CS 418: Project 2

Authors: Anusha Sagi, Fatima Kahack, Lydia Tse

Description: In this code, we will be utilizing regression, classification, and clustering to determine the party of a specified county

In [796]:
```
# Load 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 [797]:
```
# Load Election dataset
data_election = pd.read_csv('merged_train.csv')
data_election.head()
```

Out[797]:

	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	
0	AZ	apache	4001	72346	18.571863	0.486551	5.947806	1.719515	50.598513	4
1	AZ	cochise	4003	128177	56.299492	3.714395	34.403208	11.458374	49.069646	3
2	AZ	coconino	4005	138064	54.619597	1.342855	13.711033	4.825298	50.581614	4
3	AZ	gila	4007	53179	63.222325	0.552850	18.548675	4.249798	50.296170	3
4	AZ	graham	4009	37529	51.461536	1.811932	32.097844	4.385942	46.313518	4

TASK 1 - Partition in train and validation sets

Answer: We have partitioned the data into train and validation sets using the *Hold-Out Method*

In [798]:
```
x_train_full, x_validation_full, y_train, y_validation = train_test_sp
lit(data_election[['State', 'County', 'FIPS', 'Total Population', 'Per
cent White, not Hispanic or Latino', 'Percent Black, not Hispanic or L
atino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent
Female', 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Medi
an Household Income', 'Percent Unemployed', 'Percent Less than High Sc
hool Degree', 'Percent Less than Bachelor\'s Degree', 'Percent Rural',
'Democratic', 'Republican']], data_election['Party'], test_size = 0.25
, random_state = 0)
```

TASK 2 - Standardize the data

In [799]:
```
# Selecting required variables for x_train
x_train = x_train_full.select_dtypes(include=[np.int64,np.float64])
x_train = x_train.iloc[:,1:14]

# Selecting required variables for x_validation
x_validation = x_validation_full.select_dtypes(include=[np.int64,np.fl
oat64])
x_validation = x_validation.iloc[:,1:14]

# Standardizing the data
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_validation_scaled = scaler.transform(x_validation)
x_train_scaled_df = pd.DataFrame(x_train_scaled,index = x_train.index,
columns=x_train.columns)
x_validation_scaled_df = pd.DataFrame(x_validation_scaled,index = x_va
lidation.index,columns=x_validation.columns)
```

TASK 3

Using various predictor variables to develop regression models either via linear regression or LASSO.

Task 3a - Regression to predict *Democratic* values

Model 1 Linear Regression - *Including all variables*

In [800]:
```
# Create the linear regression model
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df, y = x_train_full['Demo
cratic'])
print(fitted_model.coef_)
```

```
[ 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]
```

In [801]:
```
# Predict the values
y_predicted = fitted_model.predict(x_validation_scaled_df)
```

In [802]:
```
# Determining values to calculate evaluation metrics
n = len(x_validation_scaled_df.index)
p = len(x_train_scaled_df.columns)
print(n)
print(p)
print(n-p-1)
```

```
299
13
285
```

In [803]:
```
# Generating Evaluation metrics
corr_coef = np.corrcoef(y_predicted,x_validation_full['Democratic'])[1
, 0]

R_squared = corr_coef ** 2
print("R squared:",R_squared)
```

```
adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['De
mocratic']))
print('RMSE -',rmse)
```

```
R squared: 0.9338361960241593
Adjusted R squared: 0.9308181979480683
RMSE - 14771.9947930757
```

Model 2 - Lasso Regression using all variables

In [804]:

```
# Generating model
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train_full['Demo
cratic'])
print(fitted_model.coef_)
```

```
[ 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]
```

In [805]:

```
# Predict the values
y_predicted = fitted_model.predict(x_validation_scaled_df)
```

In [806]:

```
# Determining values to calculate evaluation metrics
n = len(x_validation_scaled_df.index)
p = len(x_validation_scaled_df.columns)
n-p-1
print(n)
print(p)
print(n-p-1)
```

```
299
13
285
```

In [807]:

```
# Generating Evaluation metrics
corr_coef = np.corrcoef(y_predicted,x_validation_full['Democratic'])[1
, 0]

R_squared = corr_coef ** 2
print("R squared:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['De
mocratic']))
print('RMSE: ',rmse)
```

```
R squared: 0.9338579590814098
Adjusted R squared: 0.9308409537061758
RMSE:  14768.885350551016
```

Model 3. Linear Regression - Includes 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born' as variables.

In [808]:
```
# Generating model
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'P
ercent White, not Hispanic or Latino', 'Percent Black, not Hispanic or
Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born']], y = x
_train_full['Democratic'])
print(fitted_model.coef_)
```

```
[ 70705.8786866   -2212.85847901   -131.80192434 -10178.54695173
   9916.88242758]
```

In [809]:
```
# Predict the values
y_predicted = fitted_model.predict(x_validation_scaled_df[['Total Popu
lation', 'Percent White, not Hispanic or Latino', 'Percent Black, not
Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Bo
rn']])
```

In [810]:
```
# Determining values to calculate evaluation metrics
n = len(x_validation_scaled_df.index)
p = len(x_train_scaled_df[['Total Population', 'Percent White, not His
panic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hi
spanic or Latino', 'Percent Foreign Born']].columns)
n-p-1
print(n)
print(p)
print(n-p-1)
```

```
299
5
293
```

In [811]:
```
# Generating Evaluation metrics
corr_coef = np.corrcoef(y_predicted,x_validation_full['Democratic'])[1
, 0]
R_squared = corr_coef ** 2
print("R squared:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['De
mocratic']))
print('RMSE: ',rmse)
```

```
R squared: 0.9272983198666898
Adjusted R squared: 0.9260576768610018
RMSE:  14592.862156527432
```

Model 4. Linear Regression - Includes 'Total Population', 'Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree' as predictors.

NOTE: This is our BEST MODEL for predicting *Democratic* values

In [812]:
```
# Generating model
model = linear_model.LinearRegression()
fitted_model_democratic = model.fit(X = x_train_scaled_df[['Total Popu
lation', 'Percent Black, not Hispanic or Latino', 'Percent Less than B
achelor\'s Degree']], y = x_train_full['Democratic'])
print(fitted_model_democratic.coef_)
```

```
[70692.75301251  1827.68857508 -9335.76053975]
```

In [813]:
```
# Predicting values
y_predicted = fitted_model_democratic.predict(x_validation_scaled_df[[
'Total Population', 'Percent Black, not Hispanic or Latino', 'Percent
Less than Bachelor\'s Degree']])
```

In [814]:
```
# Determining values to calculate evaluation metrics
n = len(x_validation_scaled_df.index)
p = len(x_train_scaled_df[['Total Population', 'Percent Black, not His
panic or Latino', 'Percent Less than Bachelor\'s Degree']].columns)
n-p-1
print(n)
print(p)
print(n-p-1)
```

```
299
3
295
```

In [815]:
```
# Generating Evaluation metrics
corr_coef = np.corrcoef(y_predicted,x_validation_full['Democratic'])[1
, 0]
R_squared = corr_coef ** 2
print("R squared:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['De
mocratic']))
print('RMSE: ',rmse)
```

```
R squared: 0.9505061106430135
Adjusted R squared: 0.9500027829546374
RMSE:  12456.89252865588
```

Task 3b - Regression to predict *Republican* values

Model 1. Linear Regression including all variables

In [816]:
```
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df, y = x_train_full['Repu
blican'])
print(fitted_model.coef_)
```

```
[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]
```

In [817]:
```
y_predicted = fitted_model.predict(x_validation_scaled_df)
```

In [818]:
```
n = len(x_validation_scaled_df.index)
p = len(x_train_scaled_df.columns)
print(n)
print(p)
print(n-p-1)
```

```
299
13
285
```

In [819]:
```
corr_coef = np.corrcoef(y_predicted,x_validation_full['Republican'])[1
, 0]

R_squared = corr_coef ** 2
print("R sqaured:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['Re
publican']))
print('RMSE: ',rmse)
```

```
R sqaured: 0.7239014362949739
Adjusted R squared: 0.7113074667224639
RMSE:  15962.4313106021
```

Model 2. LASSO Regression that includes all variables

In [820]:
```
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled_df, y = x_train_full['Repu
blican'])
print(fitted_model.coef_)
```

```
[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]
```

In [821]:
```
y_predicted = fitted_model.predict(x_validation_scaled_df)
```

In [822]:
```
n = len(x_validation_scaled_df.index)
p = len(x_validation_scaled_df.columns)
n-p-1
print(n)
print(p)
print(n-p-1)
```

```
299
13
285
```

In [823]:
```
corr_coef = np.corrcoef(y_predicted,x_validation_full['Republican'])[1
, 0]

R_squared = corr_coef ** 2
print("R squared:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['Re
publican']))
print('RMSE: ',rmse)
```

```
R squared: 0.7238886663016905
Adjusted R squared: 0.7112941142382588
RMSE:  15962.567869419843
```

Model 3. Linear Regression using 'Total Population', 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born' as predictors.

In [824]:
```
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled_df[['Total Population', 'P
ercent White, not Hispanic or Latino', 'Percent Black, not Hispanic or
Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born']], y = x
_train_full['Republican'])
print(fitted_model.coef_)
```

```
[46801.58031155  2411.56062758 -1926.15808714    98.71008908
  -478.25725257]
```

In [825]:
```
y_predicted = fitted_model.predict(x_validation_scaled_df[['Total Popu
lation', 'Percent White, not Hispanic or Latino', 'Percent Black, not
Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Bo
rn']])
```

In [826]:
```
n = len(x_validation_scaled_df.index)
p = len(x_train_scaled_df[['Total Population', 'Percent White, not His
panic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hi
spanic or Latino', 'Percent Foreign Born']].columns)
n-p-1
print(n)
print(p)
print(n-p-1)
```

```
299
5
293
```

In [827]:
```
corr_coef = np.corrcoef(y_predicted,x_validation_full['Republican'])[1
, 0]
R_squared = corr_coef ** 2
print("R squared:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['Re
publican']))
print('RMSE: ',rmse)
```

```
R squared: 0.6704238187062499
Adjusted R squared: 0.6647996517899744
RMSE:  17111.714193417978
```

Model 4. Linear Regression including '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' as predictor variables

NOTE: This is our BEST MODEL for predicting *Republican* party values

In [828]:
```
model = linear_model.LinearRegression()
fitted_model_republican = model.fit(X = x_train_scaled_df[['Total Popu
lation', '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
_full['Republican'])
print(fitted_model_republican.coef_)
```

```
[45133.5738712   4612.72460625  3998.62967731 -4790.68208843
  2692.84982155  2174.86528205  6130.35899569 -5297.8335129 ]
```

In [829]:
```
y_predicted = fitted_model_republican.predict(x_validation_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 [830]:
```
n = len(x_validation_scaled_df.index)
p = len(x_train_scaled_df[['Total Population', 'Percent White, not His
panic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born'
, 'Percent Age 65 and Older', 'Percent Unemployed', 'Median Household
Income', 'Percent Rural']].columns)
n-p-1
print(n)
print(p)
print(n-p-1)
```

```
299
8
290
```

In [831]:
```
corr_coef = np.corrcoef(y_predicted,x_validation_full['Republican'])[1
, 0]
R_squared = corr_coef ** 2
print("R squared:",R_squared)

adjusted_r = 1 - (((1-R_squared)*(n-1))/(n-p-1))
print("Adjusted R squared:",adjusted_r)

rmse = math.sqrt(mean_squared_error(y_predicted, x_validation_full['Re
publican']))
print('RMSE: ',rmse)
```

```
R squared: 0.7302080671531
Adjusted R squared: 0.7227655310745649
RMSE:  15749.245925443494
```

TASK 4

Building a Classification Model

4a. Decision Tree Classifier

Model 1. Predicts each county as either Democratic or Republican using *all variables* and *entropy*

In [832]:

```
classifier = DecisionTreeClassifier(criterion = "entropy", splitter="b
est", min_weight_fraction_leaf=0.0, max_features=None, random_state=0,
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=Non
e, class_weight=None)
classifier.fit(x_train_scaled_df, y_train)
```

Out[832]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
epth=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=False,
                       random_state=0, splitter='best')
```

In [833]:

```
# Show the structure of the decision tree classifier
print(classifier.tree_.__getstate__()['nodes'])
len(classifier.tree_.__getstate__()['nodes'])
```

```
[(  1, 100, 11, -0.08037002, 0.85103407, 896, 896.)
 (  2,   7, 12, -1.57463121, 0.99675236, 328, 328.)
 (  3,   4,  1,  0.13988956, 0.28290479,  61,  61.)
 ( -1,  -1, -2, -2.        , 0.        ,  57,  57.)
 (  5,   6, 11, -2.27329141, 0.81127812,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 (  8,  15,  0, -0.34515437, 0.98895258, 267, 267.)
 (  9,  10, 11, -0.18558561, 0.30337484,  37,  37.)
 ( -1,  -1, -2, -2.        , 0.        ,  29,  29.)
 ( 11,  12,  9, -0.91212842, 0.81127812,   8,   8.)
 ( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
 ( 13,  14,  5,  0.40603571, 0.91829583,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( 16,  17,  1, -1.39712286, 1.        , 230, 230.)
 ( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
 ( 18,  99, 10,  0.07734165, 0.9985091 , 220, 220.)
 ( 19,  42,  4, -0.37913467, 0.99998349, 209, 209.)
 ( 20,  25,  8, -0.14533475, 0.8890349 ,  62,  62.)
 ( 21,  24,  7,  0.36920083, 0.86312057,  14,  14.)
 ( 22,  23,  2,  0.18771875, 0.65002242,  12,  12.)
 ( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 26,  27,  1,  0.62267354, 0.69621226,  48,  48.)
 ( -1,  -1, -2, -2.        , 0.        ,  16,  16.)
 ( 28,  41, 11, -0.1682383 , 0.85714844,  32,  32.)
 ( 29,  40,  6,  0.67939885, 0.73550858,  29,  29.)
 ( 30,  39,  8,  0.51208383, 0.60518658,  27,  27.)
 ( 31,  34,  4, -0.49395105, 0.83664074,  15,  15.)
```

```
( 32,  33, 11, -0.27043697, 0.46899559,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( 35,  36,  6, -0.97654212, 0.97095059,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 37,  38,  8,  0.11499928, 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  12,  12.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 43,  92,  3,  0.05903428, 0.97903461, 147, 147.)
( 44,  83,  8,  2.13687468, 0.93925472, 118, 118.)
( 45,  82, 12, -0.01478512, 0.88247445, 103, 103.)
( 46,  53, 11, -1.47726208, 0.93255384,  89,  89.)
( 47,  48,  8,  1.62924671, 0.42622866,  23,  23.)
( -1,  -1, -2, -2.        , 0.        ,  17,  17.)
( 49,  52,  5,  0.71812838, 0.91829583,   6,   6.)
( 50,  51, 11, -2.57235062, 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 54,  57, 10, -1.00663757, 0.98937558,  66,  66.)
( 55,  56, 11, -1.36946547, 0.43949699,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
( 58,  69,  0,  0.06232688, 0.99976152,  55,  55.)
( 59,  60,  0, -0.26622814, 0.90592822,  28,  28.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 61,  62,  9, -0.40477926, 0.79504028,  25,  25.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
( 63,  64,  0, -0.21745228, 0.97095059,  15,  15.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( 65,  66,  7, -0.58334582, 0.99403021,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( 67,  68, 10, -0.69751415, 0.86312057,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 70,  81, 12, -1.04157078, 0.91829583,  27,  27.)
( 71,  80,  6,  0.92913273, 0.99277445,  20,  20.)
( 72,  75,  7, -0.46444936, 0.89603823,  16,  16.)
( 73,  74, 10, -0.92671674, 0.50325833,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
( 76,  79,  6, -0.03350545, 0.98522814,   7,   7.)
( 77,  78,  9,  0.89954698, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( 84,  85,  3, -0.4157771 , 0.83664074,  15,  15.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( 86,  87,  4,  0.10013912, 1.        ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 88,  89,  4,  0.30861057, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 90,  91,  4,  0.71367368, 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( 93,  96,  9,  0.22570178, 0.92936363,  29,  29.)
( 94,  95,  8,  3.66894639, 0.54356444,  16,  16.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 97,  98,  7,  0.72845934, 0.9612366 ,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
```

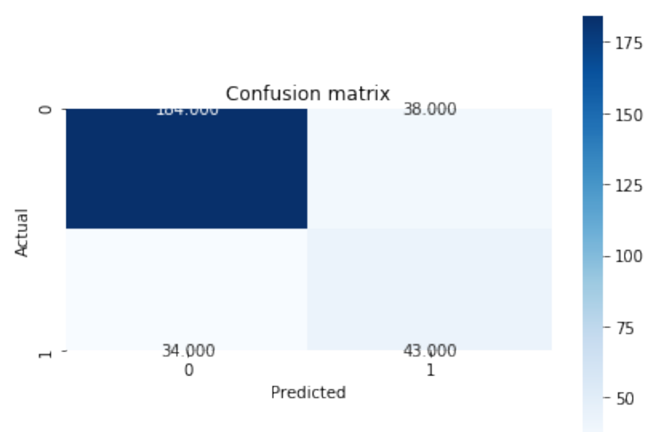
```
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)
(101, 110,  1, -1.93926793, 0.55336838, 568, 568.)
(102, 103,  3,  2.75355005, 0.99613448,  41,  41.)
( -1,  -1, -2, -2.        , 0.        ,  13,  13.)
(104, 105,  1, -3.12038493, 0.90592822,  28,  28.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(106, 109,  1, -2.37764275, 0.45371634,  21,  21.)
(107, 108,  3,  3.70863354, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,  15,  15.)
(111, 196,  3,  0.07042832, 0.45868582, 527, 527.)
(112, 193,  5,  1.17786229, 0.53017385, 424, 424.)
(113, 158, 10, -0.27290677, 0.50723272, 418, 418.)
(114, 115,  2, -0.58004016, 0.66366113, 168, 168.)
( -1,  -1, -2, -2.        , 0.        ,  24,  24.)
(116, 141,  6, -0.33900376, 0.72469719, 144, 144.)
(117, 140,  9,  0.75768894, 0.86853396,  69,  69.)
(118, 139, 10, -0.29305258, 0.93484902,  57,  57.)
(119, 120, 12, -0.6469022 , 0.89865338,  54,  54.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(121, 138,  7,  1.93100727, 0.84786175,  51,  51.)
(122, 131,  2, -0.52002695, 0.80309098,  49,  49.)
(123, 124,  1,  0.77033558, 0.56650951,  30,  30.)
( -1,  -1, -2, -2.        , 0.        ,  15,  15.)
(125, 126,  7,  0.24093483, 0.83664074,  15,  15.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(127, 128, 10, -0.54269454, 0.61938219,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
(129, 130,  3, -0.59259918, 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(132, 135, 10, -0.52860704, 0.98194079,  19,  19.)
(133, 134,  0, -0.35401343, 0.54356444,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(136, 137,  5, -1.3264969 , 0.43949699,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,  12,  12.)
(142, 155,  9,  1.04082423, 0.52936087,  75,  75.)
(143, 154,  9, -0.2200299 , 0.41786426,  71,  71.)
(144, 149, 12, -0.17743065, 0.6098403 ,  40,  40.)
(145, 146,  3, -0.42516482, 1.        ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(147, 148,  9, -1.51887619, 0.72192809,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(150, 151, 12,  1.07181042, 0.33729007,  32,  32.)
( -1,  -1, -2, -2.        , 0.        ,  24,  24.)
(152, 153,  6, -0.20031215, 0.81127812,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,  31,  31.)
(156, 157,  4, -0.40317059, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(159, 170,  4, -0.71062896, 0.37334332, 250, 250.)
(160, 169,  8, -0.68556282, 0.74248757,  38,  38.)
(161, 164, 12,  0.77434984, 0.91829583,  24,  24.)
(162, 163,  8, -0.79894298, 0.81127812,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(165, 168,  7, -0.18964487, 0.54356444,  16,  16.)
(166, 167,  2, -0.32445248, 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
```

```
( -1,  -1, -2, -2.         , 0.         ,   1,   1.)
( -1,  -1, -2, -2.         , 0.         ,  13,  13.)
( -1,  -1, -2, -2.         , 0.         ,  14,  14.)
(171, 180, 11,  0.20588898, 0.27425064, 212, 212.)
(172, 173,  9, -0.30048504, 0.73550858,  29,  29.)
( -1,  -1, -2, -2.         , 0.         ,  12,  12.)
(174, 177,  0, -0.24804918, 0.93666738,  17,  17.)
(175, 176, 12,  1.22702581, 0.86312057,   7,   7.)
( -1,  -1, -2, -2.         , 0.         ,   5,   5.)
( -1,  -1, -2, -2.         , 0.         ,   2,   2.)
(178, 179, 11, -0.00373855, 0.46899559,  10,  10.)
( -1,  -1, -2, -2.         , 0.         ,   1,   1.)
( -1,  -1, -2, -2.         , 0.         ,   9,   9.)
(181, 192,  5, -0.15476202, 0.15174889, 183, 183.)
(182, 183,  8, -1.90594471, 0.32275696,  68,  68.)
( -1,  -1, -2, -2.         , 0.         ,   1,   1.)
(184, 191, 11,  0.63255814, 0.26377744,  67,  67.)
(185, 190, 11,  0.58593363, 0.65002242,  18,  18.)
(186, 187,  0, -0.12371872, 0.33729007,  16,  16.)
( -1,  -1, -2, -2.         , 0.         ,  14,  14.)
(188, 189,  4, -0.45904274, 1.         ,   2,   2.)
( -1,  -1, -2, -2.         , 0.         ,   1,   1.)
( -1,  -1, -2, -2.         , 0.         ,   1,   1.)
( -1,  -1, -2, -2.         , 0.         ,   2,   2.)
( -1,  -1, -2, -2.         , 0.         ,  49,  49.)
( -1,  -1, -2, -2.         , 0.         , 115, 115.)
(194, 195,  0, -0.34596957, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.         , 0.         ,   2,   2.)
( -1,  -1, -2, -2.         , 0.         ,   4,   4.)
( -1,  -1, -2, -2.         , 0.         , 103, 103.)]
```

Out[833]: 197

In [834]:
```
y_pred = classifier.predict(x_validation_scaled_df)
```

In [835]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [836]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7591973244147158, 0.24080267558528423, array([0.8440367, 0.530864
2]), array([0.82882883, 0.55844156]), array([0.83636364, 0.5443038 ]
)]
```

Model 2. Predicts each county as either Democratic or Republican using *all variables* and *gini index*

In [837]:
```
classifier = DecisionTreeClassifier(criterion = "gini", splitter="best
", min_weight_fraction_leaf=0.0, max_features=None, random_state=0, ma
x_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None,
class_weight=None)
classifier.fit(x_train_scaled_df, y_train)
```

Out[837]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_dept
h=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=False,
                       random_state=0, splitter='best')
```

In [838]:
```
# Show the structure of the decision tree classifier
print(classifier.tree_.__getstate__()['nodes'])
len(classifier.tree_.__getstate__()['nodes'])
```

```
[(  1,  66, 11, -0.56057394, 0.40035077, 896, 896.)
 (  2,   7, 12, -1.57463121, 0.46602727, 211, 211.)
 (  3,   4,  1,  0.13199214, 0.07262371,  53,  53.)
 ( -1,  -1, -2, -2.        , 0.        ,  50,  50.)
 (  5,   6, 10, -0.75041622, 0.44444444,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 (  8,  21,  4, -0.37756465, 0.49927896, 158, 158.)
 (  9,  20,  6,  0.9269689 , 0.34179688,  32,  32.)
 ( 10,  15,  5,  0.08567018, 0.27777778,  30,  30.)
 ( 11,  12,  9, -0.99918535, 0.48      ,  10,  10.)
 ( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
 ( 13,  14,  0, -0.3478808 , 0.32      ,   5,   5.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)

 ( 16,  17,  7,  1.6322031 , 0.095     ,  20,  20.)
 ( -1,  -1, -2, -2.        , 0.        ,  17,  17.)
 ( 18,  19, 10, -0.77914186, 0.44444444,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 22,  51,  8,  1.57170981, 0.48185941, 126, 126.)
 ( 23,  30, 10, -0.84897971, 0.43933847,  89,  89.)
 ( 24,  25,  8, -0.60550237, 0.1171875 ,  32,  32.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( 26,  27,  0,  0.74333228, 0.06243496,  31,  31.)
 ( -1,  -1, -2, -2.        , 0.        ,  29,  29.)
 ( 28,  29,  5,  0.24840664, 0.5       ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
 ( 31,  32,  1, -1.28936231, 0.49861496,  57,  57.)
 ( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
```

```
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
( 33,  48,  3,  0.02518291, 0.49479384,  49,  49.)
( 34,  45,  0,  0.41852768, 0.48442907,  34,  34.)
( 35,  44,  4,  0.18717835, 0.43622449,  28,  28.)
( 36,  37,  0, -0.28134367, 0.5       ,  18,  18.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 38,  41,  3, -0.50456491, 0.4260355 ,  13,  13.)
( 39,  40,  8, -0.58805928, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 42,  43, 12, -1.39063555, 0.19753086,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
( 46,  47, 11, -1.2563647 , 0.27777778,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 49,  50,  0,  0.65423778, 0.23111111,  15,  15.)
( -1,  -1, -2, -2.        , 0.        ,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 52,  55,  4,  0.22627059, 0.48210373,  37,  37.)
( 53,  54,  9, -0.26112332, 0.24489796,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,  12,  12.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 56,  57,  3, -0.4157771 , 0.49149338,  23,  23.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 58,  59,  3, -0.28143749, 0.455     ,  20,  20.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( 60,  61,  0,  0.01551987, 0.49704142,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( 62,  65, 11, -2.12902129, 0.44444444,   9,   9.)
( 63,  64,  1, -1.02892289, 0.48      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( 67,  76,  1, -2.37764275, 0.27939688, 685, 685.)
( 68,  69,  0, -0.35987961, 0.32      ,  20,  20.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 70,  75,  2, -0.18057108, 0.19753086,  18,  18.)
( 71,  74,  2, -0.58732125, 0.11072664,  17,  17.)
( 72,  73, 12, -0.88343644, 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( 77, 226,  2,  3.49378169, 0.25341851, 665, 665.)
( 78, 211,  0,  0.09956769, 0.23156151, 651, 651.)
( 79, 140, 10, -0.42592379, 0.20167139, 624, 624.)
( 80,  89,  2, -0.5602732 , 0.33406191, 184, 184.)
( 81,  86, 10, -0.46300647, 0.11386593,  66,  66.)
( 82,  83,  8,  0.72398362, 0.06147644,  63,  63.)
( -1,  -1, -2, -2.        , 0.        ,  52,  52.)
( 84,  85,  8,  0.79642329, 0.29752066,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   9,   9.)
( 87,  88,  5, -0.14008594, 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 90,  97,  3, -0.58171928, 0.41726515, 118, 118.)
( 91,  92,  7, -0.41934912, 0.39111111,  15,  15.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 93,  94,  0, -0.35902697, 0.26035503,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( 95,  96, 10, -0.45566392, 0.15277778,  12,  12.)
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( 98, 117, 12, -0.17743065, 0.35743237, 103, 103.)
( 99, 106,  0, -0.23631393, 0.48389218,  39,  39.)
(100, 103,  5, -0.09738274, 0.46875   ,  16,  16.)
```

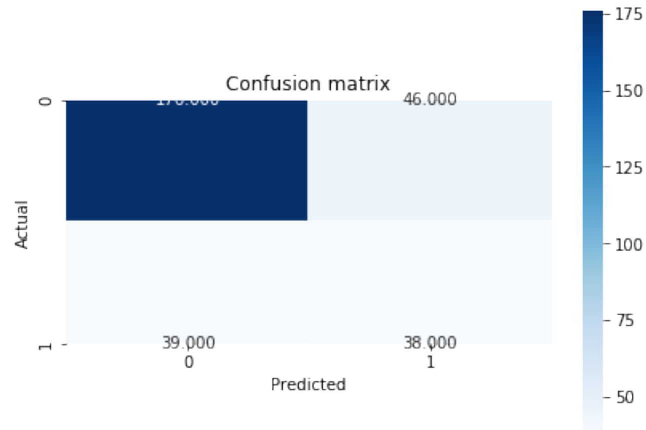
```
(100, 103,  5, -0.69758274, 0.46875   ,  16,  16.)
(101, 102,  6, -1.03581828, 0.27777778,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(104, 105, 10, -0.51484808, 0.18      ,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(107, 114,  6,  0.45889696, 0.38563327,  23,  23.)
(108, 113, 10, -0.44130473, 0.19753086,  18,  18.)
(109, 112, 12, -0.88582143, 0.11072664,  17,  17.)
(110, 111,  8, -0.12090595, 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(115, 116, 11,  0.03089635, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(118, 119,  9, -2.08196294, 0.21875   ,  64,  64.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(120, 129,  0, -0.35097449, 0.19753086,  63,  63.)
(121, 128,  0, -0.35286698, 0.39111111,  15,  15.)
(122, 123,  3, -0.55414024, 0.26035503,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(124, 125, 11,  0.59247236, 0.15277778,  12,  12.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
(126, 127, 11,  0.71725097, 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(130, 131,  8, -0.75630897, 0.1171875 ,  48,  48.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(132, 135,  7, -0.79681048, 0.08148483,  47,  47.)
(133, 134, 10, -0.79246247, 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(136, 137,  8,  0.87783855, 0.04345679,  45,  45.)
( -1,  -1, -2, -2.        , 0.        ,  41,  41.)
(138, 139,  0, -0.2610504 , 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(141, 160,  4, -0.71081272, 0.13487603, 440, 440.)
(142, 153,  9,  0.67550379, 0.32      ,  50,  50.)
(143, 144,  1, -1.5266487 , 0.15673469,  35,  35.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(145, 150, 11,  1.31604785, 0.11072664,  34,  34.)
(146, 149,  5, -0.42600691, 0.06054688,  32,  32.)
(147, 148,  3, -0.61190793, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,  27,  27.)
(151, 152,  0, -0.27763786, 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(154, 159, 10,  1.32609534, 0.49777778,  15,  15.)
(155, 156,  8, -0.88298315, 0.42      ,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
(157, 158,  3, -0.51559971, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(161, 178, 11,  0.16163053, 0.10645628, 390, 390.)
(162, 163,  9, -0.33839346, 0.29273785,  73,  73.)
( -1,  -1, -2, -2.        , 0.        ,  31,  31.)
(164, 175,  5,  0.44011837, 0.42743764,  42,  42.)
(165, 174,  1,  0.69545308, 0.49382716,  27,  27.)
(166, 167,  3, -0.45734653, 0.49586777,  22,  22.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(168, 169,  8,  0.1079289 , 0.44444444,  15,  15.)
```

```
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
(170, 171, 11, -0.08203183, 0.49382716,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(172, 173,  8,  0.33943101, 0.44444444,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(176, 177,  9,  1.13136494, 0.12444444,  15,  15.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(179, 180,  8, -1.98354959, 0.05517022, 317, 317.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(181, 208,  6,  2.17715502, 0.04935107, 316, 316.)
(182, 191, 10, -0.27290677, 0.04386506, 312, 312.)
(183, 190,  8,  0.37702683, 0.18549346,  29,  29.)
(184, 189, 10, -0.27334464, 0.13265306,  28,  28.)
(185, 188,  5, -1.17303771, 0.07133059,  27,  27.)
(186, 187, 10, -0.35334122, 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,  25,  25.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(192, 195,  5, -4.00755882, 0.027869  , 283, 283.)
(193, 194,  9,  1.43522251, 0.27777778,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(196, 199, 11,  0.20588898, 0.02142606, 277, 277.)
(197, 198,  4, -0.64108327, 0.19753086,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
(200, 205,  0, -0.06010243, 0.01481399, 268, 268.)
(201, 204,  6, -1.35780209, 0.00769219, 259, 259.)
(202, 203,  7,  1.17945796, 0.13265306,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,  13,  13.)
( -1,  -1, -2, -2.        , 0.        , 245, 245.)
(206, 207,  0, -0.05059671, 0.19753086,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
(209, 210,  7, -1.68070221, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(212, 225, 10,  0.2964929 , 0.48285322,  27,  27.)
(213, 214,  6, -0.89193663, 0.42344045,  23,  23.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(215, 218,  3, -0.4986935 , 0.36281179,  21,  21.)
(216, 217, 10, -0.44083035, 0.48      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(219, 220, 10, -0.83233967, 0.21875   ,  16,  16.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
(221, 224,  5,  0.19251465, 0.12444444,  15,  15.)
(222, 223,  8, -0.06836496, 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  12,  12.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(227, 230,  8, -1.1103785 , 0.24489796,  14,  14.)
(228, 229,  1, -1.60967112, 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   1,   1.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)]
```

Out[838]: 231

In [839]:
```
y_pred = classifier.predict(x_validation_scaled_df)
```

In [840]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [841]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7157190635451505, 0.28428093645484953, array([0.81860465, 0.45238
095]), array([0.79279279, 0.49350649]), array([0.80549199, 0.4720496
9])]
```

Model 3. Predicts each county as Democratic or Republican via the variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'

NOTE: This is the BEST classifier model

In [842]:
```
classifier_party = DecisionTreeClassifier(criterion = "entropy", split
ter="best", min_weight_fraction_leaf=0.0, max_features=None, random_st
ate=0, min_samples_leaf=2, max_leaf_nodes=None, min_impurity_decrease=
0.0, min_impurity_split=None, class_weight=None)
classifier_party.fit(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']], y_train)
```

Out[842]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
epth=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 [843]:
```
# Show the structure of the decision tree classifier
print(classifier_party.tree_.__getstate__()['nodes'])
len(classifier_party.tree_.__getstate__()['nodes'])
```

```
[(  1, 134,  3, -0.08037002, 0.85103407, 896, 896.)
 (  2,   3,  0, -1.39712286, 0.99675236, 328, 328.)
 ( -1,  -1, -2, -2.        , 0.        ,  36,  36.)
 (  4,  57,  3, -0.9321757 , 0.99834117, 292, 292.)
 (  5,  26,  0,  0.14515513, 0.90688017, 121, 121.)
 (  6,  15,  2,  0.135652  , 0.66319684,  58,  58.)
 (  7,   8,  3, -1.63488019, 0.29747225,  38,  38.)
 ( -1,  -1, -2, -2.        , 0.        ,  23,  23.)
 (  9,  10,  1,  0.79554036, 0.56650951,  15,  15.)
 ( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
 ( 11,  14,  0, -0.07996122, 0.86312057,   7,   7.)
 ( 12,  13,  1,  1.77366126, 1.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 16,  17,  1, -0.29885851, 0.97095059,  20,  20.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 18,  25,  1,  0.49151306, 0.87398105,  17,  17.)
 ( 19,  20,  1, -0.05425084, 0.97986876,  12,  12.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 21,  22,  3, -3.29062343, 0.99107606,   9,   9.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 23,  24,  3, -1.70829451, 0.86312057,   7,   7.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
 ( 27,  56,  1, -0.14673719, 0.99545158,  63,  63.)
 ( 28,  29,  3, -2.98789024, 0.97844933,  58,  58.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 30,  39,  3, -1.85677475, 0.95931603,  55,  55.)
 ( 31,  32,  0,  0.38269778, 0.70246655,  21,  21.)
 ( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
 ( 33,  34,  1, -0.46104233, 0.89049164,  13,  13.)
 ( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
 ( 35,  36,  1, -0.44390647, 1.        ,   8,   8.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 37,  38,  0,  0.52497701, 0.72192809,   5,   5.)
 ( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 40,  55,  2, -0.38122553, 1.        ,  34,  34.)
 ( 41,  54,  2, -0.43249336, 0.96661863,  28,  28.)
 ( 42,  53,  3, -1.02097434, 1.        ,  22,  22.)
 ( 43,  48,  1, -0.5319612 , 0.96407876,  18,  18.)
 ( 44,  45,  2, -0.57423112, 0.91829583,   6,   6.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 46,  47,  0,  0.79128367, 1.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 49,  50,  3, -1.4790206 , 0.81127812,  12,  12.)
 ( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
 ( 51,  52,  2, -0.54822895, 0.50325833,   9,   9.)
 ( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
 ( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
 ( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
 ( 58,  61,  1, -0.5689348 , 0.91829583, 171, 171.)
 ( 59,  60,  0,  0.93850088, 0.2108423 ,  30,  30.)
 ( -1,  -1, -2, -2.        , 0.        ,  28,  28.)
```

```
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( 62, 133,  2,  0.90451238, 0.96926692, 141, 141.)
( 63, 128,  3, -0.15711062, 0.97895964, 135, 135.)
( 64,  69,  0, -0.55136567, 0.95952128, 123, 123.)
( 65,  66,  3, -0.54595357, 0.89049164,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( 67,  68,  0, -0.65364078, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 70, 127,  2,  0.5669066 , 0.92994294, 110, 110.)
( 71, 126,  1,  1.41485691, 0.94142311, 106, 106.)
( 72,  81,  1, -0.52583343, 0.94984855, 103, 103.)
( 73,  74,  0,  0.40878092, 0.74248757,  19,  19.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( 75,  80,  1, -0.53337231, 0.91829583,  12,  12.)
( 76,  79,  0,  0.80438378, 0.98522814,   7,   7.)
( 77,  78,  1, -0.55504334, 0.72192809,   5,   5.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 82, 113,  0,  0.57266808, 0.97366806,  84,  84.)
( 83,  84,  2, -0.54875144, 0.99800088,  57,  57.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 85, 112,  3, -0.20163455, 0.99403021,  55,  55.)
( 86, 101,  0,  0.25532573, 0.98738002,  53,  53.)
( 87,  92,  1,  0.23392791, 0.98769251,  23,  23.)
( 88,  89,  2, -0.02867506, 0.50325833,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 90,  91,  0,  0.02844463, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( 93,  98,  3, -0.44412768, 0.94028596,  14,  14.)
( 94,  97,  0, -0.20509183, 0.72192809,  10,  10.)
( 95,  96,  0, -0.32428472, 0.97095059,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 99, 100,  2, -0.11826703, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
(102, 111,  2, -0.17887931, 0.91829583,  30,  30.)
(103, 108,  1, -0.08007337, 0.98769251,  23,  23.)
(104, 105,  2, -0.34610529, 0.89049164,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(106, 107,  1, -0.3339825 , 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(109, 110,  3, -0.27429329, 0.46899559,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(114, 125,  1, -0.43469122, 0.82562653,  27,  27.)
(115, 122,  2, -0.48375478, 0.97741782,  17,  17.)
(116, 117,  1, -0.51891863, 0.97095059,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(118, 121,  0,  0.7145963 , 0.81127812,   8,   8.)
(119, 120,  1, -0.47075288, 1.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(123, 124,  3, -0.3659679 , 0.59167278,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
```

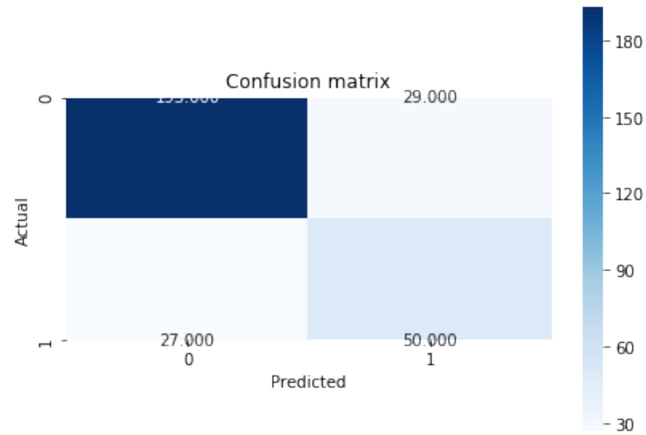
```
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(129, 132,  0,  0.47249402, 0.81127812,  12,  12.)
(130, 131,  0, -0.1117643 , 1.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
(135, 144,  0, -1.93926793, 0.55336838, 568, 568.)
(136, 137,  2,  2.75355005, 0.99613448,  41,  41.)
( -1,  -1, -2, -2.        , 0.        ,  13,  13.)
(138, 139,  0, -3.12038493, 0.90592822,  28,  28.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(140, 143,  0, -2.37764275, 0.45371634,  21,  21.)
(141, 142,  2,  3.70863354, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,  15,  15.)
(145, 262,  2,  0.07042832, 0.45868582, 527, 527.)
(146, 231,  3,  0.63485846, 0.53017385, 424, 424.)
(147, 148,  1, -0.57970834, 0.63707035, 242, 242.)
( -1,  -1, -2, -2.        , 0.        ,  28,  28.)
(149, 150,  0, -1.23085541, 0.68495936, 214, 214.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(151, 172,  2, -0.58171928, 0.65911225, 211, 211.)
(152, 163,  2, -0.58943173, 0.8812909 ,  40,  40.)
(153, 154,  1, -0.50285958, 0.63430955,  25,  25.)
( -1,  -1, -2, -2.        , 0.        ,  13,  13.)
(155, 162,  0,  0.86333296, 0.91829583,  12,  12.)
(156, 157,  1, -0.47551461, 1.        ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(158, 159,  2, -0.61531579, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(160, 161,  0,  0.82370168, 1.        ,   4,   4.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(164, 167,  0,  0.7519342 , 0.99679163,  15,  15.)
(165, 166,  2, -0.58622202, 0.65002242,   6,   6.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(168, 171,  2, -0.58491206, 0.76420451,   9,   9.)
(169, 170,  3,  0.52204853, 0.97095059,   5,   5.)
( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(173, 192,  0,  0.24085698, 0.58515699, 171, 171.)
(174, 175,  2, -0.44547236, 0.84535094,  33,  33.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(176, 181,  3,  0.10490276, 0.93058613,  26,  26.)
(177, 178,  2, -0.35087338, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(179, 180,  3,  0.05317019, 1.        ,   4,   4.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
(182, 183,  0, -0.16688394, 0.81127812,  20,  20.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(184, 191,  1,  0.85316563, 0.9612366 ,  13,  13.)
(185, 188,  1,  0.47877514, 0.99403021,  11,  11.)
(186, 187,  1, -0.21158562, 0.86312057,   7,   7.)
( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(189, 190,  1,  0.58017725, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(193, 194,  0,  0.50979313, 0.49596907, 138, 138.)
( -1,  -1, -2, -2.        , 0.        ,  33,  33.)
(195, 196,  0,  0.51963624, 0.59167278, 105, 105.)
```

```
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(197, 210,  0,  0.61753303, 0.54696174, 103, 103.)
(198, 209,  0,  0.60419577, 0.84535094,  22,  22.)
(199, 208,  2, -0.32332835, 0.72192809,  20,  20.)
(200, 207,  3,  0.58762729, 0.89049164,  13,  13.)
(201, 206,  3,  0.1584497 , 0.68403844,  11,  11.)
(202, 203,  2, -0.54801926, 0.91829583,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(204, 205,  2, -0.44306691, 1.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(211, 228,  1, -0.34305049, 0.42440514,  81,  81.)
(212, 217,  0,  0.79691407, 0.35001059,  76,  76.)
(213, 214,  2, -0.35420863, 0.15649106,  44,  44.)
( -1,  -1, -2, -2.        , 0.        ,  40,  40.)
(215, 216,  3,  0.32067127, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(218, 223,  2, -0.52877185, 0.54356444,  32,  32.)
(219, 222,  0,  0.82281923, 0.26676499,  22,  22.)
(220, 221,  2, -0.55390826, 0.72192809,   5,   5.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,  17,  17.)
(224, 225,  2, -0.5051432 , 0.8812909 ,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(226, 227,  0,  0.81409812, 0.54356444,   8,   8.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
(229, 230,  1, -0.29000814, 0.97095059,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(232, 235,  2, -0.62734044, 0.35056382, 182, 182.)
(233, 234,  2, -0.64311478, 0.9456603 ,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(236, 257,  0,  0.69196457, 0.27257363, 171, 171.)
(237, 256,  0,  0.67314881, 0.42806963,  80,  80.)
(238, 255,  1,  0.43997394, 0.34351974,  78,  78.)
(239, 254,  0,  0.62962723, 0.45079139,  53,  53.)
(240, 241,  3,  0.72731042, 0.55249511,  39,  39.)
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)
(242, 243,  3,  0.73419315, 0.67694187,  28,  28.)
( -1,  -1, -2, -2.        , 0.91829583,   3,   3.)
(244, 253,  2, -0.43490677, 0.52936087,  25,  25.)
(245, 246,  3,  0.8355791 , 0.74959526,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(247, 252,  0,  0.53458279, 0.91829583,   9,   9.)
(248, 251,  0,  0.4444977 , 1.        ,   6,   6.)
(249, 250,  3,  0.98112524, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,  25,  25.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(258, 261,  1, -0.59058732, 0.08728059,  91,  91.)
(259, 260,  2, -0.55242294, 0.81127812,   4,   4.)
( -1,  -1, -2, -2.        , 1.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  87,  87.)
( -1,  -1, -2, -2.        , 0.        , 103, 103.)]
```

Out[843]: 263

In [844]:
```
y_pred = classifier_party.predict(x_validation_scaled_df[['Percent Whi
te, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino',
'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degree']]
)
```

In [845]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [846]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.8127090301003345, 0.18729096989966554, array([0.87727273, 0.63291
139]), array([0.86936937, 0.64935065]), array([0.87330317, 0.6410256
4])]
```

Model 4. Predictions using gini index and variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'

In [847]:
```
classifier = DecisionTreeClassifier(criterion = "gini", splitter="best
", min_weight_fraction_leaf=0.0, max_features=None, random_state=0, mi
n_samples_leaf=2, max_leaf_nodes=None, min_impurity_decrease=0.0, min_
impurity_split=None, class_weight=None)
classifier.fit(x_train_scaled_df[['Percent White, not Hispanic or Lati
no', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Lat
ino', 'Percent Less than Bachelor\'s Degree']], y_train)
```

Out[847]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_dept
h=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')
```

```
                    random_state=0, splitter='best')
```

In [848]:
```
# Show the structure of the decision tree classifier
print(classifier.tree_.__getstate__()['nodes'])
len(classifier.tree_.__getstate__()['nodes'])
```

```
[(  1,  84,  3, -0.56057394, 0.40035077, 896, 896.)
 (  2,  13,  0, -0.55090189, 0.46602727, 211, 211.)
 (  3,   4,  0, -1.24638557, 0.1723356 ,  63,  63.)
 ( -1,  -1, -2, -2.        , 0.        ,  33,  33.)
 (  5,  10,  2,  0.90451238, 0.32      ,  30,  30.)
 (  6,   9,  3, -2.22614324, 0.1472    ,  25,  25.)
 (  7,   8,  2,  0.30470251, 0.40816327,   7,   7.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,  18,  18.)
 ( 11,  12,  1,  0.02679734, 0.32      ,   5,   5.)
 ( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 14,  29,  3, -1.85677475, 0.49963477, 148, 148.)
 ( 15,  16,  0,  0.21986136, 0.33240997,  38,  38.)
 ( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
 ( 17,  18,  3, -2.98789024, 0.44444444,  24,  24.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 19,  24,  2, -0.48755288, 0.36281179,  21,  21.)
 ( 20,  21,  2, -0.51566105, 0.48979592,   7,   7.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 22,  23,  1, -0.45739405, 0.48      ,   5,   5.)
 ( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
 ( 25,  26,  2, -0.38434875, 0.24489796,  14,  14.)
 ( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
 ( 27,  28,  3, -2.41806042, 0.44444444,   6,   6.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)

 ( 30,  59,  3, -0.9321757 , 0.48661157, 110, 110.)
 ( 31,  42,  3, -1.29890788, 0.49861496,  57,  57.)
 ( 32,  39,  3, -1.49121279, 0.46280992,  22,  22.)
 ( 33,  38,  3, -1.58662784, 0.49586777,  11,  11.)
 ( 34,  35,  1, -0.37129088, 0.40816327,   7,   7.)
 ( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
 ( 36,  37,  1, -0.11084155, 0.5       ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( 40,  41,  1,  0.08887938, 0.29752066,  11,  11.)
 ( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
 ( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
 ( 43,  58,  2,  0.08355056, 0.46693878,  35,  35.)
 ( 44,  47,  0,  0.16765592, 0.44444444,  33,  33.)
 ( 45,  46,  3, -0.97523135, 0.18      ,  10,  10.)
 ( -1,  -1, -2, -2.        , 0.        ,   8,   8.)
 ( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
 ( 48,  57,  2, -0.38122553, 0.49149338,  23,  23.)
 ( 49,  52,  3, -1.12407148, 0.43213296,  19,  19.)
 ( 50,  51,  2, -0.55972055, 0.19753086,   9,   9.)
 ( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
 ( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
 ( 53,  56,  3, -1.02097434, 0.5       ,  10,  10.)
 ( 54,  55,  1, -0.52965948, 0.27777778,   6,   6.)
 ( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
 ( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
 ( 60,  65,  3, -0.76883274, 0.42150231,  53,  53.)
 ( 61,  64,  3, -0.87899101, 0.23553719,  22,  22.)
 ( 62,  63,  1,  0.64434062, 0.46875   ,   8,   8.)
```

```
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,  14,  14.)
( 66,  67,  2, -0.58090663, 0.48699272,  31,  31.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( 68,  77,  2, -0.35901998, 0.5       ,  26,  26.)
( 69,  72,  0,  0.65629002, 0.4296875 ,  16,  16.)
( 70,  71,  1, -0.07269594, 0.19753086,   9,   9.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( 73,  74,  0,  0.7542271 , 0.48979592,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( 75,  76,  1, -0.52199644, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 78,  83,  2,  0.02518291, 0.32      ,  10,  10.)
( 79,  80,  2, -0.16046966, 0.44444444,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( 81,  82,  0,  0.09252132, 0.5       ,   4,   4.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( 85,  90,  0, -2.37764275, 0.27939688, 685, 685.)
( 86,  87,  0, -3.11295795, 0.32      ,  20,  20.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
( 88,  89,  2,  3.70863354, 0.48      ,  10,  10.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( 91, 248,  1,  3.49378169, 0.25341851, 665, 665.)
( 92, 139,  3, -0.08037002, 0.23156151, 651, 651.)
( 93,  96,  1, -0.55227783, 0.4338843 , 110, 110.)
( 94,  95,  0,  0.9381769 , 0.0739645 ,  26,  26.)
( -1,  -1, -2, -2.        , 0.        ,  24,  24.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( 97, 122,  3, -0.26412845, 0.48185941,  84,  84.)
( 98, 101,  1, -0.52333665, 0.41522491,  51,  51.)
( 99, 100,  3, -0.47977597, 0.14201183,  13,  13.)
( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
(102, 103,  2, -0.57037982, 0.46537396,  38,  38.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(104, 105,  2, -0.49994178, 0.44444444,  36,  36.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(106, 107,  0, -0.64234865, 0.47450572,  31,  31.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(108, 109,  0, -0.08823872, 0.49704142,  26,  26.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(110, 111,  1, -0.47412421, 0.44444444,  21,  21.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(112, 121,  1, -0.10132172, 0.48442907,  17,  17.)
(113, 118,  1, -0.3335074 , 0.49704142,  13,  13.)
(114, 115,  1, -0.46554987, 0.46875   ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(116, 117,  0,  0.64731777, 0.27777778,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
(119, 120,  3, -0.47418377, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(123, 138,  0,  0.6406737 , 0.48852158,  33,  33.)
(124, 129,  1, -0.17670867, 0.49704142,  26,  26.)
(125, 128,  0,  0.27220932, 0.29752066,  11,  11.)
(126, 127,  0, -0.03260628, 0.44444444,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(130, 131,  2, -0.53116238, 0.44444444,  15,  15.)
```

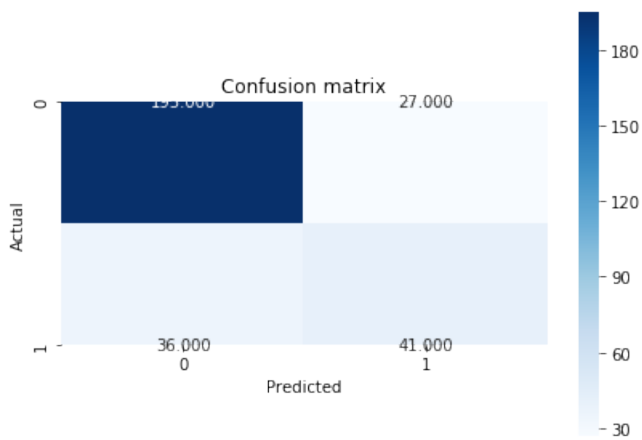
```
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(132, 133,  2, -0.29862802, 0.35502959,  13,  13.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(134, 137,  0, -0.27679405, 0.5       ,   6,   6.)
(135, 136,  1,  0.63068554, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(140, 241,  2, -0.07058155, 0.17375914, 541, 541.)
(141, 144,  0, -1.47601032, 0.2168905 , 404, 404.)
(142, 143,  3,  0.56992751, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(145, 218,  3,  0.63485846, 0.20399369, 399, 399.)
(146, 155,  1, -0.56541881, 0.26176678, 226, 226.)
(147, 150,  3,  0.03367743, 0.06887755,  56,  56.)
(148, 149,  0,  0.84800935, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(151, 152,  3,  0.53074369, 0.03844675,  51,  51.)
( -1,  -1, -2, -2.        , 0.        ,  46,  46.)
(153, 154,  3,  0.54233357, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(156, 161,  1, -0.55460158, 0.31287197, 170, 170.)
(157, 160,  3,  0.35052219, 0.5       ,  10,  10.)
(158, 159,  0,  0.60323593, 0.27777778,   6,   6.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(162, 217,  3,  0.62660956, 0.28875   , 160, 160.)
(163, 178,  3,  0.08463416, 0.2763601 , 157, 157.)
(164, 177,  3,  0.07852823, 0.40429688,  32,  32.)
(165, 170,  1, -0.24439333, 0.35777778,  30,  30.)
(166, 167,  3,  0.01224033, 0.18836565,  19,  19.)
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)
(168, 169,  3,  0.0217658 , 0.375     ,   8,   8.)
( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
(171, 174,  2, -0.35087338, 0.49586777,  11,  11.)
(172, 173,  2, -0.53437999, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(175, 176,  0, -0.42023294, 0.27777778,   6,   6.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(179, 206,  0,  0.82198247, 0.235008  , 125, 125.)
(180, 181,  2, -0.6319989 , 0.18663194,  96,  96.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
(182, 187,  1, -0.44312377, 0.17315527,  94,  94.)
(183, 186,  0,  0.52049688, 0.05259313,  37,  37.)
(184, 185,  0,  0.46973659, 0.24489796,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   5,   5.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  30,  30.)
(188, 197,  0,  0.50925177, 0.24130502,  57,  57.)
(189, 190,  1,  0.47877514, 0.14901388,  37,  37.)
( -1,  -1, -2, -2.        , 0.        ,  20,  20.)
(191, 192,  1,  0.58017725, 0.29065744,  17,  17.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(193, 196,  1,  0.84851107, 0.12444444,  15,  15.)
(194, 195,  3,  0.29107797, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  11,  11.)
(198, 199,  0,  0.55567738, 0.375     ,  20,  20.)
```

```
( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
(200, 205,  1, -0.29000814, 0.29065744,  17,  17.)
(201, 204,  1, -0.34305049, 0.42      ,  10,  10.)
(202, 203,  2, -0.44286659, 0.21875   ,   8,   8.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
(207, 210,  0,  0.83689818, 0.36623068,  29,  29.)
(208, 209,  0,  0.82436171, 0.32      ,   5,   5.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   3,   3.)
(211, 216,  3,  0.34548417, 0.21875   ,  24,  24.)
(212, 213,  0,  0.86333296, 0.39669421,  11,  11.)
( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
(214, 215,  3,  0.12288101, 0.21875   ,   8,   8.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   6,   6.)
( -1,  -1, -2, -2.        , 0.        ,  13,  13.)
( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
(219, 222,  2, -0.62734044, 0.11908183, 173, 173.)
(220, 221,  2, -0.64311478, 0.46280992,  11,  11.)
( -1,  -1, -2, -2.        , 0.        ,   7,   7.)
( -1,  -1, -2, -2.        , 0.        ,   4,   4.)
(223, 240,  3,  1.58702171, 0.08268557, 162, 162.)
(224, 235,  0,  0.69196457, 0.0721875 , 160, 160.)
(225, 234,  0,  0.67314881, 0.13442554,  69,  69.)
(226, 227,  0,  0.51050979, 0.08554244,  67,  67.)
( -1,  -1, -2, -2.        , 0.        ,  36,  36.)
(228, 229,  0,  0.52670035, 0.1748179 ,  31,  31.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(230, 231,  1, -0.04673263, 0.0665874 ,  29,  29.)
( -1,  -1, -2, -2.        , 0.        ,  25,  25.)
(232, 233,  1, -0.0084341 , 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
(236, 239,  1, -0.59058732, 0.02173651,  91,  91.)
(237, 238,  0,  0.87628293, 0.375     ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,  87,  87.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
(242, 247,  3,  0.30837013, 0.02877085, 137, 137.)
(243, 244,  0, -1.83954883, 0.13717421,  27,  27.)
( -1,  -1, -2, -2.        , 0.5       ,   2,   2.)
(245, 246,  2,  0.07790576, 0.0768    ,  25,  25.)
( -1,  -1, -2, -2.        , 0.44444444,   3,   3.)
( -1,  -1, -2, -2.        , 0.        ,  22,  22.)
( -1,  -1, -2, -2.        , 0.        , 110, 110.)
(249, 250,  0, -1.47376376, 0.24489796,  14,  14.)
( -1,  -1, -2, -2.        , 0.        ,  10,  10.)
(251, 252,  2, -0.60146526, 0.5       ,   4,   4.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)
( -1,  -1, -2, -2.        , 0.        ,   2,   2.)]
```

Out[848]: 253

In [849]:
```
y_pred = classifier.predict(x_validation_scaled_df[['Percent White, no
t Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Perce
nt Hispanic or Latino', 'Percent Less than Bachelor\'s Degree']])
```

In [850]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [851]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.7892976588628763, 0.21070234113712372, array([0.84415584, 0.60294
118]), array([0.87837838, 0.53246753]), array([0.86092715, 0.5655172
4])]
```

4b. K-Nearest Neighbours Using KNN to predict political party

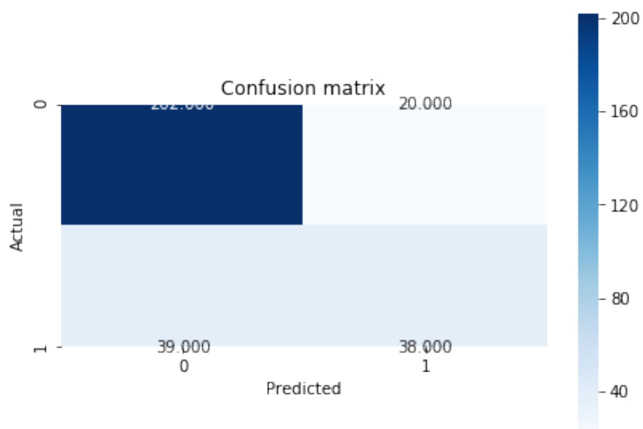
Model 1. KNN using all variables with k = 3

In [852]:
```
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train_scaled_df, y_train)
```

Out[852]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
                     metric_params=None, n_jobs=None, n_neighbors=3,
p=2,
                     weights='uniform')
```

In [853]:
```
y_pred = classifier.predict(x_validation_scaled_df)
```

In [854]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [855]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - metrics.accuracy_score(y_validation, y_pred)
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.802675585284281, 0.19732441471571904, array([0.83817427, 0.655172
41]), array([0.90990991, 0.49350649]), array([0.87257019, 0.56296296
])]
```

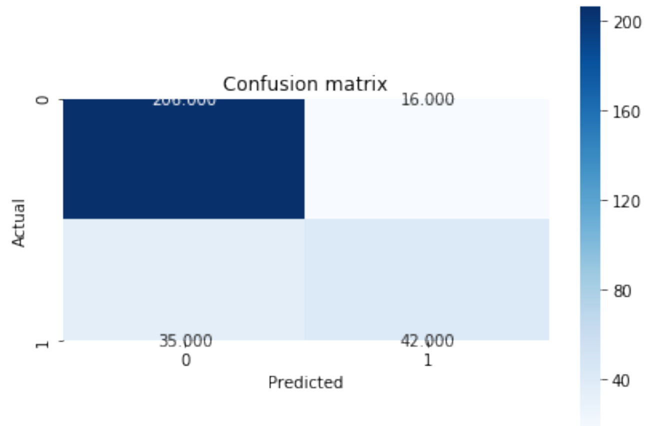
Model 2. KNN with k = 3 and the variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'

In [856]:
```
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train_scaled_df[['Percent White, not Hispanic or Lati
no', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Lat
ino', 'Percent Less than Bachelor\'s Degree']], y_train)
```

Out[856]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
                     metric_params=None, n_jobs=None, n_neighbors=3,
p=2,
                     weights='uniform')
```

In [857]:
```
y_pred = classifier.predict(x_validation_scaled_df[['Percent White, no
t Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Perce
nt Hispanic or Latino', 'Percent Less than Bachelor\'s Degree']])
```

In [858]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [859]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - metrics.accuracy_score(y_validation, y_pred)
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.8294314381270903, 0.1705685618729097, array([0.85477178, 0.724137
93]), array([0.92792793, 0.54545455]), array([0.88984881, 0.62222222
])]
```

4c. Support Vector Machines Using SVM to predict whether a county is Democratic or Republican

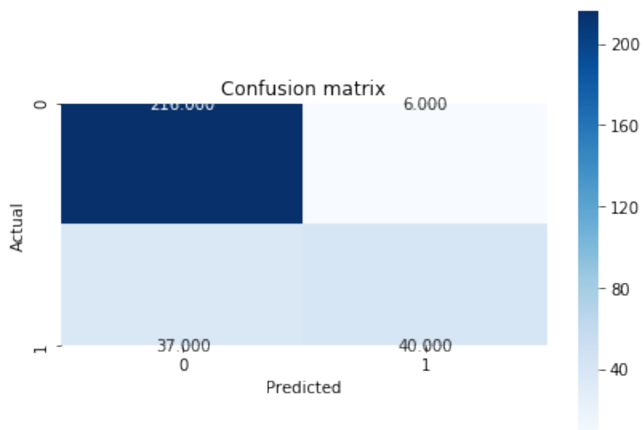
Model 1. SVM using all variables

In [860]:
```
classifier = SVC(kernel = 'rbf')
classifier.fit(x_train_scaled_df, y_train)
```

Out[860]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated'
,
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [861]:
```
y_pred = classifier.predict(x_validation_scaled_df)
```

In [862]:
```
conf_matrix = metrics.confusion_matrix(y_validation, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



In [863]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - metrics.accuracy_score(y_validation, y_pred)
precision = metrics.precision_score(y_validation, y_pred, average = No
ne)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.8561872909698997, 0.14381270903010035, array([0.85375494, 0.86956
522]), array([0.97297297, 0.51948052]), array([0.90947368, 0.6504065
])]
```

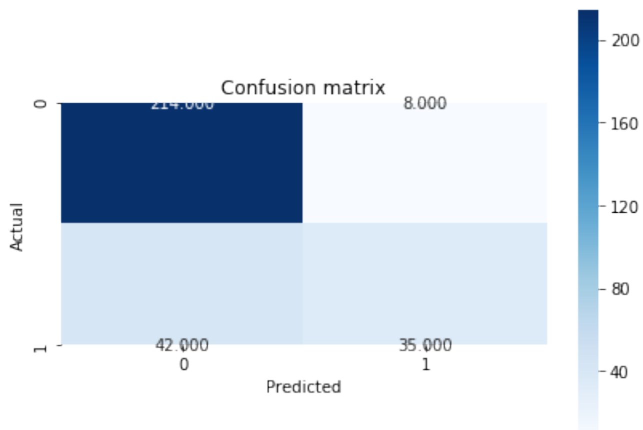
Model 2. SVM with 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degree' as predictor variables

In [864]:
```
classifier = SVC(kernel = 'rbf')
classifier.fit(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']], y_train)
```

Out[864]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated'
,
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [865]:
```
y_pred = classifier.predict(x_validation_scaled_df[['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degree']])
```

In [866]:
```
conf_matrix = metrics.confusion_matrix(y_validation, 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 [867]:
```
accuracy = metrics.accuracy_score(y_validation, y_pred)
error = 1 - metrics.accuracy_score(y_validation, y_pred)
precision = metrics.precision_score(y_validation, y_pred, average = None)
recall = metrics.recall_score(y_validation, y_pred, average = None)
F1_score = metrics.f1_score(y_validation, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.8327759197324415, 0.16722408026755853, array([0.8359375 , 0.81395
349]), array([0.96396396, 0.45454545]), array([0.89539749, 0.5833333
3])]
```

TASK 5

Creating clustering models using various clustering methods

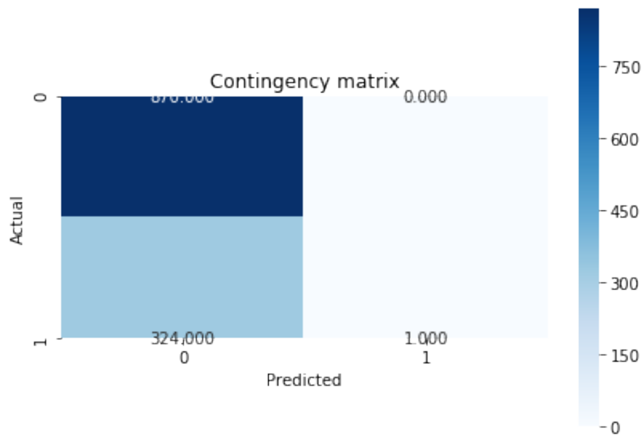
In [868]:
```
# Grab Data that we will work with
X = data_election[['FIPS', 'Total Population', 'Percent White, not His
panic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hi
spanic or Latino', 'Percent Foreign Born', 'Percent Female', 'Percent
Age 29 and Under', 'Percent Age 65 and Older', 'Median Household Incom
e', 'Percent Unemployed', 'Percent Less than High School Degree', 'Per
cent Less than Bachelor\'s Degree', 'Percent Rural', 'Party']]
Y = data_election['Party']

# Standardize the data
scaler = StandardScaler()
scaler.fit(X)
X_standardized = scaler.transform(X)
```

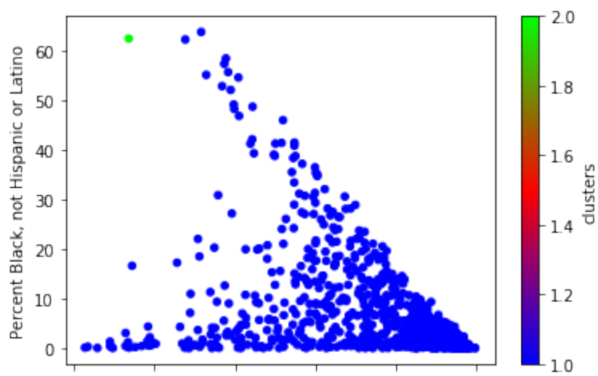
5a. Hierarchical Clustering with the Single Linkage Method

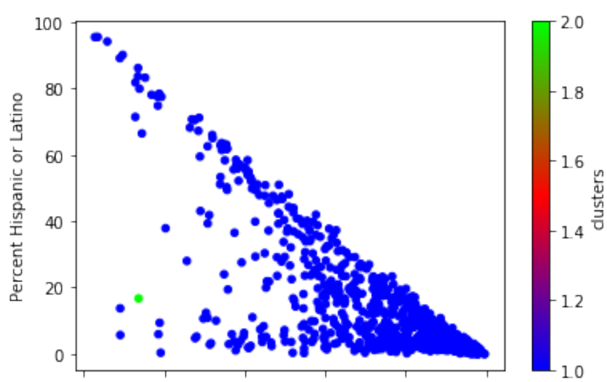
Clustering performed using variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born'

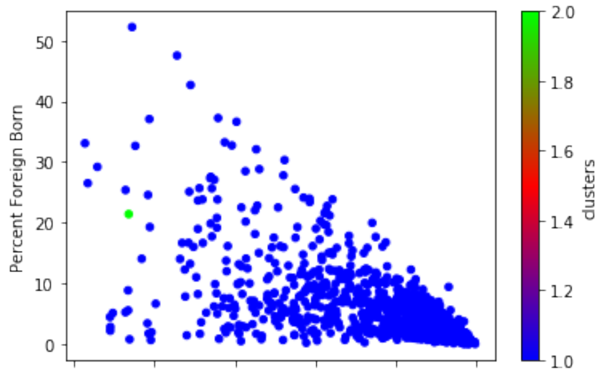
In [869]:
```
#Task 5a - Model hierarchical clustering with single linkage method
  #5a.i variables - 'Percent White, not Hispanic or Latino', 'Percent
Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent
Foreign Born'
scaler = StandardScaler()
scaler.fit(data_election[['Percent White, not Hispanic or Latino', 'Pe
rcent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'P
ercent Foreign Born']])
x = scaler.transform(data_election[['Percent White, not Hispanic or La
tino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or L
atino', 'Percent Foreign Born']])

clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using hierarchical clustering with single linkag
e method
data_election['clusters'] = clusters
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
```

```
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
```









In [870]:
```
# Evaluation Calculations for hierarchical clustering with single link
age method
adjusted_rand_index = metrics.adjusted_rand_score(Y, data_election['cl
usters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

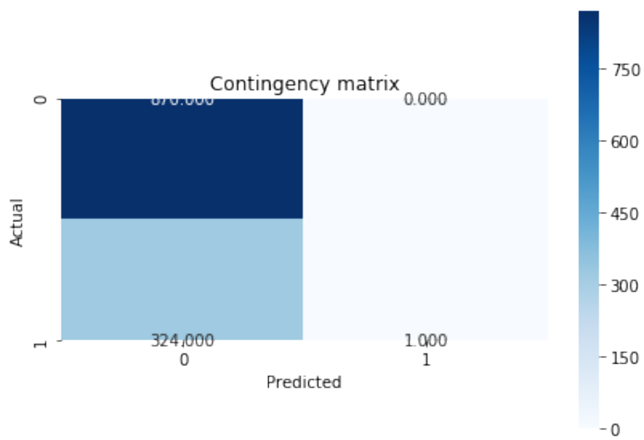
```
adjusted Rand index:  0.0028041107323011935  Silhouette coefficient:
0.6967676709484538
```

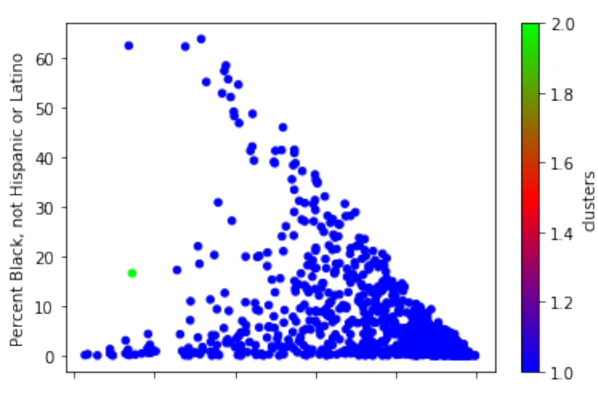
5b. Hierarchical Clustering with Single Linkage Method

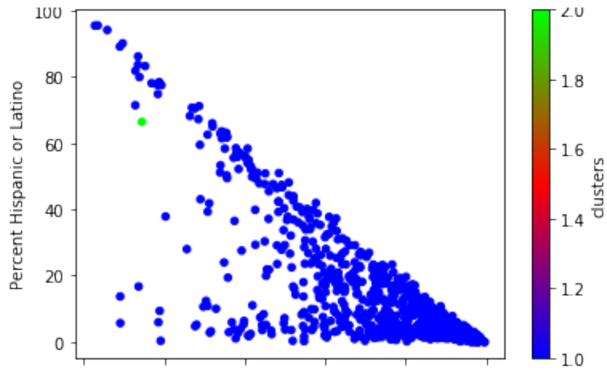
Clustering uses variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older'

In [871]:
```
#Task 5b - Model hierarchical clustering with single linkage method
  #5b.i variables - 'Percent White, not Hispanic or Latino', 'Percent
Black, not Hispanic or Latino', 'Percent Hispanic or Latino',
  #            'Percent Foreign Born', 'Percent Age 29 and Under', 'Pe
rcent Age 65 and Older'
scaler = StandardScaler()
scaler.fit(data_election[['Percent White, not Hispanic or Latino', 'Pe
rcent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'P
ercent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and
Older']])
x = scaler.transform(data_election[['Percent White, not Hispanic or La
tino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or L
atino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent A
ge 65 and Older']])
```

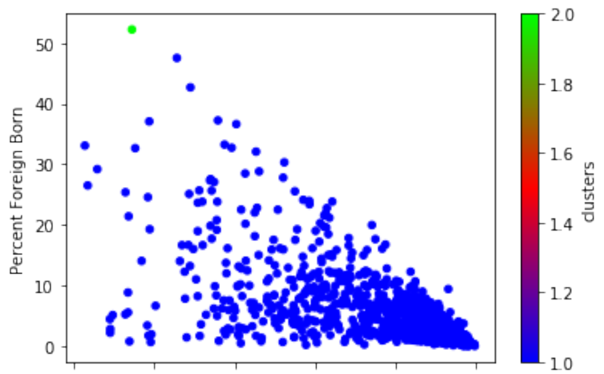
```
clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using hierarchical clustering with single linkag
e method
data_election['clusters'] = clusters

ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Age 29 and Under', c = 'clusters', color
map = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Age 65 and Older', c = 'clusters', color
map = plt.cm.brg)
```



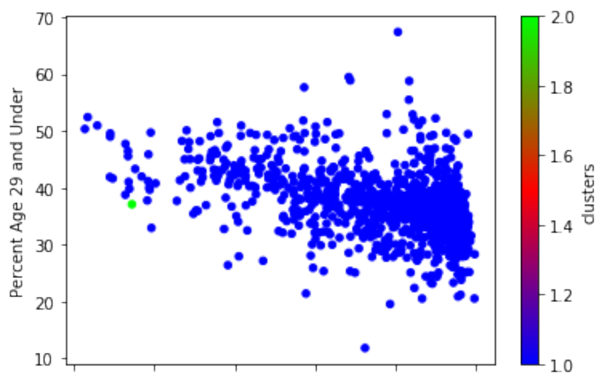




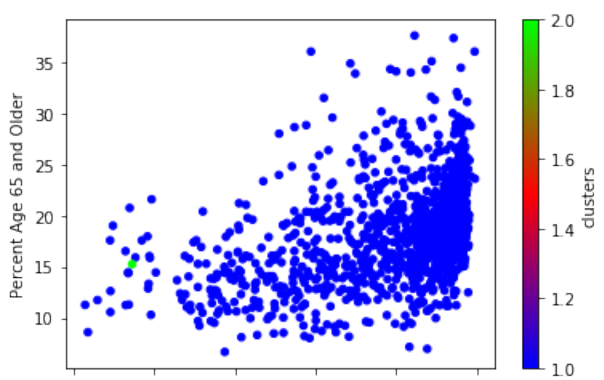
Percent Hispanic or Latino
clusters



Percent Foreign Born
clusters



Percent Age 29 and Under
clusters



Percent Age 65 and Older
clusters

In [872]:
```
# Evaluation calculations for this model
adjusted_rand_index = metrics.adjusted_rand_score(Y, data_election['cl
usters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

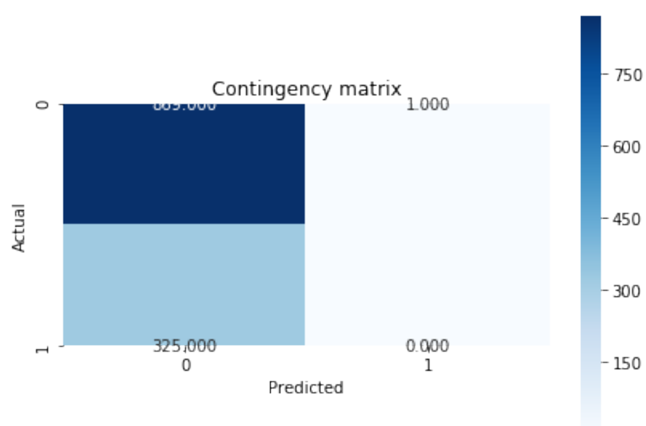
```
adjusted Rand index:  0.0028041107323011935  Silhouette coefficient:
0.670572365919519
```

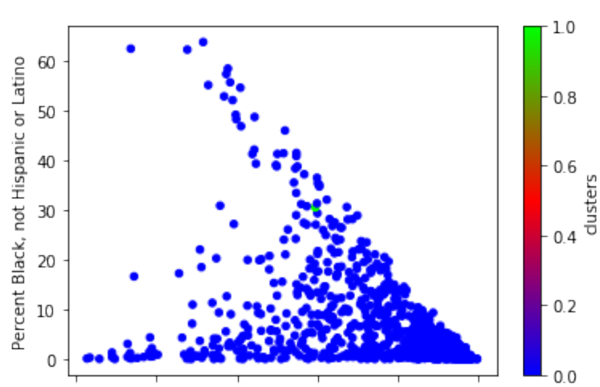
5c. Hierarchical Clustering with Single Linkage Method

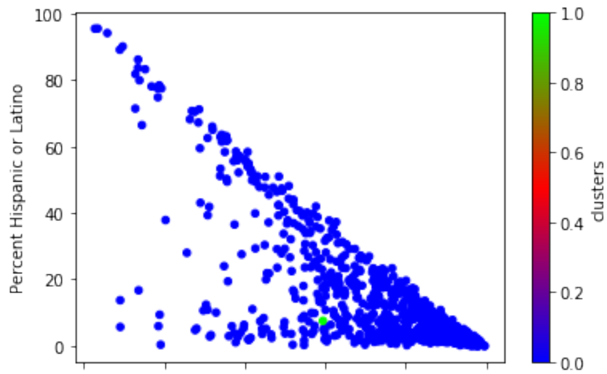
Clustering uses variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Female'

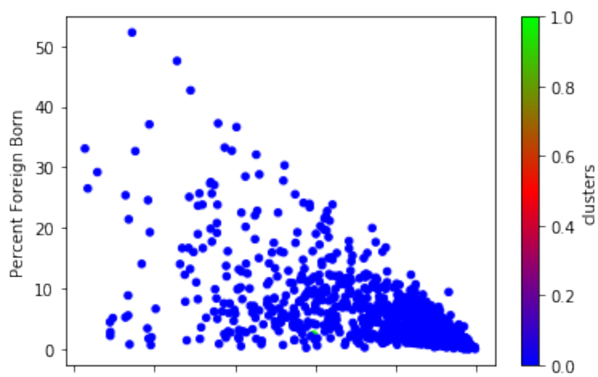
In [873]:
```
#Task 5c - Model hierarchical clustering with single linkage method
  #5c.i variables - 'Percent White, not Hispanic or Latino', 'Percent
Black, not Hispanic or Latino', 'Percent Hispanic or Latino',
  #            'Percent Foreign Born', 'Percent Female'
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black,
not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreig
n Born', 'Percent Female']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino', 'Perc
ent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Per
cent Foreign Born', 'Percent Female']])

clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
```

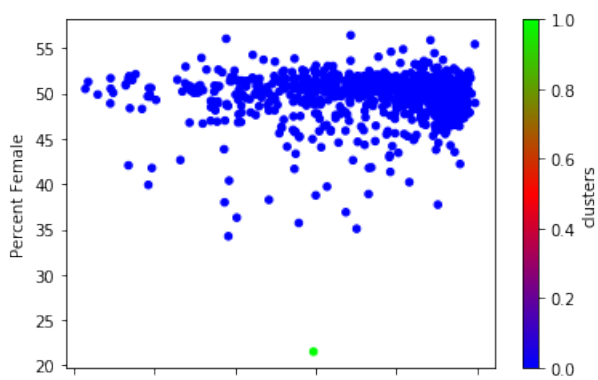
```
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using hierarchical clustering with single linkag
e method
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Female', c = 'clusters', colormap = plt.
cm.brg)
```











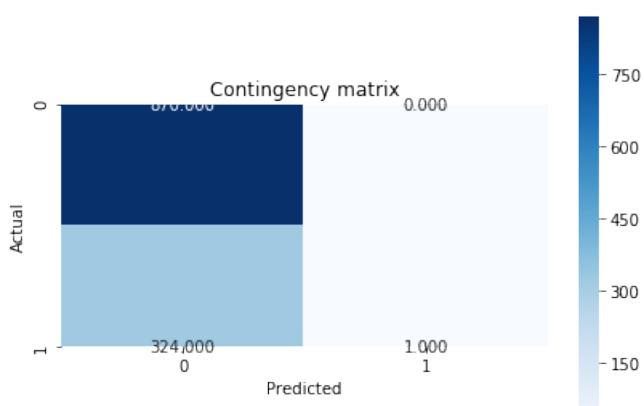
In [874]:
```
# Evaluation calculations for this model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

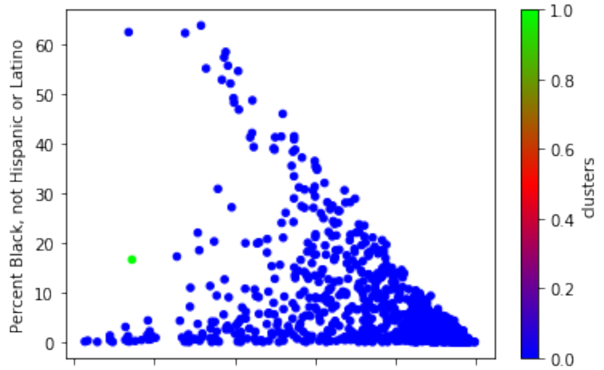
```
adjusted Rand index:  -0.001047512629882871  Silhouette coefficient:
0.7946287431336253
```

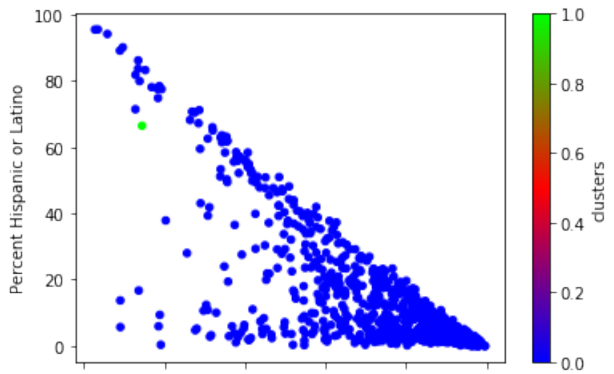
5d. Hierarchical Clustering with Single Linkage Method

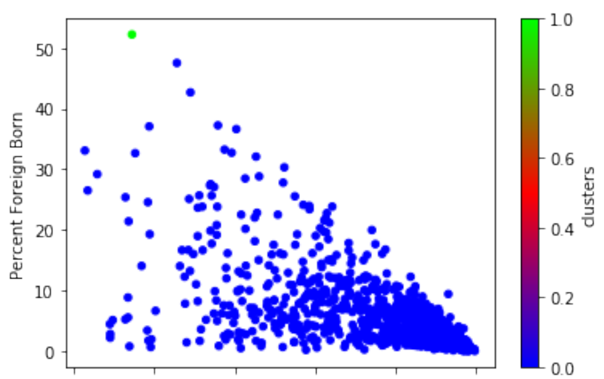
Clustering uses variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree'

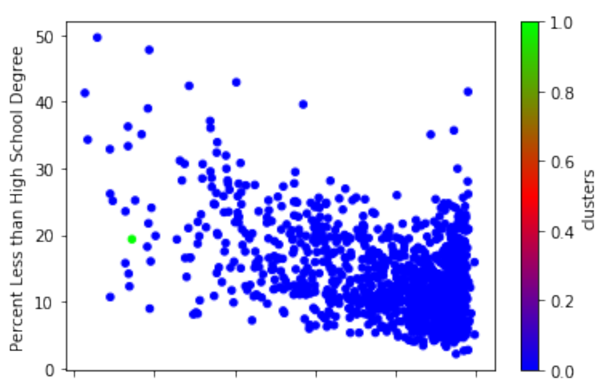
In [875]:
```
#Task 5d - Model hierarchical clustering with single linkage method
  #5d.i variables - 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino',
  #            'Percent Foreign Born', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree'
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree']])


clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
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()

# Plot clusters found using hierarchical clustering with single linkage method
data_election['clusters'] = clusters -1

ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Less than High School Degree', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'clusters', colormap = plt.cm.brg)
```

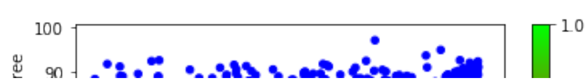


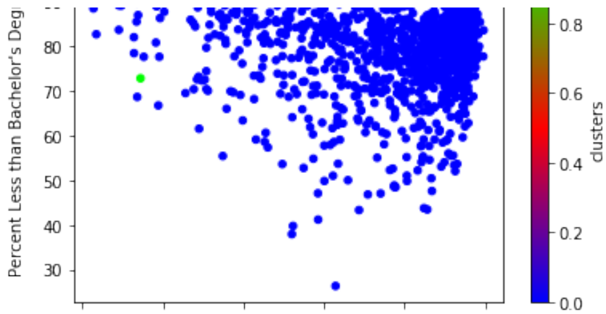

Percent Black, not Hispanic or Latino
0
10
20
30
40
50
60
clusters
0.0
0.2
0.4
0.6
0.8
1.0


Percent Hispanic or Latino
0
20
40
60
80
100
clusters
0.0
0.2
0.4
0.6
0.8
1.0


Percent Foreign Born
0
10
20
30
40
50
clusters
0.0
0.2
0.4
0.6
0.8
1.0


Percent Less than High School Degree
0
10
20
30
40
50
clusters
0.0
0.2
0.4
0.6
0.8
1.0


100
90
1.0



In [876]:
```
# Evaluation calculations for this model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

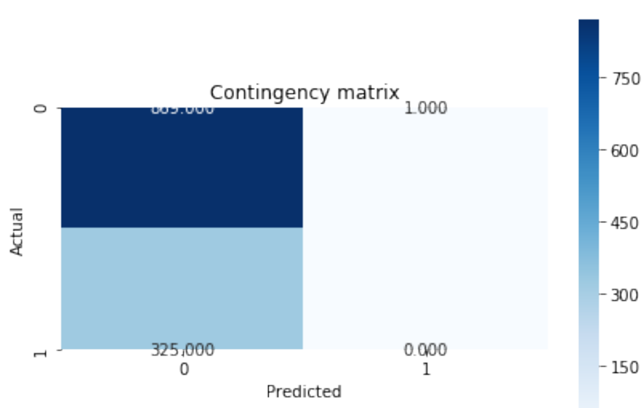
```
adjusted Rand index:  0.0028041107323011935  Silhouette coefficient:
0.6752810905295151
```

5e. Hierarchical Clustering with Single Linkage Method

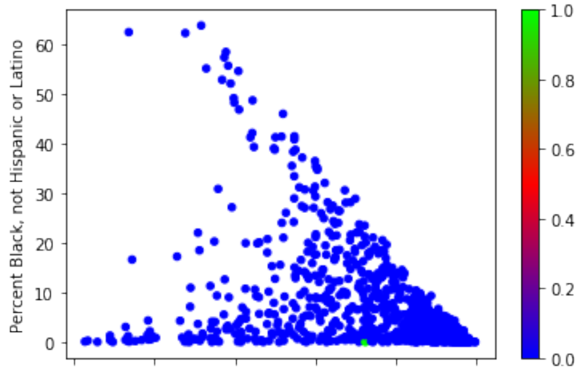
Clustering uses variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree'

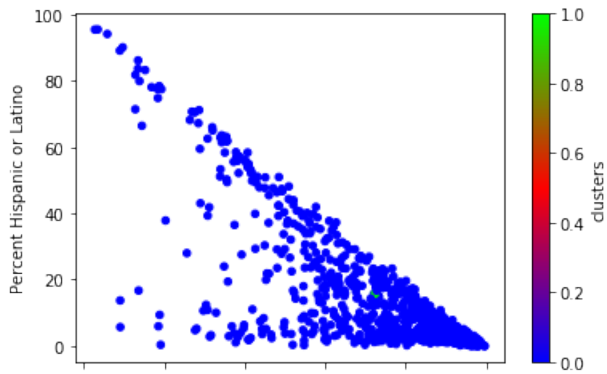
In [877]:

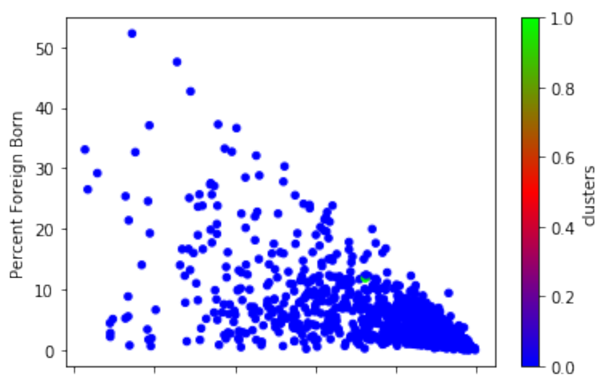
```python
# Standardize the data
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older','Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older','Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree']])


clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
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()

# Plot clusters found using hierarchical clustering with single linkage method
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Age 29 and Under', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Age 65 and Older', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Less than High School Degree', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hispanic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'clusters', colormap = plt.cm.brg)
```

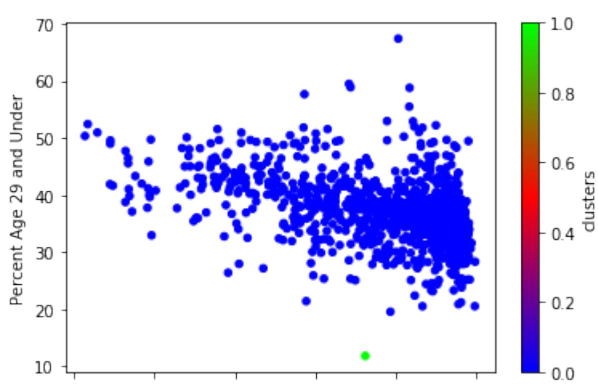


0

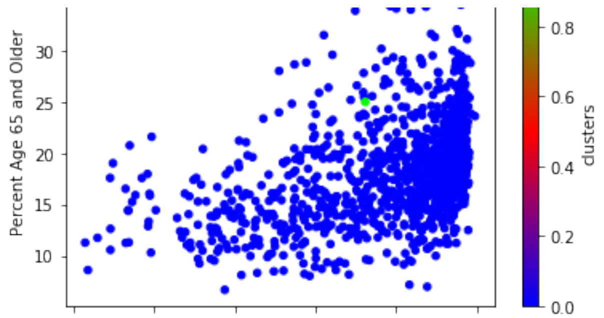

Percent Black, not Hispanic or Latino
clusters
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0

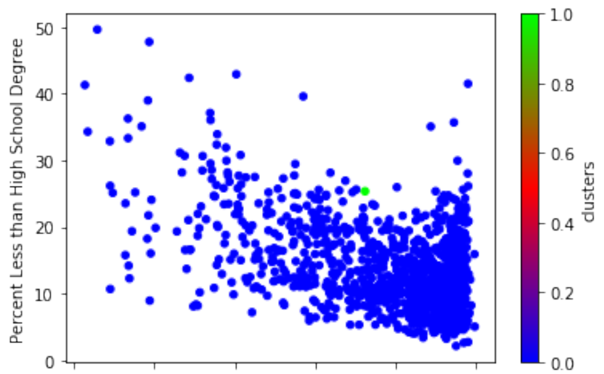

Percent Hispanic or Latino
clusters
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

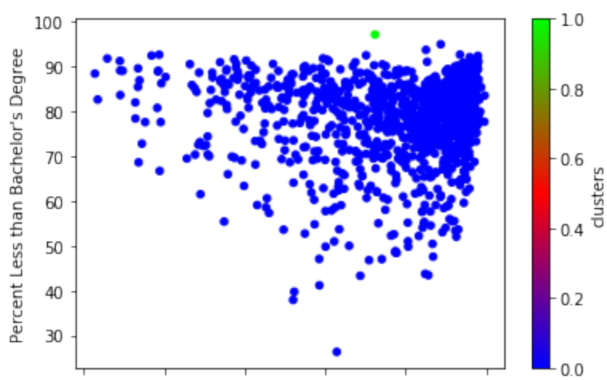

Percent Foreign Born
clusters
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0


Percent Age 29 and Under
clusters
10
20
30
40
50
60
70
0.0
0.2
0.4
0.6
0.8
1.0


35
1.0







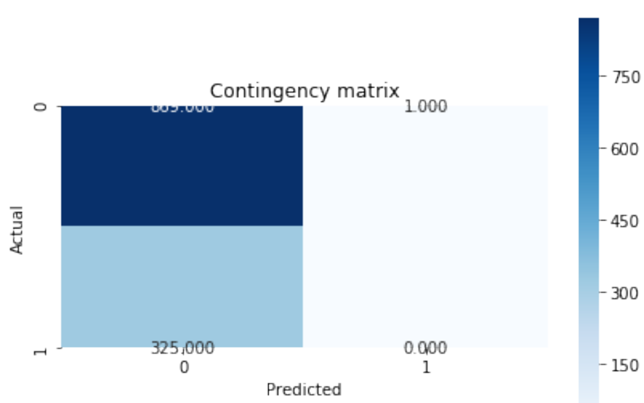
In [878]:
```
# Evaluation calculations this model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

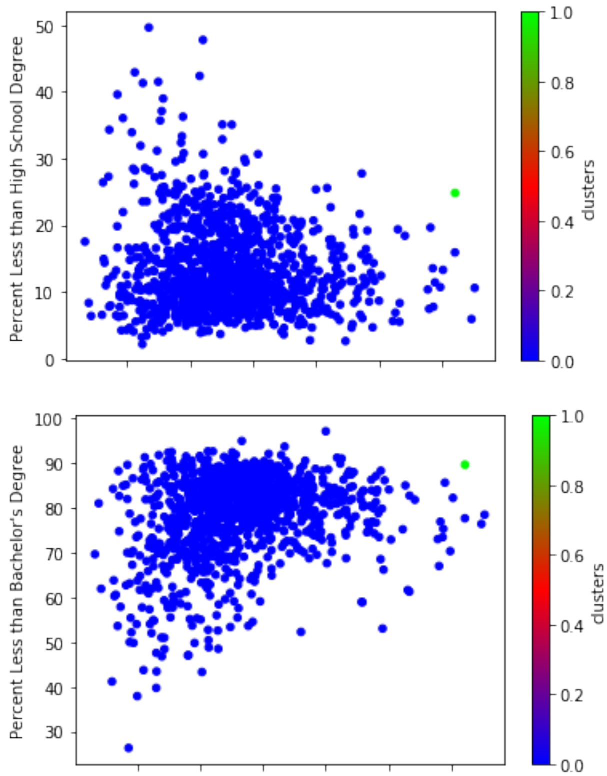
```
adjusted Rand index:  -0.001047512629882871  Silhouette coefficient:
0.4218980441370412
```

5f. Hierarchical Clustering with Single Linkage Method

Clustering uses variables 'Percent Age 65 and Older','Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree'

In [879]:
```
scaler = StandardScaler()
scaler.fit(X[['Percent Age 65 and Older','Percent Less than High Schoo
l Degree', 'Percent Less than Bachelor\'s Degree']])
x = scaler.transform(X[['Percent Age 65 and Older','Percent Less than
High School Degree', 'Percent Less than Bachelor\'s Degree' ]])


clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using hierarchical clustering with single linkag
e method
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent Age 65 and Olde
r', y = 'Percent Less than High School Degree', c = 'clusters', colorm
ap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent Age 65 and Olde
r', y = 'Percent Less than Bachelor\'s Degree', c = 'clusters', colorm
ap = plt.cm.brg)
```






```
In [880]: # Evaluation calculations for this model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

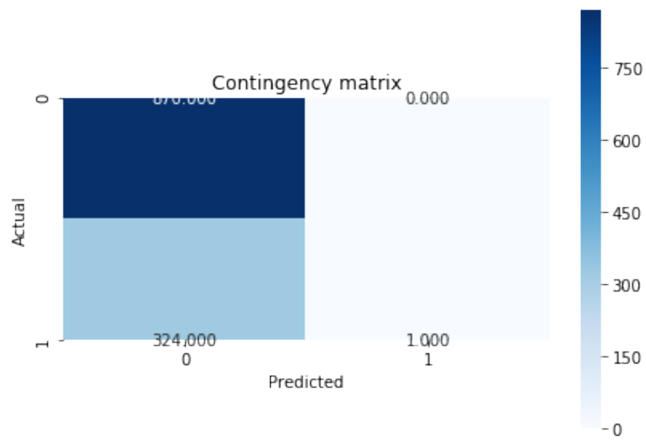
```
adjusted Rand index:  -0.001047512629882871  Silhouette coefficient:
0.5061973365950125
```

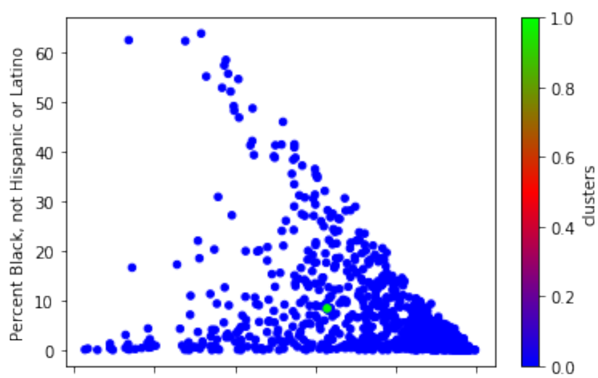
5g. Hierarchical Clustering with Single Linkage Method

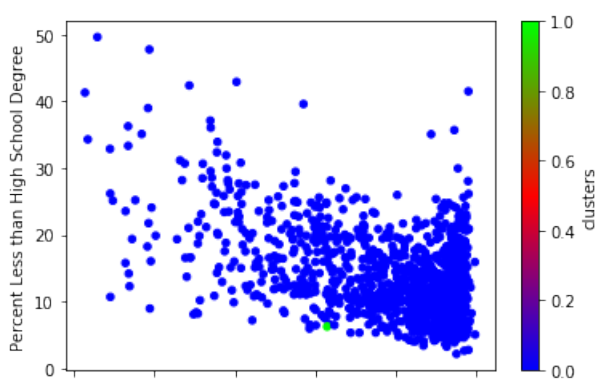
Clustering uses variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Less than High School Degree', 'Percent Less than Bachelor\'s Degree'

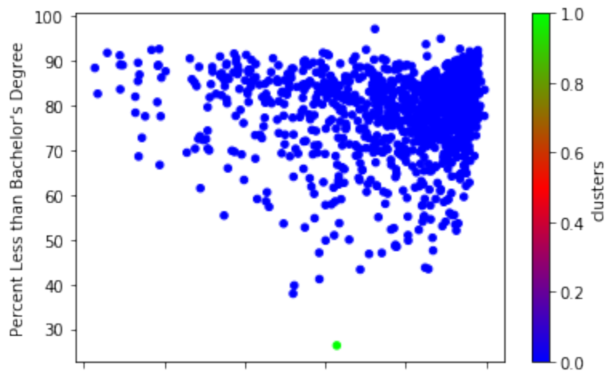
```
In [881]: # Standardizing the data
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black,
not Hispanic or Latino', 'Percent Less than High School Degree', 'Perc
ent Less than Bachelor\'s Degree']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino','Perce
nt Black, not Hispanic or Latino','Percent Less than High School Degre
e', 'Percent Less than Bachelor\'s Degree']])


clustering = linkage(x, method= 'single', metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
```

```
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using hierarchical clustering with single linkag
e method
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Less than High School Degree', c = 'clus
ters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'clus
ters', colormap = plt.cm.brg)
```









In [882]:
```
# Evaluation calculations for this model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

```
adjusted Rand index:  0.0028041107323011935  Silhouette coefficient:
0.5846363456702045
```

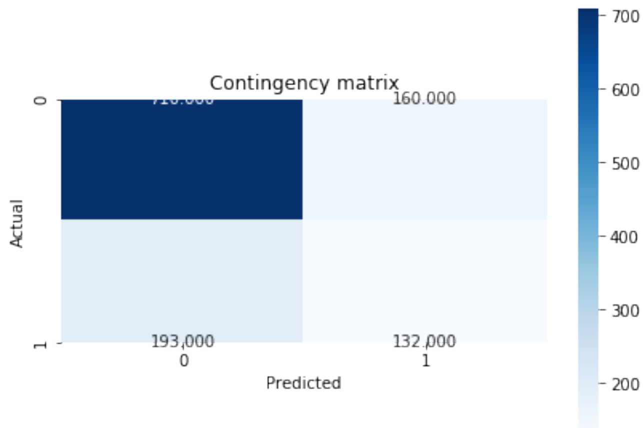
In [883]:
```
#************************************************END OF hierarchical c
lustering with single linkage method**********************************
*********
```

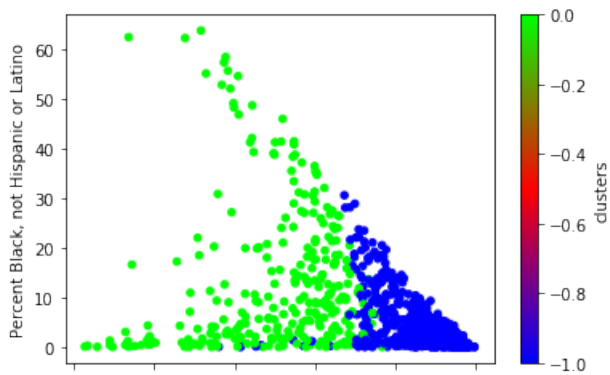
5h. K-Means Clustering

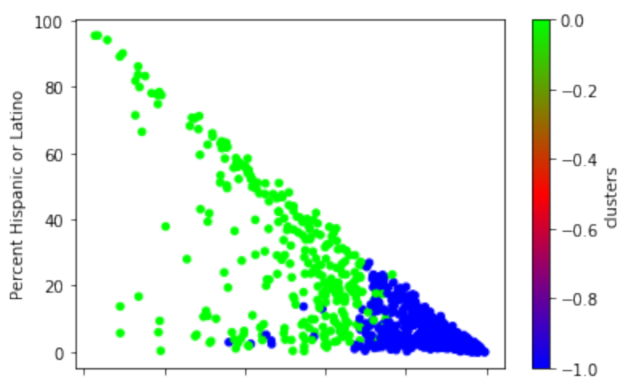
Clustering using variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born'

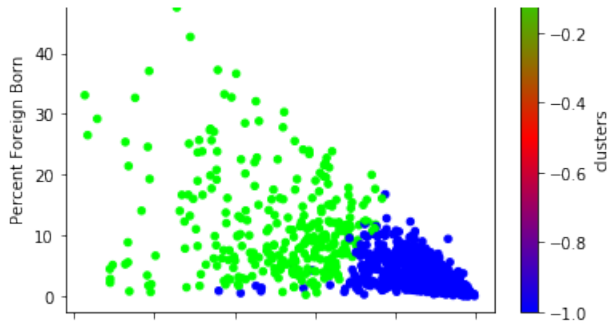
In [884]:
```
#Task 5h - Model KMeans Clustering
  #5h.i variables - 'Percent White, not Hispanic or Latino', 'Percent
Black, not Hispanic or Latino',
  #                  'Percent Hispanic or Latino', 'Percent Foreign Bor
n'
x = X_standardized[:,2:6]

clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, rand
om_state = 0).fit(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
```

```
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
```









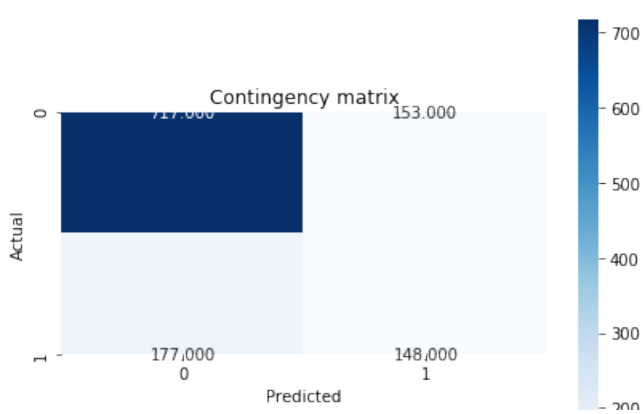
In [885]:
```
#Task 5h.ii - Evaluation Calculations for hierarchical clustering with single linkage method
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['clusters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coefficient: ", silhouette_coefficient)
```

```
adjusted Rand index:  0.11911877926404817  Silhouette coefficient:
0.5818101112791731
```

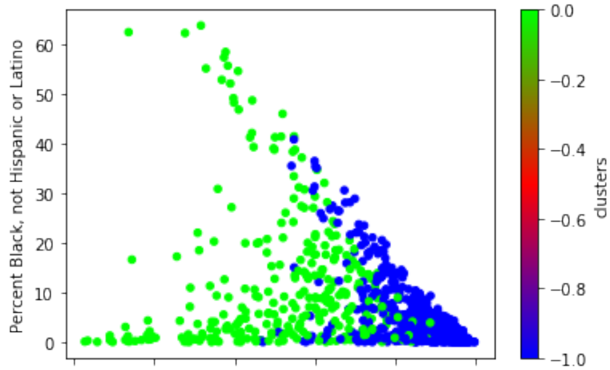
5i. K-Means Clustering

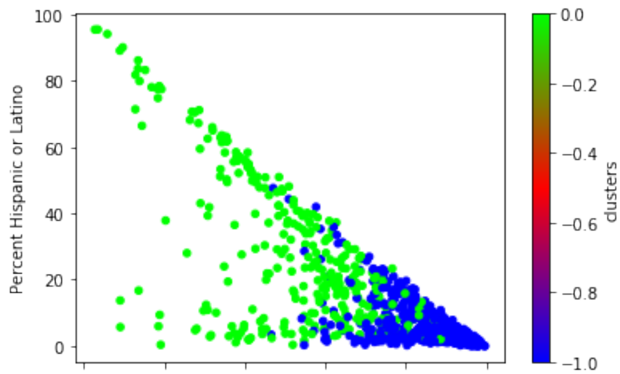
Clustering using variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older'

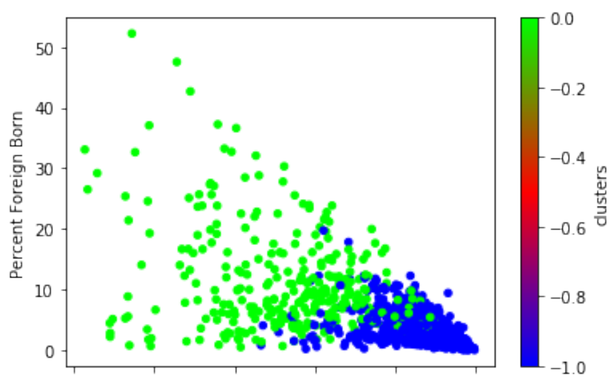
In [886]:
```
#Task 5i - Model KMeans Clustering
  #5i.i variables - 'Percent White, not Hispanic or Latino', 'Percent
Black, not Hispanic or Latino', 'Percent Hispanic or Latino',
  #            'Percent Foreign Born', 'Percent Age 29 and Under', 'Pe
rcent Age 65 and Older'
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black,
not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreig
n Born', 'Percent Age 29 and Under', 'Percent Age 65 and Older']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino', 'Perc
ent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Per
cent Foreign Born', 'Percent Age 29 and Under', 'Percent Age 65 and Ol
der']])

clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, rand
om_state = 0).fit(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Age 29 and Under', c = 'clusters', color
map = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Age 65 and Older', c = 'clusters', color
map = plt.cm.brg)
```

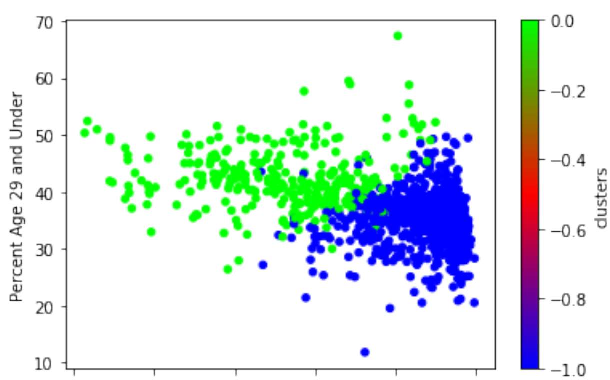


200

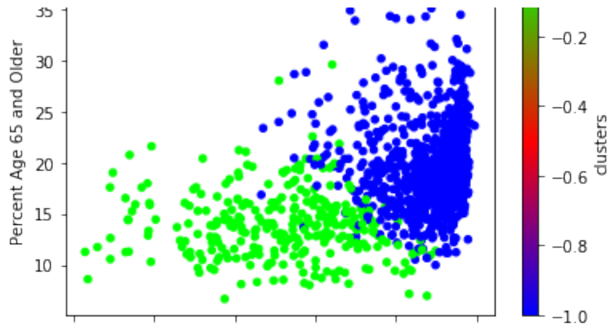

Percent Black, not Hispanic or Latino
60
50
40
30
20
10
0
clusters
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


Percent Hispanic or Latino
100
80
60
40
20
0
clusters
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


Percent Foreign Born
50
40
30
20
10
0
clusters
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


Percent Age 29 and Under
70
60
50
40
30
20
10
clusters
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


0.0



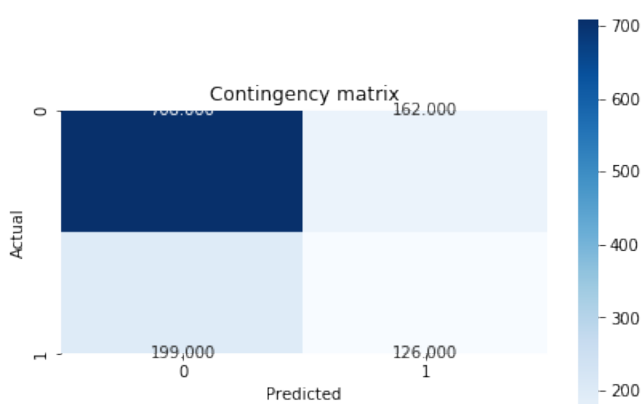
In [887]:
```
#Task 5i.ii - Evaluation Calculations for model
adjusted_rand_index = metrics.adjusted_rand_score(Y, clusters-1)
silhouette_coefficient = metrics.silhouette_score(X_standardized, clus
ters-1, metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

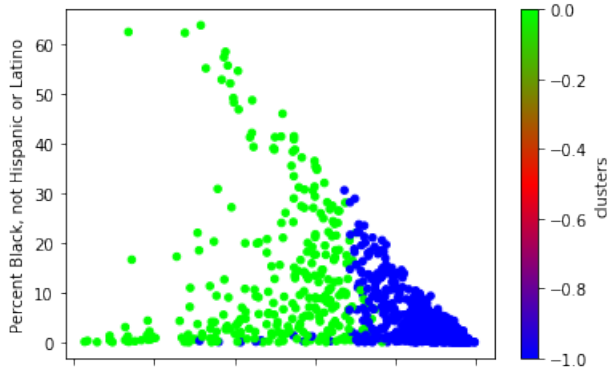
```
adjusted Rand index:  0.15682610655732362  Silhouette coefficient:
0.2906553943228404
```

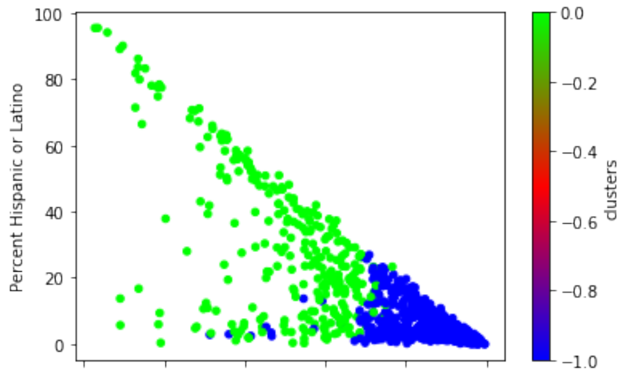
5j. K-Means Clustering

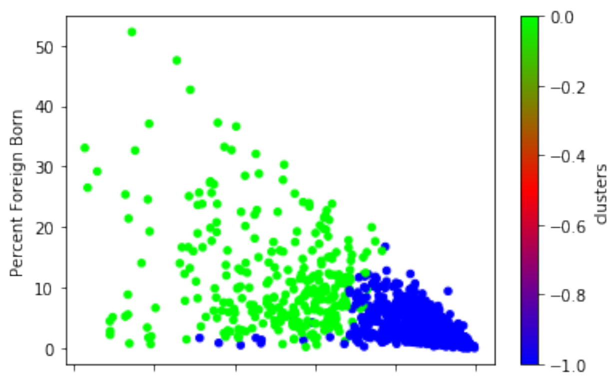
Clustering using variables 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', 'Percent Female'

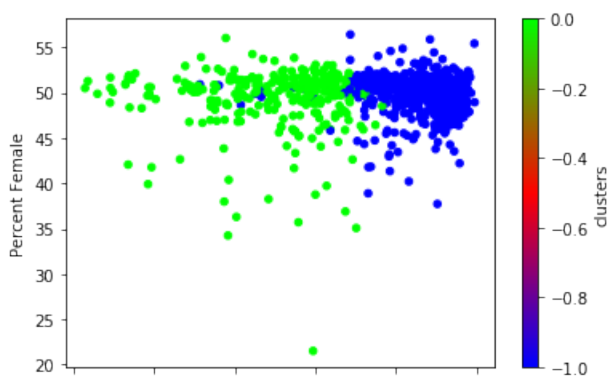
In [888]:
```
#Task 5j - Model KMeans Clustering
#5j.i variables - 'Percent White, not Hispanic or Latino', 'Percent Bl
ack, not Hispanic or Latino', 'Percent Hispanic or Latino',
  #            'Percent Foreign Born', 'Percent Female'
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black,
not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreig
n Born', 'Percent Female']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino', 'Perc
ent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Per
cent Foreign Born', 'Percent Female']])

clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, rand
om_state = 0).fit(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Female', c = 'clusters', colormap = plt.
cm.brg)
```











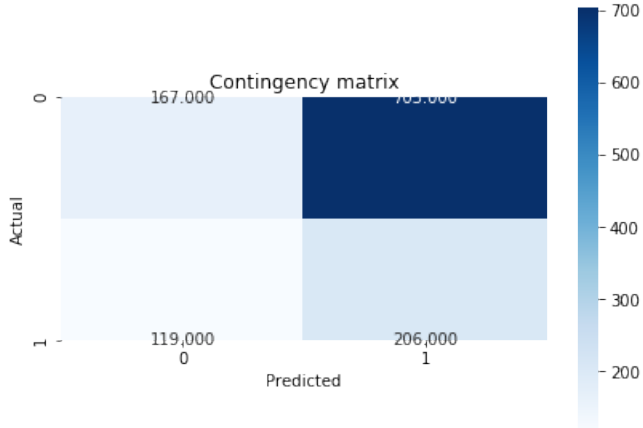
In [889]:
```
#Task 5j.ii - Evaluation Calculations for model
```

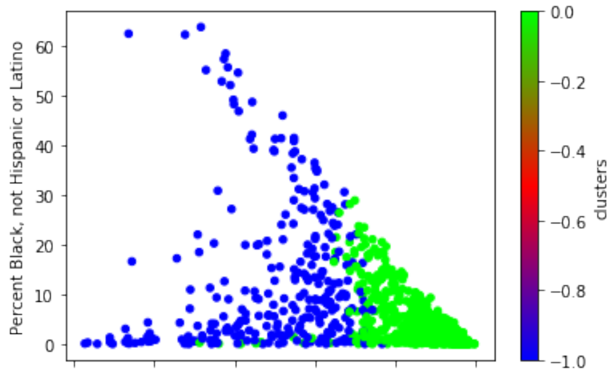
```
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

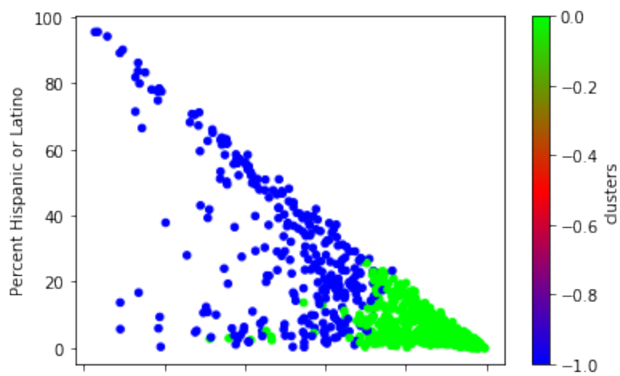
```
adjusted Rand index:  0.10636374173342371  Silhouette coefficient:
0.5223223037625178
```

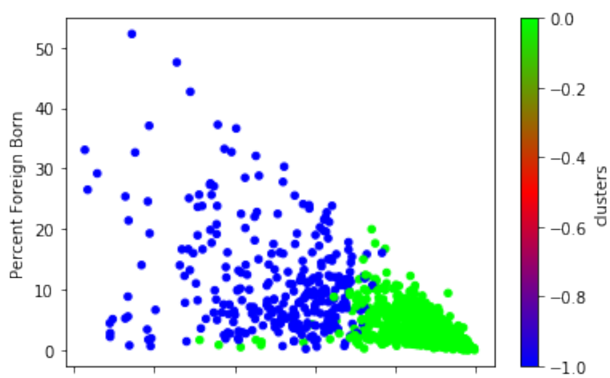
In [890]:

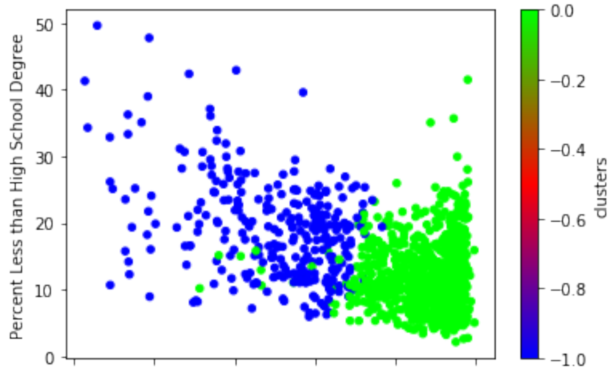
```
#Task 5k - Model KMeans Clustering
  #5k.i variables - 'Percent White, not Hispanic or Latino', 'Percent
Black, not Hispanic or Latino', 'Percent Hispanic or Latino',
  #            'Percent Foreign Born', 'Percent Less than High School
Degree', 'Percent Less than Bachelor\'s Degree'
scaler = StandardScaler()
scaler.fit(X[['Percent White, not Hispanic or Latino', 'Percent Black,
not Hispanic or Latino', 'Percent Hispanic or Latino', 'Percent Foreig
n Born', 'Percent Less than High School Degree', 'Percent Less than Ba
chelor\'s Degree']])
x = scaler.transform(X[['Percent White, not Hispanic or Latino', 'Perc
ent Black, not Hispanic or Latino', 'Percent Hispanic or Latino', 'Per
cent Foreign Born', 'Percent Less than High School Degree', 'Percent L
ess than Bachelor\'s Degree']])

clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, rand
om_state = 0).fit(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1

ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'clu
sters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'clusters', col
ormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'clusters', colormap
= plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Less than High School Degree', c = 'clus
ters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'clus
ters', colormap = plt.cm.brg)
```

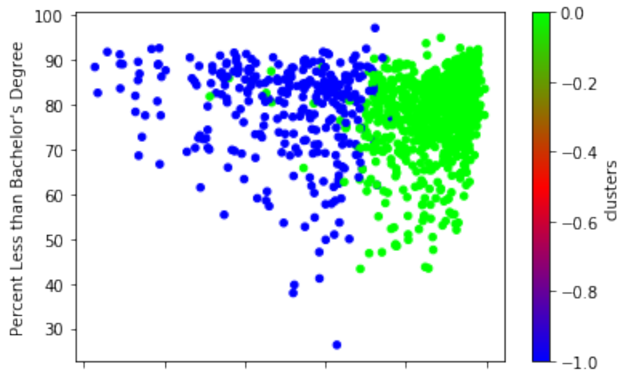

Contingency matrix
167.000
119.000
206.000
Actual
Predicted
0
1
700
600
500
400
300
200


Percent Black, not Hispanic or Latino
clusters
0
10
20
30
40
50
60
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


Percent Hispanic or Latino
clusters
0
20
40
60
80
100
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


Percent Foreign Born
clusters
0
10
20
30
40
50
0.0
−0.2
−0.4
−0.6
−0.8
−1.0





In [891]:
```
#Task 5k.ii - Evaluation Calculations for model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

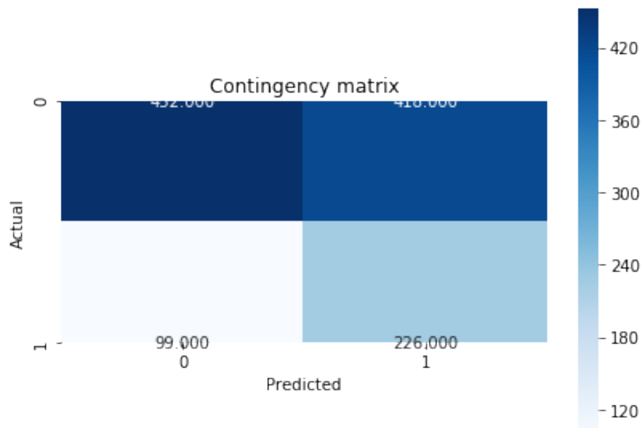
```
adjusted Rand index:  0.08924775988883304  Silhouette coefficient:
0.4635101311098164
```

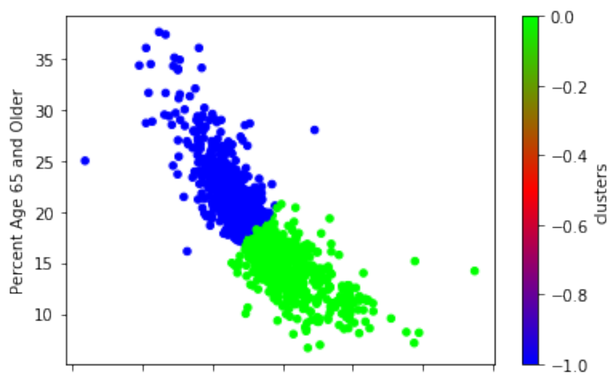
5l. K-Means Clustering

Clustering using variables 'Percent Age 29 and Under', 'Percent Age 65 and Older'

In [892]:
```
#Task 5l - Model KMeans Clustering
  #5l.i variables - 'Percent Age 29 and Under', 'Percent Age 65 and Ol
der'
scaler = StandardScaler()
scaler.fit(X[['Percent Age 29 and Under', 'Percent Age 65 and Older']]
)
x = scaler.transform(X[['Percent Age 29 and Under', 'Percent Age 65 an
d Older']])

clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, rand
om_state = 0).fit(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
sns.heatmap(cont_matrix, annot = True, fmt = ".2f", square = True, cma
```

```
sns.heatmap(cont_matrix, annot = True, fmt = '.3f', square = True, cma
p = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent Age 29 and Unde
r', y = 'Percent Age 65 and Older', c = 'clusters', colormap = plt.cm.
brg)
```





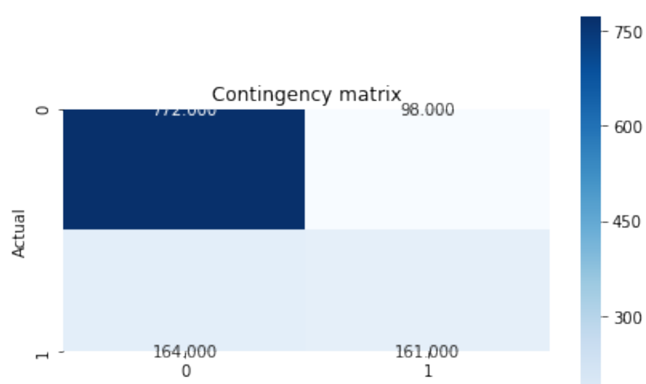
In [893]:
```
#Task 5l.ii - Evaluation Calculations for model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

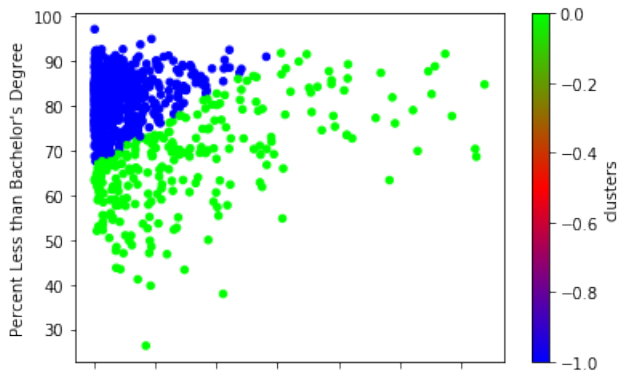
```
adjusted Rand index:  0.016263794172611586  Silhouette coefficient:
0.4621003253492598
```

5m. K-Means Clustering

Clustering using variables 'Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'

In [894]:
```
#Task 5m - Model KMeans Clustering
  #5m.i variables - 'Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'
scaler = StandardScaler()
scaler.fit(X[['Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree']])
x = scaler.transform(X[['Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree']])

clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10, random_state = 0).fit(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
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()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent Black, not Hispanic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'clusters', colormap = plt.cm.brg)
```





In [895]:
```
#Task 5m.ii - Evaluation Calculations for model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['clusters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coefficient: ", silhouette_coefficient)
```

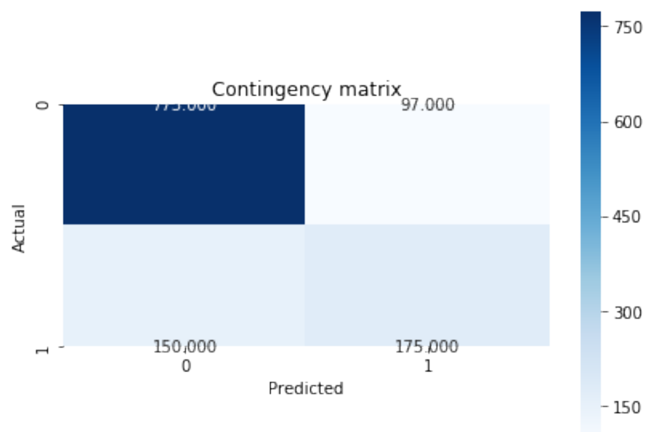
```
adjusted Rand index:  0.26595586689147394  Silhouette coefficient:
0.5445444266793265
```

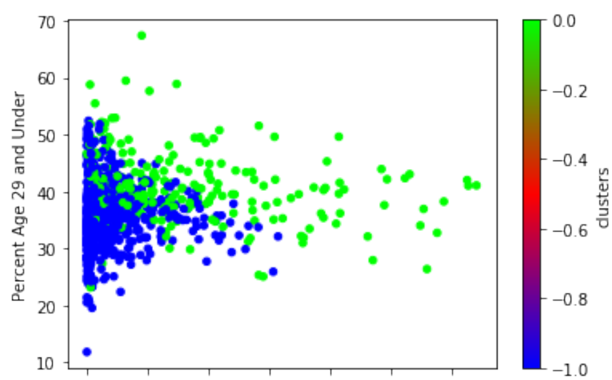
5n. K-Means Clustering

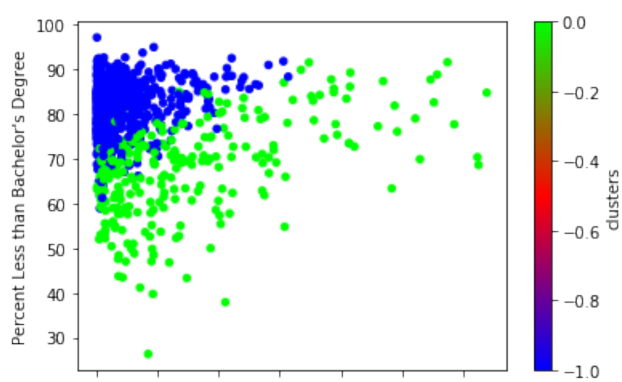
Clustering using variables 'Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'

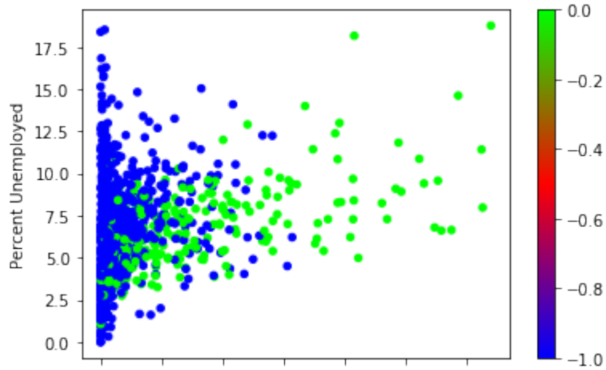
In [896]:
```
#Task 5n - Model KMeans Clustering
  #5n.i variables - 'Percent Black, not Hispanic or Latino', 'Percent Less than Bachelor\'s Degree'
scaler = StandardScaler()
scaler.fit(X[['Percent Black, not Hispanic or Latino', 'Percent Age 29 and Under','Percent Less than Bachelor\'s Degree', 'Percent Unemployed']])
x = scaler.transform(X[['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(x)
clusters = clustering.labels_

cont_matrix = metrics.cluster.contingency_matrix(Y, clusters-1)
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()

# Plot clusters found using KMeans clustering of 2 clusters
data_election['clusters'] = clusters -1
ax = data_election.plot(kind = 'scatter', x = 'Percent Black, not Hispanic or Latino', y = 'Percent Age 29 and Under', c = 'clusters', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent Black, not Hispanic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'clusters', colormap = plt.cm.brg)
```

```
ax = data_election.plot(kind = 'scatter', x = 'Percent Black, not Hisp
anic or Latino', y = 'Percent Unemployed', c = 'clusters', colormap =
plt.cm.brg)
```









In [897]:
```
#Task 5n.ii - Evaluation Calculations for model
adjusted_rand_index = metrics.adjusted_rand_score(data_election['Party
'], data_election['clusters'])
silhouette_coefficient = metrics.silhouette_score(x, data_election['cl
usters'], metric = "euclidean")
print("adjusted Rand index: ", adjusted_rand_index, " Silhouette coeff
icient: ", silhouette_coefficient)
```

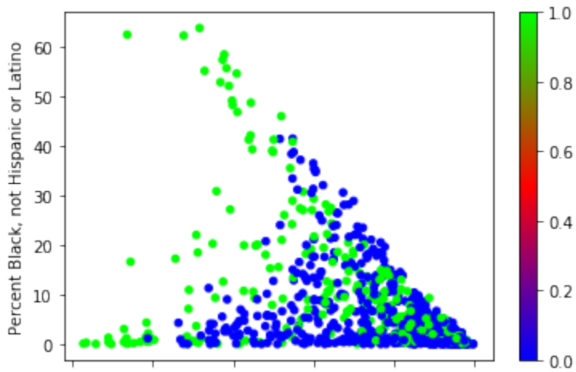
```
adjusted Rand index:  0.30061428945310187  Silhouette coefficient:
0.3250057188451443
```

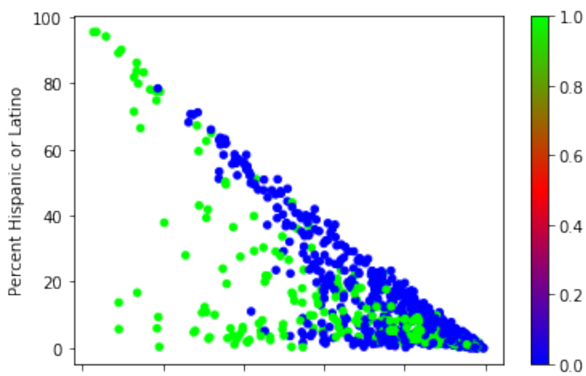
5o. Finding True Clusters

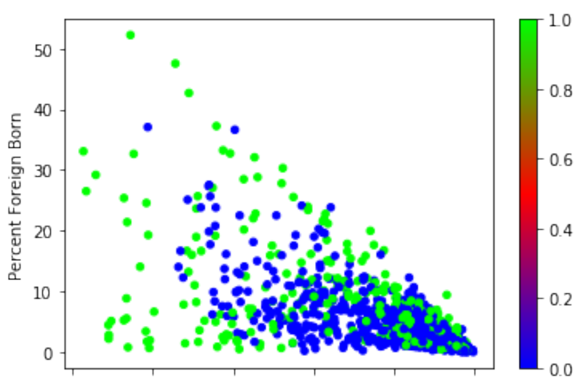
In [898]:
```
# Plot true clusters for the predictor, which is the party for each co
unty
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Black, not Hispanic or Latino', c = 'Par
ty', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Hispanic or Latino', c = 'Party', colorm
ap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Foreign Born', c = 'Party', colormap = p
lt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Age 29 and Under', c = 'Party', colormap
= plt.cm.brg)
```

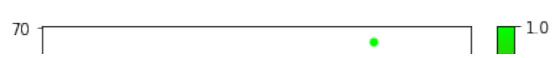
```
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Age 65 and Older', c = 'Party', colormap
= plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Less than High School Degree', c = 'Part
y', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Less than Bachelor\'s Degree', c = 'Part
y', colormap = plt.cm.brg)
ax = data_election.plot(kind = 'scatter', x = 'Percent White, not Hisp
anic or Latino', y = 'Percent Female', c = 'Party', colormap = plt.cm.
brg)
silhouette_coefficient = metrics.silhouette_score(X_standardized, Y, m
etric = "euclidean")
print("Silhouette coefficient: ",silhouette_coefficient)
```

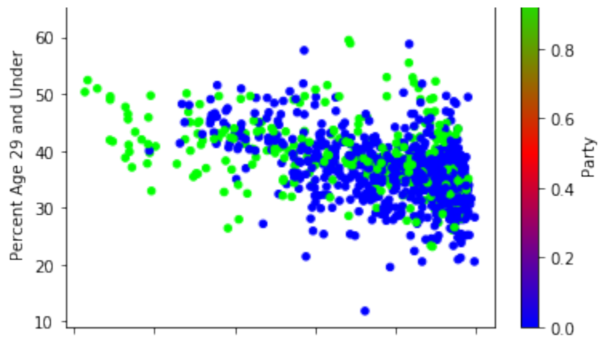
```
Silhouette coefficient:  0.21427376755203847
```

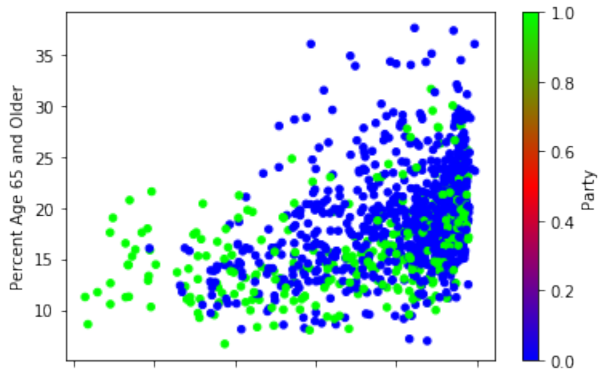


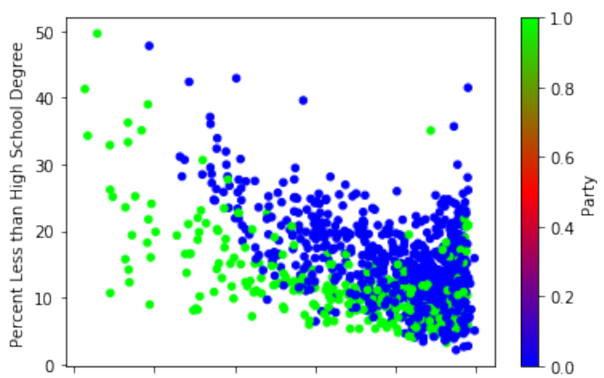


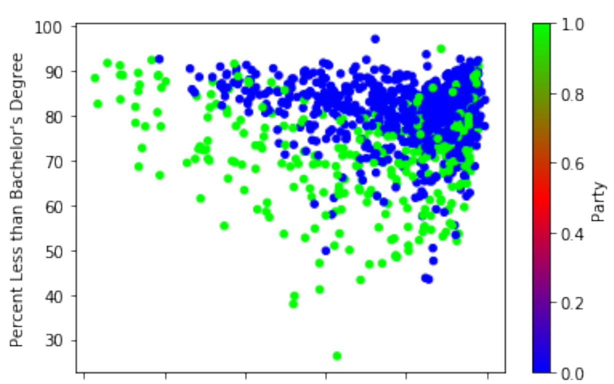


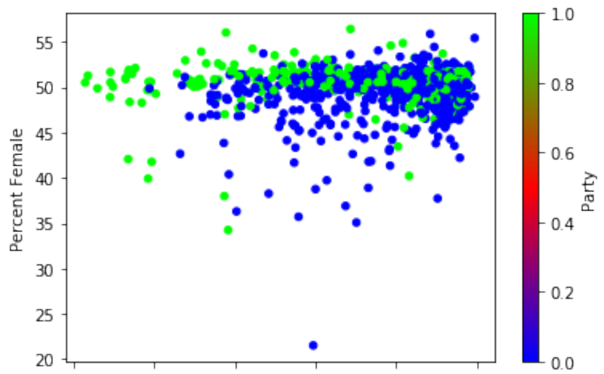



Percent Age 29 and Under
60
50
40
30
20
10
Party
0.8
0.6
0.4
0.2
0.0


Percent Age 65 and Older
35
30
25
20
15
10
Party
1.0
0.8
0.6
0.4
0.2
0.0


Percent Less than High School Degree
50
40
30
20
10
0
Party
1.0
0.8
0.6
0.4
0.2
0.0


Percent Less than Bachelor's Degree
100
90
80
70
60
50
40
30
Party
1.0
0.8
0.6
0.4
0.2
0.0



TASK 6

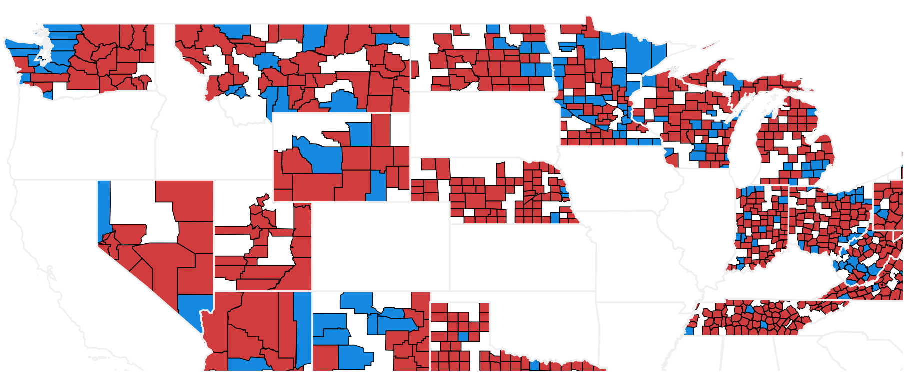
Creating a choropleth using the best classifier model from task 4. The chosen model was *Model 4* using Decision Trees.

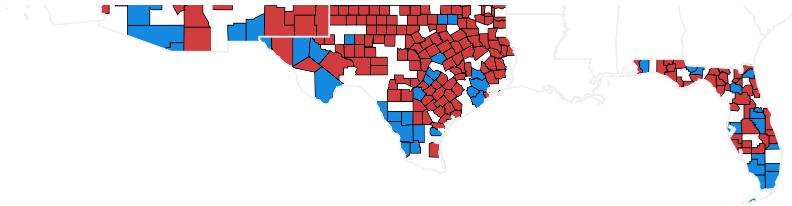
In [899]:
```
# Scaling the all of the data
data_election = pd.read_csv('merged_train.csv')
all_data = data_election[['State', 'County', 'FIPS', 'Total Population
', 'Percent White, not Hispanic or Latino', 'Percent Black, not Hispan
ic or Latino', 'Percent Hispanic or Latino', 'Percent Foreign Born', '
Percent Female', 'Percent Age 29 and Under', 'Percent Age 65 and Older
', 'Median Household Income', 'Percent Unemployed', 'Percent Less than
High School Degree', 'Percent Less than Bachelor\'s Degree', 'Percent
Rural', 'Democratic', 'Republican']]
full_data = all_data.select_dtypes(include=[np.int64,np.float64])
full_data = full_data.iloc[:,1:14]

# Standardizing the full dataset
scaler = StandardScaler()
scaler.fit(x_train)
full_data_scaled = scaler.transform(full_data)
full_data_scaled_df = pd.DataFrame(full_data_scaled,index = full_data.
index,columns=full_data.columns)

# Classifying using Classifier Model 3 (Decision Tree)
best_prediction = classifier_party.predict(full_data_scaled_df[['Perce
nt White, not Hispanic or Latino', 'Percent Black, not Hispanic or Lat
ino', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Deg
ree']])

# Merging with FIPS for choropleth preparation
best_data = pd.DataFrame({'Party': best_prediction, 'FIPS': all_data['
FIPS']})


# Create a map of Democratic & Republic counties with FIPS codes based
on the dataset
import plotly.figure_factory as ff
from plotly.offline import init_notebook_mode, iplot # Needed to rende
r the figure when exporting to HTML
init_notebook_mode(connected=True)

fips = best_data['FIPS'].tolist()
party_values = best_data['Party'].map({0: 'Republican', 1: 'Democratic
'})
colorscale = ["#1689E0", "#D13D3F"]
figure = ff.create_choropleth(fips=fips,
                              values=party_values,
```

```
                                  colorscale=colorscale,
                                  county_outline={'color': '#000000', 'wid
th': 0.5},
                                  title='Political Party by Counties via D
ecision Tree',
                                  legend_title='Political Party')
figure.layout.template = None
iplot(figure, validate=False) # Displaying figure even when exported t
o HTML
```

```
/Users/lydia/opt/anaconda3/lib/python3.7/site-packages/pandas/core/f
rame.py:7123: FutureWarning:

Sorting because non-concatenation axis is not aligned. A future vers
ion
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=T
rue'.
```

Political Party by Counties via Decisior





TASK 7

Predicting the number of votes cast using best performing regression and classification models.

In [900]:
```
# Load Election dataset
data_election = pd.read_csv('demographics_test.csv')
data_election.head()
```

Out[900]:

	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
0	NV	eureka	32011	1730	98.265896	0.057803	0.462428	0.346821	51.156069
1	TX	zavala	48507	12107	5.798299	0.594697	93.326175	9.193029	49.723301
2	VA	king george	51099	25260	73.804434	16.722090	4.441805	2.505938	50.166271
3	OH	hamilton	39061	805965	66.354867	25.654340	2.890944	5.086945	51.870615
4	TX	austin	48015	29107	63.809393	8.479060	25.502456	9.946061	50.671660

In [901]:
```
x_test = data_election.select_dtypes(include=[np.int64,np.float64])
x_test = x_test.iloc[:,1:14]
x_test_scaled = scaler.transform(x_test)
x_test_scaled_df = pd.DataFrame(x_test_scaled,index = x_test.index,col
umns=x_test.columns)
```

In [902]:
```
y_predicted_democratic = fitted_model_democratic.predict(x_test_scaled
_df[['Total Population', 'Percent Black, not Hispanic or Latino', 'Per
cent Less than Bachelor\'s Degree']])
data_election['Democratic'] = y_predicted_democratic
```

In [903]:
```
y_predicted_republican = fitted_model_republican.predict(x_test_scaled
_df[['Total Population', 'Percent White, not Hispanic or Latino', 'Per
cent Hispanic or Latino', 'Percent Foreign Born', 'Percent Age 65 and
Older', 'Percent Unemployed', 'Median Household Income', 'Percent Rura
l']])
data_election['Republican'] = y_predicted_republican
```

In [904]:

```
y_predicted_party = classifier_party.predict(x_test_scaled_df[['Percen
t White, not Hispanic or Latino', 'Percent Black, not Hispanic or Lati
no', 'Percent Hispanic or Latino', 'Percent Less than Bachelor\'s Degr
ee']])
data_election['Party'] = y_predicted_party
data_election.head()
```

Out[904]:

	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	
0	NV	eureka	32011	1730	98.265896	0.057803	0.462428	0.346821	51.156069	2
1	TX	zavala	48507	12107	5.798299	0.594697	93.326175	9.193029	49.723301	4
2	VA	king george	51099	25260	73.804434	16.722090	4.441805	2.505938	50.166271	4
3	OH	hamilton	39061	805965	66.354867	25.654340	2.890944	5.086945	51.870615	4
4	TX	austin	48015	29107	63.809393	8.479060	25.502456	9.946061	50.671660	3

In [905]:

```
sample_output = data_election[['State','County', 'Democratic', 'Republ
ican', 'Party']]
sample_output.head()
```

Out[905]:

	State	County	Democratic	Republican	Party
0	NV	eureka	-4368.133477	10279.986522	0
1	TX	zavala	-9771.647091	-87.022736	1
2	VA	king george	21823.049764	18795.181860	0
3	OH	hamilton	183669.476767	112375.441324	1
4	TX	austin	7294.738614	6193.106586	0

In [906]:

```
num_data = sample_output._get_numeric_data()
num_data[num_data < 0] = 0
sample_output.head()
```

Out[906]:

	State	County	Democratic	Republican	Party
0	NV	eureka	0.000000	10279.986522	0
1	TX	zavala	0.000000	0.000000	1
2	VA	king george	21823.049764	18795.181860	0
3	OH	hamilton	183669.476767	112375.441324	1
4	TX	austin	7294.738614	6193.106586	0

In [907]:
```
sample_output.to_excel("output.xlsx")
```