In [1]:
```
#import libraries
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
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn import metrics
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import mean_squared_error
import math
```

In [2]:
```
#load the merged_train set
merged = pd.read_csv('merged_train.csv')
merged.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	Percent Female	Pe A l
0	AZ	apache	4001	72346	18.571863	0.486551	5.947806	1.719515	50.598513	45.8
1	AZ	cochise	4003	128177	56.299492	3.714395	34.403208	11.458374	49.069646	37.9
2	AZ	coconino	4005	138064	54.619597	1.342855	13.711033	4.825298	50.581614	48.9
3	AZ	gila	4007	53179	63.222325	0.552850	18.548675	4.249798	50.296170	32.2
4	AZ	graham	4009	37529	51.461536	1.811932	32.097844	4.385942	46.313518	46.3

Task 01

Partitioned the given dataset into training set and validation set with the ratio 75:25. We have used hold-out method

In [3]:
```
#Partition the merged dataset into a training set and a validation set using t
he holdout method or the cross-validation method
x_train, x_valid, y_train, y_validate = train_test_split(merged.iloc[:,0:-1],
merged['Party'], test_size = 0.25, random_state = 0)
```

Task 02

In [4]:
```
# Standardize the training set and the validation set

std_train=x_train.select_dtypes(include=[np.int64,np.float64])
std_train=std_train.iloc[:,1:-2]
train_columns=std_train.columns
std_valid=x_valid.select_dtypes(include=[np.int64,np.float64])
std_valid=std_valid.iloc[:,1:-2]
scaler = StandardScaler()
scaler.fit(std_train)
x_train_scaled = scaler.transform(std_train)
x_validate_scaled = scaler.transform(std_valid)
x_train_scaled_df=pd.DataFrame(x_train_scaled,index=std_train.index,columns=tr
ain_columns)
x_validate_scaled_df=pd.DataFrame(x_validate_scaled,index=std_valid.index,colu
mns=std_valid.columns)
```

In [5]:
```
x_validate_scaled_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 299 entries, 1103 to 2
Data columns (total 13 columns):
Total Population                            299 non-null float64
Percent White, not Hispanic or Latino       299 non-null float64
Percent Black, not Hispanic or Latino       299 non-null float64
Percent Hispanic or Latino                  299 non-null float64
Percent Foreign Born                        299 non-null float64
Percent Female                              299 non-null float64
Percent Age 29 and Under                    299 non-null float64
Percent Age 65 and Older                    299 non-null float64
Median Household Income                     299 non-null float64
Percent Unemployed                          299 non-null float64
Percent Less than High School Degree        299 non-null float64
Percent Less than Bachelor's Degree         299 non-null float64
Percent Rural                               299 non-null float64
dtypes: float64(13)
memory usage: 32.7 KB
```

In [6]:
```
x_train_scaled_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 896 entries, 589 to 684
Data columns (total 13 columns):
Total Population                          896 non-null float64
Percent White, not Hispanic or Latino     896 non-null float64
Percent Black, not Hispanic or Latino     896 non-null float64
Percent Hispanic or Latino                896 non-null float64
Percent Foreign Born                      896 non-null float64
Percent Female                            896 non-null float64
Percent Age 29 and Under                  896 non-null float64
Percent Age 65 and Older                  896 non-null float64
Median Household Income                   896 non-null float64
Percent Unemployed                        896 non-null float64
Percent Less than High School Degree      896 non-null float64
Percent Less than Bachelor's Degree       896 non-null float64
Percent Rural                             896 non-null float64
dtypes: float64(13)
memory usage: 98.0 KB
```

Task 03

Build a linear regression model to predict the number of votes cast for the Democratic party in each county

Multi-linear regression model - predicting number of votes from Democratic party in each county

In [7]:
```
#We choose all the variables as predictors
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)

rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[ 69224.38708039  -3209.1591268   -1023.23488454  -6931.14708179
   3973.74580741    194.19056985  -5299.5676761   -1853.22320472
   1471.25963216   1467.0213699    4037.7699931  -10519.02638282
   -158.13004477]
R sqaure - 0.9338361960241587
Adjusted R square - 0.9308181979480676
RMSE - 14771.994793075706
```

In [8]:
```
# we choose the variables - 'Percent White, not Hispanic or Latino','Percent B
lack, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than
 Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Percent White, not Hispanic o
r Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino'
,'Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Ag
e 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Percent White, not His
panic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or
 Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Pe
rcent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R square -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/292)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[-20262.84720067   5551.41090784   1188.47368879 -35156.54026573
 -25404.76574525 -23595.02314639]
R square - 0.2838137093348295
Adjusted R square - 0.26909755267732605
RMSE - 43214.06927179952
```

In [9]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or
Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Per
cent Age 65 and Older'

model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Bl
ack, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent A
ge 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Per
cent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Pe
rcent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R square -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/293)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[70639.67384795  1832.13412456 -9896.9238706  -5575.62902898
 -3160.18431924]
R square - 0.9461354693637971
Adjusted R square - 0.9452162794211998
RMSE - 12978.539138584476
```

In [10]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree'

model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R square -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[70692.75301251  1827.68857508 -9335.76053975]
R square - 0.950506110643013
Adjusted R square - 0.9500027829546369
RMSE - 12456.892528655851
```

LASSO regression model - Predicting number of votes from Democratic party in each county

In [11]:
```
#with all the predictors

model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[ 69224.71479124  -3195.33996565  -1013.63916087  -6917.77376216
   3975.00309549    192.59502461  -5290.27001162  -1846.83971098
   1471.58775101   1467.72300999   4030.09531822 -10515.05282676
   -155.56176752]
R sqaure - 0.9338579590814098
Adjusted R square - 0.9308409537061758
RMSE - 14768.885350551014
```

In [12]:
```
# we choose the variables - 'Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/292)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[-20265.34792665   5549.70445054   1184.57925331 -35155.11759985
 -25398.23706304 -23589.3889991 ]
R sqaure - 0.28382122966731393
Adjusted R square - 0.26910522753719024
RMSE - 43212.16547566386
```

In [13]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/293)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[70639.59231518  1831.64211856 -9895.93640409 -5569.81209394
 -3154.74390006]
R sqaure - 0.9461445927509485
Adjusted R square - 0.9452255584975517
RMSE - 12976.947943119712
```

In [14]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or
Latino','Percent Less than Bachelor\'s Degree'

model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Bl
ack, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_t
rain['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Per
cent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[70692.50886821  1826.8889712  -9334.93014866]
R sqaure - 0.9505081178945095
Adjusted R square - 0.9500048106188604
RMSE - 12456.252078729762
```

Ridge regression model - Predicting number of votes from Democratic party in each county

In [15]:
```
#with all the predictors

model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[ 69103.28485379  -3156.22864837   -971.82311815  -6882.24436148
   4045.25218998    204.80424021  -5288.7637525   -1853.97190828
   1469.8517414    1478.13208619   3972.27630871 -10485.34826069
   -172.45175812]
R sqaure - 0.9337099729670875
Adjusted R square - 0.9306862173480424
RMSE - 14765.278891644848
```

In [16]:
```
# we choose the variables - 'Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/292)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[-20149.15991102   5611.76554918   1241.33741276 -35114.35347907
 -25217.71256808 -23443.60441099]
R sqaure - 0.28385560030907786
Adjusted R square - 0.2691403044250179
RMSE - 43167.957328237964
```

In [17]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/293)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[70546.19110868  1849.51351113 -9915.97783396 -5559.0069844
 -3160.47101088]
R sqaure - 0.9460531895143045
Adjusted R square - 0.9451325954787124
RMSE - 12961.358434277668
```

In [18]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree'

model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[70600.49484247  1846.50696155 -9358.71285411]
R sqaure - 0.9504166542075589
Adjusted R square - 0.9499124167927205
RMSE - 12440.758359506015
```

Elastic Net regression model - Predicting number of votes from Democratic party in each county

In [19]:
```
#with all the predictors

model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[41480.09632506 -1983.76861711  3594.19472878 -2314.59694751
  9774.47950057  2016.54392298 -3280.56072649 -1560.96655116
  2979.50479741  2172.37968958 -1723.49591663 -7979.8874114
 -4479.4847173 ]
R sqaure - 0.848768905919109
Adjusted R square - 0.841870645487349
RMSE - 17108.644664068634
```

In [20]:
```
# we choose the variables - 'Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df[['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/292)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[ -9958.76050958   8161.23002439   2913.67806086 -22910.95625786
  -3995.42519405  -7979.19493997]
R sqaure - 0.2816662681494376
Adjusted R square - 0.26690598598812476
RMSE - 36869.909786470125
```

In [21]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older'

model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree','Percent Age 29 and Under','Percent Age 65 and Older']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/293)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))
print('RMSE -',rmse)
```

```
[ 45426.16318698   4875.16941789 -12110.66178732  -1517.11978518
  -3704.77349815]
R sqaure - 0.8948969412272033
Adjusted R square - 0.8931033736713535
RMSE - 14219.205682751672
```

In [22]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree'

model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Democratic'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Democratic'], predicted))

print('RMSE -',rmse)
```

```
[ 45805.86273082   5176.16842415 -12339.67185178]
R sqaure - 0.9014973962488838
Adjusted R square - 0.9004956748548046
RMSE - 13929.489712866985
```

Result of prediction made by different regression models:

1. The best regression model to predict number of votes for Democratic party in each county is **LASSO Regression model**
2. The best predictors are **'Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'**

Multi-linear regression model - predicting number of votes from Republican party in each county

In [23]:
```
#Build a linear regression model to predict the number of votes cast for the R
epublican party in each county
#we choose all the variables
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45467.5097118   1769.95034533 -3141.4206375   1167.17323402
 -6463.65917143 -1121.73432851  -955.67013341  2580.74056065
  5910.97457236  2037.10575397  3530.42010898 -3156.11275644
 -5992.05181735]
R sqaure - 0.7239014362949736
Adjusted R square - 0.7113074667224637
RMSE - 15962.431310602105
```

In [24]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or
Latino','Percent Less than Bachelor\'s Degree'
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Bl
ack, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_t
rain['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Per
cent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))

print('RMSE -',rmse)
```

```
[70692.75301251  1827.68857508 -9335.76053975]
R sqaure - 0.6819847663986087
Adjusted R square - 0.6787507131755437
RMSE - 28909.245900355312
```

In [25]:
```
# we choose the variables - 'Total Population', 'Percent White, not Hispanic o
r Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age
 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rura
l','Percent Less than Bachelor\'s Degree'
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent W
hite, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign
 Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household In
come', 'Percent Rural','Percent Less than Bachelor\'s Degree']], y = x_train[
'Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Pe
rcent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent F
oreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Househ
old Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/289)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45023.1971947   4759.34637858  4407.16635812 -4978.7068913
  2485.17242773  2310.56734499  5528.01016664 -4817.64186136
 -1288.35179284]
R sqaure - 0.7310110689906854
Adjusted R square - 0.7226342510699801
RMSE - 15711.484101428712
```

In [26]:
```
#we choose variables - 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/290)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45133.5738712   4612.72460625  3998.62967731 -4790.68208843
  2692.84982155  2174.86528205  6130.35899569 -5297.8335129 ]
R sqaure - 0.7302080671530998
Adjusted R square - 0.7227655310745646
RMSE - 15749.245925443487
```

LASSO model - predicting number of votes from Republican party in each county

In [27]:
```
#with all the predictors
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45464.11625996  1763.84615535 -3141.51363944  1160.39910811
 -6454.91877737 -1119.19972956  -956.20034133  2577.09105238
  5906.62715265  2034.44712921  3523.56962737 -3151.08771664
 -5989.09353181]
R sqaure - 0.72388866630169
Adjusted R square - 0.7112941142382583
RMSE - 15962.567869419841
```

In [28]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree'
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))

print('RMSE -',rmse)
```

```
[70692.50886821  1826.8889712  -9334.93014866]
R sqaure - 0.6819862574760841
Adjusted R square - 0.6787522194165189
RMSE - 28908.58533472568
```

In [29]:
```
# we choose the variables - 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree'
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/289)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45020.63216815  4754.58652151  4399.13619837 -4972.63355335
  2483.38143536  2307.57325499  5525.26233565 -4816.12817309
 -1286.9267177 ]
R sqaure - 0.7309918999451649
Adjusted R square - 0.7226144850645645
RMSE - 15711.861460828213
```

In [30]:

```
#we choose variables - 'Total Population', 'Percent White, not Hispanic or Lat
ino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 an
d Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent W
hite, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign
 Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household In
come', 'Percent Rural']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Pe
rcent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent F
oreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Househ
old Income', 'Percent Rural']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/290)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45130.8653894   4608.08879379  3990.979774   -4784.78595721
  2690.83168918  2172.00928412  6126.93887566 -5295.79893901]
R sqaure - 0.7301892853867069
Adjusted R square - 0.7227462311904782
RMSE - 15749.597875361354
```

Ridge model - predicting number of votes from Republican party in each county

In [31]:

```
#with all the predictors
model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45372.43153194  1768.56879541 -3122.95059686  1154.57630992
 -6373.37506063 -1105.98502965  -985.3332241   2534.8701695
  5871.5568462   2034.41527943  3478.81492474 -3150.05692091
 -5980.42156648]
R sqaure - 0.7237763439914228
Adjusted R square - 0.7111766684541894
RMSE - 15961.471842340876
```

In [32]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree'
model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))

print('RMSE -',rmse)
```

```
[70600.49484247  1846.50696155 -9358.71285411]
R sqaure - 0.681928137249801
Adjusted R square - 0.6786935081370871
RMSE - 28880.59100368566
```

In [33]:
```
# we choose the variables - 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree'
model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/289)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[44936.46610876  4728.4999488   4348.4867791  -4910.19682192
  2477.65013056  2302.29935721  5507.53909423 -4819.27591403
 -1300.52648883]
R sqaure - 0.730866955276649
Adjusted R square - 0.7224856493856104
RMSE - 15711.621797161388
```

In [34]:
```
#we choose variables - 'Total Population', 'Percent White, not Hispanic or Lat
ino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 an
d Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
model = linear_model.Ridge(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent W
hite, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign
 Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household In
come', 'Percent Rural']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Pe
rcent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent F
oreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Househ
old Income', 'Percent Rural']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/290)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[45048.37114874  4581.97094435  3937.75363896 -4720.76479082
  2686.25742648  2165.51351516  6114.94401424 -5303.18486375]
R sqaure - 0.7300650510890524
Adjusted R square - 0.7226185697397849
RMSE - 15749.227555025664
```

Elastic Net model - predicting number of votes from Republican party in each county

In [35]:
```
#with all the predictors
model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df)
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/285)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[26093.3984422    177.12756605   -64.5701929   -148.7604211
  3165.10347916   945.0099141  -1759.96891039  -586.77309116
  2980.09223117  1588.76773382 -1237.52311144 -3574.03374256
 -5039.75691365]
R sqaure - 0.6536767052951091
Adjusted R square - 0.6378795023787457
RMSE - 17292.42585739893
```

In [36]:
```
# we choose the variables - 'Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree'
model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/295)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))

print('RMSE -',rmse)
```

```
[ 45805.86273082   5176.16842415 -12339.67185178]
R sqaure - 0.6483971210199565
Adjusted R square - 0.6448214985218543
RMSE - 21887.40081895172
```

In [37]:
```
# we choose the variables - 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree'
model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural','Percent Less than Bachelor\'s Degree']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/289)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[26280.07044573   631.61807456  -593.30116942  2928.22024787
   420.7892185   1549.69419897  3332.4892813  -5017.82103997
 -3955.38246393]
R sqaure - 0.6558387428820472
Adjusted R square - 0.6451209182659172
RMSE - 17225.520205220193
```

In [38]:
```
#we choose variables - 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
model = linear_model.ElasticNet(alpha = 1, l1_ratio = 0.5)
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']], y = x_train['Republican'])
print(fitted_model.coef_)
predicted = fitted_model.predict(x_validate_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
corr_coef = np.corrcoef(predicted,x_valid['Republican'])[1, 0]
R_squared = corr_coef ** 2
print("R sqaure -",R_squared)
adjusted_r = 1 - (((1-R_squared)*(298))/290)
print("Adjusted R square -",adjusted_r)
rmse = math.sqrt(mean_squared_error(x_valid['Republican'], predicted))
print('RMSE -',rmse)
```

```
[26746.38770763   651.43975159 -1131.21848954  3326.34780023
   501.6337263   1190.67039685  4606.77711254 -5939.22384719]
R sqaure - 0.6665869824704843
Adjusted R square - 0.6573893819869114
RMSE - 16963.393423905003
```

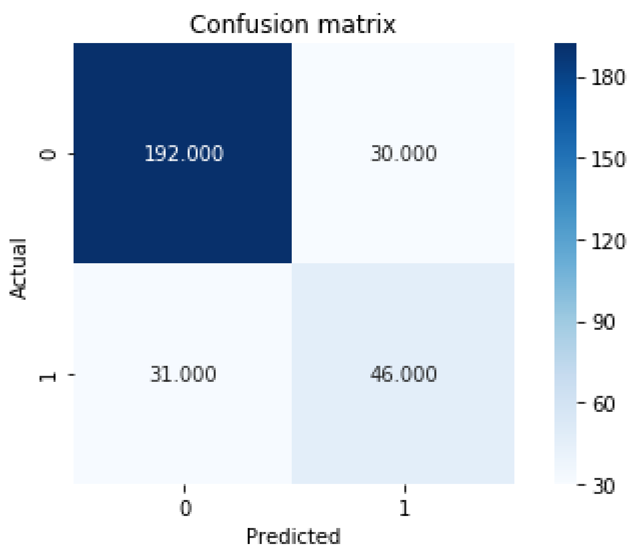
Result of prediction made by different regression models:

1. The best regression model to predict number of votes for Democratic party in each county is **multi-linear Regression model**
2. The best predictors are **'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'**

In [39]:
```
#For Decicsion Tree First Try

classifier = DecisionTreeClassifier(criterion="gini",random_state = 0,splitter='best',min_samples_leaf=2,min_samples_split=3)
classifier.fit(x_train_scaled_df.loc[:,['Median Household Income','Percent Age 29 and Under','Percent Age 65 and Older','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']],y_train)
```

Out[39]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=3,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [40]:
```
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Median Household Inco
me','Percent Age 29 and Under','Percent Age 65 and Older','Percent White, not
 Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



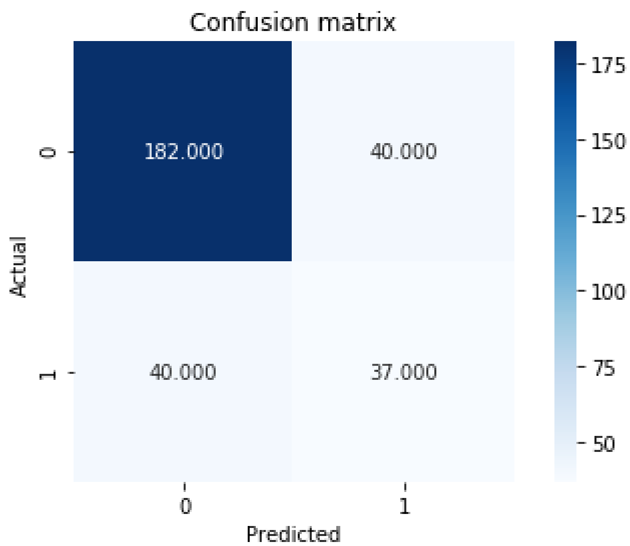
In [41]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7959866220735786, 0.20401337792642138, array([0.86098655, 0.60526316]), ar
ray([0.86486486, 0.5974026 ]), array([0.86292135, 0.60130719])]
```

In [42]:
```
#For Decicsion Tree(Trying For Every varaible) without any parameters

classifier = DecisionTreeClassifier(criterion="gini",random_state = 0,splitter
='random',presort=True)
classifier.fit(x_train_scaled_df,y_train)
```

Out[42]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=True,
                       random_state=0, splitter='random')
```

In [43]:
```
y_pred = classifier.predict(x_validate_scaled_df)
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



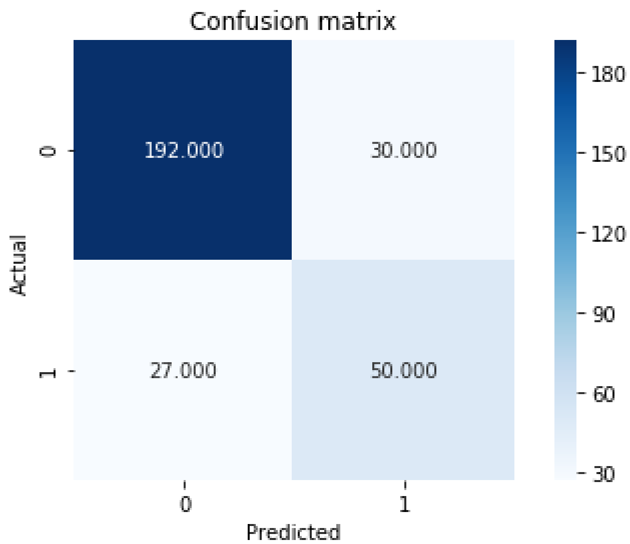
In [44]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7324414715719063, 0.2675585284280937, array([0.81981982, 0.48051948]), arr
ay([0.81981982, 0.48051948]), array([0.81981982, 0.48051948])]
```

In [45]:
```
#For Decicsion Tree 3rd Try (Best Model Till now)

classifier = DecisionTreeClassifier(criterion="entropy",random_state = 0,split
ter='best',min_samples_leaf=2)
classifier.fit(x_train_scaled_df.loc[:,['Percent Hispanic or Latino','Percent
 Black, not Hispanic or Latino','Percent White, not Hispanic or Latino','Perce
nt Less than Bachelor\'s Degree']],y_train)
```

Out[45]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [46]:
```
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Percent Hispanic or L
atino','Percent Black, not Hispanic or Latino','Percent White, not Hispanic or
Latino','Percent Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



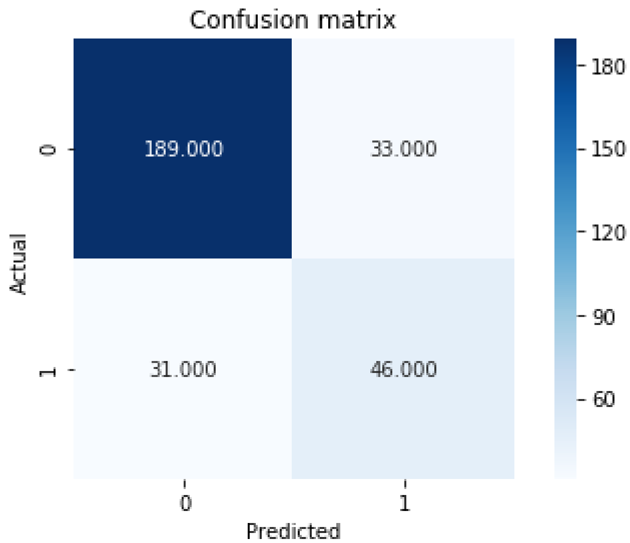
In [47]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.8093645484949833, 0.1906354515050167, array([0.87671233, 0.625     ]), arr
ay([0.86486486, 0.64935065]), array([0.8707483 , 0.63694268])]
```

In [48]:
```
#For Decicsion Tree 4th Try

classifier = DecisionTreeClassifier(criterion="gini",random_state = 0,splitter
='best',min_samples_leaf=2)
classifier.fit(x_train_scaled_df.loc[:,['Percent White, not Hispanic or Latin
o','Percent Less than Bachelor\'s Degree']],y_train)
```

Out[48]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [49]:
```
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Percent White, not Hi
spanic or Latino','Percent Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



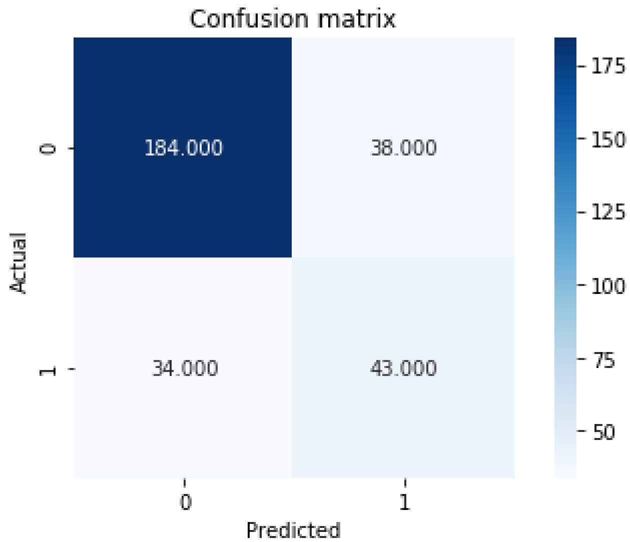
In [50]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7859531772575251, 0.21404682274247488, array([0.85909091, 0.58227848]), ar
ray([0.85135135, 0.5974026 ]), array([0.85520362, 0.58974359])]
```

In [51]:
```
#For Decicsion Tree 5th Try

classifier = DecisionTreeClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train_scaled_df,y_train)
```

Out[51]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

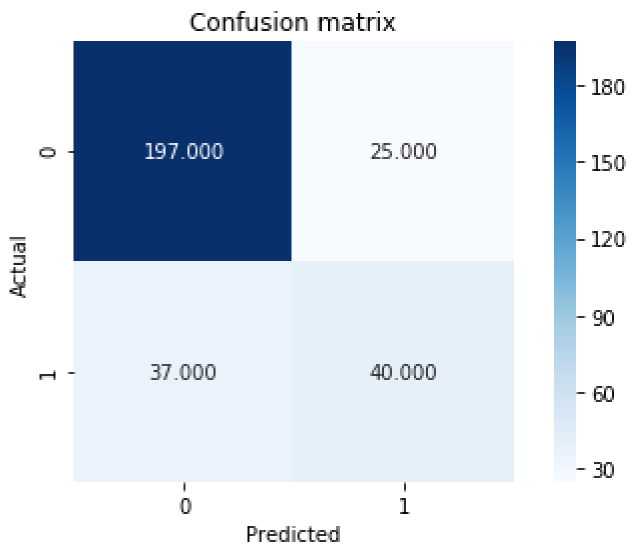
In [52]:
```
y_pred = classifier.predict(x_validate_scaled_df)
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [53]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7591973244147158, 0.24080267558528423, array([0.8440367, 0.5308642]), arra
y([0.82882883, 0.55844156]), array([0.83636364, 0.5443038 ])]
```

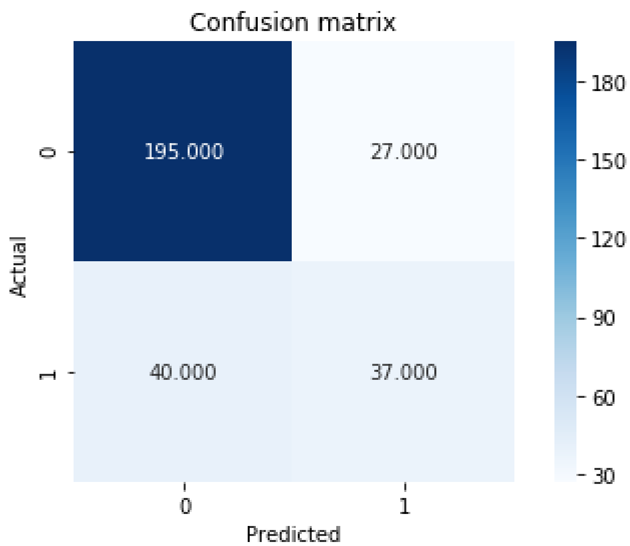
In [54]:
```
#First Try
classifier = GaussianNB()
classifier.fit(x_train_scaled_df.loc[:,['Percent Age 29 and Under','Percent Ag
e 65 and Older','Percent White, not Hispanic or Latino','Percent Less than Bac
helor\'s Degree']],y_train)
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Percent Age 29 and Un
der','Percent Age 65 and Older','Percent White, not Hispanic or Latino','Perce
nt Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [55]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7926421404682275, 0.20735785953177255, array([0.84188034, 0.61538462]), ar
ray([0.88738739, 0.51948052]), array([0.86403509, 0.56338028])]
```

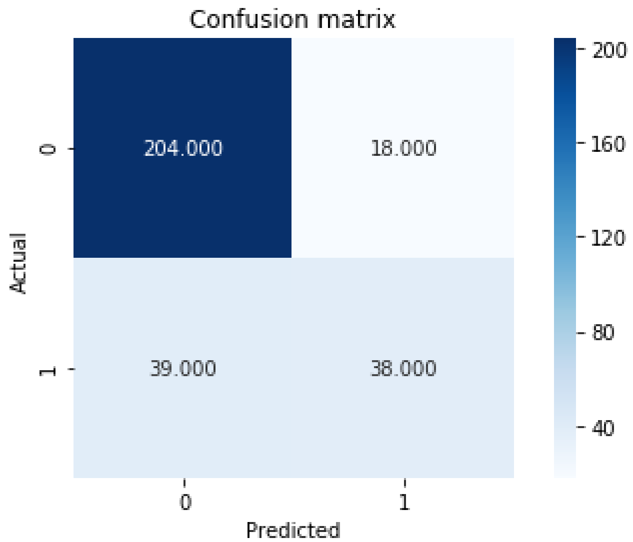
In [56]:
```
#Second Try
classifier = GaussianNB()
classifier.fit(x_train_scaled_df.loc[:,['Median Household Income','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']],y_train)
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Median Household Income','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [57]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7759197324414716, 0.2240802675585284, array([0.82978723, 0.578125  ]), array([0.87837838, 0.48051948]), array([0.85339168, 0.5248227 ])]
```

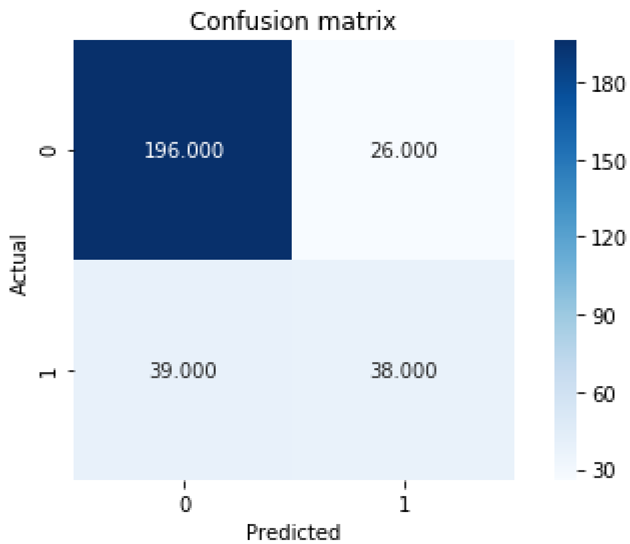
In [58]:
```
#Third Try
classifier = GaussianNB(var_smoothing=2e-09)
classifier.fit(x_train_scaled_df.loc[:,['Total Population','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']],y_train)
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Total Population','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [59]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.8093645484949833, 0.1906354515050167, array([0.83950617, 0.67857143]), array([0.91891892, 0.49350649]), array([0.87741935, 0.57142857])]
```

In [60]:
```
#Fourth Try
classifier = GaussianNB(var_smoothing=3e-09)
classifier.fit(x_train_scaled_df.loc[:,['Percent White, not Hispanic or Latin
o','Percent Less than Bachelor\'s Degree']],y_train)
y_pred = classifier.predict(x_validate_scaled_df.loc[:,['Percent White, not Hi
spanic or Latino','Percent Less than Bachelor\'s Degree']])
conf_matrix = metrics.confusion_matrix(y_validate,y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [61]:
```
accuracy = metrics.accuracy_score(y_validate,y_pred)
error = 1 - metrics.accuracy_score(y_validate,y_pred)
precision = metrics.precision_score(y_validate,y_pred, average = None)
recall = metrics.recall_score(y_validate,y_pred, average = None)
F1_score = metrics.f1_score(y_validate,y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.782608695652174, 0.21739130434782605, array([0.83404255, 0.59375   ]), arr
ay([0.88288288, 0.49350649]), array([0.85776805, 0.53900709])]
```

TASK 05

Build a clustering model to cluster the counties

In [62]:
```
data_cluster = merged.iloc[:,3:-3]
data_cluster.head()
```

Out[62]:

	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	Percent Age 65 and Older	M Hous In
0	72346	18.571863	0.486551	5.947806	1.719515	50.598513	45.854643	13.322091	
1	128177	56.299492	3.714395	34.403208	11.458374	49.069646	37.902276	19.756275	
2	138064	54.619597	1.342855	13.711033	4.825298	50.581614	48.946141	10.873943	
3	53179	63.222325	0.552850	18.548675	4.249798	50.296170	32.238290	26.397638	
4	37529	51.461536	1.811932	32.097844	4.385942	46.313518	46.393456	12.315809	

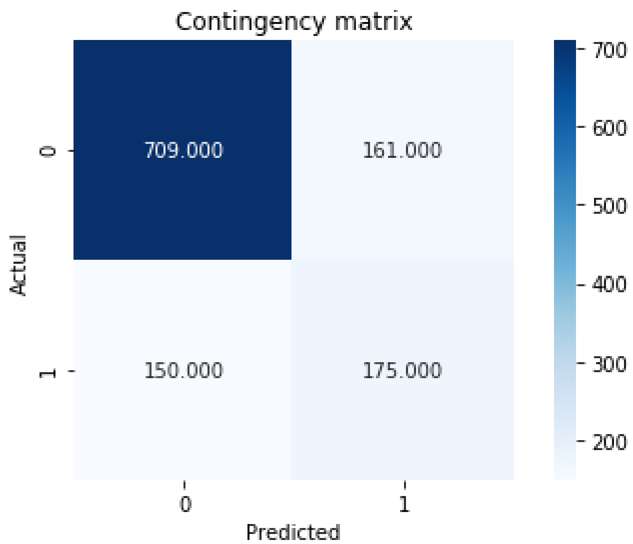
In [63]:
```
scaler_cluster = StandardScaler()

scaler_cluster.fit(data_cluster)
data_cluster_scaled = scaler_cluster.transform(data_cluster)
data_cluster_scaled_df=pd.DataFrame(data_cluster_scaled,index=data_cluster.ind
ex,columns=data_cluster.columns)
data_cluster_scaled_df.head()
```

Out[63]:

	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	Percent Age 65 and Older	Me House Inc
0	-0.152853	-3.066408	-0.546700	-0.289050	-0.553790	0.320582	1.620631	-1.006762	-1.43
1	0.022271	-1.155992	-0.199124	1.514068	1.052585	-0.322286	0.205758	0.345583	-0.38
2	0.053284	-1.241056	-0.454493	0.202878	-0.041507	0.313476	2.170666	-1.521317	0.08
3	-0.212974	-0.805440	-0.539561	0.509422	-0.136432	0.193451	-0.801971	1.741474	-0.77
4	-0.262063	-1.400971	-0.403982	1.367985	-0.113976	-1.481202	1.716496	-1.218263	-0.21

K-Means clustering --- Method : random --- n_init : 10

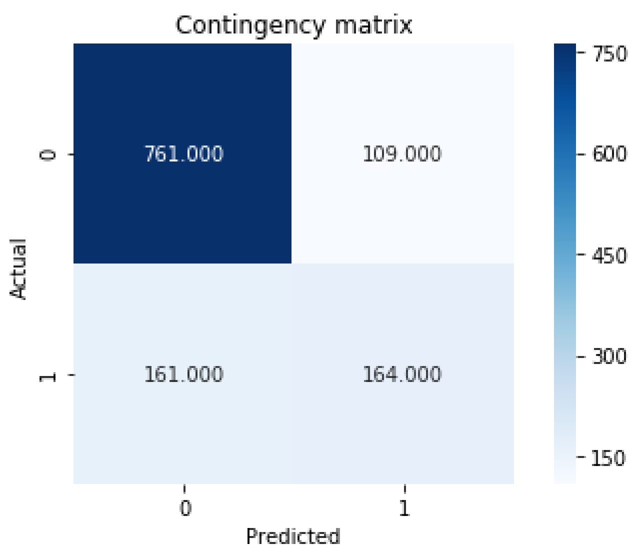
In [64]:
```
#we choose all the variables
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, random_state
= 0).fit(data_cluster_scaled_df)
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [65]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df, clus
ters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.19751656022671712
Silhouette Coefficient -  0.30700290833697047
```

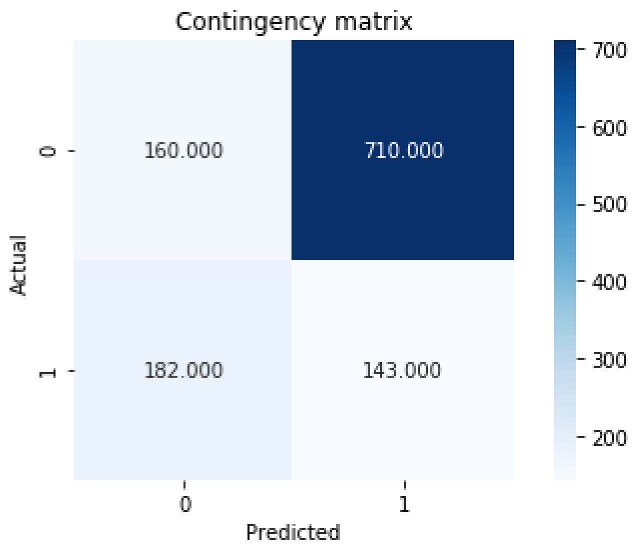
In [66]:
```
# we choose variables - 'Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, random_state = 0).fit(data_cluster_scaled_df[['Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [67]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.25433770859397
Silhouette Coefficient -  0.33450378634601924
```

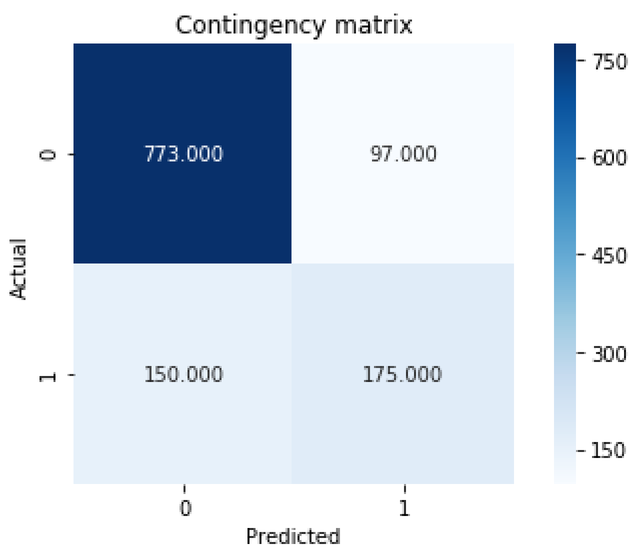
In [68]:
```
# we choose variables - 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, random_state = 0).fit(data_cluster_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [69]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.21256200142203743
Silhouette Coefficient -  0.3423898853017791
```

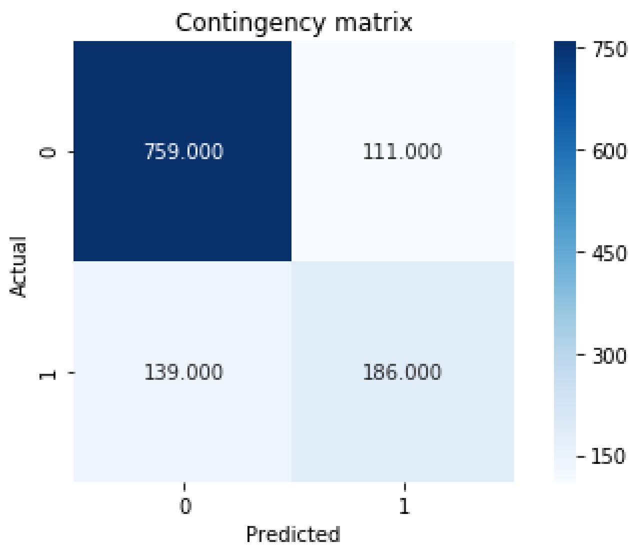
In [70]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, random_state = 0).fit(data_cluster_scaled_df[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [71]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.30061428945310187
Silhouette Coefficient -  0.3250057188451443
```

In [72]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age
 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed',
 'Percent Female'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, random_state
= 0).fit(data_cluster_scaled_df[['Percent Black, not Hispanic or Latino', 'Per
cent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemp
loyed', 'Percent Female']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

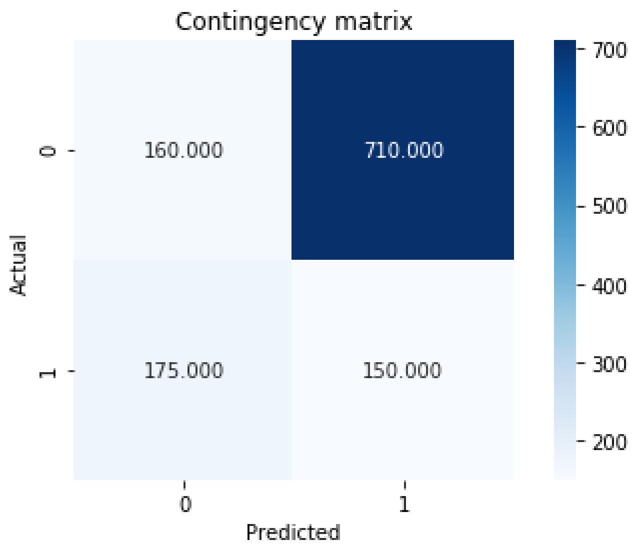


In [73]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Per
cent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less
than Bachelor\'s Degree', 'Percent Unemployed', 'Percent Female']], clusters,
metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.301132997830686
Silhouette Coefficient -  0.2596874751140196
```

K-Means clustering --- Method : random --- n_init : 3

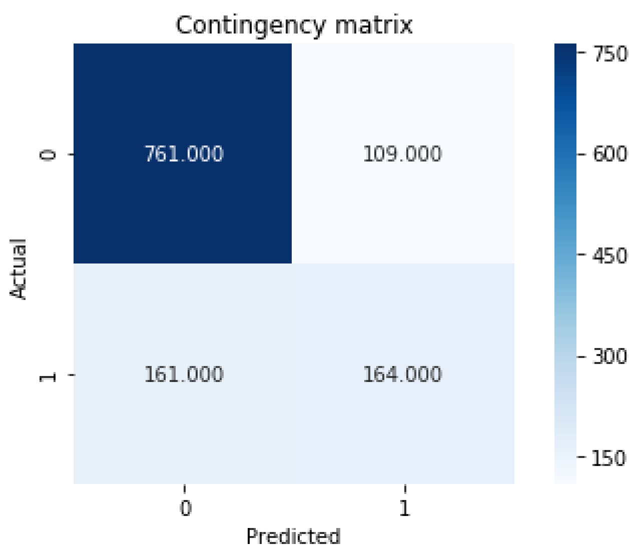
In [74]:
```
#we choose all the variables
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 3, random_state
= 0).fit(data_cluster_scaled_df)
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [75]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df, clus
ters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.19893799165776016
Silhouette Coefficient -  0.30691597445230206
```

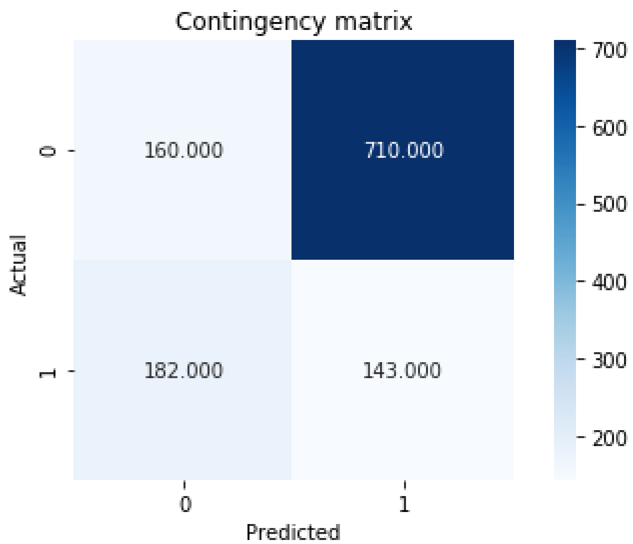
In [76]:
```
# we choose variables - 'Total Population', 'Percent Black, not Hispanic or La
tino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Pe
rcent Unemployed', 'Median Household Income', 'Percent Rural'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 3, random_state
= 0).fit(data_cluster_scaled_df[['Total Population', 'Percent Black, not Hispa
nic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Deg
ree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [77]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Tot
al Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and O
lder', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median H
ousehold Income', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.25433770859397
Silhouette Coefficient -  0.33450378634601924
```

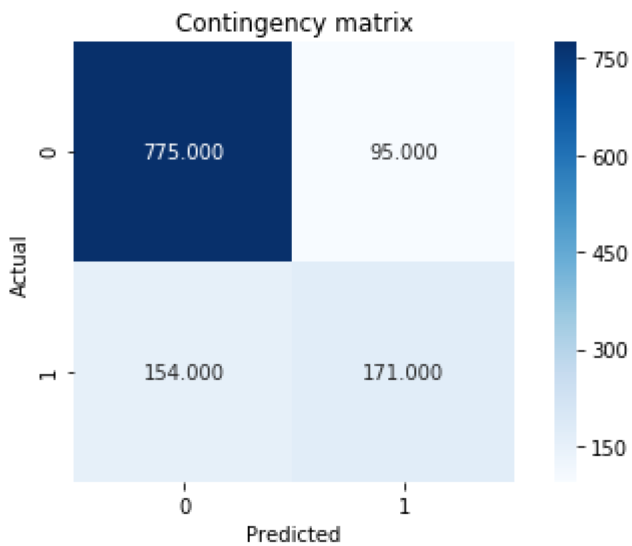
In [78]:
```
# we choose variables - 'Total Population', 'Percent White, not Hispanic or La
tino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 a
nd Under', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree',
'Percent Unemployed', 'Median Household Income', 'Percent Rural'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 3, random_state
= 0).fit(data_cluster_scaled_df[['Total Population', 'Percent White, not Hispa
nic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent
Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s
 Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [79]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Tot
al Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or
 Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65
 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Me
dian Household Income', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.21256200142203743
Silhouette Coefficient -  0.3423898853017791
```

In [80]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 3, random_state = 0).fit(data_cluster_scaled_df[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



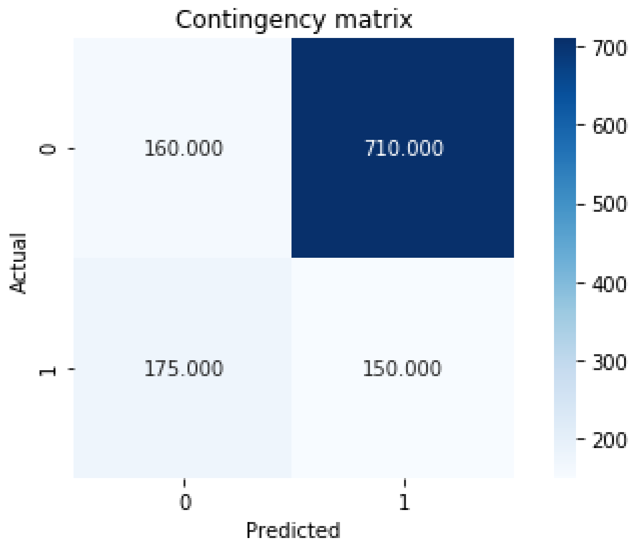
In [81]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.2947169759645422
Silhouette Coefficient -  0.32745694983411816
```

K-means clustering --- Method : k-means++ --- n_init = 10

In [82]:

```
#we choose all the variables
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 10, random_st
ate = 0).fit(data_cluster_scaled_df)
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

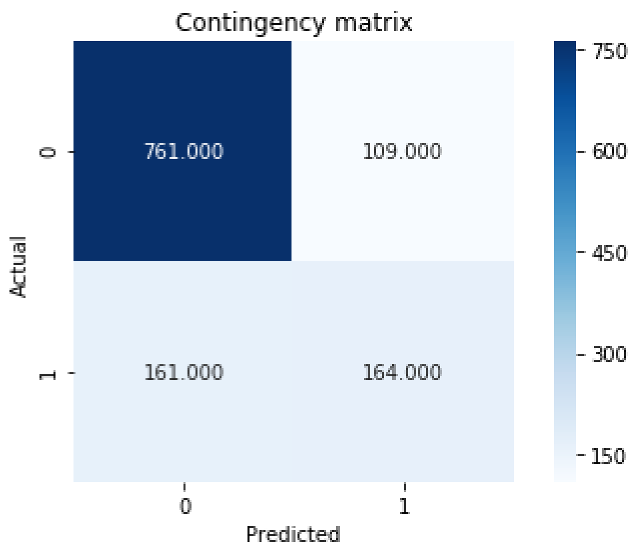


In [83]:

```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df, clus
ters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.19893799165776016
Silhouette Coefficient -  0.30691597445230206
```

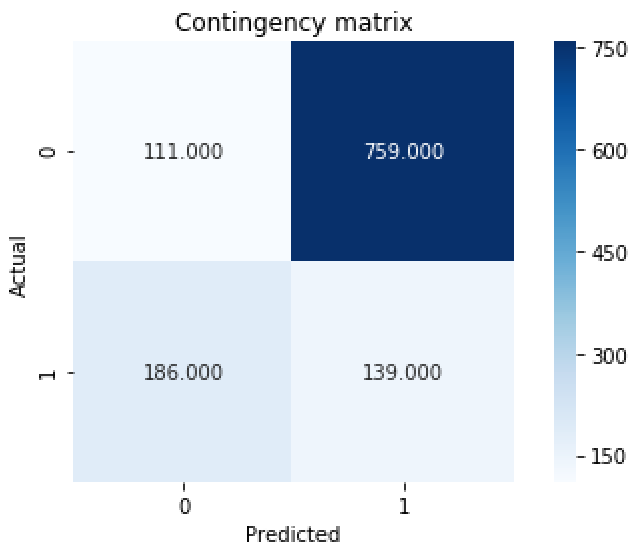
In [84]:
```
#we choose variables - 'Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural'
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 10, random_state = 0).fit(data_cluster_scaled_df[['Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [85]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Age 65 and Older', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.25433770859397
Silhouette Coefficient -  0.33450378634601924
```

In [86]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age
 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 10, random_st
ate = 0).fit(data_cluster_scaled_df[['Percent Black, not Hispanic or Latino',
'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent U
nemployed', 'Percent Female']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

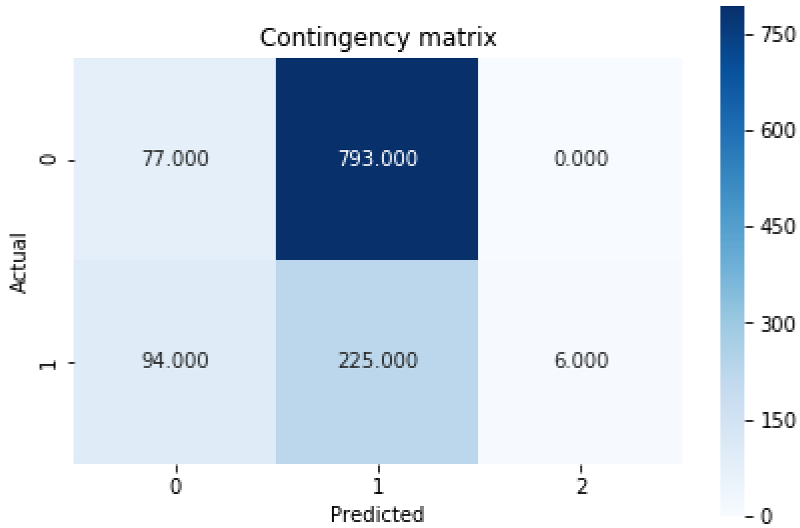


In [87]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Per
cent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less
than Bachelor\'s Degree', 'Percent Unemployed', 'Percent Female']], clusters,
metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.301132997830686
Silhouette Coefficient -  0.2596874751140196
```

DBSCAN clustering --- EPS = 2 --- min_samples = 5

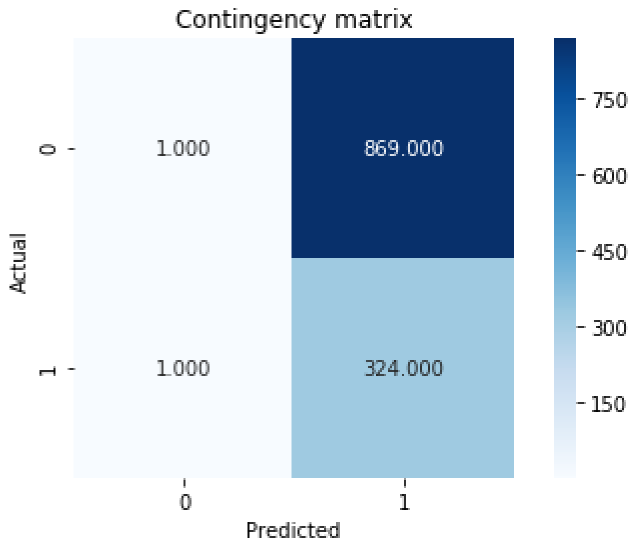
In [88]:
```
# we choose all the variables
clustering = DBSCAN(eps = 2, min_samples = 5, metric = "euclidean").fit(data_c
luster_scaled_df)
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [89]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df, clus
ters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.1577583255755803
Silhouette Coefficient -  0.31265113101181913
```

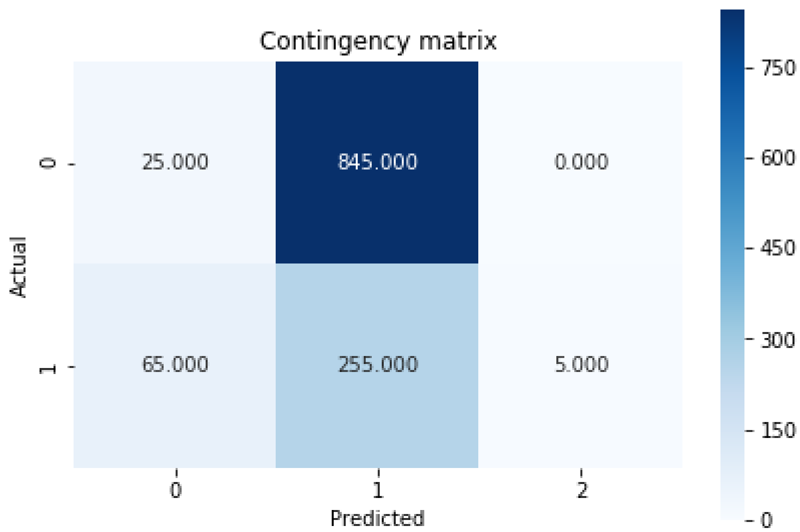
In [90]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age
 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = DBSCAN(eps = 2, min_samples = 5, metric = "euclidean").fit(data_c
luster_scaled_df[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and
Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [91]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Per
cent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less
than Bachelor\'s Degree', 'Percent Unemployed']], clusters, metric = "euclidea
n")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.0017465403807912195
Silhouette Coefficient -  0.5703181363565738
```

In [92]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age
 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = DBSCAN(eps = 2, min_samples = 5, metric = "euclidean").fit(data_c
luster_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino',
'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percen
t Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than High Schoo
l Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Income',
'Percent Unemployed', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



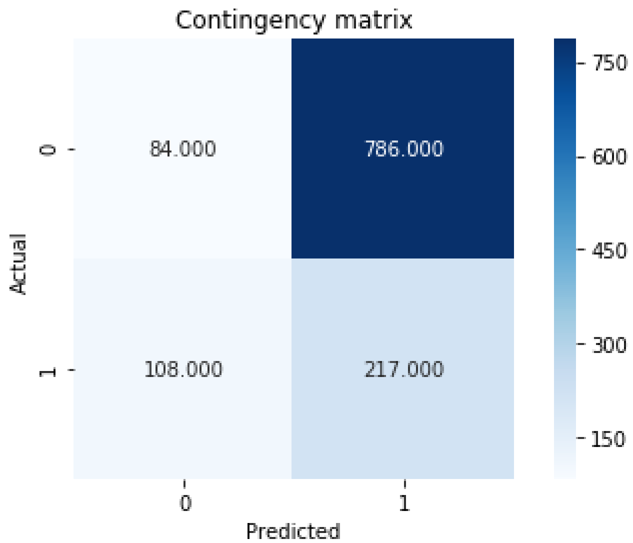
In [93]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Tot
al Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not H
ispanic or Latino', 'Percent Hispanic or Latino', 'Percent Age 29 and Under',
'Percent Age 65 and Older', 'Percent Less than High School Degree', 'Percent L
ess than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed',
'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.15799063145639403
Silhouette Coefficient -  0.252796660438697
```

DBSCAN clustering --- EPS = 2 --- min_samples = 7

In [94]:

```
# we choose all the variables
clustering = DBSCAN(eps = 2, min_samples = 7, metric = "euclidean").fit(data_c
luster_scaled_df)
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

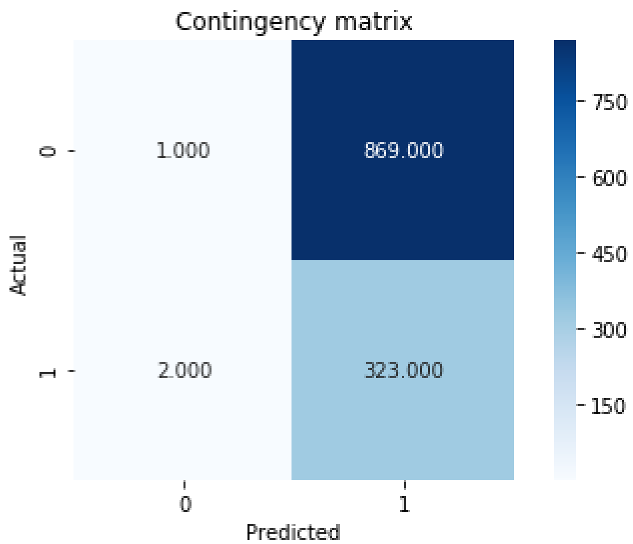


In [95]:

```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df, clus
ters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.16605899668385218
Silhouette Coefficient -  0.3542397525547686
```

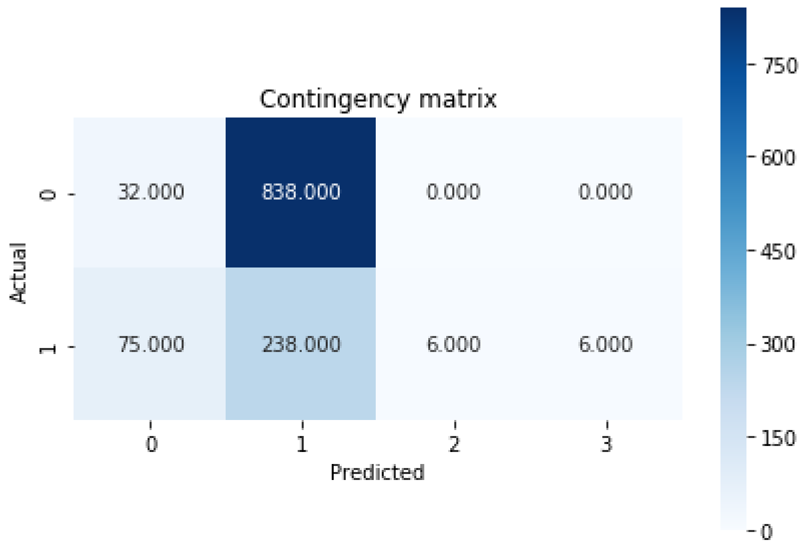
In [96]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age
 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = DBSCAN(eps = 2, min_samples = 7, metric = "euclidean").fit(data_c
luster_scaled_df[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and
Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [97]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Per
cent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less
than Bachelor\'s Degree', 'Percent Unemployed']], clusters, metric = "euclidea
n")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.0045413206116124245
Silhouette Coefficient -  0.6145052824393595
```

In [98]:
```
# we choose variables - 'Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Less than Bachelor\'s Degree', 'Percent Unemployed'
clustering = DBSCAN(eps = 2, min_samples = 7, metric = "euclidean").fit(data_cluster_scaled_df[['Total Population', '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 Older', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

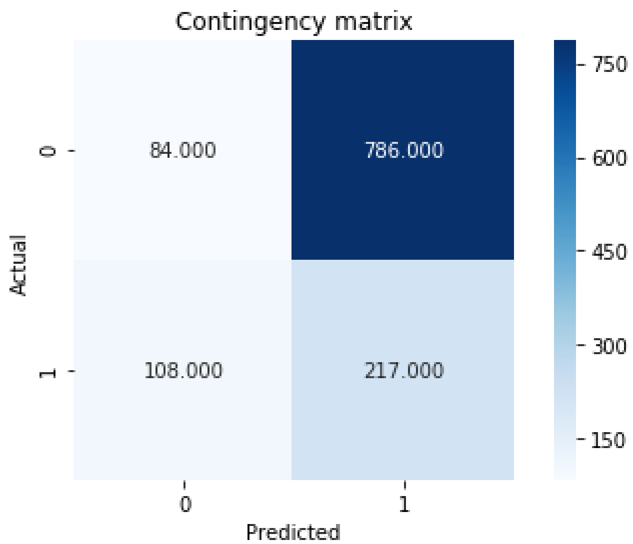


In [99]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.19145663513159158
Silhouette Coefficient -  0.2777449524833404
```

DBSCAN clustering --- EPS = 1.8 --- min_samples = 5

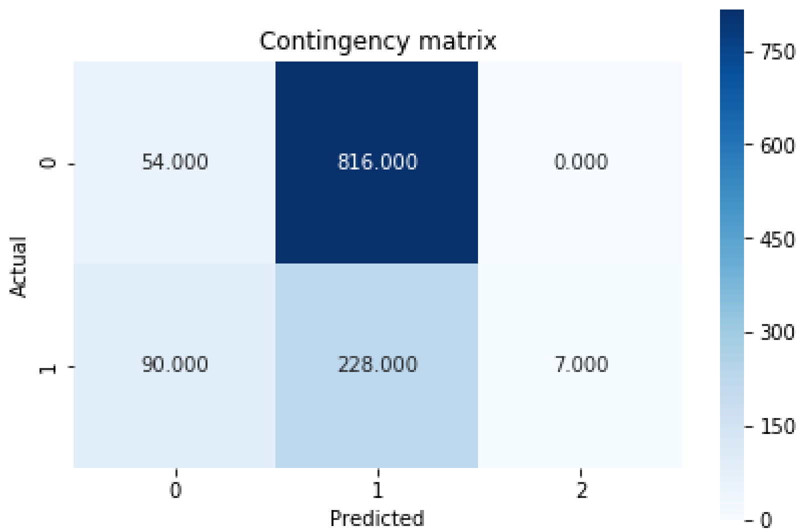
In [100]:
```
# we choose all the variables
clustering = DBSCAN(eps = 2, min_samples = 7, metric = "euclidean").fit(data_c
luster_scaled_df)
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [101]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df, clus
ters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.16605899668385218
Silhouette Coefficient -  0.3542397525547686
```

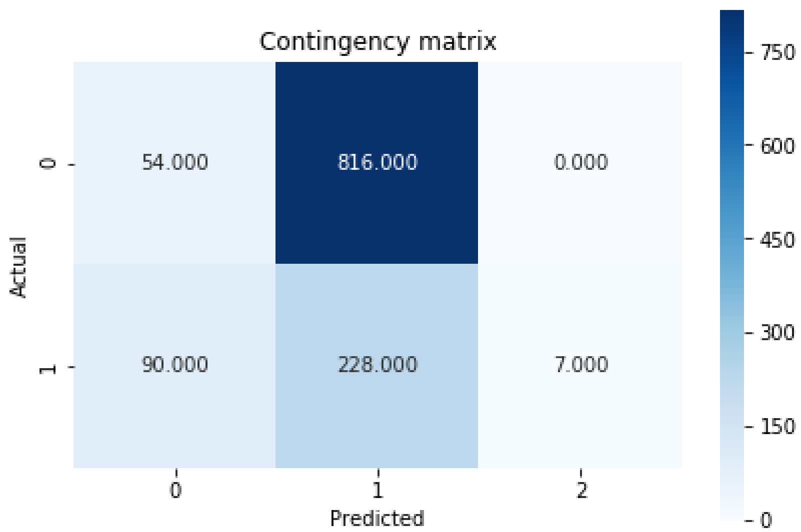
In [102]:
```
# we choose variables - 'Total Population', 'Percent White, not Hispanic or La
tino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino',
 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than Hi
gh School Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household I
ncome', 'Percent Unemployed', 'Percent Rural'
clustering = DBSCAN(eps = 1.8, min_samples = 5, metric = "euclidean").fit(data
_cluster_scaled_df[['Total Population', 'Percent White, not Hispanic or Latin
o', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Pe
rcent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than High S
chool Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Incom
e', 'Percent Unemployed', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [103]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Tot
al Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not H
ispanic or Latino', 'Percent Hispanic or Latino', 'Percent Age 29 and Under',
'Percent Age 65 and Older', 'Percent Less than High School Degree', 'Percent L
ess than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed',
'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)

print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.18351664707906754
Silhouette Coefficient -  0.2712267179561382
```

In [104]:
```
# we choose variables - 'Total Population', '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 Older', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed', 'Percent Rural'
clustering = DBSCAN(eps = 1.8, min_samples = 5, metric = "euclidean").fit(data_cluster_scaled_df[['Total Population', '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 Older', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed', 'Percent Rural']])
clusters = clustering.labels_
cont_matrix = metrics.cluster.contingency_matrix(merged['Party'], clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [105]:
```
adjusted_rand_index = metrics.adjusted_rand_score(merged['Party'], clusters)
silhouette_coefficient = metrics.silhouette_score(data_cluster_scaled_df[['Total Population', '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 Older', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree', 'Median Household Income', 'Percent Unemployed', 'Percent Rural']], clusters, metric = "euclidean")
print('Adjusted Rand Index - ',adjusted_rand_index)
print('Silhouette Coefficient - ', silhouette_coefficient)
```

```
Adjusted Rand Index -  0.18351664707906754
Silhouette Coefficient -  0.2712267179561382
```

TASK 06

In [109]:
```
#Create a map of Democratic counties and Republican counties using the countie
s' FIPS codes

#we use Desicion tree classifier to predict whether a county belongs to Democr
atic or Republican

#Classifing using Decision tree
classifier_best = DecisionTreeClassifier(criterion="entropy",random_state = 0,
splitter='best',min_samples_leaf=2)
classifier_best.fit(x_train_scaled_df.loc[:,['Percent Hispanic or Latino','Per
cent Black, not Hispanic or Latino','Percent White, not Hispanic or Latino','P
ercent Less than Bachelor\'s Degree']],y_train)
pred_valid = classifier_best.predict(x_validate_scaled_df.loc[:,['Percent Hisp
anic or Latino','Percent Black, not Hispanic or Latino','Percent White, not Hi
spanic or Latino','Percent Less than Bachelor\'s Degree']])
x_valid['Party'] = pred_valid
pred_train = classifier_best.predict(x_train_scaled_df.loc[:,['Percent Hispani
c or Latino','Percent Black, not Hispanic or Latino','Percent White, not Hispa
nic or Latino','Percent Less than Bachelor\'s Degree']])
x_train['Party'] = pred_train
```

In [110]:
```
x_merged = pd.concat([x_train,x_valid])
x_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1195 entries, 589 to 2
Data columns (total 19 columns):
State                                    1195 non-null object
County                                   1195 non-null object
FIPS                                     1195 non-null int64
Total Population                         1195 non-null int64
Percent White, not Hispanic or Latino    1195 non-null float64
Percent Black, not Hispanic or Latino    1195 non-null float64
Percent Hispanic or Latino               1195 non-null float64
Percent Foreign Born                     1195 non-null float64
Percent Female                           1195 non-null float64
Percent Age 29 and Under                 1195 non-null float64
Percent Age 65 and Older                 1195 non-null float64
Median Household Income                  1195 non-null int64
Percent Unemployed                       1195 non-null float64
Percent Less than High School Degree     1195 non-null float64
Percent Less than Bachelor's Degree      1195 non-null float64
Percent Rural                            1195 non-null float64
Democratic                               1195 non-null int64
Republican                               1195 non-null int64
Party                                    1195 non-null int64
dtypes: float64(11), int64(6), object(2)
memory usage: 186.7+ KB
```

In [111]:

```
import plotly.figure_factory as ff
change_values = {1: 'Democratic', 0: 'Republican'}
x_merged['Party'] = x_merged['Party'].map(change_values)
fips = x_merged['FIPS']

values = x_merged['Party']

fig = ff.create_choropleth(fips=fips, values=values, legend_title='Counties',t
itle='Democratic and Republican Counties in the United States of America')
fig.layout.template = None
fig.show()
```

```
C:\Users\Bharath\Anaconda3\lib\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=True'.
```

TASK 07

Use your best performing regression and classification models to predict the number of votes cast for the Democratic party in each county, the number of votes cast for the Republican party in each county, and the party (Democratic or Republican) of each county for the test dataset

In [112]:

```
#Read the demographic test data
x_test = pd.read_csv('demographics_test.csv')
x_test.head()
```

Out[112]:

	State	County	FIPS	Total Population	Percent White, not Hispanic or Latino	Percent Black, not Hispanic or Latino	Percent Hispanic or Latino	Percent Foreign Born	Percent Female	P /
0	NV	eureka	32011	1730	98.265896	0.057803	0.462428	0.346821	51.156069	27.1
1	TX	zavala	48507	12107	5.798299	0.594697	93.326175	9.193029	49.723301	49.3
2	VA	king george	51099	25260	73.804434	16.722090	4.441805	2.505938	50.166271	40.1
3	OH	hamilton	39061	805965	66.354867	25.654340	2.890944	5.086945	51.870615	40.7
4	TX	austin	48015	29107	63.809393	8.479060	25.502456	9.946061	50.671660	37.3

In [113]:

```
#standardize the test data using training scalar
std_test = x_test.iloc[:,3:]
x_test_scaled = scaler.transform(std_test)
x_test_scaled_df=pd.DataFrame(x_test_scaled,index=std_test.index,columns=std_t
est.columns)
x_test_scaled_df.head()
```

Out[113]:

	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	Percent Age 65 and Older	Me House Inc
0	-0.366536	0.968807	-0.589177	-0.637532	-0.782941	0.551596	-1.653627	-0.516838	1.69
1	-0.336645	-3.772760	-0.532876	5.568120	0.673487	-0.050039	2.151757	-1.162674	-1.97
2	-0.298758	-0.285532	1.158300	-0.371609	-0.427467	0.135968	0.588604	-1.289054	2.90
3	1.950079	-0.667532	2.094968	-0.475246	-0.002533	0.851641	0.690395	-0.815382	0.03
4	-0.287676	-0.798059	0.293906	1.035776	0.797465	0.348187	0.102609	-0.063861	0.56

In [114]:
```
#predicting Democratic values using multi-linear regression model
model = linear_model.Lasso(alpha = 1)
fitted_model_demo = model.fit(X = x_train_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']], y = x_train['Democratic'])
predicted_demo = fitted_model_demo.predict(x_test_scaled_df[['Total Population','Percent Black, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
```

In [115]:
```
fitted_model_rep = model.fit(X = x_train_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']], y = x_train['Republican'])
predicted_rep = fitted_model_rep.predict(x_test_scaled_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rural']])
```

In [116]:
```
#Classifing using Decision tree
classifier = DecisionTreeClassifier(criterion="entropy",random_state = 0,splitter='best',min_samples_leaf=2)
classifier.fit(x_train_scaled_df.loc[:,['Percent Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']],y_train)
```

Out[116]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [117]:
```
pred_party = classifier.predict(x_test_scaled_df.loc[:,['Percent Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent White, not Hispanic or Latino','Percent Less than Bachelor\'s Degree']])
```

In [118]:
```
x_test['Democratic'] = predicted_demo
x_test['Republican'] = predicted_rep
x_test['Party'] = pred_party
```

In [121]:
```
result = x_test [['State','County', 'Democratic', 'Republican','Party']]
result['Democratic'][result['Democratic']<0] = 0
result['Republican'][result['Republican']<0] = 0
result['Party'][result['Democratic'] == result['Republican']] = 'NAN'
```

In [ ]:

In [122]:
```
result.to_csv("Task07_result.csv")
```

In [ ]: