```python
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.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
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 import metrics
import plotly.figure_factory as ff
```

```python
# Load dataset
data_m = pd.read_csv('merged_train.csv')
data_m.head()
```

	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	Percent Age 29 and Under	Percent Age 65 and Older	H
0	AZ	apache	4001	72346	18.571863	0.486551	5.947806	1.719515	50.598513	45.854643	13.322091	32
1	AZ	cochise	4003	128177	56.299492	3.714395	34.403208	11.458374	49.069646	37.902276	19.756275	45
2	AZ	coconino	4005	138064	54.619597	1.342855	13.711033	4.825298	50.581614	48.946141	10.873943	51
3	AZ	gila	4007	53179	63.222325	0.552850	18.548675	4.249798	50.296170	32.238290	26.397638	40
4	AZ	graham	4009	37529	51.461536	1.811932	32.097844	4.385942	46.313518	46.393456	12.315809	47

```python
#Task 1 - Democratic
#hold out method - 75/25 split
x_train, x_val, y_train, y_val = train_test_split(data_m[['Total Population','Percent Whit
e, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or La
tino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Age 65 a
nd Older','Median Household Income','Percent Unemployed','Percent Less than High School De
gree','Percent Less than Bachelor\'s Degree','Percent Rural']],data_m['Democratic'],train_
size = 0.75, test_size = 0.25, random_state = 0)
```

```python
# Task 1 - Republican
x_trainR, x_valR, y_trainR, y_valR = train_test_split(data_m[['Total Population','Percent
 White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic
 or Latino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Ag
e 65 and Older','Median Household Income','Percent Unemployed','Percent Less than High Sch
ool Degree','Percent Less than Bachelor\'s Degree','Percent Rural']],data_m['Republican'],
train_size = 0.75, test_size = 0.25, random_state = 0)
```

```python
#Task 2 - Democratic
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_val)
```

```python
#Task 2 - Republican
scaler2 = StandardScaler()
scaler2.fit(x_trainR)
x_train_scaledR = scaler2.transform(x_trainR)
x_test_scaledR = scaler2.transform(x_valR)
```

```python
#Task 3
#Checking first for a model with 1 predictors. - Democratic

pred_variable = ['Total Population']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[0]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[0]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[0]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.8638217161080632, 0.9168242212210275]
```

```python
#Task 3
#Checking first for a model with 1 predictors. - Republican

pred_variableR = ['Total Population']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[0]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[0]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[0]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.8408841359394673, 0.6567852066304897]
```

```python
# Democratic
pred_variable = ['Percent Black, not Hispanic or Latino']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[2]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[2]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[2]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.06441005732765726, -0.029478245381724166]
```

```python
# Republican
pred_variableR = ['Percent Black, not Hispanic or Latino']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[2]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[2]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[2]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.02976562142702599, 0.008761983512348914]
```

```python
# Democratic
pred_variable = ['Percent White, not Hispanic or Latino']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[1]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[1]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[1]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.08807441622521417, -0.18066649287431247]
```

```python
# Republican
pred_variableR = ['Percent White, not Hispanic or Latino']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[4]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[4]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[4]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.2013945530434873, -0.04192994961377905]
```

```python
# Democratic
pred_variable = ['Percent Foreign Born']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[4]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[4]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[4]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.26909173090031835, -0.022049557731990577]
```

```python
# Republican
pred_variableR = ['Percent Foreign Born']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[4]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[4]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[4]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.2013945530434873, -0.04192994961377905]
```

```python
# Democratic
pred_variable = ['Percent Hispanic or Latino']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[3]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[3]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[3]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.019047759624118088, -0.15304097506195524]
```

```python
# Republican
pred_variableR = ['Percent Hispanic or Latino']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[3]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[3]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[3]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.017676681576438313, -0.09604877594940153]
```

```python
# Democratic
pred_variable = ['Percent Female']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[5]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[5]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[5]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.02659028720385059, -0.043963892167613754]
```

```python
# Republican
pred_variableR = ['Percent Female']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[5]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[5]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[5]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.027038353770389656, 0.005573819984737271]
```

```python
# Democratic
pred_variable = ['Median Household Income']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[8]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[8]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[8]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.09991989387898992, -0.12899300631884847]
```

```python
# Republican
pred_variableR = ['Median Household Income']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[8]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[8]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[8]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.10966034398267865, -0.013253496832915879]
```

```python
# Democratic
pred_variable = ['Percent Age 29 and Under']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[6]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[6]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[6]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.024201837144587124, -0.022599018827570651]
```

```
# Republican
pred_variableR = ['Percent Age 29 and Under']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[6]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[6]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[6]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.02251881639058828, 0.013893267546554289]
```

```
# Democratic
pred_variable = ['Percent Age 65 and Older']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[7]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[7]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[7]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.06520429887643853, -0.004506569367753066]
```

```
# Republican
pred_variableR = ['Percent Age 65 and Older']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[7]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[7]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[7]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.058876933946248156, 0.021148965999515212]
```

```python
# Democratic
pred_variable = ['Percent Unemployed']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[9]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[9]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[9]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.003268862518394866, -0.04574285017569246]
```

```python
# Republican
pred_variableR = ['Percent Unemployed']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[9]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[9]], y = y_trainR) # R squared (trainin
g)
score_valR = modelR.score(X = x_test_scaledR[:,[9]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.0020938545466878677, -0.009809248436253286]
```

```python
# Democratic
pred_variable = ['Percent Less than High School Degree']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[10]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[10]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[10]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.010248412551217334, -0.033057384710943 66]
```

```python
# Republican
pred_variableR = ['Percent Less than High School Degree']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[10]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[10]], y = y_trainR) # R squared (traini
ng)
score_valR = modelR.score(X = x_test_scaledR[:,[10]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.016683640421142787, 0.015264707860547233]
```

```python
# Democratic
pred_variable = ['Percent Less than Bachelor\'s Degree']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[11]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[11]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[11]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.2059503958747709, -0.031001227682307064]
```

```python
# Republican
pred_variableR = ['Percent Less than Bachelor\'s Degree']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[11]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[11]], y = y_trainR) # R squared (traini
ng)
score_valR = modelR.score(X = x_test_scaledR[:,[11]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.1836003540444735, 0.047305206065186844]
```

```
# Democratic
pred_variable = ['Percent Rural']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[12]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[12]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[12]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.20759534900254295, 0.06340750436491682]
```

```
# Republican
pred_variableR = ['Percent Rural']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[12]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[12]], y = y_trainR) # R squared (traini
ng)
score_valR = modelR.score(X = x_test_scaledR[:,[12]], y = y_valR) # R squared (Validation)

print([score_trainR, score_valR])
```

```
[0.2292279982019585, 0.22042700912636426]
```

```
# NOW CHECKING WITH TWO PARAMETERS - DEMOCRATIC
pred_variable = ['Total Population','Median Household Income']
model = linear_model.LinearRegression().fit(X = x_train_scaled[:,[0,8]], y = y_train)

score_train = model.score(X = x_train_scaled[:,[0,8]], y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled[:,[0,8]], y = y_val) # R squared (Validation)

print([score_train, score_val])
```

```
[0.8695062788139603, 0.8979431172550174]
```

```python
# NOW CHECKING WITH TWO PARAMETERS - REPUBLICAN
pred_variableR = ['Total Population','Median Household Income']
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR[:,[0,8]], y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR[:,[0,8]], y = y_trainR) # R squared (train
ing)
score_valR = modelR.score(X = x_test_scaledR[:,[0,8]], y = y_valR) # R squared (Validatio
n)

print([score_trainR, score_valR])
```

```
[0.849787736207517, 0.6586920663174032]
```

```python
# NOW CHECKING WITH ALL PARAMETERS - DEMOCRATIC
model = linear_model.LinearRegression().fit(X = x_train_scaled, y = y_train)

score_train = model.score(X = x_train_scaled, y = y_train) # R squared (training)
score_val = model.score(X = x_test_scaled, y = y_val) # R squared (validation)
print([score_train, score_val])
```

```
[0.8807901193271512, 0.867055068427187]
```

```python
# NOW CHECKING WITH ALL PARAMETERS - REPUBLICAN
modelR = linear_model.LinearRegression().fit(X = x_train_scaledR, y = y_trainR)

score_trainR = modelR.score(X = x_train_scaledR, y = y_trainR) # R squared (training)
score_valR = modelR.score(X = x_test_scaledR, y = y_valR) # R squared (validation)
print([score_trainR, score_valR])
```

```
[0.8673465255785224, 0.7004235899502084]
```

```
# Checking first for a model with mulitiple predictors.
# We see that 'Median Household Income','Percent Rural','Percent Less than Bachelor\'s Deg
ree','Percent Less than High School Degree' is reducing our score. so we won't take them a
s predictors
pred_variable = ['Percent Age 29 and Under','Percent Age 65 and Older','Total Population',
'Percent Foreign Born','Percent Hispanic or Latino','Percent Black, not Hispanic or Latin
o','Percent White, not Hispanic or Latino','Percent Female','Percent Unemployed']
x_train, x_val, y_train, y_val = train_test_split(data_m[['Total Population','Percent Whit
e, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or La
tino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Age 65 a
nd Older','Median Household Income','Percent Unemployed','Percent Less than High School De
gree','Percent Less than Bachelor\'s Degree','Percent Rural']],data_m['Democratic'],train_
size = 0.75, test_size = 0.25, random_state = 0)


model = linear_model.LinearRegression().fit(X = x_train[pred_variable], y = y_train)
model1 = linear_model.LinearRegression().fit(X = x_train[pred_variable], y = y_train)

score_train = model.score(X = x_train[pred_variable], y = y_train) # R squared (training)
score_val = model.score(X = x_val[pred_variable], y = y_val) # R squared (validation)

m= model.coef_
c= model.intercept_

for i in range (0, len(x_val[pred_variable])):
    print("For county ",i+1," the  Predicted Democratic vote = ", (m*x_val[pred_variable].
iloc[i]).sum() + c)

print("Scores are \n")
print([score_train, score_val])
```

```
For county  1  the  Predicted Democratic vote =  1565.0364196412838
For county  2  the  Predicted Democratic vote =  3660.3329259876737
For county  3  the  Predicted Democratic vote =  67526.40800868615
For county  4  the  Predicted Democratic vote =  2795.866213918667
For county  5  the  Predicted Democratic vote =  27238.542225165875
For county  6  the  Predicted Democratic vote =  13805.94398336851
For county  7  the  Predicted Democratic vote =  -16015.163997371248
For county  8  the  Predicted Democratic vote =  19193.418195326998
For county  9  the  Predicted Democratic vote =  3637.347321290107
For county  10  the  Predicted Democratic vote =  12442.137558428165
For county  11  the  Predicted Democratic vote =  1342.6920254245251
For county  12  the  Predicted Democratic vote =  12072.524744495951
For county  13  the  Predicted Democratic vote =  14427.732043476253
For county  14  the  Predicted Democratic vote =  102378.35908144905
For county  15  the  Predicted Democratic vote =  18524.02870125405
For county  16  the  Predicted Democratic vote =  91219.70613793001
For county  17  the  Predicted Democratic vote =  12264.301130007116
For county  18  the  Predicted Democratic vote =  1137.971770302618
For county  19  the  Predicted Democratic vote =  1205.1117953403054
For county  20  the  Predicted Democratic vote =  8363.340882680011
For county  21  the  Predicted Democratic vote =  9952.812712164658
For county  22  the  Predicted Democratic vote =  33006.664080849834
For county  23  the  Predicted Democratic vote =  37222.26071525557
For county  24  the  Predicted Democratic vote =  17574.812582748997
For county  25  the  Predicted Democratic vote =  49439.33343070679
For county  26  the  Predicted Democratic vote =  124126.08201472252
For county  27  the  Predicted Democratic vote =  75426.6748945872
For county  28  the  Predicted Democratic vote =  3872.6580605474483
For county  29  the  Predicted Democratic vote =  -825.527499582302
For county  30  the  Predicted Democratic vote =  -21039.870729348204
For county  31  the  Predicted Democratic vote =  9259.27894057957
For county  32  the  Predicted Democratic vote =  -7574.275659264054
For county  33  the  Predicted Democratic vote =  16693.03506794083
For county  34  the  Predicted Democratic vote =  9818.12203989738
For county  35  the  Predicted Democratic vote =  4424.706588183679
For county  36  the  Predicted Democratic vote =  -5637.824248163866
For county  37  the  Predicted Democratic vote =  102562.1718190771
For county  38  the  Predicted Democratic vote =  2800.1913421576455
For county  39  the  Predicted Democratic vote =  51608.56342427942
For county  40  the  Predicted Democratic vote =  32493.12668913796
For county  41  the  Predicted Democratic vote =  118121.64788871893
For county  42  the  Predicted Democratic vote =  69593.5122428869
```

```
For county  43  the  Predicted Democratic vote =  6249.401556340395
For county  44  the  Predicted Democratic vote =  2221.9843304326814
For county  45  the  Predicted Democratic vote =  463.3385276745912
For county  46  the  Predicted Democratic vote =  13343.346841970684
For county  47  the  Predicted Democratic vote =  -312.123007871126
For county  48  the  Predicted Democratic vote =  19772.72465438836
For county  49  the  Predicted Democratic vote =  6722.784007353463
For county  50  the  Predicted Democratic vote =  1675.9066689063939
For county  51  the  Predicted Democratic vote =  44558.05439082652
For county  52  the  Predicted Democratic vote =  161639.77422358585
For county  53  the  Predicted Democratic vote =  12972.798729758451
For county  54  the  Predicted Democratic vote =  -4510.874009171581
For county  55  the  Predicted Democratic vote =  535847.667585148
For county  56  the  Predicted Democratic vote =  -40186.69052482926
For county  57  the  Predicted Democratic vote =  -6950.962328971878
For county  58  the  Predicted Democratic vote =  8381.789556213256
For county  59  the  Predicted Democratic vote =  8648.223961668587
For county  60  the  Predicted Democratic vote =  -10858.71601119317
For county  61  the  Predicted Democratic vote =  8614.208124562527
For county  62  the  Predicted Democratic vote =  78133.23194797328
For county  63  the  Predicted Democratic vote =  6274.782207081345
For county  64  the  Predicted Democratic vote =  5408.483370674598
For county  65  the  Predicted Democratic vote =  585.6979211173539
For county  66  the  Predicted Democratic vote =  5131.018241286869
For county  67  the  Predicted Democratic vote =  6909.834472198953
For county  68  the  Predicted Democratic vote =  6041.854183707143
For county  69  the  Predicted Democratic vote =  82031.86905462455
For county  70  the  Predicted Democratic vote =  6947.755551825824
For county  71  the  Predicted Democratic vote =  -3286.785874362695
For county  72  the  Predicted Democratic vote =  18476.862632854427
For county  73  the  Predicted Democratic vote =  3281.518438346061
For county  74  the  Predicted Democratic vote =  13775.32006445753
For county  75  the  Predicted Democratic vote =  6058.868957955937
For county  76  the  Predicted Democratic vote =  1672.7549672284786
For county  77  the  Predicted Democratic vote =  14490.826500444406
For county  78  the  Predicted Democratic vote =  5858.275201676944
For county  79  the  Predicted Democratic vote =  9730.21165172911
For county  80  the  Predicted Democratic vote =  15312.5795772459
For county  81  the  Predicted Democratic vote =  11203.398081542828
For county  82  the  Predicted Democratic vote =  1601.0671485921648
For county  83  the  Predicted Democratic vote =  -1802.8873369359371
For county  84  the  Predicted Democratic vote =  8211.6237415908
For county  85  the  Predicted Democratic vote =  9033.078982208943
```

```
For county  86  the  Predicted Democratic vote =  45873.59492479136
For county  87  the  Predicted Democratic vote =  26070.454342550885
For county  88  the  Predicted Democratic vote =  5814.697929074533
For county  89  the  Predicted Democratic vote =  16238.536283288959
For county  90  the  Predicted Democratic vote =  7321.250667999421
For county  91  the  Predicted Democratic vote =  4936.770326131978
For county  92  the  Predicted Democratic vote =  3225.818097892884
For county  93  the  Predicted Democratic vote =  -5777.975676963688
For county  94  the  Predicted Democratic vote =  -1923.546191788697
For county  95  the  Predicted Democratic vote =  -15193.179766656529
For county  96  the  Predicted Democratic vote =  12835.244420574469
For county  97  the  Predicted Democratic vote =  7519.503565279351
For county  98  the  Predicted Democratic vote =  20730.88852212405
For county  99  the  Predicted Democratic vote =  14942.404628440217
For county  100  the  Predicted Democratic vote =  124960.99543028834
For county  101  the  Predicted Democratic vote =  16187.16393189721
For county  102  the  Predicted Democratic vote =  4138.9857650996455
For county  103  the  Predicted Democratic vote =  26945.53106602173
For county  104  the  Predicted Democratic vote =  -41644.516825193146
For county  105  the  Predicted Democratic vote =  6453.813594936816
For county  106  the  Predicted Democratic vote =  13936.350258905873
For county  107  the  Predicted Democratic vote =  -23098.618828710423
For county  108  the  Predicted Democratic vote =  34597.72021044028
For county  109  the  Predicted Democratic vote =  3969.623370842958
For county  110  the  Predicted Democratic vote =  8979.901292315271
For county  111  the  Predicted Democratic vote =  -783.4284713261386
For county  112  the  Predicted Democratic vote =  31.549018564779544
For county  113  the  Predicted Democratic vote =  4258.143165489144
For county  114  the  Predicted Democratic vote =  8982.27826820492
For county  115  the  Predicted Democratic vote =  73253.00721258245
For county  116  the  Predicted Democratic vote =  4348.011879092613
For county  117  the  Predicted Democratic vote =  -347.67828165226456
For county  118  the  Predicted Democratic vote =  25394.600686955055
For county  119  the  Predicted Democratic vote =  24879.27810370493
For county  120  the  Predicted Democratic vote =  27359.628119709396
For county  121  the  Predicted Democratic vote =  28019.229823441194
For county  122  the  Predicted Democratic vote =  14164.784358897676
For county  123  the  Predicted Democratic vote =  -14797.52057553469
For county  124  the  Predicted Democratic vote =  190.62453942003413
For county  125  the  Predicted Democratic vote =  14970.909576228518
For county  126  the  Predicted Democratic vote =  1351.7679468188107
For county  127  the  Predicted Democratic vote =  13182.509859955517
For county  128  the  Predicted Democratic vote =  3146.362474632829
```

```
For county  129  the  Predicted Democratic vote =  228499.34454047598
For county  130  the  Predicted Democratic vote =  12711.133276864062
For county  131  the  Predicted Democratic vote =  -10180.45870371258
For county  132  the  Predicted Democratic vote =  7053.6094353512335
For county  133  the  Predicted Democratic vote =  9516.079620498043
For county  134  the  Predicted Democratic vote =  468.8230094924229
For county  135  the  Predicted Democratic vote =  5882.6407353719815
For county  136  the  Predicted Democratic vote =  -1786.3533982335584
For county  137  the  Predicted Democratic vote =  93024.98888418544
For county  138  the  Predicted Democratic vote =  1196.960802419353
For county  139  the  Predicted Democratic vote =  -16723.9228315771
For county  140  the  Predicted Democratic vote =  2464.4569301337046
For county  141  the  Predicted Democratic vote =  18059.72222883845
For county  142  the  Predicted Democratic vote =  -1529.2385094726633
For county  143  the  Predicted Democratic vote =  -12658.624609836486
For county  144  the  Predicted Democratic vote =  11165.541583561206
For county  145  the  Predicted Democratic vote =  5056.300674261183
For county  146  the  Predicted Democratic vote =  5399.327950218802
For county  147  the  Predicted Democratic vote =  55767.20324104402
For county  148  the  Predicted Democratic vote =  6810.010359014277
For county  149  the  Predicted Democratic vote =  3795.4200475281636
For county  150  the  Predicted Democratic vote =  52622.465645128636
For county  151  the  Predicted Democratic vote =  -1453.9404778646112
For county  152  the  Predicted Democratic vote =  15845.242583922518
For county  153  the  Predicted Democratic vote =  6500.887720243147
For county  154  the  Predicted Democratic vote =  1942.7714322021166
For county  155  the  Predicted Democratic vote =  -335.4356144145495
For county  156  the  Predicted Democratic vote =  -18826.066339991758
For county  157  the  Predicted Democratic vote =  11651.855317371635
For county  158  the  Predicted Democratic vote =  66803.6240504694
For county  159  the  Predicted Democratic vote =  76256.25145826527
For county  160  the  Predicted Democratic vote =  14144.465559525794
For county  161  the  Predicted Democratic vote =  22176.969514538796
For county  162  the  Predicted Democratic vote =  115066.46980472971
For county  163  the  Predicted Democratic vote =  -5403.995480594018
For county  164  the  Predicted Democratic vote =  -8740.984633951757
For county  165  the  Predicted Democratic vote =  -493.4491571597764
For county  166  the  Predicted Democratic vote =  -3173.403606462147
For county  167  the  Predicted Democratic vote =  1598.0120856925623
For county  168  the  Predicted Democratic vote =  -13284.579926198905
For county  169  the  Predicted Democratic vote =  -4181.674856772825
For county  170  the  Predicted Democratic vote =  5801.292739824174
For county  171  the  Predicted Democratic vote =  42452.28889624962
```

```
For county  172  the  Predicted Democratic vote =  22893.55930012153
For county  173  the  Predicted Democratic vote =  1375.31471168804
For county  174  the  Predicted Democratic vote =  11705.244343683496
For county  175  the  Predicted Democratic vote =  4237.886659817401
For county  176  the  Predicted Democratic vote =  -41547.19850432464
For county  177  the  Predicted Democratic vote =  3158.8367396696785
For county  178  the  Predicted Democratic vote =  6131.49127075186
For county  179  the  Predicted Democratic vote =  123718.31287746958
For county  180  the  Predicted Democratic vote =  10293.573105127776
For county  181  the  Predicted Democratic vote =  25301.256770399043
For county  182  the  Predicted Democratic vote =  7788.786514002959
For county  183  the  Predicted Democratic vote =  38555.11746341425
For county  184  the  Predicted Democratic vote =  40479.52692837767
For county  185  the  Predicted Democratic vote =  133449.34999083774
For county  186  the  Predicted Democratic vote =  86284.91593016172
For county  187  the  Predicted Democratic vote =  -4937.610225084767
For county  188  the  Predicted Democratic vote =  20979.287083400468
For county  189  the  Predicted Democratic vote =  11514.905001111743
For county  190  the  Predicted Democratic vote =  1435.868095595618
For county  191  the  Predicted Democratic vote =  6650.050036424667
For county  192  the  Predicted Democratic vote =  6496.277715081206
For county  193  the  Predicted Democratic vote =  6555.825040533174
For county  194  the  Predicted Democratic vote =  18264.317658287473
For county  195  the  Predicted Democratic vote =  734.0218713250188
For county  196  the  Predicted Democratic vote =  -933.7661759834364
For county  197  the  Predicted Democratic vote =  4569.012946247318
For county  198  the  Predicted Democratic vote =  7706.874036150361
For county  199  the  Predicted Democratic vote =  487.3098528654509
For county  200  the  Predicted Democratic vote =  -11547.598824001234
For county  201  the  Predicted Democratic vote =  4834.932633781195
For county  202  the  Predicted Democratic vote =  12324.589806826592
For county  203  the  Predicted Democratic vote =  19652.482332393614
For county  204  the  Predicted Democratic vote =  -16452.59811520248
For county  205  the  Predicted Democratic vote =  87238.80687399808
For county  206  the  Predicted Democratic vote =  4003.106970941695
For county  207  the  Predicted Democratic vote =  36199.515527501324
For county  208  the  Predicted Democratic vote =  8910.731382923886
For county  209  the  Predicted Democratic vote =  3148.353367999353
For county  210  the  Predicted Democratic vote =  6440.910285784084
For county  211  the  Predicted Democratic vote =  4010.1262361259824
For county  212  the  Predicted Democratic vote =  13297.409712670657
For county  213  the  Predicted Democratic vote =  4491.677809286312
For county  214  the  Predicted Democratic vote =  -3881.9446219657966
```

```
For county  215  the  Predicted Democratic vote =  6141.879654701788
For county  216  the  Predicted Democratic vote =  -18935.78103761775
For county  217  the  Predicted Democratic vote =  3447.694656753219
For county  218  the  Predicted Democratic vote =  113487.82843264923
For county  219  the  Predicted Democratic vote =  5164.772683328503
For county  220  the  Predicted Democratic vote =  10343.832062309368
For county  221  the  Predicted Democratic vote =  1483.0403658249852
For county  222  the  Predicted Democratic vote =  5114.821112409329
For county  223  the  Predicted Democratic vote =  8142.0129979649755
For county  224  the  Predicted Democratic vote =  9958.408434946068
For county  225  the  Predicted Democratic vote =  1557.2648997753467
For county  226  the  Predicted Democratic vote =  3004.781476597434
For county  227  the  Predicted Democratic vote =  108245.92264822271
For county  228  the  Predicted Democratic vote =  272779.69987639535
For county  229  the  Predicted Democratic vote =  2577.556509266403
For county  230  the  Predicted Democratic vote =  8083.465881687236
For county  231  the  Predicted Democratic vote =  -19009.412812383292
For county  232  the  Predicted Democratic vote =  -11804.267362597471
For county  233  the  Predicted Democratic vote =  -1092.5214399075949
For county  234  the  Predicted Democratic vote =  1662.6077579162693
For county  235  the  Predicted Democratic vote =  10933.120199803045
For county  236  the  Predicted Democratic vote =  37045.34152379759
For county  237  the  Predicted Democratic vote =  8564.285590024489
For county  238  the  Predicted Democratic vote =  2475.538677205023
For county  239  the  Predicted Democratic vote =  4419.060178854421
For county  240  the  Predicted Democratic vote =  11392.113610566628
For county  241  the  Predicted Democratic vote =  5081.597920217434
For county  242  the  Predicted Democratic vote =  -3590.4389263122575
For county  243  the  Predicted Democratic vote =  17069.461158494807
For county  244  the  Predicted Democratic vote =  564.6307165471198
For county  245  the  Predicted Democratic vote =  186100.2065868736
For county  246  the  Predicted Democratic vote =  2309.5715938829853
For county  247  the  Predicted Democratic vote =  17483.08438065714
For county  248  the  Predicted Democratic vote =  3067.4395565183354
For county  249  the  Predicted Democratic vote =  4374.954028445802
For county  250  the  Predicted Democratic vote =  -2728.5721308943866
For county  251  the  Predicted Democratic vote =  3233.724150691909
For county  252  the  Predicted Democratic vote =  37758.860363756714
For county  253  the  Predicted Democratic vote =  -1362.0036527722114
For county  254  the  Predicted Democratic vote =  6938.377019002861
For county  255  the  Predicted Democratic vote =  11365.854758301764
For county  256  the  Predicted Democratic vote =  8397.541454941482
For county  257  the  Predicted Democratic vote =  -2633.6224581999886
```

```
For county  258  the  Predicted Democratic vote =  -1651.016786454964
For county  259  the  Predicted Democratic vote =  693.6123859782692
For county  260  the  Predicted Democratic vote =  5358.18857374526
For county  261  the  Predicted Democratic vote =  -309.7954247758771
For county  262  the  Predicted Democratic vote =  4100.230892707198
For county  263  the  Predicted Democratic vote =  42697.429078982896
For county  264  the  Predicted Democratic vote =  5894.875663530766
For county  265  the  Predicted Democratic vote =  16170.378347713855
For county  266  the  Predicted Democratic vote =  65143.25709505558
For county  267  the  Predicted Democratic vote =  6860.963974432263
For county  268  the  Predicted Democratic vote =  24881.25308255353
For county  269  the  Predicted Democratic vote =  1258.3755993015047
For county  270  the  Predicted Democratic vote =  -205.31634540078448
For county  271  the  Predicted Democratic vote =  13803.577009260158
For county  272  the  Predicted Democratic vote =  4418.880297971709
For county  273  the  Predicted Democratic vote =  -2040.1078623397598
For county  274  the  Predicted Democratic vote =  11964.627512572943
For county  275  the  Predicted Democratic vote =  10523.703715541807
For county  276  the  Predicted Democratic vote =  4723.225392868897
For county  277  the  Predicted Democratic vote =  12247.566591983355
For county  278  the  Predicted Democratic vote =  -15601.857183991673
For county  279  the  Predicted Democratic vote =  26203.686165516036
For county  280  the  Predicted Democratic vote =  6975.820902480834
For county  281  the  Predicted Democratic vote =  692.2887455043638
For county  282  the  Predicted Democratic vote =  1781.8144882622514
For county  283  the  Predicted Democratic vote =  75556.93174144771
For county  284  the  Predicted Democratic vote =  -10187.34130166523
For county  285  the  Predicted Democratic vote =  14883.123297696444
For county  286  the  Predicted Democratic vote =  4135.434817568485
For county  287  the  Predicted Democratic vote =  14893.633473080805
For county  288  the  Predicted Democratic vote =  1935.4890587778777
For county  289  the  Predicted Democratic vote =  -5782.818524915707
For county  290  the  Predicted Democratic vote =  1805.280140251772
For county  291  the  Predicted Democratic vote =  60993.481223243165
For county  292  the  Predicted Democratic vote =  -16875.321482754385
For county  293  the  Predicted Democratic vote =  13952.927611244126
For county  294  the  Predicted Democratic vote =  6550.01701487594
For county  295  the  Predicted Democratic vote =  17449.556142319685
For county  296  the  Predicted Democratic vote =  146297.86746096253
For county  297  the  Predicted Democratic vote =  8437.61212522606
For county  298  the  Predicted Democratic vote =  195626.42985482112
For county  299  the  Predicted Democratic vote =  27021.76248240393
Scores are
```

```
[0.8727864234509344, 0.8695564114856922]
```

```
# Task 3-Republican
pred_variable= ['Percent Age 29 and Under','Percent Age 65 and Older','Total Population',
'Percent Foreign Born','Percent Hispanic or Latino','Percent Black, not Hispanic or Latin
o','Percent White, not Hispanic or Latino','Percent Female','Percent Unemployed']
pred_variable2 = ['Total Population','Percent White, not Hispanic or Latino','Percent Blac
k, not Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Fe
male','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income','Per
cent Unemployed','Percent Less than High School Degree','Percent Less than Bachelor\'s Deg
ree','Percent Rural']

x_trainR, x_valR, y_trainR, y_valR = train_test_split(data_m[['Total Population','Percent
 White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic
 or Latino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Ag
e 65 and Older','Median Household Income','Percent Unemployed','Percent Less than High Sch
ool Degree','Percent Less than Bachelor\'s Degree','Percent Rural']],data_m['Republican'],
train_size = 0.75, test_size = 0.25, random_state = 0)

model = linear_model.LinearRegression().fit(X = x_trainR[pred_variable], y = y_trainR)
model2 = linear_model.LinearRegression().fit(X = x_trainR[pred_variable2], y = y_trainR)

score_train = model.score(X = x_trainR[pred_variable], y = y_trainR) # R squared (trainin
g)
score_val = model.score(X = x_valR[pred_variable], y = y_valR) # R squared (validation)

m= model.coef_
c= model.intercept_

for i in range (0, len(x_valR[pred_variable])):
    print("For county ",i+1," the  Predicted Republican vote = ", (m*x_valR[pred_variable]
.iloc[i]).sum() + c)

print("Scores are \n")
print([score_train, score_val])
```

```
For county  1  the  Predicted Republican vote =  6779.383766123723
For county  2  the  Predicted Republican vote =  1399.207566026722
For county  3  the  Predicted Republican vote =  48567.120511239715
For county  4  the  Predicted Republican vote =  7590.968828412901
For county  5  the  Predicted Republican vote =  21158.011731286315
For county  6  the  Predicted Republican vote =  443.6612765663085
For county  7  the  Predicted Republican vote =  2267.7000040892035
For county  8  the  Predicted Republican vote =  239.0731949781184
For county  9  the  Predicted Republican vote =  1335.746906730048
For county  10  the  Predicted Republican vote =  15275.41495437473
For county  11  the  Predicted Republican vote =  7198.96586136543
For county  12  the  Predicted Republican vote =  8080.239478128013
For county  13  the  Predicted Republican vote =  5115.490970481851
For county  14  the  Predicted Republican vote =  64676.85797392721
For county  15  the  Predicted Republican vote =  12708.177238373744
For county  16  the  Predicted Republican vote =  65417.458504739916
For county  17  the  Predicted Republican vote =  -1377.5298530774244
For county  18  the  Predicted Republican vote =  6904.102830696313
For county  19  the  Predicted Republican vote =  7903.690583915584
For county  20  the  Predicted Republican vote =  13307.237684466145
For county  21  the  Predicted Republican vote =  14204.593163292899
For county  22  the  Predicted Republican vote =  11535.879053343113
For county  23  the  Predicted Republican vote =  29724.467860633635
For county  24  the  Predicted Republican vote =  9068.263200044941
For county  25  the  Predicted Republican vote =  29610.632386080226
For county  26  the  Predicted Republican vote =  82880.01542117383
For county  27  the  Predicted Republican vote =  20060.854935088515
For county  28  the  Predicted Republican vote =  -1000.4000030515708
For county  29  the  Predicted Republican vote =  5925.654121325424
For county  30  the  Predicted Republican vote =  -3237.6196236523274
For county  31  the  Predicted Republican vote =  12842.503251729726
For county  32  the  Predicted Republican vote =  473.3378438021664
For county  33  the  Predicted Republican vote =  15909.163948999168
For county  34  the  Predicted Republican vote =  10281.590497424397
For county  35  the  Predicted Republican vote =  -1056.95666432068
For county  36  the  Predicted Republican vote =  -4709.722104230749
For county  37  the  Predicted Republican vote =  66368.08298671107
For county  38  the  Predicted Republican vote =  5655.105195817256
For county  39  the  Predicted Republican vote =  35249.451477817
For county  40  the  Predicted Republican vote =  19256.294038833297
For county  41  the  Predicted Republican vote =  75843.39143927833
For county  42  the  Predicted Republican vote =  43283.408979923595
```

```
For county  43  the  Predicted Republican vote =  8556.600546669866
For county  44  the  Predicted Republican vote =  7052.9490135083615
For county  45  the  Predicted Republican vote =  5584.033559082749
For county  46  the  Predicted Republican vote =  15993.943084480317
For county  47  the  Predicted Republican vote =  5550.6448181539745
For county  48  the  Predicted Republican vote =  15252.513053351468
For county  49  the  Predicted Republican vote =  13251.273852833441
For county  50  the  Predicted Republican vote =  7892.96273227561
For county  51  the  Predicted Republican vote =  28290.819317366422
For county  52  the  Predicted Republican vote =  102478.05473620785
For county  53  the  Predicted Republican vote =  10826.73395308432
For county  54  the  Predicted Republican vote =  -5185.2390563949775
For county  55  the  Predicted Republican vote =  301628.7639768296
For county  56  the  Predicted Republican vote =  -3857.43840454948
For county  57  the  Predicted Republican vote =  3482.7145098134642
For county  58  the  Predicted Republican vote =  9093.192342299062
For county  59  the  Predicted Republican vote =  13601.6212991898
For county  60  the  Predicted Republican vote =  2655.5044002969134
For county  61  the  Predicted Republican vote =  13658.003002614856
For county  62  the  Predicted Republican vote =  33059.45024628195
For county  63  the  Predicted Republican vote =  5293.101393437524
For county  64  the  Predicted Republican vote =  11529.601721721734
For county  65  the  Predicted Republican vote =  7361.2914607435905
For county  66  the  Predicted Republican vote =  8965.246967726209
For county  67  the  Predicted Republican vote =  13552.935777525883
For county  68  the  Predicted Republican vote =  7548.53290984267
For county  69  the  Predicted Republican vote =  52897.667449195345
For county  70  the  Predicted Republican vote =  12237.7835079035
For county  71  the  Predicted Republican vote =  5406.3863554775035
For county  72  the  Predicted Republican vote =  19234.284605218403
For county  73  the  Predicted Republican vote =  13251.767144770973
For county  74  the  Predicted Republican vote =  17053.621119488576
For county  75  the  Predicted Republican vote =  12911.821675459294
For county  76  the  Predicted Republican vote =  6529.658390981784
For county  77  the  Predicted Republican vote =  8241.063392315635
For county  78  the  Predicted Republican vote =  9505.048566728754
For county  79  the  Predicted Republican vote =  14387.795222670913
For county  80  the  Predicted Republican vote =  16281.893723156401
For county  81  the  Predicted Republican vote =  6791.0908041736475
For county  82  the  Predicted Republican vote =  6526.276257850632
For county  83  the  Predicted Republican vote =  8909.178118676697
For county  84  the  Predicted Republican vote =  9999.760962295975
For county  85  the  Predicted Republican vote =  11032.123226498381
```

```
For county  86  the  Predicted Republican vote =  15591.64702932291
For county  87  the  Predicted Republican vote =  22135.790580123637
For county  88  the  Predicted Republican vote =  11143.073168079274
For county  89  the  Predicted Republican vote =  17897.982716078786
For county  90  the  Predicted Republican vote =  13757.048580251714
For county  91  the  Predicted Republican vote =  12118.257469687796
For county  92  the  Predicted Republican vote =  12720.416286152209
For county  93  the  Predicted Republican vote =  -3658.803436169781
For county  94  the  Predicted Republican vote =  6622.362123174147
For county  95  the  Predicted Republican vote =  3376.507727552529
For county  96  the  Predicted Republican vote =  7771.516523507595
For county  97  the  Predicted Republican vote =  7521.390273979778
For county  98  the  Predicted Republican vote =  -3168.4545880604874
For county  99  the  Predicted Republican vote =  16241.869166135588
For county  100  the  Predicted Republican vote =  77350.01750690198
For county  101  the  Predicted Republican vote =  13689.39969602759
For county  102  the  Predicted Republican vote =  11202.14079727677
For county  103  the  Predicted Republican vote =  27958.56307866755
For county  104  the  Predicted Republican vote =  -2493.359023255647
For county  105  the  Predicted Republican vote =  14108.950322425728
For county  106  the  Predicted Republican vote =  18240.355252010657
For county  107  the  Predicted Republican vote =  -1134.1425483798812
For county  108  the  Predicted Republican vote =  28280.868656804338
For county  109  the  Predicted Republican vote =  8437.334234747806
For county  110  the  Predicted Republican vote =  12677.550669678629
For county  111  the  Predicted Republican vote =  7618.116549083057
For county  112  the  Predicted Republican vote =  6726.119912132899
For county  113  the  Predicted Republican vote =  11961.925419382318
For county  114  the  Predicted Republican vote =  8911.788479750878
For county  115  the  Predicted Republican vote =  45409.33430371084
For county  116  the  Predicted Republican vote =  11700.591303838439
For county  117  the  Predicted Republican vote =  7637.229951704576
For county  118  the  Predicted Republican vote =  6108.024255767074
For county  119  the  Predicted Republican vote =  4456.965702226602
For county  120  the  Predicted Republican vote =  26158.43017059204
For county  121  the  Predicted Republican vote =  23656.669959420753
For county  122  the  Predicted Republican vote =  12232.249033299682
For county  123  the  Predicted Republican vote =  3853.1560927677856
For county  124  the  Predicted Republican vote =  11813.54871210628
For county  125  the  Predicted Republican vote =  5585.163740865499
For county  126  the  Predicted Republican vote =  9374.004991695478
For county  127  the  Predicted Republican vote =  14049.276449346275
For county  128  the  Predicted Republican vote =  10824.90490817869
```

```
For county  129  the  Predicted Republican vote =  150965.90945346747
For county  130  the  Predicted Republican vote =  6558.376944740987
For county  131  the  Predicted Republican vote =  1536.0836116916853
For county  132  the  Predicted Republican vote =  12859.774611998244
For county  133  the  Predicted Republican vote =  -11263.274629228996
For county  134  the  Predicted Republican vote =  5864.922155531176
For county  135  the  Predicted Republican vote =  11796.202730041306
For county  136  the  Predicted Republican vote =  10477.627487735142
For county  137  the  Predicted Republican vote =  43761.715525453226
For county  138  the  Predicted Republican vote =  6024.339979706952
For county  139  the  Predicted Republican vote =  3153.624999629532
For county  140  the  Predicted Republican vote =  9161.96062441382
For county  141  the  Predicted Republican vote =  13678.706197271207
For county  142  the  Predicted Republican vote =  7319.818178852969
For county  143  the  Predicted Republican vote =  1302.5334160933653
For county  144  the  Predicted Republican vote =  11055.306653584186
For county  145  the  Predicted Republican vote =  10609.39420414162
For county  146  the  Predicted Republican vote =  13102.230830419252
For county  147  the  Predicted Republican vote =  41016.15925423823
For county  148  the  Predicted Republican vote =  14192.37981096319
For county  149  the  Predicted Republican vote =  4457.728751134546
For county  150  the  Predicted Republican vote =  38279.906185320855
For county  151  the  Predicted Republican vote =  6213.700347620961
For county  152  the  Predicted Republican vote =  2741.6604707269635
For county  153  the  Predicted Republican vote =  12924.69672359655
For county  154  the  Predicted Republican vote =  10095.014405983758
For county  155  the  Predicted Republican vote =  8559.478768138462
For county  156  the  Predicted Republican vote =  4895.333664175576
For county  157  the  Predicted Republican vote =  -4376.702108025156
For county  158  the  Predicted Republican vote =  40645.34169256214
For county  159  the  Predicted Republican vote =  36418.77003276875
For county  160  the  Predicted Republican vote =  18401.835196578
For county  161  the  Predicted Republican vote =  17368.38882576331
For county  162  the  Predicted Republican vote =  73874.06702489883
For county  163  the  Predicted Republican vote =  3182.662688338498
For county  164  the  Predicted Republican vote =  10490.110498976166
For county  165  the  Predicted Republican vote =  3264.5189732394356
For county  166  the  Predicted Republican vote =  -2041.7354695970953
For county  167  the  Predicted Republican vote =  9212.119607061992
For county  168  the  Predicted Republican vote =  -1471.3880681597602
For county  169  the  Predicted Republican vote =  10898.631502297609
For county  170  the  Predicted Republican vote =  11400.229130813044
For county  171  the  Predicted Republican vote =  35911.490405297955
```

```
For county  172  the  Predicted Republican vote =  17572.441462823626
For county  173  the  Predicted Republican vote =  6114.878430602426
For county  174  the  Predicted Republican vote =  14571.460011652955
For county  175  the  Predicted Republican vote =  7428.850846577994
For county  176  the  Predicted Republican vote =  -878.8736540675218
For county  177  the  Predicted Republican vote =  10444.743395658297
For county  178  the  Predicted Republican vote =  11210.905489156288
For county  179  the  Predicted Republican vote =  64770.16516881548
For county  180  the  Predicted Republican vote =  16043.259290010486
For county  181  the  Predicted Republican vote =  22330.71988625976
For county  182  the  Predicted Republican vote =  14178.089523663322
For county  183  the  Predicted Republican vote =  27390.699943765485
For county  184  the  Predicted Republican vote =  33998.312082624834
For county  185  the  Predicted Republican vote =  85027.03945417145
For county  186  the  Predicted Republican vote =  57689.59110905172
For county  187  the  Predicted Republican vote =  3378.832875632166
For county  188  the  Predicted Republican vote =  21052.89832974337
For county  189  the  Predicted Republican vote =  10404.668025049312
For county  190  the  Predicted Republican vote =  7948.335498989265
For county  191  the  Predicted Republican vote =  5251.655232923511
For county  192  the  Predicted Republican vote =  3987.0763132798256
For county  193  the  Predicted Republican vote =  13629.744945388898
For county  194  the  Predicted Republican vote =  18375.718281158755
For county  195  the  Predicted Republican vote =  3766.3429584617825
For county  196  the  Predicted Republican vote =  6317.984245260848
For county  197  the  Predicted Republican vote =  11937.00202377042
For county  198  the  Predicted Republican vote =  6725.896865866824
For county  199  the  Predicted Republican vote =  9027.586467548983
For county  200  the  Predicted Republican vote =  4665.971629149293
For county  201  the  Predicted Republican vote =  4335.764003235447
For county  202  the  Predicted Republican vote =  2221.8952717977354
For county  203  the  Predicted Republican vote =  21902.452083215616
For county  204  the  Predicted Republican vote =  97.64653554530196
For county  205  the  Predicted Republican vote =  49552.66840045655
For county  206  the  Predicted Republican vote =  8748.131283711944
For county  207  the  Predicted Republican vote =  27638.840111649555
For county  208  the  Predicted Republican vote =  7140.127277034369
For county  209  the  Predicted Republican vote =  9737.00631388248
For county  210  the  Predicted Republican vote =  10909.57369058919
For county  211  the  Predicted Republican vote =  7349.9362426545995
For county  212  the  Predicted Republican vote =  17504.410744238437
For county  213  the  Predicted Republican vote =  996.9280123962217
For county  214  the  Predicted Republican vote =  4679.700580711673
```

```
For county  215  the  Predicted Republican vote =  9040.245592935076
For county  216  the  Predicted Republican vote =  1144.8193848103056
For county  217  the  Predicted Republican vote =  12634.653050965517
For county  218  the  Predicted Republican vote =  73863.22729626337
For county  219  the  Predicted Republican vote =  11093.860859084365
For county  220  the  Predicted Republican vote =  13943.149990364494
For county  221  the  Predicted Republican vote =  5627.823468501767
For county  222  the  Predicted Republican vote =  11638.827022262314
For county  223  the  Predicted Republican vote =  12696.768164778136
For county  224  the  Predicted Republican vote =  10825.556998104727
For county  225  the  Predicted Republican vote =  7525.240622258956
For county  226  the  Predicted Republican vote =  7335.308545191609
For county  227  the  Predicted Republican vote =  51031.416529234426
For county  228  the  Predicted Republican vote =  165584.06065498793
For county  229  the  Predicted Republican vote =  979.4377741082717
For county  230  the  Predicted Republican vote =  9764.206264752997
For county  231  the  Predicted Republican vote =  488.03857487045207
For county  232  the  Predicted Republican vote =  7779.056618188257
For county  233  the  Predicted Republican vote =  7526.591168016272
For county  234  the  Predicted Republican vote =  10167.053547791002
For county  235  the  Predicted Republican vote =  13746.725575089524
For county  236  the  Predicted Republican vote =  29270.231703417176
For county  237  the  Predicted Republican vote =  8953.862731691039
For county  238  the  Predicted Republican vote =  8416.856312551976
For county  239  the  Predicted Republican vote =  10483.486721042573
For county  240  the  Predicted Republican vote =  15934.991111274063
For county  241  the  Predicted Republican vote =  12458.959162249212
For county  242  the  Predicted Republican vote =  -2212.520916123711
For county  243  the  Predicted Republican vote =  16443.814022910625
For county  244  the  Predicted Republican vote =  6897.909927203422
For county  245  the  Predicted Republican vote =  117335.03618061768
For county  246  the  Predicted Republican vote =  5963.732063583864
For county  247  the  Predicted Republican vote =  15504.626088643825
For county  248  the  Predicted Republican vote =  12559.680335330137
For county  249  the  Predicted Republican vote =  8711.433103993182
For county  250  the  Predicted Republican vote =  8516.568700546508
For county  251  the  Predicted Republican vote =  9546.158710088115
For county  252  the  Predicted Republican vote =  27095.086919793768
For county  253  the  Predicted Republican vote =  6581.893394907071
For county  254  the  Predicted Republican vote =  12362.203642632565
For county  255  the  Predicted Republican vote =  9386.848695208279
For county  256  the  Predicted Republican vote =  13100.048503437665
For county  257  the  Predicted Republican vote =  3808.439986403775
```

```
For county  258  the  Predicted Republican vote =  7084.779530414817
For county  259  the  Predicted Republican vote =  7405.207818049663
For county  260  the  Predicted Republican vote =  405.7902536328402
For county  261  the  Predicted Republican vote =  11211.52832838277
For county  262  the  Predicted Republican vote =  5895.628171142276
For county  263  the  Predicted Republican vote =  32124.155709211136
For county  264  the  Predicted Republican vote =  9636.762366208797
For county  265  the  Predicted Republican vote =  18380.392923330073
For county  266  the  Predicted Republican vote =  42194.1173151572
For county  267  the  Predicted Republican vote =  11883.0176562593
For county  268  the  Predicted Republican vote =  20684.951305019542
For county  269  the  Predicted Republican vote =  8439.074998963762
For county  270  the  Predicted Republican vote =  3787.9811884430837
For county  271  the  Predicted Republican vote =  14815.152200782268
For county  272  the  Predicted Republican vote =  14064.564674581688
For county  273  the  Predicted Republican vote =  7430.593218348105
For county  274  the  Predicted Republican vote =  14354.253389124202
For county  275  the  Predicted Republican vote =  14453.278167201537
For county  276  the  Predicted Republican vote =  12399.102453016334
For county  277  the  Predicted Republican vote =  13024.696307477105
For county  278  the  Predicted Republican vote =  701.3997292489948
For county  279  the  Predicted Republican vote =  22081.315028793477
For county  280  the  Predicted Republican vote =  13813.99449998701
For county  281  the  Predicted Republican vote =  9313.019010937402
For county  282  the  Predicted Republican vote =  6391.183383811378
For county  283  the  Predicted Republican vote =  45534.856579603525
For county  284  the  Predicted Republican vote =  3182.1625110977457
For county  285  the  Predicted Republican vote =  14802.446134477843
For county  286  the  Predicted Republican vote =  10242.64660212113
For county  287  the  Predicted Republican vote =  7436.639558607232
For county  288  the  Predicted Republican vote =  6301.888641593896
For county  289  the  Predicted Republican vote =  3758.612170045406
For county  290  the  Predicted Republican vote =  7971.4025988251415
For county  291  the  Predicted Republican vote =  46034.66750198096
For county  292  the  Predicted Republican vote =  1893.6350752762391
For county  293  the  Predicted Republican vote =  18742.109675395273
For county  294  the  Predicted Republican vote =  11911.133838399473
For county  295  the  Predicted Republican vote =  13549.847290193607
For county  296  the  Predicted Republican vote =  86513.20105096197
For county  297  the  Predicted Republican vote =  11682.16470657333
For county  298  the  Predicted Republican vote =  126481.42046156351
For county  299  the  Predicted Republican vote =  21069.39271968274
Scores are
```

```
[0.8471998509246637, 0.6573876421888164]
```

```
x_train, x_test, y_train, y_test = train_test_split(data_m[['FIPS','Total Population','Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income','Percent Unemployed','Percent Less than High School Degree','Percent Less than Bachelor\'s Degree','Percent Rural']],data_m['Party'],test_size = 0.25, random_state = 0)
```

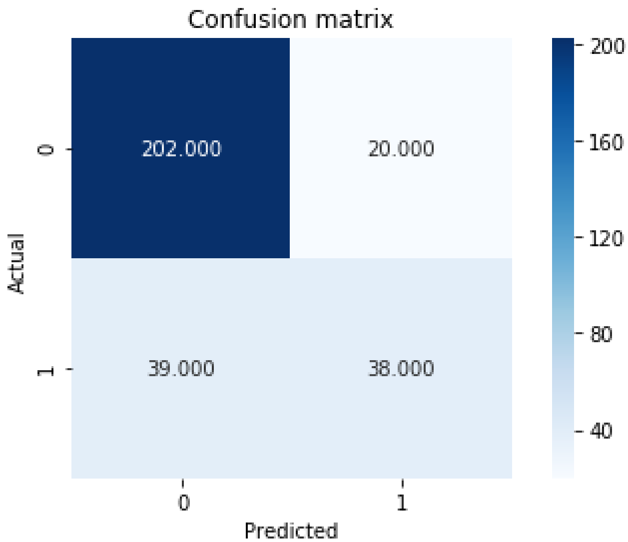
```
x_train1, x_test1, y_train1, y_test1 = train_test_split(data_m[['Total Population','Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income','Percent Unemployed','Percent Less than High School Degree','Percent Less than Bachelor\'s Degree','Percent Rural']],data_m['Party'],test_size = 0.25, random_state = 0)
```

```
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

```
scaler = StandardScaler()
scaler.fit(x_train1)
x_train_scaled1 = scaler.transform(x_train1)
x_test_scaled1 = scaler.transform(x_test1)
```

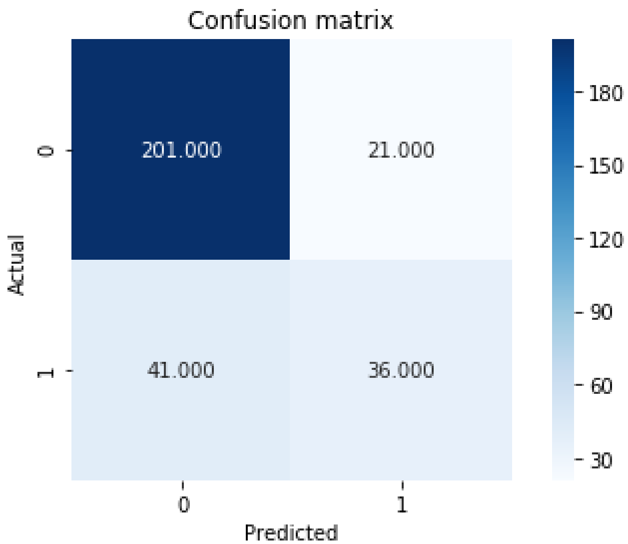
```python
#TASK 4
# CLASSIFIER 1: K-nearest neighbors #1 .1
# Uses k=3 as the number of nearest neighbors, using all variables

# Build k-nearest neighbors classifier
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train_scaled1, y_train1)

# Predict class labels using decision tree classifier
y_pred = classifier.predict(x_test_scaled1)

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
#print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[0.802675585284281, 0.19732441471571904, array([0.83817427, 0.65517241]), array([0.90990991, 0.49350649]), arra
y([0.87257019, 0.56296296])]
```



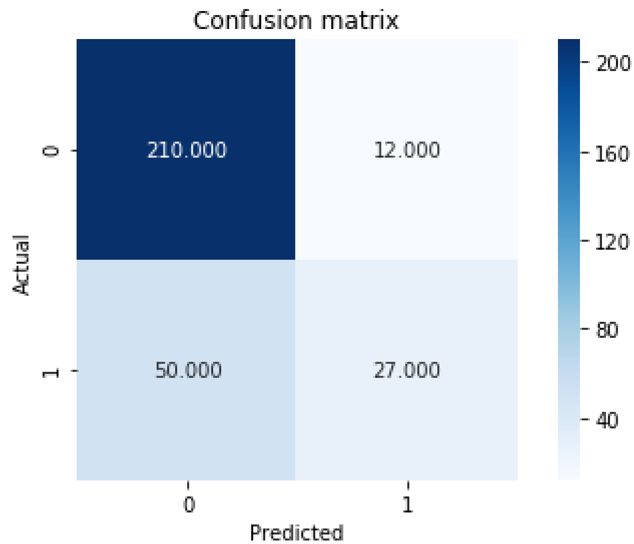
```python
# CLASSIFIER 1: K-nearest neighbors #1.2
# Uses k=3 as the number of nearest neighbors, using only variables:
# Percent White, not Hispanic or Latino,
# Percent Black, not Hispanic or Latino, and
# Percent Less than Bachelor's Degree

# Build k-nearest neighbors classifier
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train_scaled1[:,[0,1,10]], y_train)

# Predict class labels using decision tree classifier
y_pred = classifier.predict(x_test_scaled1[:,[0,1,10]])

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[201  21]
 [ 41  36]]
[0.7926421404682275, 0.20735785953177255, array([0.83057851, 0.63157895]), array([0.90540541, 0.46753247]), arr
ay([0.86637931, 0.53731343])]
```



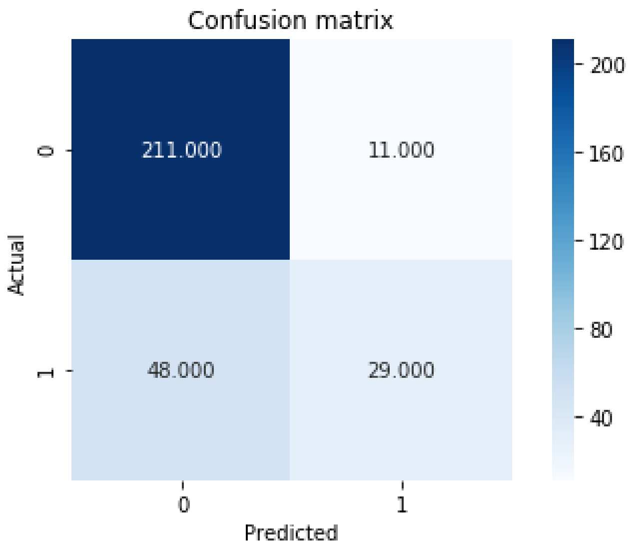
```python
# CLASSIFIER 1: K-nearest neighbors #2.1
# Uses k=4 as the number of nearest neighbors, using all variables

# Build k-nearest neighbors classifier
classifier = KNeighborsClassifier(n_neighbors = 4)
classifier.fit(x_train_scaled1, y_train)

# Predict class labels using k-nearest neighbors classifier
y_pred = classifier.predict(x_test_scaled1)

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[210  12]
 [ 50  27]]
[0.7926421404682275, 0.20735785953177255, array([0.80769231, 0.69230769]), array([0.94594595, 0.35064935]), arr
ay([0.87136929, 0.46551724])]
```



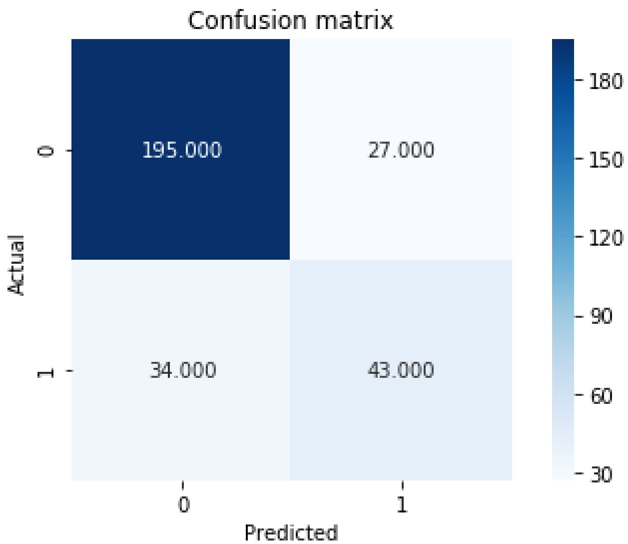
```python
# CLASSIFIER 1: K-nearest neighbors #2.2
# Uses k=4 as the number of nearest neighbors, using only variables:
# Percent White, not Hispanic or Latino,
# Percent Black, not Hispanic or Latino, and
# Percent Less than Bachelor's Degree

# Build k-nearest neighbors classifier
classifier = KNeighborsClassifier(n_neighbors = 4)
classifier.fit(x_train_scaled1[:,[0,1,10]], y_train)

# Predict class labels using k-nearest neighbors classifier
y_pred = classifier.predict(x_test_scaled1[:,[0,1,10]])

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[211  11]
 [ 48  29]]
[0.802675585284281, 0.19732441471571904, array([0.81467181, 0.725     ]), array([0.95045045, 0.37662338]), arra
y([0.87733888, 0.4957265 ])]
```



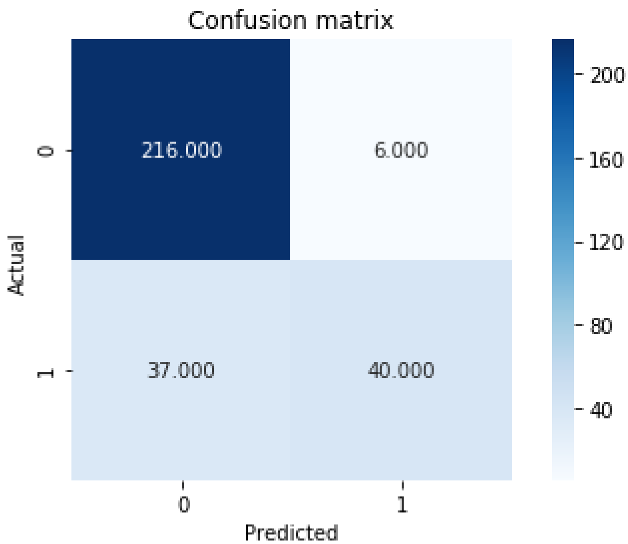
```python
# CLASSIFIER 2: Naive Bayes
# Using all variables

# Build Naive Bayes classifier
classifier = GaussianNB()
classifier.fit(x_train_scaled1, y_train)

# Predict class labels using Naive Bayes classifier
y_pred = classifier.predict(x_test_scaled1)

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[195  27]
 [ 34  43]]
[0.7959866220735786, 0.20401337792642138, array([0.85152838, 0.61428571]), array([0.87837838, 0.55844156]), arr
ay([0.86474501, 0.58503401])]
```



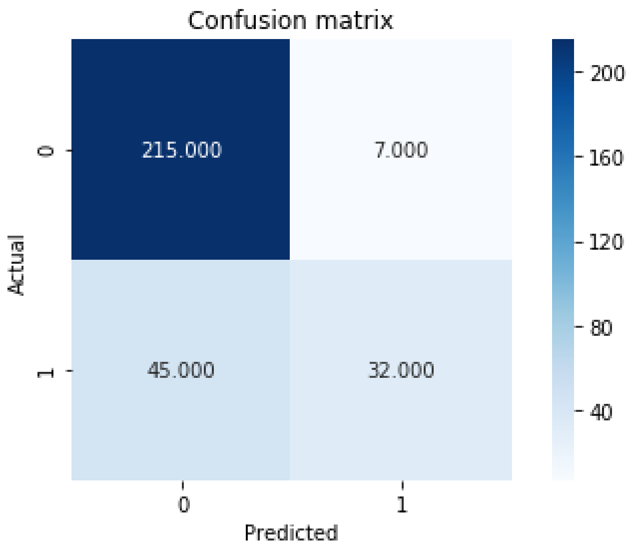
```python
# CLASSIFIER 3: SVM #1.1
# Using kernel as 'rbf' or radial basis function, using all variables

# Build SVM classifier
bestclassifier = SVC(kernel = 'rbf')
bestclassifier.fit(x_train_scaled1, y_train)

# Predict class labels using SVM classifier
y_pred = bestclassifier.predict(x_test_scaled1)

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[216   6]
 [ 37  40]]
[0.8561872909698997, 0.14381270903010035, array([0.85375494, 0.86956522]), array([0.97297297, 0.51948052]), arr
ay([0.90947368, 0.6504065 ])]
```



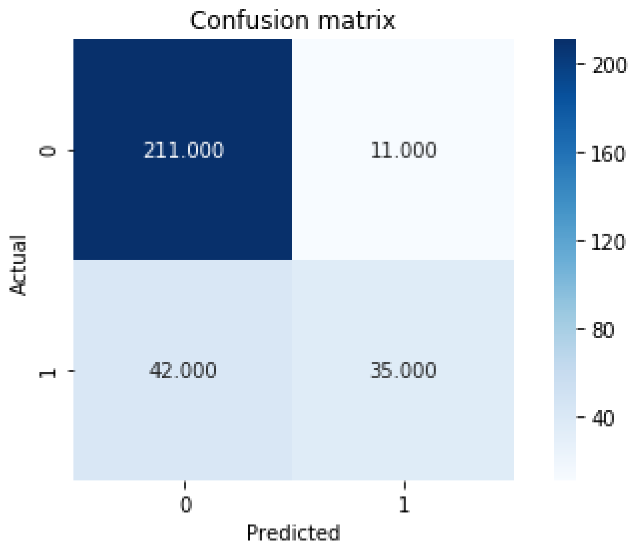
```python
# CLASSIFIER 3: SVM #1.2
# Using kernel as 'rbf' or radial basis function, using only variables:
# Percent White, not Hispanic or Latino,
# Percent Black, not Hispanic or Latino, and
# Percent Less than Bachelor's Degree

# Build SVM classifier
classifier = SVC(kernel = 'rbf')
classifier.fit(x_train_scaled1[:,[0,1,10]], y_train)

# Predict class labels using SVM classifier
y_pred = classifier.predict(x_test_scaled1[:,[0,1,10]])

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[215   7]
 [ 45  32]]
[0.8260869565217391, 0.17391304347826086, array([0.82692308, 0.82051282]), array([0.96846847, 0.41558442]), arr
ay([0.89211618, 0.55172414])]
```



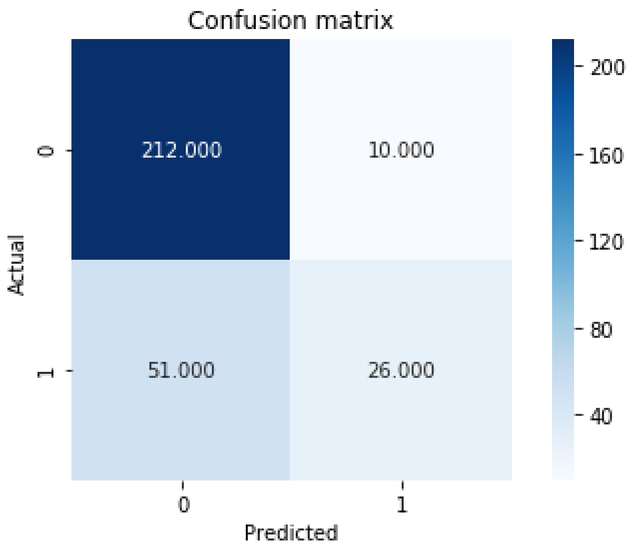
```python
# CLASSIFIER 3: SVM #2.1
# Using kernel as 'linear', using all variables

# Build SVM classifier
classifier = SVC(kernel = 'linear')
classifier.fit(x_train_scaled1, y_train)

# Predict class labels using SVM classifier
y_pred = classifier.predict(x_test_scaled1)

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[211  11]
 [ 42  35]]
[0.822742474916388, 0.17725752508361203, array([0.83399209, 0.76086957]), array([0.95045045, 0.45454545]), arra
y([0.88842105, 0.56910569])]
```



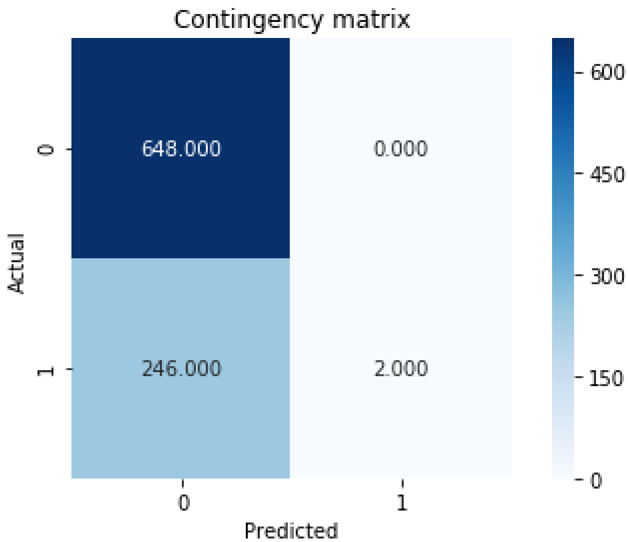
```python
# CLASSIFIER 3: SVM #2.2
# Using kernel as 'linear', using only variables:
# Percent White, not Hispanic or Latino,
# Percent Black, not Hispanic or Latino, and
# Percent Less than Bachelor's Degree

# Build SVM classifier
classifier = SVC(kernel = 'linear')
classifier.fit(x_train_scaled1[:,[0,1,10]], y_train)

# Predict class labels using SVM classifier
y_pred = classifier.predict(x_test_scaled1[:,[0,1,10]])

# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test1, y_pred)
print(conf_matrix)

# Plot confusion matrix
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()

# Evaluation metrics
accuracy = metrics.accuracy_score(y_test1, y_pred)
error = 1 - metrics.accuracy_score(y_test1, y_pred)
precision = metrics.precision_score(y_test1, y_pred, average = None)
recall = metrics.recall_score(y_test1, y_pred, average = None)
F1_score = metrics.f1_score(y_test1, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

```
[[212  10]
 [ 51  26]]
[0.7959866220735786, 0.20401337792642138, array([0.80608365, 0.72222222]), array([0.95495495, 0.33766234]), arr
ay([0.8742268 , 0.46017699])]
```



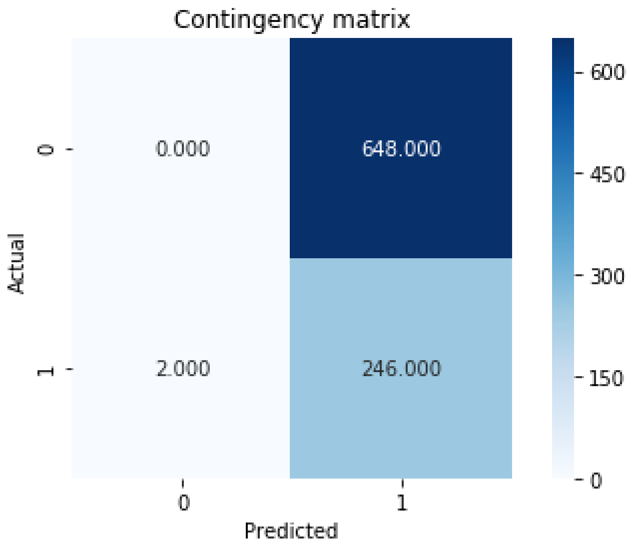
```python
# Task 5
# 5.1 - Single Linkage Hierarchical Clustering
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispani
c or Latino'[1],
# 'Percent Hispanic or Latino'[2]

# pick certain variables from x_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2]])


# cluster observations
clustering = linkage(X, method='single', metric='euclidean')
clusters = fcluster(clustering, 2, criterion = 'maxclust')

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.007194555265724431, 0.8360571913376201]
```



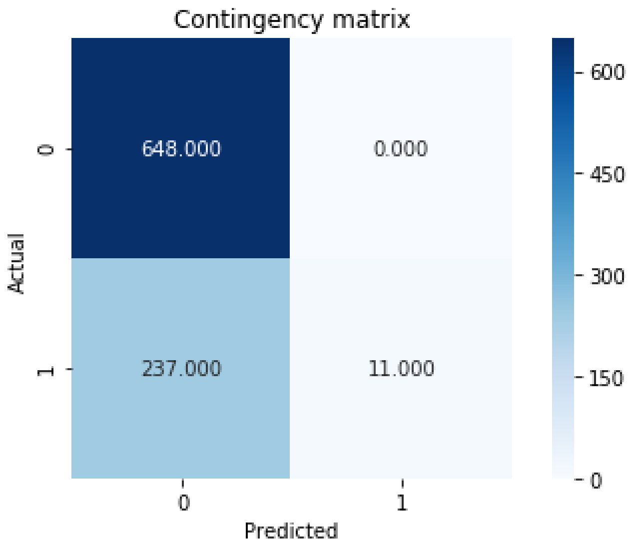
```python
# 5.2 - Single Linkage Hierarchical Clustering
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispanic or Latino'[1],
# 'Percent Hispanic or Latino'[2], 'Percent Age 29 and Under'[5], 'Percent Age 65 and Older'[6],
#'Percent Less than High School Degree'[9], 'Percent Less than High School Degree'[10],

# pick certain variables from x_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2], i[5], i[6], i[9], i[10]])


# cluster observations
clustering = linkage(X, method='single', metric='euclidean')
clusters = fcluster(clustering, 2, criterion = 'maxclust')

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.007194555265724431, 0.739203044845962]
```



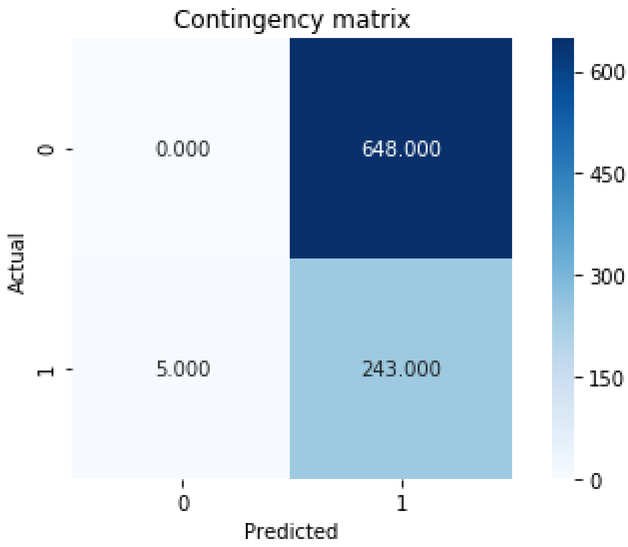
```python
# 5.3 - Complete Linkage Hierarchical Clustering
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispanic or Latino'[1],
# 'Percent Hispanic or Latino'[2]

# pick certain variables from x_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2]])


# cluster observations
clustering = linkage(X, method='complete', metric='euclidean')
clusters = fcluster(clustering, 2, criterion = 'maxclust')

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.03967438123028132, 0.7463661353840195]
```



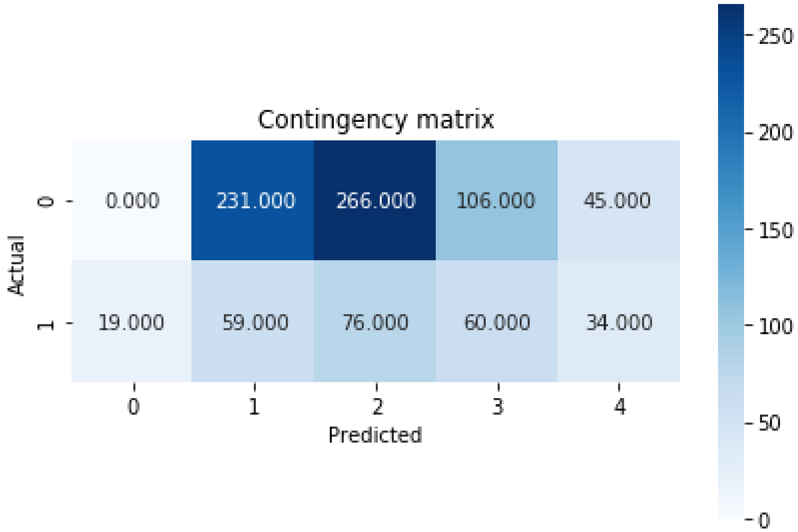
```python
# 5.4 - Complete Linkage Hierarchical Clustering
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispanic or Latino'[1],
# 'Percent Hispanic or Latino'[2], 'Percent Age 29 and Under'[5], 'Percent Age 65 and Older'[6],
#'Percent Less than High School Degree'[9], 'Percent Less than High School Degree'[10],

# pick certain variables from x_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2], i[5], i[6], i[9], i[10]])


# cluster observations
clustering = linkage(X, method='complete', metric='euclidean')
clusters = fcluster(clustering, 2, criterion = 'maxclust')

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.018001900285042876, 0.6977811688632597]
```



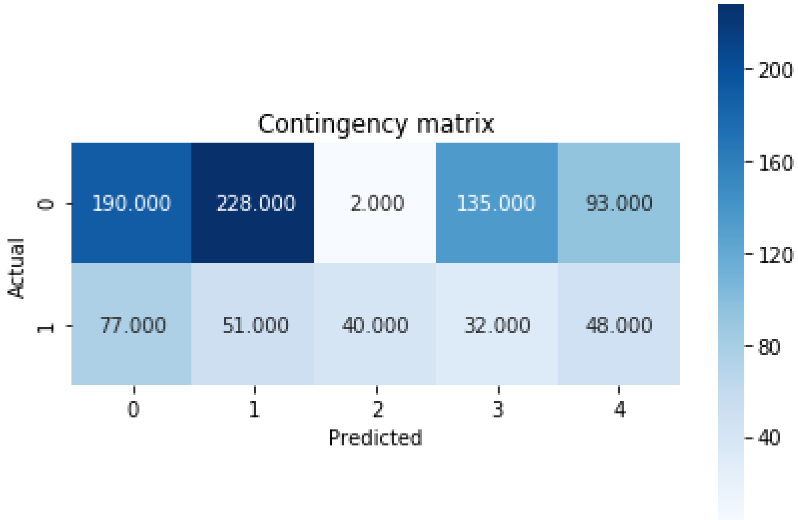
```
# 5.5 - K-Means with 5 clusters, 10 iterations
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispani
c or Latino'[1],
# 'Percent Hispanic or Latino'[2]

# pick certain variables from x_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2]])


# cluster observations
clustering = KMeans(n_clusters = 5, init='random', max_iter = 10, random_state=0).fit(X, y
_train)
clusters = clustering.labels_

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0481424456843671, 0.4262958247397856]
```



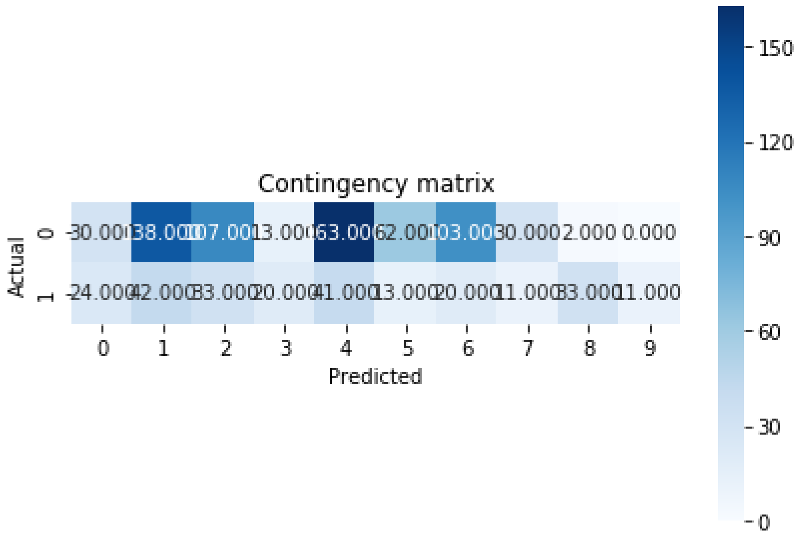
```python
# 5.6 - K-Means with 5 clusters, 10 iterations
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispani
c or Latino'[1],
# 'Percent Hispanic or Latino'[2], 'Percent Age 29 and Under'[5], 'Percent Age 65 and Olde
r'[6],
#'Percent Less than High School Degree'[9], 'Percent Less than High School Degree'[10],

# pick certain variables from x_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2], i[5], i[6], i[9], i[10]])


# cluster observations
clustering = KMeans(n_clusters = 5, init='random', max_iter = 10, random_state=0).fit(X, y
_train)
clusters = clustering.labels_

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.040886316162937586, 0.1966930179215471]
```



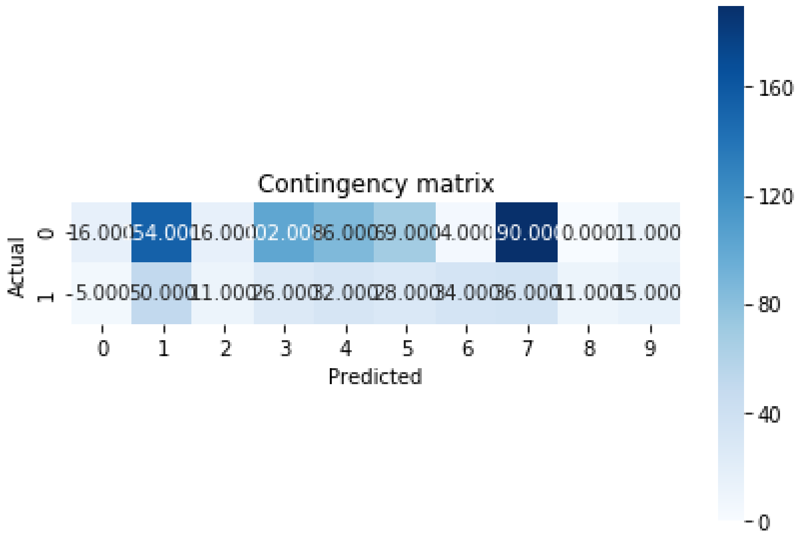
```python
# 5.7 - K-Means with 10 clusters, 25 iterations
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispani
c or Latino'[1],
# 'Percent Hispanic or Latino'[2]

# pick certain variables from X_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2]])

# cluster observations
clustering = KMeans(n_clusters = 10, init='random', max_iter=25, random_state=0).fit(X, y_
train)
clusters = clustering.labels_

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.03414156918182958, 0.40973910068783354]
```



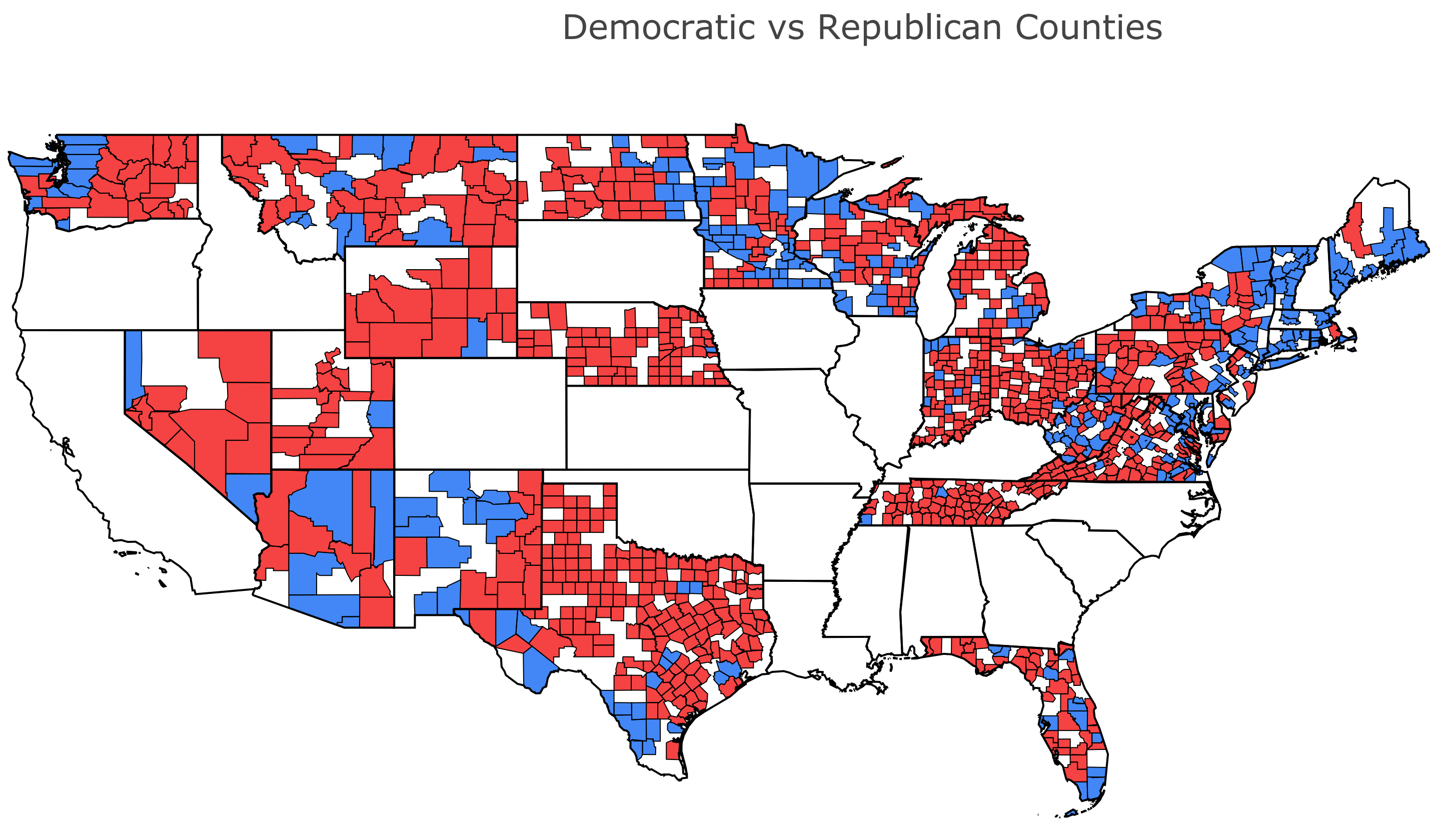
```
# 5.8 - K-Means with 10 clusters, 25 iterations
# Variables used: 'Percent White, not Hispanic or Latino'[0] , 'Percent Black, not Hispani
c or Latino'[1],
# 'Percent Hispanic or Latino'[2], 'Percent Age 29 and Under'[5], 'Percent Age 65 and Olde
r'[6],
#'Percent Less than High School Degree'[9], 'Percent Less than High School Degree'[10],

# pick certain variables from X_train_scaled
X = []
for i in x_train_scaled:
    X.append([i[0], i[1], i[2], i[5], i[6], i[9], i[10]])

# cluster observations
clustering = KMeans(n_clusters = 10, init='random', max_iter=25, random_state=0).fit(X, y_
train)
clusters = clustering.labels_

# build and display contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(y_train, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# determine adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(y_train, clusters)
silhouette_coefficient = metrics.silhouette_score(X, clusters)
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.04060135216345601, 0.22366240015269565]
```



```python
#Task 6
mergeFips = pd.concat([x_test['FIPS'],x_train['FIPS']])
fips = mergeFips
merged = pd.concat([y_test,y_train])
values = merged.tolist()

colorscale = ['#f54242', '#4287f5']

fig = ff.create_choropleth(fips=fips, values=values, scope=['usa'],
                           title_text = 'Democratic vs Republican Counties',
                           colorscale=colorscale,
                           state_outline={'color': 'rgb(0,0,0)', 'width': 1},
                           county_outline={'color': 'rgb(0,0,0)', 'width': 0.5},
                           legend_title='Party')
fig.layout.template = None
fig.show()
```

```
/Users/karanahuja/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:6211: FutureWarning:

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

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

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



```python
#Task 7
# Load dataset
data = pd.read_csv('demographics_test.csv')
data.head()
```

	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	Percent Age 29 and Under	Percent Age 65 and Older	Ho
0	NV	eureka	32011	1730	98.265896	0.057803	0.462428	0.346821	51.156069	27.109827	15.606936	70
1	TX	zavala	48507	12107	5.798299	0.594697	93.326175	9.193029	49.723301	49.302057	12.480383	26
2	VA	king george	51099	25260	73.804434	16.722090	4.441805	2.505938	50.166271	40.186065	11.868567	84
3	OH	hamilton	39061	805965	66.354867	25.654340	2.890944	5.086945	51.870615	40.779686	14.161657	50
4	TX	austin	48015	29107	63.809393	8.479060	25.502456	9.946061	50.671660	37.351840	17.799842	56

```python
# Regression
# Predicting the number of votes cast for the Democratic party in each county
pred_variable = ['Percent Age 29 and Under','Percent Age 65 and Older','Total Population',
'Percent Foreign Born','Percent Hispanic or Latino','Percent Black, not Hispanic or Latin
o','Percent White, not Hispanic or Latino','Percent Female','Percent Unemployed']
print("Democratic votes for each county for test dataset: \n")
predicted = model1.predict(X = data[pred_variable])

for i in range(len(predicted)):
    predicted[i]=int(predicted[i])
    if predicted[i]<0:
        predicted[i]=0
print(predicted)
```

Democratic votes for each county for test dataset:

```
[6.55400e+03 0.00000e+00 7.93100e+03 1.71680e+05 8.27000e+03 1.11170e+04
 0.00000e+00 3.24190e+04 1.04044e+05 2.72880e+04 0.00000e+00 0.00000e+00
 4.69000e+03 0.00000e+00 1.51373e+05 1.31400e+03 0.00000e+00 6.55020e+04
 4.29510e+04 1.13960e+04 0.00000e+00 1.27200e+03 0.00000e+00 9.51600e+03
 0.00000e+00 0.00000e+00 6.67700e+03 4.38700e+04 0.00000e+00 1.40580e+04
 8.88550e+04 1.27280e+04 2.77530e+04 1.98400e+03 1.80200e+03 1.21560e+04
 4.97600e+03 0.00000e+00 0.00000e+00 6.39500e+03 3.21300e+03 9.72400e+03
 4.71560e+04 4.38380e+04 0.00000e+00 1.28251e+05 0.00000e+00 2.47600e+03
 2.36110e+04 1.33520e+04 2.34700e+03 1.46503e+05 4.57200e+03 6.22100e+03
 0.00000e+00 7.09400e+03 1.63430e+04 2.43780e+04 5.63500e+03 1.27640e+04
 2.68500e+03 7.43400e+03 1.26888e+05 1.04580e+04 1.81810e+04 1.07780e+04
 1.28511e+05 1.46100e+03 0.00000e+00 3.66600e+03 1.36710e+04 0.00000e+00
 3.16500e+03 2.70370e+04 0.00000e+00 0.00000e+00 5.61600e+03 0.00000e+00
 7.04000e+03 1.13170e+04 5.20300e+03 5.42600e+03 2.07550e+04 8.60000e+02
 4.86400e+03 1.57800e+03 0.00000e+00 1.34970e+04 1.44390e+04 8.86900e+03
 1.01060e+04 0.00000e+00 9.64210e+04 0.00000e+00 1.91970e+04 0.00000e+00
 0.00000e+00 6.29200e+03 7.66100e+03 1.90740e+04 7.28580e+04 3.95100e+03
 6.41000e+04 7.23510e+04 0.00000e+00 8.76290e+04 0.00000e+00 8.68100e+03
 0.00000e+00 4.72670e+04 0.00000e+00 3.19200e+03 1.25400e+03 0.00000e+00
 2.51100e+03 1.36770e+04 1.80500e+04 9.67280e+04 1.05200e+04 3.63100e+04
 3.92490e+04 6.59540e+04 2.50800e+03 9.75200e+03 3.26200e+03 1.44824e+05
 4.39100e+03 1.26350e+04 3.29010e+04 3.45040e+04 2.22070e+04 0.00000e+00
 6.46340e+04 7.59300e+03 3.19720e+04 8.62600e+03 1.98120e+04 5.75900e+03
 1.59110e+04 1.06235e+05 3.03677e+05 1.58400e+03 1.59540e+04 1.67400e+04
 0.00000e+00 4.18700e+03 2.20940e+04 0.00000e+00 4.05400e+03 8.47000e+02
 5.63180e+04 1.90000e+01 0.00000e+00 6.79400e+03 1.45980e+04 7.38880e+04
 0.00000e+00 7.40000e+01 5.36490e+04 1.17190e+04 3.34000e+02 2.96300e+03
 8.36800e+03 6.04400e+03 3.76600e+03 3.05900e+03 5.05200e+03 2.77200e+03
 5.44200e+03 3.89880e+04 1.31540e+04 2.18800e+03 1.03420e+04 4.36300e+03
 6.34700e+03 1.58210e+04 3.12780e+04 3.82260e+04 0.00000e+00 6.31600e+03
 1.85360e+04 1.62730e+04 1.04086e+05 4.97800e+03 5.76300e+03 8.36080e+04
 4.42600e+03 2.16700e+03 7.25300e+03 0.00000e+00 1.50860e+04 4.43600e+03
 1.94190e+04 6.45900e+03 4.75940e+04 5.98220e+04 7.51030e+04 1.38922e+05
 2.43100e+03 6.74000e+03 2.21950e+04 1.18890e+04 1.12500e+03 0.00000e+00
 2.79300e+03 1.06720e+05 9.60880e+04 3.68400e+03 0.00000e+00 0.00000e+00
 9.18200e+03 1.71000e+03 5.70800e+03 1.11980e+04 0.00000e+00 8.18000e+02
 1.94740e+04 0.00000e+00 1.12060e+04 3.98800e+03 8.70400e+03 1.33500e+04
 2.01060e+04 2.39780e+04 0.00000e+00 1.12740e+04 0.00000e+00 6.64000e+02
 1.88600e+03 8.49900e+03 1.40180e+04 3.97410e+04 6.37400e+03 9.21300e+03
 1.12150e+04 1.26850e+04 2.13550e+04 1.82560e+04 8.75700e+03 1.49200e+03
```

```
6.72300e+03 7.48100e+03 0.00000e+00 1.08410e+04 2.40480e+04 0.00000e+00
0.00000e+00 1.21590e+04 3.20044e+05 0.00000e+00 3.11578e+05 9.72600e+03
1.97970e+04 0.00000e+00 1.92120e+04 1.67800e+03 1.75900e+03 1.93837e+05
1.53300e+03 2.83810e+04 0.00000e+00 1.28890e+04 1.09027e+05 2.86170e+04
0.00000e+00 6.26210e+04 0.00000e+00 8.81500e+03 8.40800e+03 0.00000e+00
0.00000e+00 2.37380e+04 8.38300e+03 0.00000e+00 2.49690e+04 2.95500e+03
0.00000e+00 8.34200e+03 1.24660e+04 8.78000e+03 0.00000e+00 2.45400e+03
3.94900e+03 2.24470e+04 2.77890e+04 1.59438e+05 2.55860e+05 3.66300e+03
1.69920e+04 0.00000e+00 6.96000e+03 7.97000e+02 8.46900e+03 6.53400e+03
1.12700e+04 3.00000e+02 0.00000e+00 7.00800e+03 1.17900e+04 0.00000e+00
2.68600e+03 2.65080e+04 2.78400e+03 1.35120e+04 0.00000e+00 1.34160e+04
4.38400e+03 2.68200e+03 9.12000e+03 6.88100e+03 3.71580e+04 8.91200e+03
1.61500e+03 5.90360e+04 4.13300e+03 0.00000e+00 1.86410e+04 1.63400e+03
1.26763e+05 0.00000e+00 0.00000e+00 4.56400e+03 1.59810e+04 3.49800e+03
9.11300e+03 5.14800e+03 1.07100e+03 2.27200e+03 1.97290e+04 8.81600e+03
3.16000e+03 2.64606e+05 8.14700e+03 0.00000e+00 2.11600e+03 1.77735e+05
4.66030e+04 0.00000e+00 9.74800e+03 4.99700e+03 3.02000e+03 5.88200e+03
6.13450e+04 1.03594e+05 2.20500e+03 3.29740e+04 4.92300e+03 1.66300e+03
1.47890e+04 2.98200e+03 1.49100e+03 1.57150e+04 1.89740e+04 3.32000e+03
1.14463e+05 2.44300e+03 1.80700e+03 0.00000e+00 4.99500e+03 0.00000e+00
7.36800e+03 9.55000e+03 0.00000e+00 2.97800e+03 5.94100e+03 7.28200e+03
0.00000e+00 1.14210e+04 2.12750e+04 2.34600e+03 1.79000e+03 0.00000e+00
2.98590e+04 1.33431e+05 4.39110e+04 4.35170e+04 3.69268e+05 3.68400e+03
8.76100e+03 4.80000e+02 3.95700e+03 4.63200e+03 6.13900e+03 0.00000e+00
4.54440e+04 5.43690e+04 0.00000e+00 0.00000e+00 2.76800e+03 0.00000e+00
2.54400e+04 3.87930e+04 5.66970e+04 2.63964e+05 7.06000e+03 4.28830e+04
8.06710e+04 0.00000e+00 2.36790e+04 4.33600e+03]
```

```
# Predicting the number of votes cast for the Republican party in each county.
pred_variable = ['Total Population','Percent White, not Hispanic or Latino','Percent Blac
k, not Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Fe
male','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income','Per
cent Unemployed','Percent Less than High School Degree','Percent Less than Bachelor\'s Deg
ree','Percent Rural']

print("Republican votes for each county for test dataset: \n")
predicted2 = model2.predict(X= data[pred_variable])
print(predicted2)
for i in range(len(predicted2)):
    predicted2[i]=int(predicted2[i])
    if predicted2[i]<0:
        predicted2[i]=0
print(predicted2)
```

```
Republican votes for each county for test dataset:

[ 7.83564553e+03  4.87809621e+03  1.71401901e+04  1.12207485e+05
  4.56270113e+03  1.41163207e+04  1.81422185e+04  2.80877296e+04
  5.46252566e+04  2.91670165e+04 -7.44993963e+02  1.58061333e+04
  7.24219979e+03  2.11601535e+03  9.87892268e+04  3.55463175e+03
  1.99688204e+03  6.18108051e+04  3.50682141e+04  1.63812825e+04
  5.49010428e+03 -1.08335920e+04 -3.00155791e+03  5.14342693e+03
  6.06695539e+03  5.91088597e+03  1.24973413e+04  3.70910219e+04
  6.34099986e+02  2.71523356e+04  5.83823493e+04 -7.00634762e+03
  4.58872690e+04  4.88652267e+02  7.08695032e+03  1.57956321e+04
  1.14514470e+04  1.19132134e+04  7.60301941e+03  7.88982733e+03
  1.18317299e+04  1.64239739e+04  3.65667012e+04  2.22168790e+04
 -6.86380296e+03  8.61309848e+04  3.80188344e+03 -1.37041871e+04
  2.49607535e+04  5.03433664e+03  9.76358840e+03  6.67995999e+04
 -9.68188877e+03  1.24501862e+04  9.05360283e+02  2.05995475e+04
  1.97868441e+04  2.22877805e+04  8.70698351e+03  1.86141533e+04
  8.59818654e+03 -1.06035845e+03  8.02245069e+04  1.83289508e+04
  1.88496796e+04  1.61683909e+04  9.72867927e+04  2.88217062e+03
 -4.85176074e+03  1.50645575e+04  7.90989579e+03 -2.70348781e+03
  5.11481949e+03  1.61248664e+04 -6.60165491e+03 -7.76066735e+03
  1.23624033e+04  9.78755809e+03  1.69147042e+03  2.36247153e+02
  4.79034040e+03  1.45785945e+04  1.54414269e+04  1.23685250e+04
  2.24073938e+01  4.17642656e+02  1.56243419e+03  2.02845915e+04
  2.81553668e+04  4.24353974e+03  1.18790042e+04  7.44872855e+02
  7.12235539e+04 -2.78296640e+02  1.97807709e+04  8.54944409e+03
 -6.78566257e+03  7.29437534e+03  1.72474458e+04  2.14936875e+04
  4.83314036e+04  9.71888890e+03  6.26081398e+04  5.34973877e+04
  4.60302836e+03  4.58369752e+04  3.99809096e+03  7.04570310e+03
  7.90511381e+03  2.83093531e+04  5.16608144e+03  7.17843124e+03
 -2.14636612e+03 -3.56803928e+03  6.58998406e+02  1.54838592e+04
  2.65402275e+04  6.67265518e+04  1.33390242e+04  3.40876914e+03
  4.18061879e+04  5.58549712e+04  1.25487578e+03  8.23708445e+03
  3.08835581e+03  1.08057690e+05  9.27843932e+03  1.13092708e+04
  9.57471213e+03  4.22762192e+04  2.30772204e+04  5.19107030e+03
  3.99670235e+04  6.86589660e+03  3.25404655e+04  2.51613713e+03
  2.01054765e+04  7.63917964e+03 -1.17788169e+04  7.40662615e+04
  2.01074822e+05  2.71845466e+03  1.95273177e+04 -9.92141126e+03
  7.18296652e+03  3.52574000e+03  2.27021464e+04  1.32229362e+04
  1.12963500e+04 -1.36625138e+03  3.99391382e+04 -4.21382516e+03
 -8.65060483e+03  1.41714546e+04  1.88272073e+04  3.84262418e+04
  1.25428053e+04  1.29962396e+04  4.72147327e+04  2.17861246e+04
```

```
-6.78245349e+03  1.14233377e+04  8.09794579e+03  1.04098475e+04
 5.52491799e+03  1.17928995e+03 -2.02187682e+03  4.99169093e+03
 1.33369629e+04  2.31119985e+04  1.28879014e+04  8.33192251e+02
 1.73826652e+04  6.04521507e+03  7.72976312e+03  1.68100823e+04
 3.49929357e+04  4.88828901e+04 -5.88614336e+03  1.89232361e+04
 1.52306349e+04  2.04521010e+04  7.98059708e+04  8.10651107e+03
 9.86311766e+03  6.75608839e+04  1.08496805e+04  1.09692771e+04
 9.96369073e+03  1.28289615e+04  3.01456295e+04 -7.01892209e+03
 2.04330445e+04  1.71197276e+04  4.50363957e+04  4.44364554e+04
 5.10267801e+04  1.08036507e+05  1.84616893e+03  1.02034069e+04
 3.63105744e+04 -1.57517269e+04  3.17361031e+03  9.69993099e+03
 1.84752021e+03  7.92773496e+04  7.00379090e+04  9.88455151e+03
 1.25755310e+04 -8.37648397e+02  1.30951565e+04  1.22467779e+04
-1.05271634e+04  7.57966594e+03  5.30859266e+03  3.75275655e+03
 2.78828430e+04 -3.41562914e+03  8.35547229e+03  6.99653512e+03
 9.81658957e+03  3.99822287e+03  6.71389958e+03  4.83552731e+04
-9.76159668e+03  3.67635552e+03  4.80421222e+03  1.53514023e+04
 4.74397174e+03  7.41119891e+02  2.05594470e+04  4.23746438e+04
 3.66314889e+02  7.67899097e+03  1.54473725e+04  1.42803729e+04
 2.83169729e+04  1.91937150e+04  1.11163852e+04  2.66040505e+04
 1.68065655e+04 -2.60300241e+03  9.72516311e+03  1.92790194e+04
 2.82823624e+04 -2.81616234e+03  1.19951715e+04  4.67159094e+04
 1.59202562e+05  1.69742217e+04  1.82556101e+05  4.78934533e+03
 2.32125597e+04 -4.16854217e+03  1.72365258e+04  1.44646608e+03
-1.40369675e+03  9.02176607e+04  4.11505859e+03  3.28474688e+04
-6.00780163e+03  9.45045698e+03  7.44532445e+04  3.06716238e+04
 1.03322031e+03  5.17712747e+04  7.06596215e+03 -5.61679643e+03
 8.09923180e+03  1.03695383e+04  6.92948479e+03  2.34849729e+04
 1.49580853e+04  1.13176074e+04  3.30311106e+04  1.42361077e+04
 4.01196372e+03 -7.80071222e+03 -7.65631457e+03  4.18611217e+03
-1.14215109e+04  4.45486854e+03  3.27219842e+03  2.00228214e+04
 2.21092716e+04  9.89215809e+04  1.44803720e+05  8.70925578e+03
 3.02323355e+04 -5.17151220e+03  1.92248040e+04  9.05006874e+03
 1.32605762e+04  1.25621072e+04 -2.94555725e+02 -6.09585028e+03
 2.70146830e+03  6.03399187e+03  1.56880139e+04  4.79185665e+03
 7.10238672e+03  3.53568701e+04 -8.68836793e+03  1.72864977e+04
 6.03650314e+03 -1.60651257e+04  3.38807344e+03  8.46872309e+03
 1.56375829e+04  1.11024307e+04  3.45781058e+04  8.70526914e+03
 1.28600272e+03  5.45655788e+04  1.10424018e+04 -4.15886165e+03
 2.10839774e+04  7.17343862e+03  9.51330399e+04  1.13162871e+04
 8.61833439e+03  2.04782757e+04  1.83721230e+04  3.94541953e+03
 8.91052700e+03  7.67671819e+03  1.49302108e+03  4.68460309e+03
-1.37934464e+04  1.38004866e+04  5.62451340e+03  1.70422042e+05
```

```
 -9.58655891e+03 -2.75803110e+03  5.08741149e+03  1.20294480e+05
  4.24240687e+04 -2.38577099e+03  1.04927765e+04  1.60434418e+03
 -5.01283095e+03  1.17477432e+04  2.38852865e+04  6.81203129e+04
  6.17327334e+03  3.20255680e+04 -7.81531367e+03 -2.76124313e+03
 -1.18656491e+04  8.18786722e+03 -8.04867590e+03  1.76742421e+04
  3.44773010e+04  5.34655368e+03  7.58308832e+04  1.32631863e+04
  6.71610291e+02 -1.07873523e+03  2.37857059e+04  1.17380424e+04
 -2.12092869e+03  4.78366120e+03 -5.60830746e+03 -3.70715976e+03
  5.21685111e+03  6.52212892e+03 -7.03907753e+03  1.31773951e+04
  2.96861471e+04  8.72457401e+03  2.85171262e+03  2.81466225e+03
  1.42785872e+04  9.01522180e+04  2.55634880e+04  3.23894225e+04
  2.22756220e+05  8.49822727e+03  8.90596578e+03  3.16779819e+03
  4.30596312e+03  1.30826497e+04  1.16759209e+04 -4.83994120e+02
  3.32169749e+04  4.13482865e+04  5.00453876e+03  8.77799459e+03
 -4.57617062e+02  1.42035069e+04  2.73254436e+04  3.58283079e+04
  5.08981961e+04  1.64072080e+05 -3.26393690e+02  3.75863093e+04
  6.25139961e+04  1.64680708e+03  2.80746046e+04  1.10009177e+04]
[7.83500e+03 4.87800e+03 1.71400e+04 1.12207e+05 4.56200e+03 1.41160e+04
 1.81420e+04 2.80870e+04 5.46250e+04 2.91670e+04 0.00000e+00 1.58060e+04
 7.24200e+03 2.11600e+03 9.87890e+04 3.55400e+03 1.99600e+03 6.18100e+04
 3.50680e+04 1.63810e+04 5.49000e+03 0.00000e+00 0.00000e+00 5.14300e+03
 6.06600e+03 5.91000e+03 1.24970e+04 3.70910e+04 6.34000e+02 2.71520e+04
 5.83820e+04 0.00000e+00 4.58870e+04 4.88000e+02 7.08600e+03 1.57950e+04
 1.14510e+04 1.19130e+04 7.60300e+03 7.88900e+03 1.18310e+04 1.64230e+04
 3.65660e+04 2.22160e+04 0.00000e+00 8.61300e+04 3.80100e+03 0.00000e+00
 2.49600e+04 5.03400e+03 9.76300e+03 6.67990e+04 0.00000e+00 1.24500e+04
 9.05000e+02 2.05990e+04 1.97860e+04 2.22870e+04 8.70600e+03 1.86140e+04
 8.59800e+03 0.00000e+00 8.02240e+04 1.83280e+04 1.88490e+04 1.61680e+04
 9.72860e+04 2.88200e+03 0.00000e+00 1.50640e+04 7.90900e+03 0.00000e+00
 5.11400e+03 1.61240e+04 0.00000e+00 0.00000e+00 1.23620e+04 9.78700e+03
 1.69100e+03 2.36000e+02 4.79000e+03 1.45780e+04 1.54410e+04 1.23680e+04
 2.20000e+01 4.17000e+02 1.56200e+03 2.02840e+04 2.81550e+04 4.24300e+03
 1.18790e+04 7.44000e+02 7.12230e+04 0.00000e+00 1.97800e+04 8.54900e+03
 0.00000e+00 7.29400e+03 1.72470e+04 2.14930e+04 4.83310e+04 9.71800e+03
 6.26080e+04 5.34970e+04 4.60300e+03 4.58360e+04 3.99800e+03 7.04500e+03
 7.90500e+03 2.83090e+04 5.16600e+03 7.17800e+03 0.00000e+00 0.00000e+00
 6.58000e+02 1.54830e+04 2.65400e+04 6.67260e+04 1.33390e+04 3.40800e+03
 4.18060e+04 5.58540e+04 1.25400e+03 8.23700e+03 3.08800e+03 1.08057e+05
 9.27800e+03 1.13090e+04 9.57400e+03 4.22760e+04 2.30770e+04 5.19100e+03
 3.99670e+04 6.86500e+03 3.25400e+04 2.51600e+03 2.01050e+04 7.63900e+03
 0.00000e+00 7.40660e+04 2.01074e+05 2.71800e+03 1.95270e+04 0.00000e+00
 7.18200e+03 3.52500e+03 2.27020e+04 1.32220e+04 1.12960e+04 0.00000e+00
 3.99390e+04 0.00000e+00 0.00000e+00 1.41710e+04 1.88270e+04 3.84260e+04
```

```
1.25420e+04 1.29960e+04 4.72140e+04 2.17860e+04 0.00000e+00 1.14230e+04
8.09700e+03 1.04090e+04 5.52400e+03 1.17900e+03 0.00000e+00 4.99100e+03
1.33360e+04 2.31110e+04 1.28870e+04 8.33000e+02 1.73820e+04 6.04500e+03
7.72900e+03 1.68100e+04 3.49920e+04 4.88820e+04 0.00000e+00 1.89230e+04
1.52300e+04 2.04520e+04 7.98050e+04 8.10600e+03 9.86300e+03 6.75600e+04
1.08490e+04 1.09690e+04 9.96300e+03 1.28280e+04 3.01450e+04 0.00000e+00
2.04330e+04 1.71190e+04 4.50360e+04 4.44360e+04 5.10260e+04 1.08036e+05
1.84600e+03 1.02030e+04 3.63100e+04 0.00000e+00 3.17300e+03 9.69900e+03
1.84700e+03 7.92770e+04 7.00370e+04 9.88400e+03 1.25750e+04 0.00000e+00
1.30950e+04 1.22460e+04 0.00000e+00 7.57900e+03 5.30800e+03 3.75200e+03
2.78820e+04 0.00000e+00 8.35500e+03 6.99600e+03 9.81600e+03 3.99800e+03
6.71300e+03 4.83550e+04 0.00000e+00 3.67600e+03 4.80400e+03 1.53510e+04
4.74300e+03 7.41000e+02 2.05590e+04 4.23740e+04 3.66000e+02 7.67800e+03
1.54470e+04 1.42800e+04 2.83160e+04 1.91930e+04 1.11160e+04 2.66040e+04
1.68060e+04 0.00000e+00 9.72500e+03 1.92790e+04 2.82820e+04 0.00000e+00
1.19950e+04 4.67150e+04 1.59202e+05 1.69740e+04 1.82556e+05 4.78900e+03
2.32120e+04 0.00000e+00 1.72360e+04 1.44600e+03 0.00000e+00 9.02170e+04
4.11500e+03 3.28470e+04 0.00000e+00 9.45000e+03 7.44530e+04 3.06710e+04
1.03300e+03 5.17710e+04 7.06500e+03 0.00000e+00 8.09900e+03 1.03690e+04
6.92900e+03 2.34840e+04 1.49580e+04 1.13170e+04 3.30310e+04 1.42360e+04
4.01100e+03 0.00000e+00 0.00000e+00 4.18600e+03 0.00000e+00 4.45400e+03
3.27200e+03 2.00220e+04 2.21090e+04 9.89210e+04 1.44803e+05 8.70900e+03
3.02320e+04 0.00000e+00 1.92240e+04 9.05000e+03 1.32600e+04 1.25620e+04
0.00000e+00 0.00000e+00 2.70100e+03 6.03300e+03 1.56880e+04 4.79100e+03
7.10200e+03 3.53560e+04 0.00000e+00 1.72860e+04 6.03600e+03 0.00000e+00
3.38800e+03 8.46800e+03 1.56370e+04 1.11020e+04 3.45780e+04 8.70500e+03
1.28600e+03 5.45650e+04 1.10420e+04 0.00000e+00 2.10830e+04 7.17300e+03
9.51330e+04 1.13160e+04 8.61800e+03 2.04780e+04 1.83720e+04 3.94500e+03
8.91000e+03 7.67600e+03 1.49300e+03 4.68400e+03 0.00000e+00 1.38000e+04
5.62400e+03 1.70422e+05 0.00000e+00 0.00000e+00 5.08700e+03 1.20294e+05
4.24240e+04 0.00000e+00 1.04920e+04 1.60400e+03 0.00000e+00 1.17470e+04
2.38850e+04 6.81200e+04 6.17300e+03 3.20250e+04 0.00000e+00 0.00000e+00
0.00000e+00 8.18700e+03 0.00000e+00 1.76740e+04 3.44770e+04 5.34600e+03
7.58300e+04 1.32630e+04 6.71000e+02 0.00000e+00 2.37850e+04 1.17380e+04
0.00000e+00 4.78300e+03 0.00000e+00 0.00000e+00 5.21600e+03 6.52200e+03
0.00000e+00 1.31770e+04 2.96860e+04 8.72400e+03 2.85100e+03 2.81400e+03
1.42780e+04 9.01520e+04 2.55630e+04 3.23890e+04 2.22756e+05 8.49800e+03
8.90500e+03 3.16700e+03 4.30500e+03 1.30820e+04 1.16750e+04 0.00000e+00
3.32160e+04 4.13480e+04 5.00400e+03 8.77700e+03 0.00000e+00 1.42030e+04
2.73250e+04 3.58280e+04 5.08980e+04 1.64072e+05 0.00000e+00 3.75860e+04
6.25130e+04 1.64600e+03 2.80740e+04 1.10000e+04]
```

```python
# Classification
# Deciding the party of each county
# Standardize sets
data_scaled = scaler.transform(data[['Total Population','Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign Born','Percent Female','Percent Age 29 and Under','Percent Age 65 and Older','Median Household Income','Percent Unemployed','Percent Less than High School Degree','Percent Less than Bachelor\'s Degree','Percent Rural']])

# Predict class labels using SVM classifier
party = bestclassifier.predict(data_scaled)
print(party)
```

```
[0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0
 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0
 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0
 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0
 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0]
```

```
import csv

with open('output.csv', mode='w') as csv_file:
    fieldnames = ['State', 'County', 'Democratic', 'Republican', 'Party']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    writer.writeheader()

    with open('demographics_test.csv', 'r') as readFile:
        reader = csv.reader(readFile)
        next(reader)
        for i,row in enumerate(reader):
            writer.writerow({'State': row[0], 'County': row[1], 'Democratic': int(round(predicted[i])),'Republican': int(round(predicted2[i])),'Party':int(round(party[i]))})
```