In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.cluster import KMeans, DBSCAN
import seaborn as sns
import matplotlib.pyplot as plt
from plotly.figure_factory._county_choropleth import create_choropleth
```

In [2]:

```
# load data
data = pd.read_csv("merged_train.csv")
data.head()
```

Out[2]:

	State	County	FIPS	Total Population	Percent White, not Hispanic or Latino	Percent Black, not Hispanic or Latino	Percent Hispanic or Latino	Percent Foreign Born	Percen Femal
0	AZ	apache	4001	72346	18.571863	0.486551	5.947806	1.719515	50.59851
1	AZ	cochise	4003	128177	56.299492	3.714395	34.403208	11.458374	49.06964
2	AZ	coconino	4005	138064	54.619597	1.342855	13.711033	4.825298	50.58161
3	AZ	gila	4007	53179	63.222325	0.552850	18.548675	4.249798	50.29617
4	AZ	graham	4009	37529	51.461536	1.811932	32.097844	4.385942	46.31351

In [3]:

```
# task 1
# Partition dataset into training, validation sets using holdout method 75/25 split

# Xvariables has all the names except for State, County, FIPS, Party, Democratic, Rep
ublican
# Yvariables has Party, Democratic, Republican
Yvariables = ['Party','Democratic','Republican']
Xvariables = ['Total Population', 'Percent White, not Hispanic or Latino', 'Percent B
lack, not Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Pe
rcent Female','Percent Age 29 and Under','Percent Age 65 and Older','Median Household
Income','Percent Unemployed','Percent Less than High School Degree','Percent Less tha
n Bachelor\'s Degree','Percent Rural']
X_train, X_val, Y_train, Y_val = train_test_split(data[Xvariables], data[Yvariables],
test_size = 0.25, random_state = 1)
X_train.head()
```

Out[3]:

	Total Population	Percent White, not Hispanic or Latino	Percent Black, not Hispanic or Latino	Percent Hispanic or Latino	Percent Foreign Born	Percent Female	Percent Age 29 and Under	Perc Age and Ol
943	15919	91.940449	5.207614	1.432251	1.300333	51.077329	31.660280	23.902
853	76	72.368421	0.000000	15.789474	11.842105	47.368421	11.842105	25.000
578	60878	95.579684	0.877164	1.404448	1.342028	50.962581	40.464536	16.324
1035	54562	95.484037	1.268282	1.414904	1.611011	50.500348	32.005792	20.301
822	54217	85.327480	0.673221	11.634727	3.225925	50.709187	32.939853	23.861

In [4]:

```
# task 2
# standardize the training and validation sets by using X_train as the scalar and app
lying to the training and validation sets
scaler = StandardScaler()
scaler.fit(X_train) # find the mean and standard diviation for the columns in X_train
x_train_scaled = scaler.transform(X_train) # scales X_train using the results from fi
t method
x_val_scaled = scaler.transform(X_val)  # scales X_val using the results from fit met
hod
x_train_scaled
```

Out[4]:

```
array([[-0.345068  ,  0.65924303, -0.04363108, ...,  0.33240678,
         0.56141981, -0.09521071],
       [-0.39488816, -0.34555147, -0.59845393, ...,  1.87960605,
         2.01297272,  1.36549077],
       [-0.20368919,  0.84607517, -0.50500024, ..., -0.47046883,
        -0.04999222,  0.00623229],
       ...,
       [-0.30811876,  0.79654542, -0.54416126, ..., -1.11037852,
        -1.07924513,  0.41391348],
       [ 0.2187051 , -0.47707575,  1.33755335, ..., -0.29816431,
         0.07718104, -0.74801204],
       [-0.38811151,  0.67491051, -0.59845393, ..., -1.54891473,
        -0.30144791,  1.36549077]])
```

In [5]:

```
# task 3
# predict democratic votes using 2 predictors
model = linear_model.LinearRegression()
fitted_model = model.fit(X=X_train[['Percent Foreign Born', 'Median Household Income'
]],y=Y_train['Democratic'])
predicted = fitted_model.predict(X_train[['Percent Foreign Born','Median Household In
come']])
print (predicted)

# Evaluate linear regression model using 2 predictors on democratic votes
X_train_dummy = pd.get_dummies(X_train, drop_first = True)
X_val_dummy = pd.get_dummies(X_val, drop_first = True)
model = linear_model.LinearRegression().fit(X = X_train_dummy[['Percent Foreign Born'
, 'Median Household Income']], y = Y_train['Democratic'])
score_val = model.score(X = X_val_dummy[['Percent Foreign Born', 'Median Household In
come']], y = Y_val['Democratic']) # R squared (validation)
print(score_val)

# Evaluate LASSO regression model on democratic votes
model = linear_model.Lasso(alpha = 1).fit(X = X_train_dummy, y = Y_train['Democratic'
])
score_val = model.score(X = X_val_dummy, y = Y_val['Democratic']) # R squared (valida
tion)
print(score_val)

# predict republican votes using 2 predictors
model = linear_model.LinearRegression()
fitted_model = model.fit(X=X_train[['Percent Foreign Born', 'Median Household Income'
]],y=Y_train['Republican'])
predicted = fitted_model.predict(X_train[['Percent Foreign Born', 'Median Household I
ncome']])
print (predicted)

# Evaluate linear regression model using 2 predictors on republican votes
X_train_dummy = pd.get_dummies(X_train, drop_first = True)
X_val_dummy = pd.get_dummies(X_val, drop_first = True)
model = linear_model.LinearRegression().fit(X = X_train_dummy[['Percent Foreign Born'
, 'Median Household Income']], y = Y_train['Republican'])
score_val = model.score(X = X_val_dummy[['Percent Foreign Born', 'Median Household In
come']], y = Y_val['Republican']) # R squared (validation)
print(score_val)

# Evaluate LASSO regression model on republican votes
model = linear_model.Lasso(alpha = 1).fit(X = X_train_dummy, y = Y_train['Republican'
])
score_val = model.score(X = X_val_dummy, y = Y_val['Republican']) # R squared (valida
tion)
print(score_val)

# it seems like lasso regression is better
```

```
[-8.64500143e+02  7.34376317e+04  2.55502563e+03 -7.71882399e+02
  2.37294394e+04  2.55936755e+04 -3.96595520e+03  1.27399410e+04
  4.61918278e+03  8.78250688e+03  4.59768939e+04 -1.31442070e+04
  3.58378972e+03  1.03745826e+05  1.53726698e+05 -8.47852415e+03
  1.28514163e+04  3.21290605e+04  4.86999821e+04  1.73089390e+04
  9.98973568e+02  2.31113216e+04 -9.04997898e+03  1.25519808e+03
  1.15681834e+04  5.22840254e+04  2.17122790e+03  9.62024998e+02
  1.35764442e+04  5.95582311e+03  8.47764802e+04 -4.44991529e+03
 -7.16743463e+02  1.38431985e+05  1.32033382e+05  3.25366947e+04
  2.58102023e+04  5.07805093e+03  1.38532115e+05  3.71478571e+04
 -6.21961324e+02 -8.40920238e+03  1.54228764e+05  1.98910791e+04
  1.24689928e+05  3.19162925e+04  1.09811268e+05  1.09691847e+04
  3.30036029e+04  2.62358862e+04 -3.09282466e+03  4.94857099e+04
  1.24629910e+04  5.87986862e+04 -3.18127421e+03  6.66043275e+03
  7.95044189e+02  3.29243621e+04  3.09045540e+04  3.04105439e+04
  5.22325038e+04  2.50667952e+04 -1.23423026e+04  1.05285635e+05
  5.39507549e+04 -1.77248492e+03  5.38492640e+04  9.44396314e+03
  2.51422531e+04  1.42307673e+04  1.05748028e+04 -1.00816202e+04
 -4.47722920e+03  6.98354761e+03  1.05008635e+04 -1.23735536e+04
  2.93744911e+04  1.54737615e+05  9.87016680e+04  9.54605455e+03
 -7.07037914e+03 -7.74492633e+03  1.24386024e+04  8.32679134e+04
 -4.94976685e+03  3.09174112e+03 -1.08238600e+03  4.36603622e+04
 -9.45280822e+02  6.82335705e+03  5.16461566e+03 -8.95362925e+03
  3.14781138e+04 -1.14685497e+04 -4.20944845e+03  4.66811493e+04
  3.63833838e+04 -7.17871329e+03  2.66213956e+04  5.88723210e+04
  5.24906453e+04  2.26477991e+03  6.65357018e+04  6.48067729e+04
  1.98091328e+04  1.32513366e+04  7.00818371e+04  4.22886513e+03
 -1.73723927e+04 -1.65109311e+04  2.29112317e+04  5.75357554e+03
  1.07820485e+04  6.21776089e+04 -5.85753888e+03  2.68791861e+03
  6.89277009e+04  3.49446058e+04  8.70792881e+03  4.19388694e+04
  8.41410596e+04  6.64172251e+04  7.18981368e+03 -3.90725987e+03
  3.49737147e+04  4.00579513e+04  4.99676843e+04 -9.98075920e+03
 -1.67016598e+04  8.50356290e+03  1.32412073e+04  5.11399064e+04
  3.07080558e+04  2.05660189e+04  3.93230309e+03  4.09112144e+03
  5.89700671e+04 -2.95637545e+03 -3.95947680e+03 -1.10418932e+04
  6.78046855e+04  2.42325634e+03  9.70703496e+03  3.35123826e+04
 -3.11408900e+03  2.49661251e+04  3.05472559e+03  7.71363306e+03
 -5.79594462e+03  1.59435501e+05 -1.32341392e+04  1.19234012e+04
  3.84950978e+04  4.56536502e+04  7.11404861e+04  2.09611713e+04
  1.39538351e+04  2.04235870e+04  1.41300865e+03  5.57329891e+04
 -7.69776259e+03 -6.65420468e+03 -1.98687728e+04  1.72870568e+04
  6.52447493e+03 -2.13206306e+03 -1.17593407e+03  1.87526318e+03
  3.28678573e+04 -7.40558989e+03  4.30022371e+04  1.52907068e+04
 -1.93927644e+04  4.21840403e+04 -8.54305309e+02 -1.49966991e+04
  1.90436737e+03 -4.72896017e+03  1.26638670e+04  1.93255773e+04
 -1.44281693e+03 -1.25968576e+03  2.48211748e+04  3.38242604e+02
  3.56228877e+04 -4.20285629e+01  1.96812414e+04  8.61970306e+03
  6.72372116e+04  4.99437982e+04  4.93915386e+04  4.63307275e+03
 -7.57044975e+03  8.19116617e+03  1.76044647e+04  1.12445809e+04
 -9.45168247e+03  9.56881765e+03  2.71548475e+04 -4.74206613e+03
  1.13670224e+03  3.50928640e+04  2.62154742e+04  4.72036012e+04
  3.39969377e+03  2.70436820e+04  2.06308768e+04  2.62469238e+04
  6.13496257e+04  6.25182427e+03  1.62630036e+04  3.40680046e+04
```

```
 1.70558822e+04 -9.61882431e+03  5.44786127e+04  6.92464149e+03
 3.47128440e+04  1.27962367e+05  2.21716159e+04  9.92273303e+04
-7.62519192e+03  1.38139720e+05  2.83254790e+04  7.97725173e+03
-4.44317159e+03  7.08759135e+03 -4.34129620e+03  8.88964884e+02
 9.62898994e+01  9.35338148e+04  3.95498899e+04 -7.98530920e+03
 3.87366749e+02 -2.14585626e+04  1.28851738e+05  6.77094342e+03
 2.52143629e+03  2.94522777e+04  7.16344191e+04  1.09788504e+05
 1.05772602e+04  1.09207431e+05  5.48187113e+04  6.56260484e+04
 7.80853821e+02  1.27047658e+04  7.25029328e+04  2.50894799e+04
 3.27390607e+04  5.25586140e+03  9.19996060e+04  6.11217693e+04
 8.62056941e+04  3.93315685e+04  4.08146635e+04  1.10527486e+05
 3.25918281e+04  6.44497603e+03  1.19080007e+04  1.90870504e+04
 9.36483302e+04  3.55710565e+04  1.21685605e+04  6.08332932e+03
 8.36302851e+03  1.63387170e+03  1.06365920e+05  3.59765344e+04
 9.43166598e+03  2.85349078e+04  2.09356575e+04 -1.18374885e+04
-1.29138864e+03  1.60687667e+04  2.02118980e+04  4.03984088e+04
 5.21547081e+04  2.23989346e+04  1.05467360e+04 -4.82741298e+03
 7.42607556e+04 -7.70819810e+03  1.52776201e+04  1.81326894e+04
 2.75301791e+04 -5.65754967e+02  4.95901829e+04  1.52599479e+04
-6.28435998e+03  3.72380609e+03  4.78772610e+03  1.22279582e+04
-3.47469698e+03  2.63661867e+04  2.89314192e+04 -1.36039270e+02
-1.00444096e+04  7.26883896e+03  1.13348484e+05  2.39536787e+03
-1.00347707e+03  9.21617597e+03 -3.15041455e+03  1.18747147e+03
 1.28216393e+03  1.94474116e+03  1.66471083e+03 -9.41378724e+03
 4.60181828e+03  2.34658297e+04 -8.22255657e+03 -8.85730354e+03
 5.23587538e+03  3.85280539e+04  6.50490953e+03  2.77299169e+03
-1.52731658e+04  1.55962382e+05  9.43946887e+03 -6.32922325e+03
 1.05955436e+04  1.69000777e+04  7.21863831e+04  4.59244404e+04
-2.39093573e+03  5.44725252e+04  2.49937607e+04  1.62800039e+04
 4.10679346e+03  1.25784835e+04  1.09139699e+05  7.73033815e+04
 1.51837511e+04  1.02211677e+04  2.02951009e+04  6.41292936e+03
-8.02008889e+03  2.38204039e+04  2.63884916e+04 -3.24317332e+03
-1.04918627e+04  4.12598695e+04  6.49813336e+04  9.34092217e+03
 2.17908010e+04  5.31223621e+04  1.59909175e+03  6.43329784e+04
 3.49507476e+04  4.56717892e+04  2.60304977e+04  1.04304657e+04
-6.85816968e+03  1.35606209e+04  4.58950339e+04 -1.23375920e+04
 3.82447414e+04  1.19866566e+05 -3.76489009e+03 -1.07418887e+04
 2.88705345e+03  2.54470319e+04  1.33978374e+04  3.50987711e+04
 2.55329139e+04 -5.83674445e+03  7.58689481e+04  8.37510357e+03
 4.65484955e+04  5.00692740e+04  6.69553616e+03 -3.50984679e+03
 2.45006534e+04  3.67160643e+03  2.35377658e+04 -7.56371875e+03
-1.29655235e+03 -8.70319080e+03  1.02564572e+04  6.54309649e+04
-1.42700792e+04  5.02706132e+04  7.67026061e+03  5.42308645e+04
-7.49850414e+03 -3.26689217e+03 -4.98146549e+03  1.45435808e+05
 1.30323240e+04  3.65048249e+04  1.40087575e+04  7.64623653e+03
 9.38922743e+04  3.16560788e+03  1.12435453e+05  6.69027561e+04
 1.77941784e+04  2.61729414e+04  6.49823627e+03 -5.41900168e+03
-5.41916222e+03 -1.41413875e+04  3.21082462e+04 -6.88943064e+03
-8.60025807e+03 -8.75560628e+03  6.93341925e+04  2.96739168e+04
 5.29346330e+04  4.31273922e+03  2.47476169e+04  5.36652889e+04
-1.10314228e+04 -5.97750843e+03  1.77151948e+05  2.69651216e+04
 1.21290454e+04  9.08923259e+03  2.26304320e+04 -1.25456845e+04
 5.95283594e+04  1.96704218e+04  3.27839705e+04 -5.38098939e+03
 1.57811944e+04  5.43817712e+04  2.21001615e+03 -9.80207738e+03
```

```
 2.41587904e+03  6.29393372e+03  5.36960681e+04  5.54672152e+04
 4.62302843e+04  1.74115163e+04  5.87631395e+02  9.56524015e+03
-4.21534167e+02  1.94073098e+04  2.21025488e+04  5.81285546e+04
 8.47099462e+04  9.69486324e+03  1.35818036e+04  2.95237773e+03
-1.21838194e+04  9.41795695e+04  8.31442571e+03  2.09919664e+04
 3.54965436e+03 -9.92114434e+03  1.22368847e+04  5.49421495e+04
-9.31627771e+03 -6.99185287e+03  3.23604523e+04 -2.00785665e+04
 6.10248269e+04  2.98801815e+04  9.21013259e+03 -8.92221867e+03
 1.36887464e+05  8.28249871e+03  3.59012982e+03  3.26411729e+04
 1.67141161e+04  5.96365335e+03 -4.50699706e+03 -9.59754381e+03
-2.36518120e+03  5.40037145e+04  9.03335461e+04  8.50870601e+04
 1.95951187e+04 -3.24859275e+03 -9.90337016e+02  3.33958786e+04
 7.38320008e+03  4.99755120e+03 -5.52680886e+03  1.45852494e+04
 8.77255231e+03  1.47425639e+04  2.29299694e+04  2.11099010e+04
 8.60664144e+03  2.49729962e+04  2.36575950e+04  1.37947172e+05
 4.98475010e+04 -7.60933106e+03  2.41789317e+04  4.81156760e+03
 3.50242045e+04  1.86111807e+04  2.53033457e+04  9.72748051e+03
 3.05112582e+04  3.38798517e+04  1.91561402e+03  3.49433136e+04
 2.12432296e+04  1.17811228e+04  6.29848987e+03  3.32627001e+04
 2.58704553e+04  1.00977887e+04 -1.42040772e+03  3.04713507e+04
-5.27403353e+03  1.84955546e+05  2.74948005e+04 -1.44216325e+03
 1.29687955e+04 -1.49998801e+04  6.93066701e+04  1.65396111e+04
 4.35818940e+04  2.17858991e+05  3.26302382e+03 -9.67627205e+03
 6.36958376e+04 -6.12859228e+03  7.85538811e+03  2.32621446e+04
-1.65371791e+02  4.17997621e+04  3.79270625e+04 -6.87462593e+03
 4.10974068e+02 -3.43982630e+03 -1.87569929e+04  3.35649818e+04
 5.89704963e+03  3.07347229e+03  2.93159106e+04  3.83673434e+04
 1.60138735e+05  1.23048070e+03  1.25365313e+04  3.26049326e+04
 8.88180985e+04 -1.24106493e+04  8.87394435e+04  3.99803368e+04
 5.97728925e+04  3.46503151e+02  2.85292922e+03  3.50072358e+04
 2.51409649e+03  6.71462818e+03  6.47684777e+04  5.18942751e+04
-4.50291513e+03  1.52380767e+04  4.09644661e+04  6.92211775e+03
 4.66914640e+04 -1.13585154e+04  9.31931628e+03  1.22544481e+03
 1.46128279e+05 -1.04840030e+04  8.45950928e+03  2.16354576e+05
-7.06063615e+03  5.03760855e+03  3.72068400e+04  8.61624963e+04
 2.37961719e+04  4.07494076e+04  5.24170017e+04  1.98101753e+03
 9.67611551e+03  7.54159849e+04 -1.07361334e+04  1.87620199e+04
-1.17913140e+03  4.03431306e+04 -1.67382464e+04  1.94060616e+04
 2.10040997e+04  3.39952807e+03 -9.40878819e+03  3.42981492e+03
 1.29531116e+05  2.27299229e+04  1.00180979e+05 -7.07742429e+03
 4.44174615e+04  3.35709594e+04  2.42793124e+04 -2.21831921e+02
 8.43319098e+04 -6.61965603e+03  5.84971015e+03  4.75080667e+04
-1.59663233e+04 -1.16239532e+04  6.68000275e+04  9.43137628e+03
 5.74069106e+03  1.29007447e+02 -2.96776657e+01  3.33238572e+04
-8.87124233e+02 -2.54804421e+03  8.57948574e+04  5.25872096e+04
-1.07580799e+04  1.37478627e+04  1.41494003e+04  1.73115979e+04
 8.38438700e+03  9.74213006e+02  5.93271030e+04  1.48138247e+04
 7.62300767e+04 -1.03647711e+04  6.22611145e+04  3.23674745e+04
 2.93255515e+04  3.63127080e+03  3.07351028e+04  1.34140860e+04
 1.52433816e+04  1.18300063e+05  6.71481338e+04  5.11574250e+04
 1.23192032e+04  2.12824045e+05 -5.68414283e+03  2.18186870e+05
 1.08056463e+03  3.26806326e+04 -3.37639622e+03  2.95664456e+04
 2.49148263e+04 -6.64163803e+03  1.47435528e+04  4.56906055e+04
-9.76821948e+03  1.24858127e+04  6.15553473e+04  5.55985656e+04
```

-1.75077274e+03 -7.53844568e+03 1.13834097e+04 4.53240646e+04
-7.46834802e+03 6.42879199e+04 2.34406378e+04 4.05659741e+04
 6.19424376e+03 -1.35635095e+03 8.21695546e+03 8.28489039e+04
 7.62634559e+03 1.36360179e+04 3.61981462e+04 2.01393338e+05
-1.58785504e+04 -1.38135641e+04 5.69205421e+04 2.95701702e+04
-9.61985623e+03 1.52120369e+04 4.81547345e+03 -3.19155108e+03
 4.17168863e+04 3.22574678e+04 3.82547494e+04 3.48222962e+04
 4.84898586e+03 3.72979534e+04 3.00366419e+05 -2.51628475e+03
 1.28777111e+03 1.11635230e+05 -3.70220127e+02 1.68476064e+04
 2.30009997e+03 -3.93549358e+03 -3.59203810e+03 -5.96259104e+03
-4.13578131e+03 2.48707836e+04 1.01869020e+04 2.12702537e+04
 7.00313022e+04 7.81277558e+04 -1.16670060e+04 1.06281367e+05
-6.10406476e+03 -1.24595799e+04 2.60761496e+05 1.82895431e+04
 7.60378832e+04 7.55982629e+04 1.34892642e+05 8.09369880e+04
 9.15670884e+04 1.17542204e+04 2.32315529e+04 2.90229082e+03
 2.57793250e+04 1.13067883e+04 6.71895128e+03 2.18390001e+04
 1.56294323e+04 1.34123539e+04 8.81807914e+04 -1.45731093e+04
 1.62570094e+04 2.71621645e+04 5.66748444e+04 1.52053946e+04
-2.76437986e+03 7.34739949e+03 1.09496160e+04 1.66277932e+04
 6.02608411e+04 7.03366395e+03 -4.73225635e+03 1.49800696e+05
 1.58663587e+04 -9.65650878e+03 -8.73249489e+02 2.24824050e+04
 2.67500143e+04 4.14229222e+03 7.12845875e+03 -1.48304606e+03
 7.51943632e+04 3.43863080e+04 4.58400016e+03 4.20168026e+03
 2.31119148e+04 4.67301733e+04 1.48897207e+04 -2.10596121e+04
 7.52334657e+03 -8.35919381e+02 6.44244464e+03 1.93415824e+05
 4.18372875e+04 5.61857858e+03 -1.24425947e+04 -4.92014729e+03
 1.26189504e+04 -1.07899061e+04 -1.92833482e+03 1.15135581e+05
-4.28031889e+03 2.43184290e+04 4.11611724e+03 1.04405549e+04
 7.15006052e+04 6.74298363e+03 1.26976021e+04 5.70146914e+04
-7.70266664e+03 2.03475760e+04 7.56558144e+03 1.23966541e+04
 3.13428316e+01 -2.15196302e+03 8.81692075e+03 -2.20695270e+04
-1.52116075e+03 2.42286295e+04 2.70957850e+04 4.43370682e+04
 4.34996676e+03 -1.03876238e+04 -9.35663644e+03 1.97797389e+04
 2.55205602e+04 2.47546723e+04 -6.68799142e+03 4.94196603e+04
-1.62143251e+04 5.75092350e+04 1.54077721e+04 1.21923965e+04
 9.81603904e+04 4.13818864e+03 -7.48272302e+03 -1.69487553e+03
 5.90239379e+04 4.39121606e+04 7.54340160e+04 2.37482106e+04
-8.37413236e+03 8.19501505e+03 4.92152235e+04 1.35542667e+04
-1.10935802e+04 1.85769049e+04 2.79143131e+03 -5.02434395e+03
 3.80170685e+04 3.89927302e+04 3.91277212e+03 -4.42850989e+03
 1.15772977e+04 3.94801378e+04 -7.77983661e+03 -5.33966299e+03
 2.76479092e+04 2.71266750e+03 8.32035940e+04 1.41373199e+05
-1.75422263e+04 4.12383036e+04 1.79762519e+04 -1.12417612e+04
-5.62975153e+02 7.93735274e+04 2.56791178e+04 1.69767160e+04
-4.15847742e+03 5.61720416e+04 1.40390517e+04 -7.84550451e+03
-2.32293222e+03 -1.85366062e+04 -8.37608094e+03 1.80190098e+04
 3.30919058e+03 3.34433128e+04 2.11061232e+04 1.11439870e+04
 5.68133868e+04 4.05499480e+04 -8.92255331e+03 1.83294401e+04
 6.75382456e+03 -3.77662164e+03 3.16165273e+03 3.45990966e+04
 1.25466824e+04 2.21829852e+04 1.28428449e+04 -1.11921457e+04
 1.74795118e+05 2.19220655e+05 1.37678328e+04 5.12318098e+04
 2.28034133e+04 3.36669941e+04 8.08240879e+04 1.38239594e+05
 9.08318095e+04 2.22613693e+04 1.71775128e+04 3.00445801e+04
 1.81430658e+04 -1.36621835e+04 3.24817771e+04 1.14320110e+04

```
  1.06492460e+04  7.48386393e+04  1.12281032e+03  1.27073960e+04
  4.38481997e+04 -1.68299170e+03  1.68236592e+04  3.20700594e+03
  1.97968918e+04  4.69594483e+04  1.18817452e+04  3.39969462e+04
  2.40475266e+04  2.68689863e+02  1.31488206e+04 -4.71262556e+03
  5.14463916e+03  6.70544503e+03  5.46110753e+04 -1.01051010e+04
 -1.97312039e+02  4.61204955e+03  6.51347146e+04  1.41758006e+05
  2.49855742e+04  4.57526001e+04  7.68268111e+04  3.99969289e+02
  7.13692487e+03  6.89524885e+04  3.46201881e+04 -1.97482257e+03
 -2.36201267e+03  1.50670604e+04  3.64957704e+03  5.69494851e+03]
0.2157831839685247
0.9046204240628879
[ 5.88273135e+03  4.64499172e+04  8.58447709e+03  5.37321085e+03
  2.22539995e+04  1.95027160e+04  2.28037291e+03  1.75824388e+04
  9.74833220e+03  1.10757446e+04  3.10799362e+04 -3.31314831e+03
  4.00138963e+03  6.51266088e+04  9.34414235e+04  1.29763028e+03
  1.31678342e+04  2.21226308e+04  3.33698901e+04  1.68667311e+04
  6.93330629e+03  2.39822981e+04 -3.84758757e+03  7.45609880e+03
  1.04389890e+04  3.14684011e+04  6.59386669e+03  5.16503642e+03
  1.15075894e+04  1.34195816e+04  5.45963844e+04  3.12248082e+03
  3.71293522e+03  8.53174587e+04  7.40450935e+04  2.20773344e+04
  2.09350864e+04  1.11535227e+04  7.39229232e+04  2.36047878e+04
  6.77905444e+03 -6.45578414e+02  8.61180444e+04  2.13115937e+04
  7.36052413e+04  2.74945316e+04  6.82005873e+04  1.56632728e+04
  2.47191345e+04  1.97610435e+04  4.85595316e+03  2.78605507e+04
  1.50583822e+04  4.55382356e+04  3.03994809e+03  1.07041363e+04
  7.05163149e+03  2.16104959e+04  2.20055897e+04  2.80520719e+04
  3.83564678e+04  2.36146451e+04 -1.96443293e+03  5.85546441e+04
  3.79611276e+04  2.74057530e+03  3.00564425e+04  1.35727106e+04
  1.92839353e+04  1.83623187e+04  1.45128593e+04 -6.98167769e+02
  3.87674637e+03  8.93220029e+03  7.74976300e+03 -5.13373830e+03
  2.24701734e+04  9.72762436e+04  5.90539833e+04  1.40894751e+04
  2.66176294e+03  1.15803135e+03  1.35243419e+04  4.53786544e+04
  1.90407598e+03  8.32337268e+03  5.24686140e+03  3.19070824e+04
  4.75289240e+03  8.98742323e+03  8.45007260e+03  2.55897681e+02
  2.31069402e+04 -8.16543580e+02  3.92677259e+03  3.15445337e+04
  2.80965663e+04  1.73548513e+03  2.36090986e+04  3.78352632e+04
  3.71335376e+04  7.26980816e+03  4.26770931e+04  3.95653064e+04
  1.67507805e+04  1.46458674e+04  4.19616889e+04  9.25620210e+03
 -6.02507234e+03 -4.91655200e+03  1.80506540e+04  6.84201636e+03
  1.41919934e+04  3.86710235e+04  2.86597532e+03  9.18241317e+03
  4.21274601e+04  2.46740652e+04  1.17041379e+04  3.46890767e+04
  5.65065842e+04  4.54906122e+04  1.14433630e+04  2.88715618e+03
  3.36032329e+04  2.14827152e+04  3.25911576e+04 -7.92682967e+01
 -5.79238531e+03  7.99098006e+03  1.22613853e+04  4.09974663e+04
  2.31514697e+04  1.82428550e+04  1.01606379e+04  1.06895681e+04
  3.85181785e+04  4.74132784e+03  2.16812114e+03 -2.34335265e+03
  4.91978427e+04  7.83705751e+03  1.38712298e+04  2.64563496e+04
  5.69182904e+03  2.21748547e+04  9.75779437e+03  1.34793162e+04
  2.14144083e+03  9.88411850e+04 -2.93798451e+03  1.47661701e+04
  2.84007342e+04  3.10007579e+04  4.72684171e+04  2.03852266e+04
  1.63591744e+04  1.41135678e+04  6.72092208e+03  3.60834262e+04
  1.19643850e+03  1.68523078e+03 -8.21557892e+03  2.10664824e+04
  1.14436330e+04  5.20515600e+03  5.56706421e+03  6.44276628e+03
  2.22044759e+04  2.42138661e+03  2.93468602e+04  1.78239562e+04
```

-7.19231636e+03 3.42409025e+04 5.86626248e+03 -5.68352825e+03
 7.05704618e+03 2.26088367e+03 1.73049508e+04 1.84985545e+04
 5.62805728e+03 6.29267253e+03 2.30830888e+04 6.58163531e+03
 2.90692455e+04 3.52356901e+03 1.74016458e+04 1.27236942e+04
 5.09581210e+04 3.37134200e+04 4.23289923e+04 9.76500980e+03
 2.05352521e+03 1.05386628e+04 1.91495017e+04 1.17501221e+04
-1.42021864e+03 1.02049210e+04 1.99557724e+04 3.25541536e+03
 5.27650209e+03 2.94282616e+04 1.96589333e+04 3.11618637e+04
 7.25106323e+03 2.01785134e+04 1.97111921e+04 2.00581221e+04
 4.51721728e+04 5.72257289e+03 1.76260141e+04 2.69460254e+04
 1.78038445e+04 -7.99267393e+02 4.12144176e+04 1.07837030e+04
 2.64244652e+04 7.73876867e+04 1.89325795e+04 5.56019567e+04
 1.12448431e+03 7.90152048e+04 2.04012862e+04 1.35357577e+04
 3.79677168e+03 1.07976939e+04 2.86520633e+03 6.91786023e+03
 5.30299829e+03 5.47276481e+04 3.07777904e+04 1.06386698e+03
 7.29228598e+03 -9.69496079e+03 8.47931894e+04 1.07436423e+04
 6.35502997e+03 2.11518784e+04 4.42246754e+04 5.91754106e+04
 1.42175845e+04 6.96309973e+04 3.91612645e+04 5.29340825e+04
 4.21338010e+03 1.34088466e+04 5.57107313e+04 2.43415321e+04
 2.32766379e+04 7.21151968e+03 5.25109626e+04 3.71470508e+04
 6.33164612e+04 2.88904922e+04 2.70871338e+04 6.26590401e+04
 2.40085829e+04 1.11690119e+04 1.50060960e+04 1.75996557e+04
 5.67532715e+04 2.48462902e+04 1.44523749e+04 9.88643169e+03
 1.17195684e+04 6.63676742e+03 5.81764184e+04 2.66565192e+04
 1.42819132e+04 2.59732255e+04 1.83682133e+04 -3.82788846e+03
 2.62344090e+03 1.15038518e+04 1.65298686e+04 2.59363045e+04
 3.74574120e+04 1.87919379e+04 1.43426939e+04 3.94467892e+03
 5.81107870e+04 8.25616713e+02 1.23419414e+04 1.24751466e+04
 2.46416167e+04 3.29402608e+03 3.32607033e+04 1.59397034e+04
 1.83645929e+03 9.02450589e+03 1.08369139e+04 1.61466522e+04
 5.21346259e+03 2.26967033e+04 2.95951547e+04 8.92275912e+03
-3.25698181e+01 9.84158313e+03 6.40683013e+04 9.17913378e+03
 5.11673118e+03 1.38569634e+04 2.91076483e+03 2.62672261e+03
 6.89606159e+03 7.16677846e+03 6.68631856e+03 1.11716999e+02
 7.53476421e+03 2.17425465e+04 1.15587424e+03 1.55611258e+02
 1.07272417e+04 3.53453974e+04 1.25726058e+04 1.06607812e+04
-3.49638120e+03 8.09087916e+04 1.38020657e+04 1.04311707e+03
 1.28984671e+04 1.25827333e+04 5.33186969e+04 3.04612460e+04
 6.49426333e+03 3.60949959e+04 2.56603879e+04 1.32630355e+04
 9.48918778e+03 1.47653772e+04 7.06248434e+04 5.36274000e+04
 1.43796293e+04 1.28483880e+04 1.67391353e+04 1.28736899e+04
 7.09342754e+02 1.79329413e+04 2.13707746e+04 4.71937543e+03
-1.34761642e+03 2.81877460e+04 4.19524461e+04 1.42316666e+04
 2.14776256e+04 3.91440958e+04 8.14666662e+03 4.34302414e+04
 2.88984861e+04 3.43119672e+04 1.92063954e+04 1.39796986e+04
 1.97934732e+03 1.50266817e+04 3.31604409e+04 -3.71529297e+03
 2.35133532e+04 6.83121426e+04 3.65679698e+03 -1.62531038e+03
 7.03811130e+03 1.74141808e+04 1.44977410e+04 2.43431691e+04
 2.59699968e+04 3.28988519e+03 5.20489721e+04 1.16296643e+04
 2.77358371e+04 4.31558592e+04 5.78911733e+03 2.92159385e+03
 1.94502662e+04 1.07694150e+04 2.31889541e+04 6.97362498e+02
 3.25594383e+03 -1.54101094e+03 1.12976435e+04 4.15183116e+04
-3.00810052e+03 2.98277407e+04 1.23259628e+04 4.38972512e+04
 8.81394484e+02 1.13401874e+03 1.07062295e+03 7.58732971e+04

```
 1.52495606e+04  2.40094951e+04  1.49124368e+04  1.12441837e+04
 5.55829867e+04  9.95437558e+03  6.69190873e+04  4.52259634e+04
 2.16997465e+04  1.71840041e+04  1.00401237e+04  3.32201592e+03
 2.73025250e+03 -4.65536337e+03  2.74508295e+04  2.28511900e+03
-7.59952385e+02  4.32440602e+02  4.11157888e+04  2.56900266e+04
 3.37714540e+04  9.14908473e+03  1.98624561e+04  3.47491799e+04
-2.27683603e+03  6.07365043e+02  9.94758672e+04  1.78386837e+04
 1.34245563e+04  1.26347426e+04  2.05709602e+04 -1.59761594e+03
 3.86647372e+04  2.22129220e+04  2.24746751e+04  2.78999002e+00
 1.43110942e+04  3.15046968e+04  8.23616685e+03 -1.43132550e+03
 9.38338647e+03  7.29574080e+03  3.39891318e+04  3.15801327e+04
 3.09176569e+04  1.57241517e+04  4.07216305e+03  1.57484685e+04
 7.08334673e+03  1.65562853e+04  1.78147503e+04  4.18819746e+04
 4.67002384e+04  1.30223506e+04  1.55995206e+04  1.14337857e+04
-1.18810127e+03  6.12858718e+04  9.20379078e+03  1.65128125e+04
 8.54309092e+03  8.14359412e+01  1.52139772e+04  3.20022777e+04
-1.39738966e+03  1.45016850e+03  2.96772038e+04 -7.84672471e+03
 4.01464120e+04  2.05777968e+04  1.44074858e+04 -2.37677290e+02
 8.88125475e+04  1.22572670e+04  7.61922105e+03  2.60380738e+04
 1.16084769e+04  8.65509616e+03  3.07262874e+03 -1.27634136e+03
 4.26099315e+03  3.37311325e+04  6.29970216e+04  4.52626814e+04
 1.75210206e+04  3.25476423e+03  4.19919401e+03  2.27949439e+04
 1.42433241e+04  4.73944138e+03  2.10146242e+03  1.37688662e+04
 9.29510144e+03  1.55975150e+04  1.79653607e+04  1.79572555e+04
 1.33499949e+04  1.78896986e+04  1.88809141e+04  7.66778514e+04
 3.97654589e+04  2.51678672e+02  2.46575296e+04  9.89696185e+03
 3.28908018e+04  1.78906261e+04  2.62207483e+04  8.10740271e+03
 2.79653793e+04  2.59997114e+04  7.83892638e+03  3.16434847e+04
 1.52838598e+04  1.28706142e+04  8.40569621e+03  2.51396284e+04
 2.66218594e+04  9.48691488e+03  5.88152588e+03  2.22016128e+04
 2.83097006e+03  1.11359618e+05  2.36324857e+04  1.74166522e+03
 1.06404794e+04 -4.73789079e+03  5.18587730e+04  1.64142040e+04
 2.78476971e+04  1.16079268e+05  5.80678263e+03 -1.61418442e+03
 3.87041838e+04  3.71264921e+02  1.35940162e+04  1.98360906e+04
 6.00262587e+03  3.22161314e+04  2.70075879e+04  5.00452794e+02
 5.51068300e+03  3.40809271e+03 -7.64824818e+03  2.21619188e+04
 1.16476579e+04  8.01683478e+03  2.77731087e+04  2.77003191e+04
 8.74093014e+04  6.25655221e+03  1.59064001e+04  2.24093197e+04
 5.34555722e+04 -1.37567664e+03  5.82749384e+04  2.89814835e+04
 4.94334485e+04  8.27565351e+03  9.11751346e+03  2.78560055e+04
 1.04468571e+04  1.32621563e+04  4.05075454e+04  3.69074156e+04
 3.24263500e+03  1.25659734e+04  2.79814742e+04  1.11875616e+04
 3.73787480e+04 -4.57983688e+03  1.11878324e+04  5.35706862e+03
 9.20808161e+04 -1.12354739e+03  1.26755177e+04  1.16007298e+05
 4.03912708e+02  1.06043969e+04  2.40730889e+04  5.67153984e+04
 2.43302808e+04  2.58244164e+04  2.86046697e+04  7.79277166e+03
 1.43588511e+04  4.85040598e+04 -9.10737935e+02  1.33063719e+04
 4.83777841e+03  2.86053742e+04 -4.77427789e+03  1.96742372e+04
 2.07959825e+04  4.38650102e+03 -3.51582144e+02  9.46150206e+03
 7.28990183e+04  1.97370431e+04  6.01367118e+04  2.26979114e+03
 3.24262788e+04  2.53428204e+04  2.11934424e+04  6.44267284e+03
 4.76226315e+04  3.65127731e+03  8.56724369e+03  3.72881651e+04
-4.48289886e+03 -8.79956690e+02  5.07728258e+04  1.33717377e+04
 1.06678680e+04  4.02463939e+03  6.39254674e+03  2.21632822e+04
```

```
 5.61871848e+03  1.60748808e+03  4.84278189e+04  4.31296041e+04
-1.72826187e+03  1.41198623e+04  1.40751287e+04  1.71240988e+04
 1.27011934e+04  5.70405623e+03  3.56997398e+04  1.52864760e+04
 4.69411759e+04 -1.31097281e+03  3.86837561e+04  3.01295068e+04
 2.24980774e+04  9.95777029e+03  2.53753872e+04  1.76067615e+04
 1.67792151e+04  5.79879775e+04  4.47224419e+04  3.93260252e+04
 1.58921727e+04  1.38054544e+05  5.91585285e+01  1.26052041e+05
 3.56394572e+03  2.69963893e+04  2.94214843e+03  2.14120635e+04
 1.91443378e+04  3.48804534e+03  1.95023357e+04  3.23039037e+04
 4.18259525e+02  1.43749511e+04  3.71586617e+04  3.94679953e+04
 3.90470836e+03  1.86717660e+03  1.26957719e+04  3.24012301e+04
 8.07964431e+02  4.10395171e+04  2.14635601e+04  2.80042344e+04
 1.15159212e+04  3.73151831e+03  1.18829112e+04  5.60239277e+04
 1.27122183e+04  1.72491491e+04  3.36109676e+04  1.20486233e+05
-6.36672080e+03 -3.84164690e+03  3.87752319e+04  2.11921118e+04
 7.38467843e+02  1.53275076e+04  9.25274477e+03  1.91704256e+03
 2.96135455e+04  2.37180713e+04  2.67784511e+04  2.98977559e+04
 7.99141528e+03  3.03958886e+04  1.54927381e+05  1.75306554e+03
 7.48074239e+03  6.64665604e+04  7.40008459e+03  1.63168566e+04
 6.91353626e+03  2.73060736e+03  4.09385939e+03  9.46488441e+02
 4.65351582e+03  2.27635892e+04  1.41811025e+04  1.48217044e+04
 4.30009262e+04  5.06067178e+04 -2.85772506e+03  6.07125595e+04
 2.78115743e+03 -3.28688614e+03  1.40712453e+05  1.28109841e+04
 4.43985074e+04  5.28129454e+04  7.39041069e+04  6.01237756e+04
 6.48528255e+04  8.99599031e+03  1.97993832e+04  8.45997251e+03
 2.32193074e+04  1.22125211e+04  7.66548584e+03  2.08938759e+04
 1.79799660e+04  1.42697711e+04  6.78776099e+04 -3.69306946e+03
 1.60094457e+04  2.17435874e+04  4.06514706e+04  1.57939054e+04
 2.98261171e+03  7.19189452e+03  1.28213741e+04  1.73683187e+04
 4.24876261e+04  1.15159213e+04  1.34889011e+03  9.05555520e+04
 1.82991078e+04 -5.31910150e+02  6.16933909e+03  1.71359353e+04
 1.84788893e+04  8.49435142e+03  1.29938945e+04  6.62348198e+03
 4.19282971e+04  2.92091118e+04  8.86128835e+03  9.98182817e+03
 2.03193068e+04  3.88817318e+04  1.45409870e+04 -8.42224749e+03
 7.01075783e+03  4.27320789e+03  1.19727588e+04  1.04705140e+05
 3.14936793e+04  1.20489739e+04 -2.21968977e+03  1.52018605e+03
 1.34301949e+04 -1.77993369e+03  3.01192374e+03  6.46996774e+04
 3.66784918e+03  2.11521836e+04  9.39563525e+03  1.41259393e+04
 3.58505662e+04  9.47725154e+03  1.08983011e+04  4.24261974e+04
 2.38813639e+03  2.03009246e+04  1.11036997e+04  1.39657186e+04
 6.03989467e+03  5.06089685e+03  1.38480320e+04 -1.02075315e+04
 4.42799492e+03  2.01052847e+04  2.07481032e+04  2.96151934e+04
 5.97951048e+03 -1.10051265e+03 -2.58511707e+02  1.89799616e+04
 1.75560211e+04  1.89268955e+04  6.57645266e+02  3.06570293e+04
-4.96899356e+03  3.27193036e+04  1.57116681e+04  1.36794224e+04
 5.64564984e+04  9.86812669e+03  1.20659471e+03  4.93413233e+03
 4.56010809e+04  3.36795432e+04  4.69496441e+04  2.20751102e+04
 5.16702323e+02  1.29187238e+04  3.33323246e+04  9.65466730e+03
-2.18927706e+03  1.81316236e+04  8.72428968e+03  8.71263124e+01
 2.40775220e+04  2.91007223e+04  9.62449795e+03  4.22546667e+03
 1.43765450e+04  2.78646733e+04  9.54461942e+02  3.28147124e+03
 1.99753721e+04  8.13201149e+03  4.94953746e+04  9.07781172e+04
-6.64557897e+03  2.41301759e+04  1.94837004e+04 -1.64763723e+03
 6.84909589e+03  5.94764872e+04  2.34507478e+04  1.67142739e+04
```

```
  3.89791957e+03  3.34834509e+04  1.49313832e+04 -9.64453227e+01
  5.50381374e+03 -7.31643828e+03 -4.52877618e+01  1.63256772e+04
  6.49019790e+03  2.64321880e+04  1.37258533e+04  1.52082323e+04
  4.02338876e+04  2.34236170e+04  5.59983816e+02  1.47516333e+04
  1.28210450e+04  4.10697011e+03  1.00603773e+04  3.39278102e+04
  1.67930916e+04  1.95661104e+04  1.44483893e+04 -1.99029206e+03
  9.69282945e+04  1.24709397e+05  1.26283823e+04  3.15138298e+04
  2.11933549e+04  2.58337471e+04  5.20200066e+04  7.52093596e+04
  6.55578323e+04  2.03669530e+04  1.45052353e+04  2.01528210e+04
  1.30341427e+04 -2.11085983e+03  2.42152419e+04  1.47806699e+04
  1.35239743e+04  4.91520712e+04  6.73185702e+03  1.20966399e+04
  3.11453809e+04  5.06364999e+03  1.74765990e+04  7.88053980e+03
  1.75193751e+04  3.32218538e+04  1.61389251e+04  2.96711954e+04
  2.34644078e+04  5.78060844e+03  1.19050224e+04  2.17994292e+03
  1.06904502e+04  1.05320707e+04  3.60058763e+04 -2.59283471e+03
  6.99270447e+03  8.56403453e+03  3.72680123e+04  7.44263799e+04
  2.54025605e+04  3.37042801e+04  4.66566866e+04  6.78089773e+03
  1.18199468e+04  4.22190208e+04  2.41112373e+04  6.62278595e+03
  3.14997815e+03  1.65416820e+04  7.52705550e+03  1.10479978e+04]
0.12344379156818297
0.8328536880997107
```

In [6]:

```
# task 4
# Partitioning the data into training and validation using only variables that are st
ronger predictors
X = ['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino'
, 'Percent Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Age 65 and Olde
r', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree',
'Percent Rural']
Y = ['Party']
X_train_class, X_val_class, Y_train_class, Y_val_class = train_test_split(data[X], da
ta[Y], test_size=0.25, random_state=0)
scaler.fit(X_train_class)
x_train_scaled_class = scaler.transform(X_train_class)
x_val_scaled_class = scaler.transform(X_val_class)
```

In [7]:

```
# task 4 - continuation
# Build a classification model to classify each county as Democratic or Republican
# kNN classifier
classifier = KNeighborsClassifier(n_neighbors=5, weights='uniform')
classifier.fit(x_train_scaled_class, Y_train_class.values.ravel())
```

Out[7]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=
2,
                     weights='uniform')
```

In [8]:

```
y_pred_class = classifier.predict(x_val_scaled_class)
```

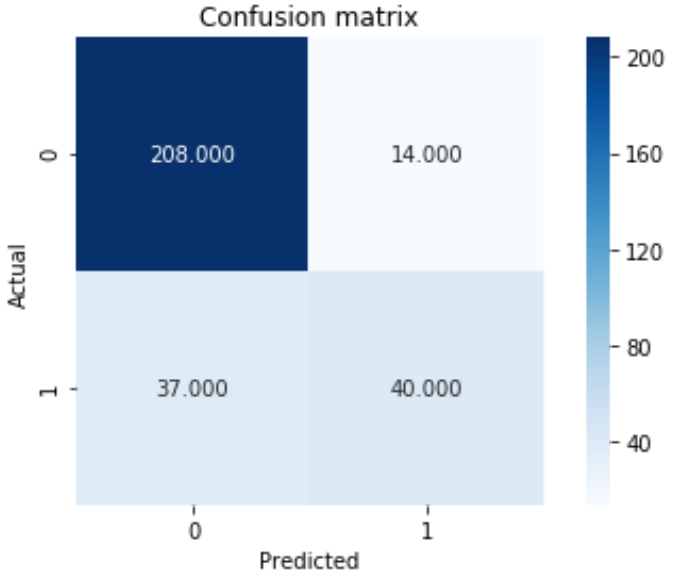
In [9]:

```
conf_matrix = metrics.confusion_matrix(Y_val_class, y_pred_class)
print(conf_matrix)
```

```
[[208  14]
 [ 37  40]]
```

In [10]:

```
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [23]:

```
accuracy = metrics.accuracy_score(Y_val_class, y_pred_class)
error = 1 - accuracy
precision = metrics.precision_score(Y_val_class, y_pred_class, average=None)
recall = metrics.recall_score(Y_val_class, y_pred_class, average=None)
F1_score = metrics.f1_score(Y_val_class, y_pred_class, average=None)
print("Results of kNN classifier\n")
print("Accuracy: " + str(accuracy) + "\n" + "Error: " + str(error) + "\n" + "Precisio
n: " + str(precision) + "\n" + "Recall: " + str(recall) + "\n" + "F1 score: " + str(F
1_score))
```

```
Results of kNN classifier

Accuracy: 0.8394648829431438
Error: 0.1605351170568562
Precision: [0.84251969 0.82222222]
Recall: [0.96396396 0.48051948]
F1 score: [0.89915966 0.60655738]
```

In [24]:

```
# task 4 - continuation
# SVM Classifier
classifier = SVC(kernel='rbf')
classifier.fit(x_train_scaled_class, Y_train_class.values.ravel())

svm_class = classifier
```

In [25]:

```
y_pred_class = classifier.predict(x_val_scaled_class)
```

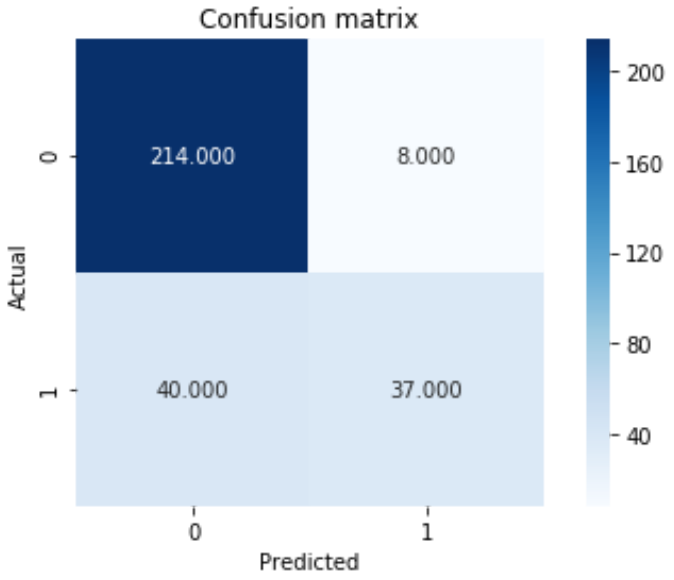
In [26]:

```
conf_matrix = metrics.confusion_matrix(Y_val_class, y_pred_class)
print(conf_matrix)
```

```
[[214   8]
 [ 40  37]]
```

In [27]:

```
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [28]:

```
accuracy = metrics.accuracy_score(Y_val_class, y_pred_class)
error = 1 - accuracy
precision = metrics.precision_score(Y_val_class, y_pred_class, average=None)
recall = metrics.recall_score(Y_val_class, y_pred_class, average=None)
F1_score = metrics.f1_score(Y_val_class, y_pred_class, average=None)
print("Results of SVM classifier\n")
print("Accuracy: " + str(accuracy) + "\n" + "Error: " + str(error) + "\n" + "Precisio
n: " + str(precision) + "\n" + "Recall: " + str(recall) + "\n" + "F1 score: " + str(F
1_score))
```

```
Results of SVM classifier

Accuracy: 0.8394648829431438
Error: 0.1605351170568562
Precision: [0.84251969 0.82222222]
Recall: [0.96396396 0.48051948]
F1 score: [0.89915966 0.60655738]
```

In [29]:

```
# task 4
# Decision trees
classifier = DecisionTreeClassifier(criterion="entropy", random_state=0)
classifier.fit(x_train_scaled_class, Y_train_class.values.ravel())
```

Out[29]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth
=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [30]:

```
# Number of nodes in the decision tree
print("Number of nodes in the decision tree: " + str(len(classifier.tree_.__getstate_
_()['nodes'])))
```

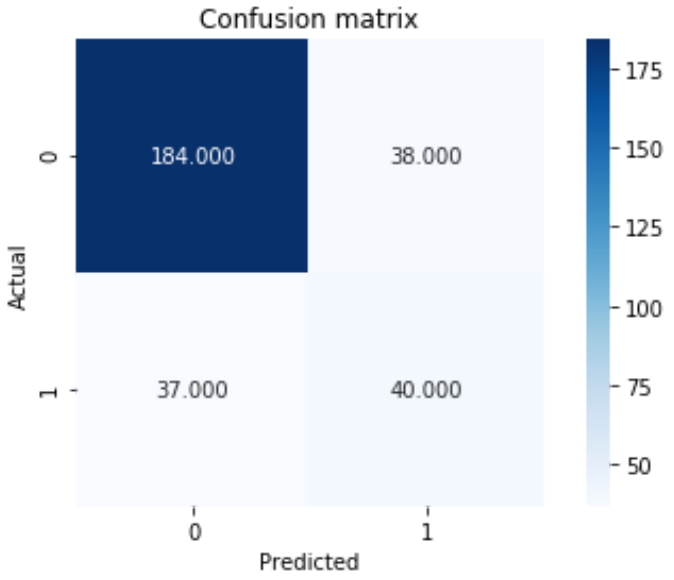
```
Number of nodes in the decision tree: 227
```

In [31]:

```
y_pred_class = classifier.predict(x_val_scaled_class)
```

In [32]:

```
conf_matrix = metrics.confusion_matrix(Y_val_class, y_pred_class)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [33]:

```
accuracy = metrics.accuracy_score(Y_val_class, y_pred_class)
error = 1 - accuracy
precision = metrics.precision_score(Y_val_class, y_pred_class, average=None)
recall = metrics.recall_score(Y_val_class, y_pred_class, average=None)
F1_score = metrics.f1_score(Y_val_class, y_pred_class, average=None)
print("Results of Decision trees classifier\n")
print("Accuracy: " + str(accuracy) + "\n" + "Error: " + str(error) + "\n" + "Precisio
n: " + str(precision) + "\n" + "Recall: " + str(recall) + "\n" + "F1 score: " + str(F
1_score))
```

```
Results of Decision trees classifier

Accuracy: 0.7491638795986622
Error: 0.25083612040133785
Precision: [0.83257919 0.51282051]
Recall: [0.82882883 0.51948052]
F1 score: [0.83069977 0.51612903]
```

In [34]:

```
# task 4
# Naives Bayes Classifier
classifier = GaussianNB()
classifier.fit(x_train_scaled_class, Y_train_class.values.ravel())
```

Out[34]:

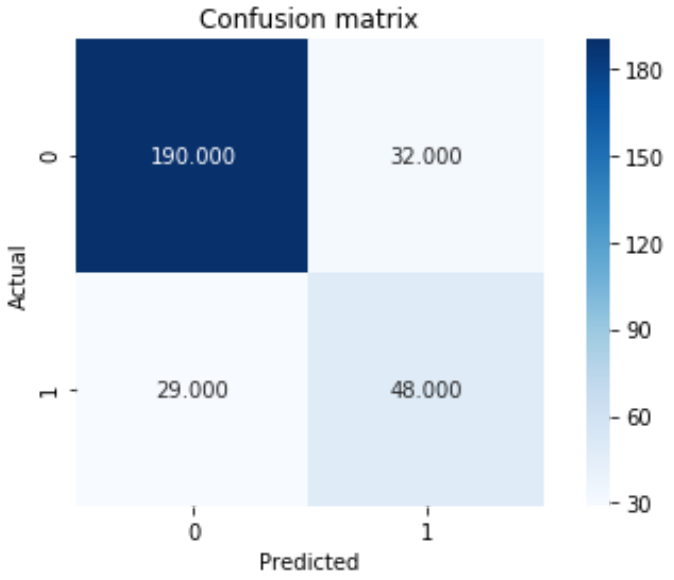
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [35]:

```
y_pred_class = classifier.predict(x_val_scaled_class)
```

In [36]:

```
conf_matrix = metrics.confusion_matrix(Y_val_class, y_pred_class)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [37]:

```
accuracy = metrics.accuracy_score(Y_val_class, y_pred_class)
error = 1 - accuracy
precision = metrics.precision_score(Y_val_class, y_pred_class, average=None)
recall = metrics.recall_score(Y_val_class, y_pred_class, average=None)
F1_score = metrics.f1_score(Y_val_class, y_pred_class, average=None)
print("Results of Naive Bayes classifier\n")
print("Accuracy: " + str(accuracy) + "\n" + "Error: " + str(error) + "\n" + "Precisio
n: " + str(precision) + "\n" + "Recall: " + str(recall) + "\n" + "F1 score: " + str(F
1_score))
```

```
Results of Naive Bayes classifier

Accuracy: 0.7959866220735786
Error: 0.20401337792642138
Precision: [0.86757991 0.6       ]
Recall: [0.85585586 0.62337662]
F1 score: [0.861678   0.61146497]
```

In [38]:

```
# SVM Classifier has the highest accuracy among all the 4 classification methods for
 the selected predictors.
```

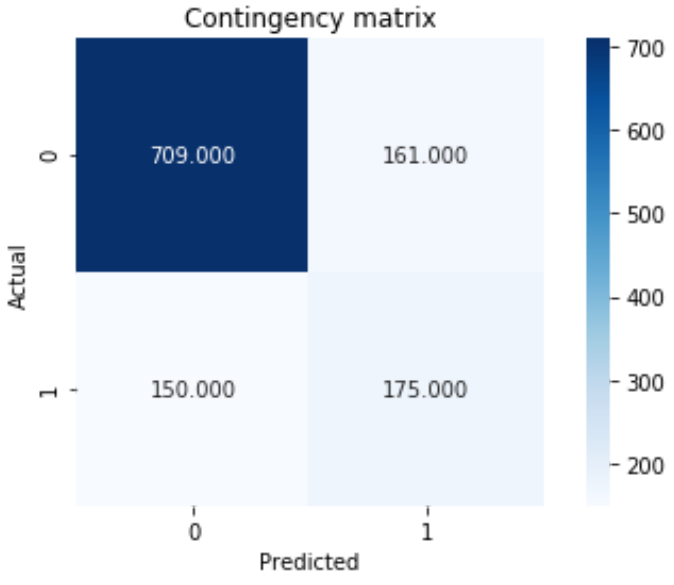
In [39]:

```
# Task 5 Start Kmeans 2 clusters all vars
```

In [40]:

```
X = ['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not
Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Fema
le','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income',
'Percent Unemployed','Percent Less than High School Degree','Percent Less than Bachel
or\'s Degree','Percent Rural']
Y = ['Party']

train_class = data[X]
lables = data[Y].to_numpy().reshape(-1)

scaler = StandardScaler()
scaler.fit(train_class)
train_scaled_class = scaler.transform(train_class)

clustering = KMeans(n_clusters=2, init='random', random_state=0).fit(train_scaled_cla
ss)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(lables, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

adjusted_rand_index = metrics.adjusted_rand_score(lables, clusters)
silhouette_coefficient = np.average(metrics.silhouette_samples(train_scaled_class, cl
usters, metric='euclidean'))
print([adjusted_rand_index, silhouette_coefficient])
```

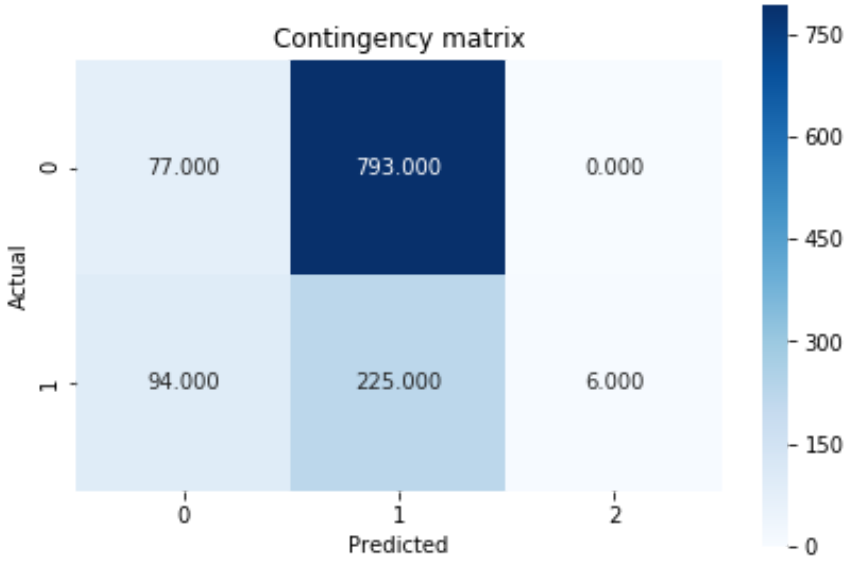
```
[0.19751656022671712, 0.30700290833697047]
```



In [41]:

```
# DB Scan eps = 2 min_samples = 5
clustering = DBSCAN(eps=2,min_samples=5,metric='euclidean').fit(train_scaled_class)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(lables, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

adjusted_rand_index = metrics.adjusted_rand_score(lables, clusters)
silhouette_coefficient = np.average(metrics.silhouette_samples(train_scaled_class, cl
usters, metric='euclidean'))
print([adjusted_rand_index, silhouette_coefficient])
```

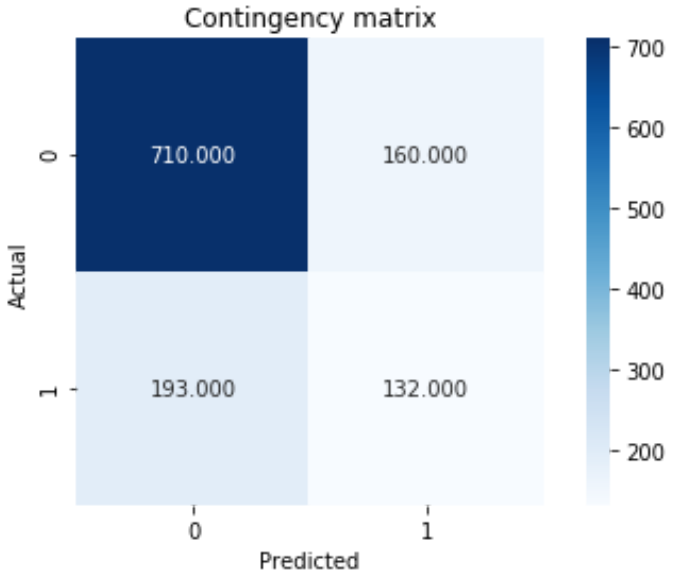
```
[0.1577583255755803, 0.31265113101181913]
```



In [42]:

```
# Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino','Pe
rcent Hispanic or Latino','Percent Foreign Born'
# Kmeans init = 'k-means++' n init = 15
X = ['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino'
,'Percent Hispanic or Latino','Percent Foreign Born']
Y = ['Party']


train_class = data[X]
lables = data[Y].to_numpy().reshape(-1)

scaler = StandardScaler()
scaler.fit(train_class)
train_scaled_class = scaler.transform(train_class)

clustering = KMeans(n_clusters=2, n_init= 15, init='k-means++', random_state=0).fit(t
rain_scaled_class)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(lables, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

adjusted_rand_index = metrics.adjusted_rand_score(lables, clusters)
silhouette_coefficient = np.average(metrics.silhouette_samples(train_scaled_class, cl
usters, metric='euclidean'))
print([adjusted_rand_index, silhouette_coefficient])
```

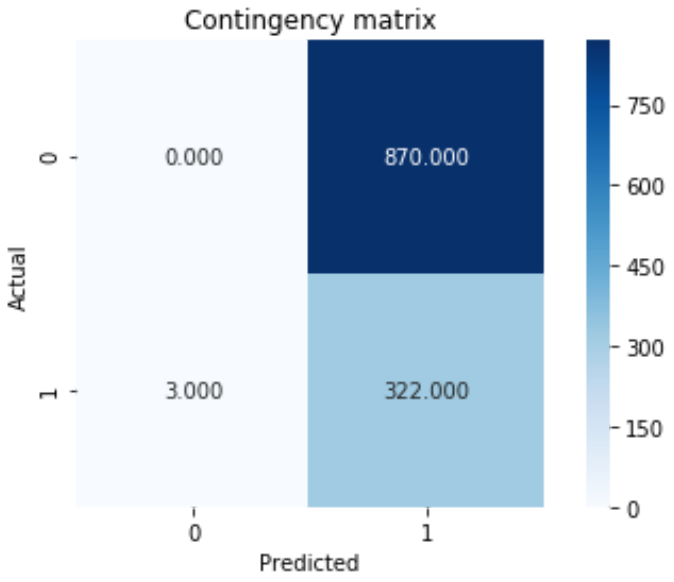
```
[0.11911877926404817, 0.5818101112791731]
```



In [43]:

```
# DB Scan eps = 2 min samples = 8
clustering = DBSCAN(eps=2,min_samples=8,metric='euclidean').fit(train_scaled_class)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(lables, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

adjusted_rand_index = metrics.adjusted_rand_score(lables, clusters)
silhouette_coefficient = np.average(metrics.silhouette_samples(train_scaled_class, cl
usters, metric='euclidean'))
print([adjusted_rand_index, silhouette_coefficient])
```

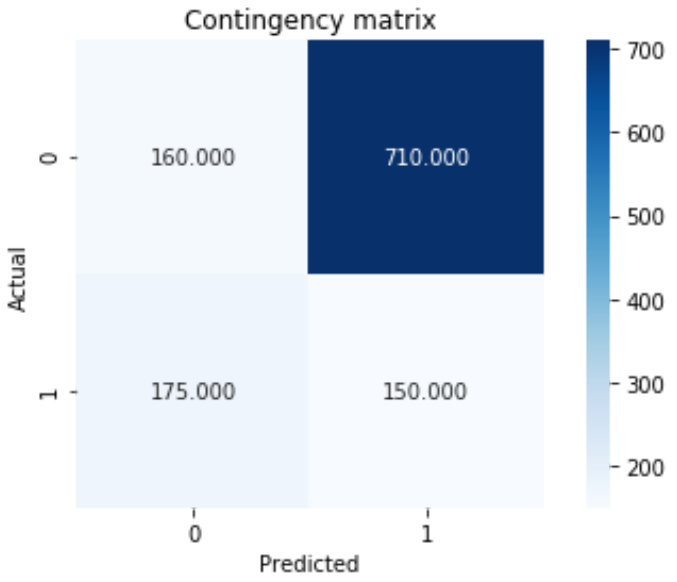
```
[0.008414453628750557, 0.7233179178571693]
```



In [44]:

```
# 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not Hi
spanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Femal
e','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income','P
ercent Unemployed','Percent Less than High School Degree','Percent Less than Bachelor
\'s Degree','Percent Rural'
# Kmeans init = 'k-means++' n init = 15
X = ['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not
Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Fema
le','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income',
'Percent Unemployed','Percent Less than High School Degree','Percent Less than Bachel
or\'s Degree','Percent Rural']

train_class = data[X]
lables = data[Y].to_numpy().reshape(-1)

scaler = StandardScaler()
scaler.fit(train_class)
train_scaled_class = scaler.transform(train_class)

clustering = KMeans(n_clusters=2, n_init= 15, init='k-means++', random_state=0).fit(t
rain_scaled_class)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(lables, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

adjusted_rand_index = metrics.adjusted_rand_score(lables, clusters)
silhouette_coefficient = np.average(metrics.silhouette_samples(train_scaled_class, cl
usters, metric='euclidean'))
print([adjusted_rand_index, silhouette_coefficient])
```

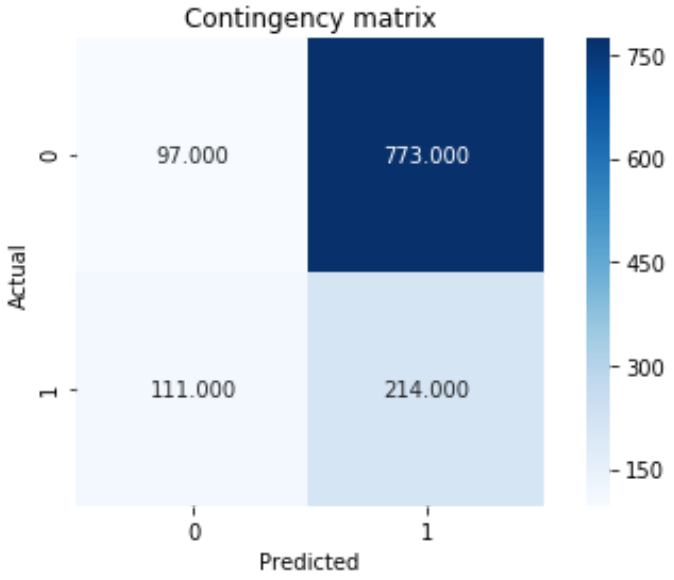
[0.19893799165776016, 0.30691597445230206]



In [45]:

```
# DB Scan eps =2 min samples =8
clustering = DBSCAN(eps=2,min_samples=8,metric='euclidean').fit(train_scaled_class)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(lables, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blue
s)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

adjusted_rand_index = metrics.adjusted_rand_score(lables, clusters)
silhouette_coefficient = np.average(metrics.silhouette_samples(train_scaled_class, cl
usters, metric='euclidean'))
print([adjusted_rand_index, silhouette_coefficient])
```

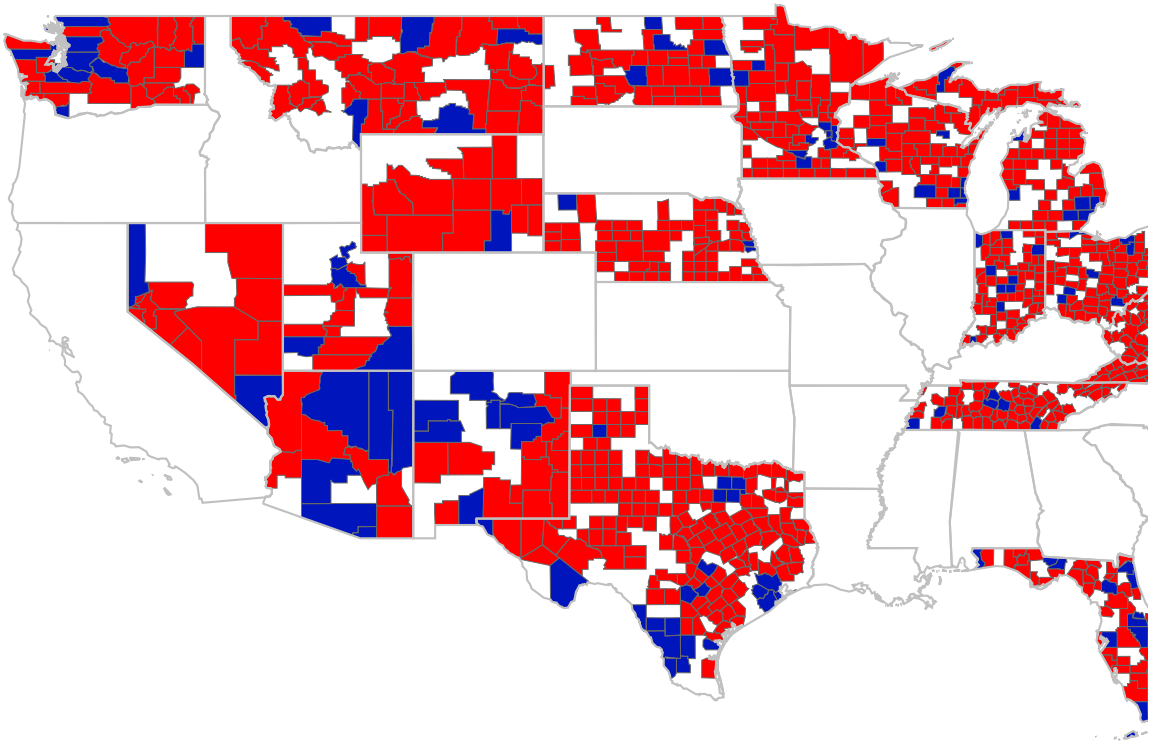
```
[0.15489854564128577, 0.3468850007721493]
```



In [46]:

```
# task 6
X = ['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino'
, 'Percent Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Age 65 and Olde
r', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree',
'Percent Rural']

_data = data[X]
_fips = data[['FIPS']]

scaler = StandardScaler()
scaler.fit(_data)
scaled_data = scaler.transform(_data)

party = svm_class.predict(scaled_data).tolist()
fips = _fips.values

colorscale = ["#ff0000","#0015bc"]
fig = create_choropleth(fips=fips, values=party, colorscale=colorscale, county_outlin
e={'color': 'rgb(105,105,105)', 'width': 0.25}, state_outline={'color': 'rgb(192,192,
192)','width': 1})
fig.layout.template = None
fig.show()
```

```
//anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:6692: Futur
eWarning:

Sorting because non-concatenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=Tru
e'.
```



In [48]:

```
# task 7
test_data = pd.read_csv("demographics_test.csv")

scaler.fit(X_train)
x_test_scaled = scaler.transform(test_data[Xvariables])

model = linear_model.Lasso(alpha=1).fit(X=X_train_dummy, y=Y_train['Democratic'])
dem_predicted = model.predict(x_test_scaled)
dem_predicted = dem_predicted.astype(int)

model = linear_model.Lasso(alpha=1).fit(X=X_train_dummy, y=Y_train['Republican'])
rep_predicted = model.predict(x_test_scaled)
rep_predicted = rep_predicted.astype(int)

class_test = test_data.iloc[:, [4, 5, 6, 9, 10, 13, 14, 15]]
scaler.fit(class_test)
class_test_scaled = scaler.transform(class_test)

test_pred = svm_class.predict(class_test_scaled)

a = np.array([test_data['State'], test_data['County'], dem_predicted, rep_predicted,
test_pred])
a = np.column_stack(a)
np.savetxt("output.csv", a, delimiter=',', header="State,County,Democratic,Republica
n,Party", fmt="%s", comments="")

# the output of this task is stored in the file 'output.csv'
```