',
 #'PKB_na_1_mieszkanca',
 'przestepstwa_ludnosc',
 'emeryci_i_rencisci_ludnosc',
 'bezrobocie_zarejsestrowane_ludnosc',
 'malzenstwa_zawarte_ludnosc'
 #'dochody_gminy_ludnosc',
 #'rozwody_powiat_ludnosc'
 #'rozwody_malzenstwa'
]
```

[ ]:
```
fig, ax = plt.subplots(figsize=(7,5))
sn.heatmap(pool_stat_df_mm.corr().loc[high_cols,high_cols], annot=True,␣
 ↪cmap='RdBu_r')
plt.savefig('dane_pdf/dane_stat/corr_between_choosen.pdf',  bbox_inches='tight')
```

[ ]:
```
stat_list = [st.loc[:,high_cols] for st in stat_list]
```

2 Model

2.1 Prepare input (X) (parameters which we will multiply)

Input - wagi, na outpucie mnożenie wag razy wartości i na tej podstawie ocena. - poprzednie wybory (par in reg/ reg in par) - wpływ sąsiadów (avg over nighbours/ weighted avg) - pole zewnętrzne

[ ]:
```
X = []
# iterate over years [from 2002 - 2019]
for y in range(len(stat_list)):
    # iterate over districts
    tmp_x = []
    st_data = stat_list[y]
    for d in range(vote_list[0].shape[0]):
```

```
        # 1. last election: Blue, Red
        #    Blue/All
        # 2. neighbours
        # 3. one (1)
        lo = pool_d[y].iloc[d,:]
        neigh = neighbours[lo.name.lower()]
        avg_n = [pool_d[y].loc[n.upper()][0]/pool_d[y].loc[n.upper()].sum() for␣
↪n in neigh]
        avg_n = sum(avg_n)/len(neigh)

        tmp_y = st_data.iloc[d,:].values.tolist()
        tmp_y.append(lo[0]/lo.sum())
        tmp_y.append(avg_n)

        tmp_x.append(tmp_y)
    X.append(tmp_x)
```

[ ]:
```
X = np.array(X)
X.shape
```

2.2 Prepare Y

[ ]:
```
Y = []
for y in range(1,pool_data_middle.shape[0]):
    # iterate over districts
    tmp_y = []
    for d in range(vote_list[0].shape[0]):
        # 1. last election: Blue, Red, Gray
        #    Blue/All
        # 2. neighbours
        # 3. one (1)
        lo = pool_d[y].iloc[d,:]
        tmp_y.append([lo[0]/lo.sum()])
    Y.append(tmp_y)
```

[ ]:
```
Y = np.array(Y)
Y.shape
```

2.3 Parameters to be estimated

- waga poprzednie wybory
- waga wpływu sąsiadów (avg over nighbours/ weighted avg)
- pole zewnętrzne

2.4 Process

- input,

- warstwy,
- output (parametry lub wagi),
- output*parameters (the real output - wynik wyborów)

OR - input (parameters), - simple network to get the weight = parameters to multiply, - output (next election)

2.5 Training phase (looking for parameters)

Functions for models

[ ]:
```
X.shape
```

2.5.1 Models with percentage of Blue support per district

[ ]:
```
def model_percent(a,x):
    '''
    INPUT:
    a - vector of weights 16x14
    x - vector of input data 18x16x14
    OUTPUT:
    y - predicted value in (0,1)
    '''
    #d0 = x.shape[0] if (len(x.shape) == 3) else 1
    #d1 = x.shape[-1]

    #a = np.repeat(a, d0, 0)
    #x = x.reshape(-1, d1)

    #return 1 / (1+np.exp(-np.sum(x.dot(a.T))))
    y = 1 / (1+np.exp(-np.sum(x*a, 1, keepdims=True)))
    return y

def grad_percent(a,x,y):
    '''
    INPUT:
    a - vector of weights 16x14
    x - vector of input data 18x16x14
    '''
    #return a * np.exp(-x.T.dot(a)) / (1+np.exp(-x.T.dot(a)))**2
    #return a*np.exp(-np.sum(x*a,1,keepdims=True)) / (1+np.exp(-np.
↪sum(x*a,1,keepdims=True)))**2
    d0 = x.shape[0] if (len(x.shape) == 3) else 1
    d1 = x.shape[-1]

    a = np.repeat(a, d0, 0)
    x = x.reshape(-1, d1)
    y = y.reshape(-1, 1)
```

```
    y1 = -(2 *
          ( y - 1/(1+np.exp(-np.sum(x.dot(a.T),1,keepdims=True))) ) *
          1/(1+np.exp(-np.sum(x.dot(a.T),1,keepdims=True)))**2 *
          np.exp(-np.sum(x.dot(a.T),1,keepdims=True)) *
          x)
    return y1
```

[ ]:
```
"""
def grad_percent(a,x,y):
    '''
    INPUT:
    a - vector of weights 16x14
    x - vector of input data 18x16x14
    '''
    #return a * np.exp(-x.T.dot(a)) / (1+np.exp(-x.T.dot(a)))**2
    #return a*np.exp(-np.sum(x*a,1,keepdims=True)) / (1+np.exp(-np.
 ↪sum(x*a,1,keepdims=True)))**2
    d0 = x.shape[0] if (len(x.shape) == 3) else 1
    d1 = x.shape[-1]

    a = np.repeat(a, d0, 0)
    x = x.reshape(-1, d1)
    y = y.reshape(-1, 1)
    m = np.sign(y - model_percent(a,x))
    y1 =m*(2 *
          ( y - 1/(1+np.exp(-np.sum(x.dot(a.T),1,keepdims=True))) ) *
          1/(1+np.exp(-np.sum(x.dot(a.T),1,keepdims=True)))**2 *
          np.exp(-np.sum(x.dot(a.T),1,keepdims=True)) *
          x)
    return y1
"""
```

2.5.2 Setup for testing model

[ ]:
```
neigh_ndx = []
for d in range(X.shape[1]):
    # 1. last election: Blue, Red, Gray
    #    Blue/All
    # 2. neighbours
    # 3. one (1)
    lo = par_in_reg_list[0].iloc[d,:]
    neigh = neighbours[lo.name.lower()]
    indexs = par_in_reg_list[0].index.values
    neigh_ndx.append(np.searchsorted(indexs, np.char.upper(neigh)))
```

[ ]:
```
st_data.shape
```

[ ]:
```python
def prepare_input(y, st_data):
    tmp_x = np.zeros((y.shape[0],st_data.shape[-1]+2))
    for d in range(y.shape[0]):
        neigh = neigh_ndx[d]
        avg_n = [y[n,0]/np.sum(y[neigh,0]) for n in neigh]
        avg_n = sum(avg_n)/len(neigh)

        tmp_y = st_data.iloc[d,:].values.tolist()
        tmp_y.append(y[d,0])
        tmp_y.append(avg_n)

        tmp_x[d] = np.array(tmp_y)
    return(tmp_x)
```

[ ]:
```python
def model(a,x,Y,st_list):
    y = Y[0]
    loss = []
    out = np.zeros(Y.shape)
    out[0] = y
    for year in range(1,X.shape[0]):
        st_data = stat_list[year-1]
        xi = prepare_input(y,st_list[year-1])
        y = model_percent(a,xi)
        loss.append(np.sum((y - Y[year])**2))
        #print(y.shape,'loss:', np.sum((y - Y[year])**2))
        out[year] = y
    return loss, out
```

[ ]:
```python
loss_p = np.inf
loss_v = np.inf

a_avg = np.random.rand(X.shape[1],X.shape[2])
a_all = a_avg
#av = np.random.rand(X.shape[1],X.shape[2])

step = 0.01
beta = 0.1
a_avg.shape
```

2.6 Use output -> next input (grad each step)

[ ]:
```python
a_step = np.random.rand(X.shape[1],X.shape[2]) - 0.5
step = 0.1

loss_arr = np.zeros((10**3,))

loss_prev = np.inf
```

```python
for epoch in range(10**3):
    print(epoch,end='\r')
    loss_p = 0
    y = Y[0]
    for i in range(X.shape[0]):
        xi = prepare_input(y,stat_list[i])
        y = model_percent(a_step,xi)

        grad = grad_percent(a_step,xi,Y[i])#.reshape(18,16,3)
        #grad = np.sum(grad, axis=0)
        a_step = a_step - step*grad

        loss_p += np.sum((model_percent(a_step,xi) - Y[i].reshape(-1,1))**2)

    loss_arr[epoch] = loss_p
    #if loss_p < loss_prev:
    #    step = step*0.95
    #else:
    #    step = step*1.05

    #loss_prev = loss_p

    if epoch%100==0:
        l, o = model(a_step,X,Y,stat_list)
        print('loss:', np.mean(l))
        plt.plot(np.average(o,1, voter_w[1]),'b:', linewidth=1)

l, o = model(a_step,X,Y,stat_list)
plt.plot(np.average(o,1, voter_w[1]),'bo', linewidth=1)

plt.plot(pool_data_middle['Blue'].values[1:],'b')
plt.ylim(0,1)
plt.show()
```

[ ]:
```python
plt.plot(loss_arr)
plt.show()
```

[ ]:
```python
plt.plot(o.reshape(-1,16),'o-')
plt.show()
```

2.6.1 Compare of parameters per each district

```
[ ]: plt.plot(np.abs(a_step.T), 'o-')
     plt.xticks(range(a_step.shape[1]), ['prev','neigh']+high_cols,␣
      ↪size='small',rotation=90)
     plt.show()
```

```
[ ]: plt.plot(np.abs(a_step).mean(0), 'o-')
     plt.errorbar(np.arange(9), np.abs(a_step).mean(0), e, linestyle='None',␣
      ↪marker='', ecolor='tab:blue')
     e = np.abs(a_step).std(0)
     plt.xticks(range(a_step.shape[1]), ['prev','neigh']+high_cols,␣
      ↪size='small',rotation=90)
     plt.show()
```

```
[ ]: a_step = np.random.rand(X.shape[1],X.shape[2]) - 0.5
     step = 1

     for epoch in range(10**3):
         print(epoch,end='\r')
         loss_p = 0
         y = Y[0]
         for i in range(X.shape[0]):
             xi = prepare_input(y,stat_list[i])
             y = model_percent(a_step,xi)

             grad = grad_percent(a_step,xi,Y[i])#.reshape(18,16,3)
             #grad = np.sum(grad, axis=0)
             a_step = a_step - step*grad

             loss_p += np.sum((model_percent(a_step,xi) - Y[i].reshape(-1,1))**2)

         if epoch%100==0:
             l, o = model(a_step,X,Y,stat_list)
             plt.plot(np.average(o,1, voter_w[1]),'b:', linewidth=1)

     l, o = model(a_step,X,Y,stat_list)
     plt.plot(np.average(o,1, voter_w[1]),'bo', linewidth=1)

     plt.plot(pool_data_middle['Blue'].values[1:],'b')
     plt.ylim(0,1)
     plt.show()
```

```
[ ]: a_step = np.random.rand(X.shape[1],X.shape[2]) - 0.5
     step = 10

     for epoch in range(10**3):
         print(epoch,end='\r')
         loss_p = 0
```

```
    y = Y[0]
    for i in range(X.shape[0]):
        xi = prepare_input(y,stat_list[i])
        y = model_percent(a_step,xi)

        grad = grad_percent(a_step,xi,Y[i])#.reshape(18,16,3)
        #grad = np.sum(grad, axis=0)
        a_step = a_step - step*grad

        loss_p += np.sum((model_percent(a_step,xi) - Y[i].reshape(-1,1))**2)

    if epoch%100==0:
        l, o = model(a_step,X,Y,stat_list)
        plt.plot(np.average(o,1, voter_w[1]),'b:', linewidth=1)

l, o = model(a_step,X,Y,stat_list)
plt.plot(np.average(o,1, voter_w[1]),'bo', linewidth=1)

plt.plot(pool_data_middle['Blue'].values[1:],'b')
plt.ylim(0,1)
plt.show()
```

2.7 All at once + evolution strategies (TO DO)

3 https://towardsdatascience.com/introduction-to-evolution-strategy-1b78b9d48385

for epoch in range(10**3): grad = grad_percent(a_avg,X).reshape(18,16,3) grad = np.sum(grad, axis=0) a_avg = a_avg - step*grad

```
#if epoch%50==0:
#    if np.sum((model_percent(ap,X) - Y.reshape(-1,1))**2) < loss_p: step *= (1+beta)
#    else: step /= (1-beta)

loss_p = np.sum((model_percent(a_avg,X) - Y.reshape(-1,1))**2)

if epoch%100==0:
    print('loss sum:',loss_p)
```

a_avg.shape

[ ]:
```
l_avg, o = model(a_avg,X,Y)
l_all, o = model(a_all,X,Y)
plt.plot(l_avg,'bo')
plt.plot(l_all,'ro')
plt.plot()
```

3.1 Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)

https://stackabuse.com/time-series-prediction-using-lstm-with-pytorch-in-python/
http://proceedings.mlr.press/v57/zhao16.pdf https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816 https://developer.apple.com/documentation/coreml/cor
https://www.youtube.com/watch?v=WCUNPb-5EYI

[ ]:
```
(a_avg-a_all).max()
```

[ ]:
```

```