

```
In [1]: import pandas as pd

# Task 1
from sklearn.model_selection import train_test_split, KFold, cross_val_score

# TASK 2
from sklearn.preprocessing import MinMaxScaler

# TASK 3a
from sklearn import linear_model
import statsmodels.api as sm

# Task 4
from sklearn import tree
from sklearn.externals.six import StringIO
from sklearn import metrics
import pydot
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier

# Task 5
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.cluster import KMeans, DBSCAN
from sklearn import metrics

# Task 6
import plotly as py
import plotly.graph_objs as go
import plotly.figure_factory as ff

# Task 7
import numpy as np
```

C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\externals\six.py:31: DeprecationWarning: The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on the official version of six (<https://pypi.org/project/six/>).  
"(<https://pypi.org/project/six/>).", DeprecationWarning)

```
In [2]: # Dateset from PProject 01  
merged = pd.read_csv("merged.csv")  
merged.head()
```

Out[2]:

Unnamed: 0		State	County	Democratic	Republican	FIPS	Total Population	Citizen Voting-Age Population	Percent White, not Hispanic or Latino	Percent Black, not Hispanic or Latino	...	Percent Unemployed	Percent Less than High School Degree
0	0	AZ	apache	16298.0	7810.0	4001	72346	NaN	18.571863	0.486551	...	15.807433	21.758252
1	1	AZ	cochise	17383.0	26929.0	4003	128177	92915.0	56.299492	3.714395	...	8.567108	13.409171
2	2	AZ	coconino	34240.0	19249.0	4005	138064	104265.0	54.619597	1.342855	...	8.238305	11.085381
3	3	AZ	gila	7643.0	12180.0	4007	53179	NaN	63.222325	0.552850	...	12.129932	15.729958
4	4	AZ	graham	3368.0	6870.0	4009	37529	NaN	51.461536	1.811932	...	14.424104	14.580797

5 rows × 26 columns



```
In [3]: # Dateset from PProject 01
```

```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 26 columns):
Unnamed: 0                      1200 non-null int64
State                           1200 non-null object
County                          1200 non-null object
Democratic                      1200 non-null float64
Republican                      1200 non-null float64
FIPS                            1200 non-null int64
Total Population                 1200 non-null int64
Citizen Voting-Age Population   520 non-null float64
Percent White, not Hispanic or Latino 1200 non-null float64
Percent Black, not Hispanic or Latino 1200 non-null float64
Percent Hispanic or Latino      1200 non-null float64
Percent Foreign Born            1200 non-null float64
Percent Female                   1200 non-null float64
Percent Age 29 and Under       1200 non-null float64
Percent Age 65 and Older       1200 non-null float64
Median Household Income         1200 non-null int64
Percent Unemployed              1200 non-null float64
Percent Less than High School Degree 1200 non-null float64
Percent Less than Bachelor's Degree 1200 non-null float64
Percent Rural                   1200 non-null float64
Party                            1200 non-null object
Percent Age between 30 and 64 inclusive 1200 non-null float64
Percent Male                     1200 non-null float64
Percent more than High School Degree 1200 non-null float64
Percent more than Bachelor's Degree 1200 non-null float64
Citizen Voting-Age Population voted 1200 non-null float64
dtypes: float64(19), int64(4), object(3)
memory usage: 243.8+ KB
```

```
In [4]: # Dateset from PProject 01
```

```
# converting our dataset into the format of data set merged_train
merged = merged.drop(["Unnamed: 0", "Citizen Voting-Age Population", "Citizen Voting-Age Population voted",
                     "Percent Male", "Percent Age between 30 and 64 inclusive",
                     "Percent more than High School Degree", "Percent more than Bachelor's Degree"], axis = 1
)
```

```
In [5]: # Dateset from PProject 01
# converting our dataset into the format of data set merged_train
merged = merged[(merged[["Democratic", "Republican"]] != 0).all(axis = 1)]
```

```
In [6]: # Dateset from PProject 01
# converting our dataset into the format of data set merged_train
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1195 entries, 0 to 1199
Data columns (total 19 columns):
State                      1195 non-null object
County                     1195 non-null object
Democratic                 1195 non-null float64
Republican                 1195 non-null float64
FIPS                       1195 non-null int64
Total Population            1195 non-null int64
Percent White, not Hispanic or Latino 1195 non-null float64
Percent Black, not Hispanic or Latino 1195 non-null float64
Percent Hispanic or Latino   1195 non-null float64
Percent Foreign Born        1195 non-null float64
Percent Female               1195 non-null float64
Percent Age 29 and Under    1195 non-null float64
Percent Age 65 and Older    1195 non-null float64
Median Household Income     1195 non-null int64
Percent Unemployed          1195 non-null float64
Percent Less than High School Degree 1195 non-null float64
Percent Less than Bachelor's Degree 1195 non-null float64
Percent Rural                1195 non-null float64
Party                       1195 non-null object
dtypes: float64(13), int64(3), object(3)
memory usage: 186.7+ KB
```

In [7]: # Dataset given by the PROFESSOR

```
merged2 = pd.read_csv("merged_train.csv")
merged2.head()
```

Out[7]:

	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	Median Household Income	Perc Unemplo
0	AZ	apache	4001	72346	18.571863	0.486551	5.947806	1.719515	50.598513	45.854643	13.322091	32460	15.807
1	AZ	cochise	4003	128177	56.299492	3.714395	34.403208	11.458374	49.069646	37.902276	19.756275	45383	8.567
2	AZ	coconino	4005	138064	54.619597	1.342855	13.711033	4.825298	50.581614	48.946141	10.873943	51106	8.238
3	AZ	gila	4007	53179	63.222325	0.552850	18.548675	4.249798	50.296170	32.238290	26.397638	40593	12.129
4	AZ	graham	4009	37529	51.461536	1.811932	32.097844	4.385942	46.313518	46.393456	12.315809	47422	14.424

In [8]: # Dataset given by the PROFESSOR

```
merged2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1195 entries, 0 to 1194
Data columns (total 19 columns):
State                      1195 non-null object
County                     1195 non-null object
FIPS                       1195 non-null int64
Total Population           1195 non-null int64
Percent White, not Hispanic or Latino 1195 non-null float64
Percent Black, not Hispanic or Latino 1195 non-null float64
Percent Hispanic or Latino    1195 non-null float64
Percent Foreign Born        1195 non-null float64
Percent Female              1195 non-null float64
Percent Age 29 and Under   1195 non-null float64
Percent Age 65 and Older   1195 non-null float64
Median Household Income     1195 non-null int64
Percent Unemployed          1195 non-null float64
Percent Less than High School Degree 1195 non-null float64
Percent Less than Bachelor's Degree 1195 non-null float64
Percent Rural               1195 non-null float64
Democratic                 1195 non-null int64
Republican                 1195 non-null int64
Party                      1195 non-null int64
dtypes: float64(11), int64(6), object(2)
memory usage: 177.5+ KB
```

In [9]: # Dateset from PProject 01

```
# converting our dataset into the format of data set merged_train
merged = merged[["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", "Median Household Income",
                 "Percent Unemployed", "Percent Less than High School Degree",
                 "Percent Less than Bachelor's Degree",
                 "Percent Rural",
                 "Democratic", "Republican", "Party"]]
```

```
In [10]: # Dateset from PProject 01
# converting our dataset into the format of data set merged_train
merged.columns
```

```
Out[10]: Index(['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', 'Median Household Income',
       'Percent Unemployed', 'Percent Less than High School Degree',
       'Percent Less than Bachelor's Degree', 'Percent Rural', 'Democratic',
       'Republican', 'Party'],
      dtype='object')
```

```
In [11]: # Dateset from PProject 01
# converting our dataset into the format of data set merged_train
merged.head()
```

```
Out[11]:
```

	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	Median Household Income	Perc Unemplo
0	AZ	apache	4001	72346	18.571863	0.486551	5.947806	1.719515	50.598513	45.854643	13.322091	32460	15.807
1	AZ	cochise	4003	128177	56.299492	3.714395	34.403208	11.458374	49.069646	37.902276	19.756275	45383	8.567
2	AZ	coconino	4005	138064	54.619597	1.342855	13.711033	4.825298	50.581614	48.946141	10.873943	51106	8.238
3	AZ	gila	4007	53179	63.222325	0.552850	18.548675	4.249798	50.296170	32.238290	26.397638	40593	12.129
4	AZ	graham	4009	37529	51.461536	1.811932	32.097844	4.385942	46.313518	46.393456	12.315809	47422	14.424

◀ ▶

```
In [12]: #X = merged.iloc[:, :17]
#Y = merged.iloc[:, 17:19]
```

In [13]: # TASK 1

```
X_train, X_val= train_test_split(merged, test_size = 0.2, random_state = 1)
x_train = X_train.iloc[:,3:16]
x_val = X_val.iloc[:,3:16]
y_train = X_train.iloc[:,16:19]
y_val = X_val.iloc[:,16:19]
```

In [14]: # TASK 1

```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 956 entries, 1002 to 1065
Data columns (total 13 columns):
Total Population                956 non-null int64
Percent White, not Hispanic or Latino 956 non-null float64
Percent Black, not Hispanic or Latino 956 non-null float64
Percent Hispanic or Latino        956 non-null float64
Percent Foreign Born             956 non-null float64
Percent Female                  956 non-null float64
Percent Age 29 and Under        956 non-null float64
Percent Age 65 and Older        956 non-null float64
Median Household Income         956 non-null int64
Percent Unemployed              956 non-null float64
Percent Less than High School Degree 956 non-null float64
Percent Less than Bachelor's Degree 956 non-null float64
Percent Rural                  956 non-null float64
dtypes: float64(11), int64(2)
memory usage: 104.6 KB
```

In [15]: # TASK 1

```
x_val.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 239 entries, 49 to 778
Data columns (total 13 columns):
Total Population                239 non-null int64
Percent White, not Hispanic or Latino 239 non-null float64
Percent Black, not Hispanic or Latino 239 non-null float64
Percent Hispanic or Latino        239 non-null float64
Percent Foreign Born             239 non-null float64
Percent Female                   239 non-null float64
Percent Age 29 and Under        239 non-null float64
Percent Age 65 and Older         239 non-null float64
Median Household Income          239 non-null int64
Percent Unemployed               239 non-null float64
Percent Less than High School Degree 239 non-null float64
Percent Less than Bachelor's Degree 239 non-null float64
Percent Rural                    239 non-null float64
dtypes: float64(11), int64(2)
memory usage: 26.1 KB
```

In [16]: # TASK 2

```
scaler = MinMaxScaler(feature_range=(0, 1))
x_train[["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"]] = scaler.fit_transform(x_train[["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"]])
x_val[["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"]] = scaler.fit_transform(x_val[["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"]])
```

In [17]: # TASK 2  
x\_train.head()

Out[17]:

	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
1002	0.002942	0.850025	0.113687	0.023226	0.034434	0.762428	0.410657	0.444647	0.238719	0.292156	0.245331	0.616610
879	0.027857	0.845456	0.023428	0.116834	0.085801	0.718393	0.479555	0.259951	0.432558	0.306275	0.172667	0.593255
401	0.002025	0.887518	0.001235	0.100628	0.086903	0.685378	0.449038	0.471617	0.247669	0.139900	0.205674	0.712950
134	0.004725	0.923838	0.070571	0.017434	0.015298	0.536575	0.428021	0.311917	0.194606	0.310777	0.234054	0.836307
510	0.010969	0.929594	0.022788	0.026875	0.041112	0.764089	0.561861	0.275287	0.264675	0.284376	0.147896	0.631202

◀ ▶

In [ ]:

In [18]: # TASK 3a  
# Evaluate model with one predictor

```
#model = linear_model.LinearRegression().fit(X = X_train["Total Population"], y = Y_train["Democratic"])
#score_train = model.score(X = X_train["Total Population"], y = Y_train["Democratic"]) # R squared (training)
#score_val = model.score(X = X_train["Total Population"], y = Y_train["Democratic"]) # R squared (validation)
##score_test = model.score(X = X_test[['total_bill', 'size']], y = Y_test) # R squared (test)
#print([score_train, score_val])
```

```
In [19]: # TASK 3a
# Build model with 1 predictor (using statsmodels)
X = x_train["Total Population"]
Y = y_train["Democratic"]
X = sm.add_constant(X)
results = sm.OLS(Y,X).fit()
print(results.summary())
#SSE = round(results.ssr,3)
#SSE
```

OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.862			
Model:	OLS	Adj. R-squared:	0.861			
Method:	Least Squares	F-statistic:	5940.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:10	Log-Likelihood:	-11122.			
No. Observations:	956	AIC:	2.225e+04			
Df Residuals:	954	BIC:	2.226e+04			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	-1025.4901	950.169	-1.079	0.281	-2890.152	839.172
Total Population	9.586e+05	1.24e+04	77.073	0.000	9.34e+05	9.83e+05
Omnibus:	1537.753	Durbin-Watson:			1.990	
Prob(Omnibus):	0.000	Jarque-Bera (JB):			1446704.953	
Skew:	9.498	Prob(JB):			0.00	
Kurtosis:	192.626	Cond. No.			14.1	

#### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [20]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population"]])
y_pred = results.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[20]: 602303211.9622428

In [ ]:

In [21]:

```
# TASK 3a
# Build model with 2 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino"]]
Y = y_train["Democratic"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())
#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

```
=====
Dep. Variable: Democratic R-squared: 0.862
Model: OLS Adj. R-squared: 0.861
Method: Least Squares F-statistic: 2971.
Date: Mon, 18 Nov 2019 Prob (F-statistic): 0.00
Time: 17:48:10 Log-Likelihood: -11121.
No. Observations: 956 AIC: 2.225e+04
Df Residuals: 953 BIC: 2.226e+04
Df Model: 2
Covariance Type: nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	-4783.1995	3891.161	-1.229	0.219	-1.24e+04	2853.035
Total Population	9.629e+05	1.31e+04	73.228	0.000	9.37e+05	9.89e+05
Percent White, not Hispanic or Latino	4612.1499	4631.369	0.996	0.320	-4476.710	1.37e+04

```
=====
Omnibus: 1533.072 Durbin-Watson: 1.991
Prob(Omnibus): 0.000 Jarque-Bera (JB): 1434274.766
Skew: 9.438 Prob(JB): 0.00
Kurtosis: 191.814 Cond. No. 19.4
=====
```

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [22]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[22]: 608651158.1019803

In [ ]:

In [23]:

```
# TASK 3a
# Build model with 2 predictors (using statsmodels)
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino"]]
Y = y_train["Democratic"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())
#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.862			
Model:	OLS	Adj. R-squared:	0.862			
Method:	Least Squares	F-statistic:	2977.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:10	Log-Likelihood:	-11120.			
No. Observations:	956	AIC:	2.225e+04			
Df Residuals:	953	BIC:	2.226e+04			
Df Model:	2					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	-1788.7257	1050.875	-1.702	0.089	-3851.022	273.571
Total Population	9.524e+05	1.3e+04	73.512	0.000	9.27e+05	9.78e+05
Percent Black, not Hispanic or Latino	1.071e+04	6325.941	1.693	0.091	-1705.707	2.31e+04
Omnibus:	1552.951	Durbin-Watson:	1.985			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1500240.441			
Skew:	9.691	Prob(JB):	0.00			
Kurtosis:	196.099	Cond. No.	14.9			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [24]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[24]: 593895417.487927

In [ ]:

```
In [25]: # TASK 3a  
# Build model with 3 predictors (using statsmodels)  
X = x_train[["Total Population",  
             "Percent Black, not Hispanic or Latino","Percent Less than Bachelor's Degree"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results4 = sm.OLS(Y,X).fit()  
print(results4.summary())  
#SSE = round(results4.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.871			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	2146.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:10	Log-Likelihood:	-11087.			
No. Observations:	956	AIC:	2.218e+04			
Df Residuals:	952	BIC:	2.220e+04			
Df Model:	3					
Covariance Type:	nonrobust					
<hr/>						
	coef	std err	t	P> t	[0.025	0.975]
<hr/>						
const	3.322e+04	4378.327	7.588	0.000	2.46e+04	4.18e+04
Total Population	9.118e+05	1.35e+04	67.712	0.000	8.85e+05	9.38e+05
Percent Black, not Hispanic or Latino	1.126e+04	6116.282	1.840	0.066	-747.291	2.33e+04
Percent Less than Bachelor's Degree	-4.933e+04	6000.645	-8.220	0.000	-6.11e+04	-3.76e+04
<hr/>						
Omnibus:	1578.465	Durbin-Watson:	1.993			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1552611.775			
Skew:	10.034	Prob(JB):	0.00			
Kurtosis:	199.405	Cond. No.	19.9			
<hr/>						

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [26]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino","Percent Less than Bachelor's Degree"]])
y_pred = results4.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[26]: 509661642.5656613

In [ ]:

```
In [27]: # TASK 3a  
# Build model with 4 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Hispanic or Latino"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.872			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	1617.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:10	Log-Likelihood:	-11085.			
No. Observations:	956	AIC:	2.218e+04			
Df Residuals:	951	BIC:	2.220e+04			
Df Model:	4					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
const	3.295e+04	4371.859	7.536	0.000	2.44e+04	4.15e+04
Total Population	9.183e+05	1.38e+04	66.656	0.000	8.91e+05	9.45e+05
Percent Black, not Hispanic or Latino	1.072e+04	6109.783	1.754	0.080	-1274.902	2.27e+04
Percent Less than Bachelor's Degree	-4.727e+04	6065.003	-7.794	0.000	-5.92e+04	-3.54e+04
Percent Hispanic or Latino	-1.182e+04	5487.825	-2.154	0.032	-2.26e+04	-1048.814
Omnibus:	1576.786	Durbin-Watson:	1.988			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1553193.819			
Skew:	10.010	Prob(JB):	0.00			
Kurtosis:	199.447	Cond. No.	20.6			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [28]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Hispanic or Latino"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

```
Out[28]: 525000539.7274302
```

```
In [ ]:
```

```
In [29]: # TASK 3a  
# Build model with 4 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.871			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	1608.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:10	Log-Likelihood:	-11087.			
No. Observations:	956	AIC:	2.218e+04			
Df Residuals:	951	BIC:	2.221e+04			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	3.252e+04	4552.943	7.143	0.000	2.36e+04	4.15e+04
Total Population	9.081e+05	1.5e+04	60.636	0.000	8.79e+05	9.37e+05
Percent Black, not Hispanic or Latino	1.106e+04	6128.797	1.804	0.072	-971.347	2.31e+04
Percent Less than Bachelor's Degree	-4.884e+04	6064.651	-8.054	0.000	-6.07e+04	-3.69e+04
Percent Foreign Born	4948.1036	8816.051	0.561	0.575	-1.24e+04	2.22e+04
Omnibus:	1579.283	Durbin-Watson:	1.997			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1551700.635			
Skew:	10.046	Prob(JB):	0.00			
Kurtosis:	199.344	Cond. No.	22.6			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [30]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[30]: 508647738.0798381

In [ ]:

```
In [31]: # TASK 3a  
# Build model with 5 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Percent Female"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.871		
Model:	OLS	Adj. R-squared:	0.871		
Method:	Least Squares	F-statistic:	1286.		
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00		
Time:	17:48:10	Log-Likelihood:	-11087.		
No. Observations:	956	AIC:	2.219e+04		
Df Residuals:	950	BIC:	2.222e+04		
Df Model:	5				
Covariance Type:	nonrobust				
coef	std err	t	P> t	[0.025	0.975]
const	3.491e+04	8831.912	3.953	0.000	1.76e+04 5.22e+04
Total Population	9.088e+05	1.51e+04	60.058	0.000	8.79e+05 9.38e+05
Percent Black, not Hispanic or Latino	1.093e+04	6144.858	1.779	0.076	-1129.585 2.3e+04
Percent Less than Bachelor's Degree	-4.94e+04	6315.140	-7.822	0.000	-6.18e+04 -3.7e+04
Percent Foreign Born	4516.6429	8925.771	0.506	0.613	-1.3e+04 2.2e+04
Percent Female	-2800.2334	8882.108	-0.315	0.753	-2.02e+04 1.46e+04
Omnibus:	1578.040	Durbin-Watson:	2.000		
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1547819.860		
Skew:	10.029	Prob(JB):	0.00		
Kurtosis:	199.100	Cond. No.	26.5		

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [32]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Percent Female"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

```
Out[32]: 511206897.4273276
```

```
In [ ]:
```

```
In [33]: # TASK 3a  
# Build model with 5 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Percent Age 29 and Under"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.873			
Model:	OLS	Adj. R-squared:	0.872			
Method:	Least Squares	F-statistic:	1305.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-11081.			
No. Observations:	956	AIC:	2.217e+04			
Df Residuals:	950	BIC:	2.220e+04			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	4.773e+04	6209.097	7.688	0.000	3.55e+04	5.99e+04
Total Population	9.063e+05	1.49e+04	60.854	0.000	8.77e+05	9.36e+05
Percent Black, not Hispanic or Latino	1.212e+04	6098.408	1.988	0.047	154.428	2.41e+04
Percent Less than Bachelor's Degree	-5.154e+04	6074.192	-8.484	0.000	-6.35e+04	-3.96e+04
Percent Foreign Born	1.445e+04	9155.592	1.578	0.115	-3516.559	3.24e+04
Percent Age 29 and Under	-3.196e+04	8931.909	-3.578	0.000	-4.95e+04	-1.44e+04
Omnibus:	1582.562	Durbin-Watson:	2.013			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1562636.922			
Skew:	10.089	Prob(JB):	0.00			
Kurtosis:	200.034	Cond. No.	24.3			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [34]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Percent Age 29 and Under"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[34]: 528999176.2284078

In [ ]:

```
In [35]: # TASK 3a  
# Build model with 5 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Percent Age 65 and Older"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

OLS Regression Results						
Dep. Variable:	Democratic	R-squared:	0.872			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	1293.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-11085.			
No. Observations:	956	AIC:	2.218e+04			
Df Residuals:	950	BIC:	2.221e+04			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	2.804e+04	4984.223	5.626	0.000	1.83e+04	3.78e+04
Total Population	9.1e+05	1.5e+04	60.780	0.000	8.81e+05	9.39e+05
Percent Black, not Hispanic or Latino	1.253e+04	6153.407	2.036	0.042	451.074	2.46e+04
Percent Less than Bachelor's Degree	-5.029e+04	6088.486	-8.260	0.000	-6.22e+04	-3.83e+04
Percent Foreign Born	9781.7121	9071.389	1.078	0.281	-8020.565	2.76e+04
Percent Age 65 and Older	1.291e+04	5898.234	2.189	0.029	1335.994	2.45e+04
Omnibus:	1580.219	Durbin-Watson:	1.997			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1559712.307			
Skew:	10.057	Prob(JB):	0.00			
Kurtosis:	199.854	Cond. No.	23.6			

### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [36]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Percent Age 65 and Older"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[36]: 514138529.2944029

In [ ]:

```
In [37]: # TASK 3a  
# Build model with 5 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.871			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	1285.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-11087.			
No. Observations:	956	AIC:	2.219e+04			
Df Residuals:	950	BIC:	2.222e+04			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	3.185e+04	7414.761	4.295	0.000	1.73e+04	4.64e+04
Total Population	9.081e+05	1.5e+04	60.602	0.000	8.79e+05	9.38e+05
Percent Black, not Hispanic or Latino	1.113e+04	6167.772	1.805	0.071	-971.183	2.32e+04
Percent Less than Bachelor's Degree	-4.829e+04	7734.308	-6.243	0.000	-6.35e+04	-3.31e+04
Percent Foreign Born	4861.2090	8852.628	0.549	0.583	-1.25e+04	2.22e+04
Median Household Income	1153.6110	9983.856	0.116	0.908	-1.84e+04	2.07e+04
Omnibus:	1578.806	Durbin-Watson:	1.998			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1549099.610			
Skew:	10.040	Prob(JB):	0.00			
Kurtosis:	199.179	Cond. No.	23.3			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [38]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[38]: 508441650.7225877

In [ ]:

```
In [39]: # TASK 3a  
# Build model with 6 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Perce  
nt Unemployed"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.871			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	1072.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-11087.			
No. Observations:	956	AIC:	2.219e+04			
Df Residuals:	949	BIC:	2.222e+04			
Df Model:	6					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	2.853e+04	7894.089	3.614	0.000	1.3e+04	4.4e+04
Total Population	9.061e+05	1.51e+04	60.122	0.000	8.77e+05	9.36e+05
Percent Black, not Hispanic or Latino	9192.6150	6367.320	1.444	0.149	-3303.040	2.17e+04
Percent Less than Bachelor's Degree	-4.869e+04	7739.368	-6.292	0.000	-6.39e+04	-3.35e+04
Percent Foreign Born	4145.9649	8869.665	0.467	0.640	-1.33e+04	2.16e+04
Median Household Income	5209.6278	1.05e+04	0.495	0.621	-1.54e+04	2.59e+04
Percent Unemployed	8036.3892	6576.586	1.222	0.222	-4869.943	2.09e+04
Omnibus:	1579.851	Durbin-Watson:	1.995			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1545711.817			
Skew:	10.056	Prob(JB):	0.00			
Kurtosis:	198.959	Cond. No.	24.2			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [40]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Percent Unemployed"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
```

Out[40]: 503655991.1485221

In [ ]:

```
In [41]: # TASK 3a  
# Build model with 7 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Perce-  
             nt Unemployed",  
             "Percent Less than High School Degree"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.871			
Model:	OLS	Adj. R-squared:	0.870			
Method:	Least Squares	F-statistic:	917.9			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-11087.			
No. Observations:	956	AIC:	2.219e+04			
Df Residuals:	948	BIC:	2.223e+04			
Df Model:	7					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	2.875e+04	8052.946	3.570	0.000	1.29e+04	4.46e+04
Total Population	9.066e+05	1.55e+04	58.540	0.000	8.76e+05	9.37e+05
Percent Black, not Hispanic or Latino	8975.6993	6560.331	1.368	0.172	-3898.750	2.19e+04
Percent Less than Bachelor's Degree	-4.943e+04	9387.904	-5.265	0.000	-6.79e+04	-3.1e+04
Percent Foreign Born	3082.3601	1.17e+04	0.263	0.793	-1.99e+04	2.61e+04
Median Household Income	5539.7901	1.08e+04	0.513	0.608	-1.56e+04	2.67e+04
Percent Unemployed	7957.5739	6604.552	1.205	0.229	-5003.658	2.09e+04
Percent Less than High School Degree	1483.4648	1.07e+04	0.138	0.890	-1.95e+04	2.25e+04
Omnibus:	1578.847	Durbin-Watson:	1.995			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1542095.769			
Skew:	10.043	Prob(JB):	0.00			
Kurtosis:	198.730	Cond. No.	28.1			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [42]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
                           "Percent Less than High School Degree"]])

y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
print(y_pred.head())
MSE
```

```
49      72140.189464
314     2347.234241
331     -400.616410
1022    1977.622496
808     -4097.635004
dtype: float64
```

Out[42]: 502922217.6645136

In [ ]:

In [43]: best\_regression\_Demo = results5

In [44]: # TASK 3a
# The above model is the best performing model

In [ ]:

```
In [45]: # TASK 3a  
# Build model with 8 predictors (using statsmodels)  
X = x_train[["Total Population", "Percent Black, not Hispanic or Latino",  
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Perce-  
nt Unemployed",  
             "Percent Less than High School Degree",  
             "Percent Rural"]]  
Y = y_train["Democratic"]  
X = sm.add_constant(X)  
results5 = sm.OLS(Y,X).fit()  
print(results5.summary())  
  
#SSE = round(results5.ssr,3)  
#SSE
```

## OLS Regression Results

Dep. Variable:	Democratic	R-squared:	0.872			
Model:	OLS	Adj. R-squared:	0.871			
Method:	Least Squares	F-statistic:	808.0			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-11084.			
No. Observations:	956	AIC:	2.219e+04			
Df Residuals:	947	BIC:	2.223e+04			
Df Model:	8					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	2.551e+04	8144.140	3.132	0.002	9525.467	4.15e+04
Total Population	9.116e+05	1.56e+04	58.488	0.000	8.81e+05	9.42e+05
Percent Black, not Hispanic or Latino	1.054e+04	6575.799	1.603	0.109	-2364.918	2.34e+04
Percent Less than Bachelor's Degree	-5.476e+04	9621.752	-5.691	0.000	-7.36e+04	-3.59e+04
Percent Foreign Born	1.067e+04	1.21e+04	0.881	0.379	-1.31e+04	3.45e+04
Median Household Income	7048.4622	1.08e+04	0.654	0.513	-1.41e+04	2.82e+04
Percent Unemployed	1.097e+04	6705.356	1.636	0.102	-2187.376	2.41e+04
Percent Less than High School Degree	267.2934	1.07e+04	0.025	0.980	-2.07e+04	2.13e+04
Percent Rural	8464.8785	3510.109	2.412	0.016	1576.388	1.54e+04
Omnibus:	1568.926	Durbin-Watson:	2.005			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1519582.891			
Skew:	9.909	Prob(JB):	0.00			
Kurtosis:	197.308	Cond. No.	30.6			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [46]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent Black, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
                           "Percent Less than High School Degree",
                           "Percent Rural"]])
y_pred = results5.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Democratic"], y_pred)
MSE
# Mean squared error
```

Out[46]: 511406826.89655924

In [ ]:

In [47]: X\_train.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 956 entries, 1002 to 1065
Data columns (total 19 columns):
State                      956 non-null object
County                     956 non-null object
FIPS                       956 non-null int64
Total Population           956 non-null int64
Percent White, not Hispanic or Latino 956 non-null float64
Percent Black, not Hispanic or Latino 956 non-null float64
Percent Hispanic or Latino 956 non-null float64
Percent Foreign Born       956 non-null float64
Percent Female              956 non-null float64
Percent Age 29 and Under   956 non-null float64
Percent Age 65 and Older   956 non-null float64
Median Household Income    956 non-null int64
Percent Unemployed         956 non-null float64
Percent Less than High School Degree 956 non-null float64
Percent Less than Bachelor's Degree 956 non-null float64
Percent Rural               956 non-null float64
Democratic                 956 non-null float64
Republican                 956 non-null float64
Party                      956 non-null object
dtypes: float64(13), int64(3), object(3)
memory usage: 149.4+ KB
```

In [48]: # TASK 3a

```
# The model with 7 predictor variables :
# "Total Population", "Percent Black, not Hispanic or Latino",
#          "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
#          "Percent Less than High School Degree" is the best performing model.
# The Adjusted R2 value of this model is 0.870 and root mean square error for validation set is 502922217.
```

In [ ]:

```
In [49]: # TASK 3b
# Build model with 1 predictor (using statsmodels)
X = x_train["Total Population"]
Y = y_train["Republican"]
X = sm.add_constant(X)
results = sm.OLS(Y,X).fit()
print(results.summary())

#SSE = round(results5.ssr,3)
#SSE
```

OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.830			
Model:	OLS	Adj. R-squared:	0.830			
Method:	Least Squares	F-statistic:	4662.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-10719.			
No. Observations:	956	AIC:	2.144e+04			
Df Residuals:	954	BIC:	2.145e+04			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	5227.8666	623.827	8.380	0.000	4003.635	6452.099
Total Population	5.576e+05	8165.879	68.281	0.000	5.42e+05	5.74e+05
Omnibus:	375.794	Durbin-Watson:	1.917			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	189166.599			
Skew:	-0.185	Prob(JB):	0.00			
Kurtosis:	71.912	Cond. No.	14.1			

#### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [50]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population"]])
y_pred = results.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[50]: 521027312.25435406

In [ ]:

```
In [51]: # Build model with 2 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.834				
Model:	OLS	Adj. R-squared:	0.833				
Method:	Least Squares	F-statistic:	2388.				
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00				
Time:	17:48:11	Log-Likelihood:	-10709.				
No. Observations:	956	AIC:	2.142e+04				
Df Residuals:	953	BIC:	2.144e+04				
Df Model:	2						
Covariance Type:	nonrobust						
		coef	std err	t	P> t	[0.025	0.975]
const		-5800.8358	2529.358	-2.293	0.022	-1.08e+04	-837.082
Total Population		5.7e+05	8547.058	66.695	0.000	5.53e+05	5.87e+05
Percent White, not Hispanic or Latino	1.354e+04	3010.512		4.496	0.000	7628.446	1.94e+04

Omnibus: 386.634 Durbin-Watson: 1.931  
 Prob(Omnibus): 0.000 Jarque-Bera (JB): 198931.513  
 Skew: -0.301 Prob(JB): 0.00  
 Kurtosis: 73.666 Cond. No. 19.4

#### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [52]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[52]: 508124329.243582

In [ ]:

```
In [53]: # Build model with 3 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree e"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.843			
Model:	OLS	Adj. R-squared:	0.842			
Method:	Least Squares	F-statistic:	1701.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-10683.			
No. Observations:	956	AIC:	2.137e+04			
Df Residuals:	952	BIC:	2.139e+04			
Df Model:	3					
Covariance Type:	nonrobust					
<hr/>						
	coef	std err	t	P> t	[0.025	0.975]
<hr/>						
const	1.654e+04	3889.181	4.254	0.000	8910.383	2.42e+04
Total Population	5.444e+05	9005.783	60.447	0.000	5.27e+05	5.62e+05
Percent White, not Hispanic or Latino	1.163e+04	2939.940	3.954	0.000	5855.828	1.74e+04
Percent Less than Bachelor's Degree	-2.926e+04	3943.581	-7.419	0.000	-3.7e+04	-2.15e+04
<hr/>						
Omnibus:	406.471	Durbin-Watson:	1.943			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	189205.286			
Skew:	-0.499	Prob(JB):	0.00			
Kurtosis:	71.912	Cond. No.	24.8			
<hr/>						

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [54]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino",
                           "Percent Less than Bachelor's Degree"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[54]: 510966729.2939641

In [ ]:

```
In [55]: # Build model with 4 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

### OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.844	
Model:	OLS	Adj. R-squared:	0.843	
Method:	Least Squares	F-statistic:	1283.	
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00	
Time:	17:48:11	Log-Likelihood:	-10680.	
No. Observations:	956	AIC:	2.137e+04	
Df Residuals:	951	BIC:	2.139e+04	
Df Model:	4			
Covariance Type:	nonrobust			
<hr/>				
	coef	std err	t	
			P> t	
			[0.025 0.975]	
<hr/>				
const	2.011e+04	4156.437	4.839	0.000 1.2e+04 2.83e+04
Total Population	5.473e+05	9065.399	60.371	0.000 5.29e+05 5.65e+05
Percent White, not Hispanic or Latino	8396.4717	3227.998	2.601	0.009 2061.650 1.47e+04
Percent Less than Bachelor's Degree	-2.952e+04	3935.404	-7.502	0.000 -3.72e+04 -2.18e+04
Percent Black, not Hispanic or Latino	-1.052e+04	4396.702	-2.394	0.017 -1.92e+04 -1896.273
<hr/>				
Omnibus:	401.353	Durbin-Watson:	1.946	
Prob(Omnibus):	0.000	Jarque-Bera (JB):	185278.744	
Skew:	-0.466	Prob(JB):	0.00	
Kurtosis:	71.194	Cond. No.	24.9	
<hr/>				

### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [56]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[56]: 510760773.46957666

In [ ]:

```
In [57]: # Build model with 5 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.844			
Model:	OLS	Adj. R-squared:	0.843			
Method:	Least Squares	F-statistic:	1027.			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-10679.			
No. Observations:	956	AIC:	2.137e+04			
Df Residuals:	950	BIC:	2.140e+04			
Df Model:	5					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	1.404e+04	6761.205	2.076	0.038	770.093	2.73e+04
Total Population	5.472e+05	9064.580	60.363	0.000	5.29e+05	5.65e+05
Percent White, not Hispanic or Latino	1.511e+04	6719.423	2.249	0.025	1922.740	2.83e+04
Percent Less than Bachelor's Degree	-3.026e+04	3987.297	-7.588	0.000	-3.81e+04	-2.24e+04
Percent Black, not Hispanic or Latino	-6314.6884	5743.351	-1.099	0.272	-1.76e+04	4956.432
Percent Hispanic or Latino	8513.9023	7474.735	1.139	0.255	-6154.997	2.32e+04
Omnibus:	398.231	Durbin-Watson:	1.943			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	185347.346			
Skew:	-0.440	Prob(JB):	0.00			
Kurtosis:	71.208	Cond. No.	32.3			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [58]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[58]: 507476811.4934242

In [ ]:

```
In [59]: # Build model with 6 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Foreign Born"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.845			
Model:	OLS	Adj. R-squared:	0.844			
Method:	Least Squares	F-statistic:	860.8			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-10676.			
No. Observations:	956	AIC:	2.137e+04			
Df Residuals:	949	BIC:	2.140e+04			
Df Model:	6					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	1.893e+04	7070.673	2.677	0.008	5053.374	3.28e+04
Total Population	5.565e+05	9909.609	56.159	0.000	5.37e+05	5.76e+05
Percent White, not Hispanic or Latino	1.303e+04	6764.670	1.925	0.054	-250.127	2.63e+04
Percent Less than Bachelor's Degree	-3.364e+04	4239.583	-7.935	0.000	-4.2e+04	-2.53e+04
Percent Black, not Hispanic or Latino	-6386.1092	5730.391	-1.114	0.265	-1.76e+04	4859.594
Percent Hispanic or Latino	1.551e+04	8050.713	1.927	0.054	-284.740	3.13e+04
Percent Foreign Born	-1.971e+04	8538.460	-2.309	0.021	-3.65e+04	-2955.118
Omnibus:	375.088	Durbin-Watson:	1.926			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	174095.387			
Skew:	-0.231	Prob(JB):	0.00			
Kurtosis:	69.109	Cond. No.	32.6			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [60]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Foreign Born"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[60]: 486097511.811946

In [ ]:

```
In [61]: # Build model with 7 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Foreign Born",
             "Percent Female"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.845			
Model:	OLS	Adj. R-squared:	0.844			
Method:	Least Squares	F-statistic:	737.2			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:11	Log-Likelihood:	-10676.			
No. Observations:	956	AIC:	2.137e+04			
Df Residuals:	948	BIC:	2.141e+04			
Df Model:	7					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	2.047e+04	8548.921	2.395	0.017	3696.446	3.73e+04
Total Population	5.569e+05	1e+04	55.687	0.000	5.37e+05	5.77e+05
Percent White, not Hispanic or Latino	1.306e+04	6768.590	1.929	0.054	-226.032	2.63e+04
Percent Less than Bachelor's Degree	-3.399e+04	4377.957	-7.764	0.000	-4.26e+04	-2.54e+04
Percent Black, not Hispanic or Latino	-6457.3983	5737.382	-1.125	0.261	-1.77e+04	4802.039
Percent Hispanic or Latino	1.548e+04	8055.390	1.921	0.055	-332.050	3.13e+04
Percent Foreign Born	-1.991e+04	8565.792	-2.325	0.020	-3.67e+04	-3104.534
Percent Female	-1862.3732	5789.661	-0.322	0.748	-1.32e+04	9499.660
Omnibus:	374.878	Durbin-Watson:	1.928			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	173550.031			
Skew:	-0.230	Prob(JB):	0.00			
Kurtosis:	69.005	Cond. No.	37.0			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [62]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Foreign Born",
                           "Percent Female"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[62]: 487097856.33708

In [ ]:

```
In [63]: # Build model with 7 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Foreign Born", "Percent Age 29 and Under"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

### OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.845			
Model:	OLS	Adj. R-squared:	0.844			
Method:	Least Squares	F-statistic:	737.8			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:12	Log-Likelihood:	-10676.			
No. Observations:	956	AIC:	2.137e+04			
Df Residuals:	948	BIC:	2.141e+04			
Df Model:	7					
Covariance Type:	nonrobust					
<hr/>						
	coef	std err	t	P> t	[0.025	0.975]
<hr/>						
const	2.315e+04	8599.603	2.692	0.007	6273.410	4e+04
Total Population	5.566e+05	9911.267	56.157	0.000	5.37e+05	5.76e+05
Percent White, not Hispanic or Latino	1.158e+04	6970.404	1.661	0.097	-2100.580	2.53e+04
Percent Less than Bachelor's Degree	-3.448e+04	4349.489	-7.927	0.000	-4.3e+04	-2.59e+04
Percent Black, not Hispanic or Latino	-6942.8602	5767.403	-1.204	0.229	-1.83e+04	4375.492
Percent Hispanic or Latino	1.566e+04	8053.559	1.944	0.052	-145.166	3.15e+04
Percent Foreign Born	-2.004e+04	8548.002	-2.344	0.019	-3.68e+04	-3262.930
Percent Age 29 and Under	-5442.8311	6310.553	-0.862	0.389	-1.78e+04	6941.436
<hr/>						
Omnibus:	376.497	Durbin-Watson:	1.931			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	173344.987			
Skew:	-0.253	Prob(JB):	0.00			
Kurtosis:	68.966	Cond. No.	35.9			
<hr/>						

### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [64]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                           "Percent Foreign Born", "Percent Age 29 and Under"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[64]: 488121515.17466944

In [ ]:

```
In [65]: # Build model with 7 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Foreign Born", "Percent Age 65 and Older"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

### OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.845	
Model:	OLS	Adj. R-squared:	0.844	
Method:	Least Squares	F-statistic:	737.4	
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00	
Time:	17:48:12	Log-Likelihood:	-10676.	
No. Observations:	956	AIC:	2.137e+04	
Df Residuals:	948	BIC:	2.141e+04	
Df Model:	7			
Covariance Type:	nonrobust			
<hr/>				
	coef	std err	t	
			P> t	
			[0.025 0.975]	
<hr/>				
const	1.898e+04	7073.861	2.683	0.007 5099.673 3.29e+04
Total Population	5.561e+05	9941.626	55.936	0.000 5.37e+05 5.76e+05
Percent White, not Hispanic or Latino	1.37e+04	6875.160	1.993	0.047 211.627 2.72e+04
Percent Less than Bachelor's Degree	-3.329e+04	4286.207	-7.768	0.000 -4.17e+04 -2.49e+04
Percent Black, not Hispanic or Latino	-6266.2968	5736.474	-1.092	0.275 -1.75e+04 4991.359
Percent Hispanic or Latino	1.571e+04	8061.488	1.949	0.052 -107.009 3.15e+04
Percent Foreign Born	-1.994e+04	8551.308	-2.332	0.020 -3.67e+04 -3158.113
Percent Age 65 and Older	-2206.9942	3948.349	-0.559	0.576 -9955.509 5541.521
<hr/>				
Omnibus:	373.422	Durbin-Watson:	1.925	
Prob(Omnibus):	0.000	Jarque-Bera (JB):	173873.314	
Skew:	-0.207	Prob(JB):	0.00	
Kurtosis:	69.067	Cond. No.	33.7	
<hr/>				

### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [66]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                           "Percent Foreign Born", "Percent Age 65 and Older"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[66]: 486505527.32842785

In [ ]:

```
In [67]: # Build model with 7 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Foreign Born", "Median Household Income"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

## OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.848			
Model:	OLS	Adj. R-squared:	0.847			
Method:	Least Squares	F-statistic:	754.2			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:12	Log-Likelihood:	-10667.			
No. Observations:	956	AIC:	2.135e+04			
Df Residuals:	948	BIC:	2.139e+04			
Df Model:	7					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
const	4656.5878	7749.749	0.601	0.548	-1.06e+04	1.99e+04
Total Population	5.573e+05	9820.784	56.746	0.000	5.38e+05	5.77e+05
Percent White, not Hispanic or Latino	1.112e+04	6717.438	1.656	0.098	-2059.243	2.43e+04
Percent Less than Bachelor's Degree	-2.062e+04	5175.218	-3.984	0.000	-3.08e+04	-1.05e+04
Percent Black, not Hispanic or Latino	-5548.4466	5681.411	-0.977	0.329	-1.67e+04	5601.150
Percent Hispanic or Latino	1.524e+04	7977.481	1.910	0.056	-415.868	3.09e+04
Percent Foreign Born	-2.382e+04	8514.053	-2.798	0.005	-4.05e+04	-7110.336
Median Household Income	2.786e+04	6466.365	4.309	0.000	1.52e+04	4.06e+04
Omnibus:	372.564	Durbin-Watson:	1.935			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	168459.277			
Skew:	-0.216	Prob(JB):	0.00			
Kurtosis:	68.030	Cond. No.	33.5			

## Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [68]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                           "Percent Foreign Born", "Median Household Income"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[68]: 483202952.3469636

In [ ]:

```
In [69]: # Build model with 8 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Foreign Born", "Median Household Income", "Percent Unemployed"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

### OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.851			
Model:	OLS	Adj. R-squared:	0.850			
Method:	Least Squares	F-statistic:	678.5			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:12	Log-Likelihood:	-10655.			
No. Observations:	956	AIC:	2.133e+04			
Df Residuals:	947	BIC:	2.137e+04			
Df Model:	8					
Covariance Type:	nonrobust					
<hr/>						
	coef	std err	t	P> t	[0.025	0.975]
<hr/>						
const	-1.043e+04	8267.148	-1.261	0.208	-2.67e+04	5797.176
Total Population	5.528e+05	9749.570	56.704	0.000	5.34e+05	5.72e+05
Percent White, not Hispanic or Latino	1.847e+04	6809.930	2.713	0.007	5107.688	3.18e+04
Percent Less than Bachelor's Degree	-2.235e+04	5127.406	-4.360	0.000	-3.24e+04	-1.23e+04
Percent Black, not Hispanic or Latino	-6083.5427	5616.240	-1.083	0.279	-1.71e+04	4938.172
Percent Hispanic or Latino	2.286e+04	8039.535	2.843	0.005	7081.055	3.86e+04
Percent Foreign Born	-2.561e+04	8422.835	-3.040	0.002	-4.21e+04	-9076.646
Median Household Income	3.801e+04	6725.115	5.652	0.000	2.48e+04	5.12e+04
Percent Unemployed	2.087e+04	4304.821	4.848	0.000	1.24e+04	2.93e+04
<hr/>						
Omnibus:	370.104	Durbin-Watson:	1.924			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	169715.753			
Skew:	-0.169	Prob(JB):	0.00			
Kurtosis:	68.273	Cond. No.	35.9			
<hr/>						

#### Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [70]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                           "Percent Foreign Born", "Median Household Income", "Percent Unemployed"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[70]: 478997825.18020195

In [ ]:

```
In [71]: # Build model with 9 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Foreign Born", "Median Household Income", "Percent Unemployed", "Percent Less than High
School Degree"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

### OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.852			
Model:	OLS	Adj. R-squared:	0.850			
Method:	Least Squares	F-statistic:	603.3			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:12	Log-Likelihood:	-10655.			
No. Observations:	956	AIC:	2.133e+04			
Df Residuals:	946	BIC:	2.138e+04			
Df Model:	9					
Covariance Type:	nonrobust					
<hr/>						
	coef	std err	t	P> t	[0.025	0.975]
<hr/>						
const	-9325.3536	8335.446	-1.119	0.264	-2.57e+04	7032.750
Total Population	5.546e+05	9902.828	56.007	0.000	5.35e+05	5.74e+05
Percent White, not Hispanic or Latino	1.818e+04	6815.425	2.668	0.008	4808.448	3.16e+04
Percent Less than Bachelor's Degree	-2.556e+04	5991.729	-4.265	0.000	-3.73e+04	-1.38e+04
Percent Black, not Hispanic or Latino	-7580.7326	5800.309	-1.307	0.192	-1.9e+04	3802.228
Percent Hispanic or Latino	2.028e+04	8419.400	2.408	0.016	3753.510	3.68e+04
Percent Foreign Born	-2.841e+04	8849.114	-3.210	0.001	-4.58e+04	-1.1e+04
Median Household Income	3.95e+04	6877.842	5.743	0.000	2.6e+04	5.3e+04
Percent Unemployed	2.035e+04	4334.587	4.694	0.000	1.18e+04	2.89e+04
Percent Less than High School Degree	7686.5115	7446.220	1.032	0.302	-6926.508	2.23e+04
<hr/>						
Omnibus:	372.331	Durbin-Watson:		1.922		
Prob(Omnibus):	0.000	Jarque-Bera (JB):		168866.541		
Skew:	-0.211	Prob(JB):		0.00		
Kurtosis:	68.109	Cond. No.		37.5		
<hr/>						

### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [72]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                           "Percent Foreign Born", "Median Household Income", "Percent Unemployed", "Percent Less than High
                           School Degree"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[72]: 478525026.73129755

In [ ]:

```
In [73]: # Build model with 10 predictors (using statsmodels)
X = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
             "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
             "Percent Less than High School Degree", "Percent Rural"]]
Y = y_train["Republican"]
X = sm.add_constant(X)
results3 = sm.OLS(Y,X).fit()
print(results3.summary())

#SSE = round(results3.ssr,3)
#SSE
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
 Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

### OLS Regression Results

Dep. Variable:	Republican	R-squared:	0.857			
Model:	OLS	Adj. R-squared:	0.855			
Method:	Least Squares	F-statistic:	565.2			
Date:	Mon, 18 Nov 2019	Prob (F-statistic):	0.00			
Time:	17:48:12	Log-Likelihood:	-10638.			
No. Observations:	956	AIC:	2.130e+04			
Df Residuals:	945	BIC:	2.135e+04			
Df Model:	10					
Covariance Type:	nonrobust					
<hr/>						
	coef	std err	t	P> t	[0.025	0.975]
<hr/>						
const	-3573.5291	8253.238	-0.433	0.665	-1.98e+04	1.26e+04
Total Population	5.459e+05	9848.876	55.430	0.000	5.27e+05	5.65e+05
Percent White, not Hispanic or Latino	1.706e+04	6702.549	2.545	0.011	3902.976	3.02e+04
Percent Less than Bachelor's Degree	-1.728e+04	6058.999	-2.851	0.004	-2.92e+04	-5386.088
Percent Black, not Hispanic or Latino	-1.175e+04	5746.648	-2.045	0.041	-2.3e+04	-474.653
Percent Hispanic or Latino	1.265e+04	8379.460	1.509	0.132	-3797.530	2.91e+04
Percent Foreign Born	-3.501e+04	8772.515	-3.991	0.000	-5.22e+04	-1.78e+04
Median Household Income	3.731e+04	6771.567	5.510	0.000	2.4e+04	5.06e+04
Percent Unemployed	1.512e+04	4354.554	3.471	0.001	6569.423	2.37e+04
Percent Less than High School Degree	1.308e+04	7378.056	1.772	0.077	-1403.682	2.76e+04
Percent Rural	-1.311e+04	2250.760	-5.826	0.000	-1.75e+04	-8696.606
<hr/>						
Omnibus:	367.549	Durbin-Watson:	1.915			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	177546.464			
Skew:	0.002	Prob(JB):	0.00			
Kurtosis:	69.763	Cond. No.	40.4			
<hr/>						

### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
In [74]: # Checking the model with y_val
X = sm.add_constant(x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                           "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
                           "Percent Less than High School Degree", "Percent Rural"]])
y_pred = results3.predict(X)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
MSE = mean_squared_error(y_val["Republican"], y_pred)
MSE
# Mean squared error
```

Out[74]: 470853100.34529376

```
In [75]: best_regression_Rep = results3
```

```
In [76]: # TASK 3b
```

```
# The model with 10 predictor variables :
#           "Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
#           "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
#           "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
#           "Percent Less than High School Degree", "Percent Rural"
# is the best performing model.
# The Adjusted R2 value of this model is 0.855 and root mean squared error is 470853100
```

```
In [ ]:
```

In [77]: `X_train.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 956 entries, 1002 to 1065
Data columns (total 19 columns):
State                      956 non-null object
County                     956 non-null object
FIPS                       956 non-null int64
Total Population           956 non-null int64
Percent White, not Hispanic or Latino 956 non-null float64
Percent Black, not Hispanic or Latino 956 non-null float64
Percent Hispanic or Latino    956 non-null float64
Percent Foreign Born        956 non-null float64
Percent Female              956 non-null float64
Percent Age 29 and Under   956 non-null float64
Percent Age 65 and Older   956 non-null float64
Median Household Income    956 non-null int64
Percent Unemployed          956 non-null float64
Percent Less than High School Degree 956 non-null float64
Percent Less than Bachelor's Degree 956 non-null float64
Percent Rural               956 non-null float64
Democratic                 956 non-null float64
Republican                 956 non-null float64
Party                      956 non-null object
dtypes: float64(13), int64(3), object(3)
memory usage: 149.4+ KB
```

In [78]: `# Task 4`

```
# Build decision tree with one parameter
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)
x = x_train[["Total Population"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

Out[78]: `DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False,
random_state=0, splitter='best')`

```
In [79]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[79]: array([( 1, 442, 0, 3.39954551e-02, 0.83826453, 956, 956.),
( 2, 35, 0, 1.47163589e-03, 0.69437466, 793, 793.),
( 3, 4, 0, 4.26798084e-04, 0.35149019, 136, 136.),
(-1, -1, -2, -2.00000000e+00, 0. , 35, 35.),
( 5, 6, 0, 4.27812935e-04, 0.43348846, 101, 101.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 7, 8, 0, 6.57956902e-04, 0.40217919, 100, 100.),
(-1, -1, -2, -2.00000000e+00, 0. , 20, 20.),
( 9, 10, 0, 6.86936342e-04, 0.46899559, 80, 80.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(11, 34, 0, 1.38954632e-03, 0.43180499, 79, 79.),
(12, 33, 0, 1.37421093e-03, 0.46012789, 72, 72.),
(13, 32, 0, 1.23686879e-03, 0.41786426, 71, 71.),
(14, 31, 0, 1.20180030e-03, 0.49716776, 55, 55.),
(15, 30, 0, 9.89923486e-04, 0.3860189 , 53, 53.),
(16, 29, 0, 9.78534721e-04, 0.49418293, 37, 37.),
(17, 28, 0, 8.95430276e-04, 0.41381685, 36, 36.),
(18, 27, 0, 8.86409456e-04, 0.51594693, 26, 26.),
(19, 24, 0, 8.57430045e-04, 0.40217919, 25, 25.),
(20, 23, 0, 7.52224587e-04, 0.27619543, 21, 21.),
(21, 22, 0, 7.43429293e-04, 0.72192809, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 16, 16.),
(25, 26, 0, 8.65210517e-04, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(36, 327, 0, 1.22619034e-02, 0.74446874, 657, 657.),
(37, 54, 0, 1.58259214e-03, 0.68724193, 491, 491.),
(38, 41, 0, 1.47716113e-03, 0.99572745, 13, 13.),
(39, 40, 0, 1.47344009e-03, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(42, 43, 0, 1.52316742e-03, 0.91829583, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.)),
```

```
( 44, 45, 0, 1.53602206e-03, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 46, 47, 0, 1.54391531e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 48, 49, 0, 1.55744655e-03, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 50, 51, 0, 1.57142885e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 52, 53, 0, 1.57706690e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 55, 326, 0, 1.17108435e-02, 0.66596458, 478, 478.),  
( 56, 325, 0, 1.16156735e-02, 0.67761925, 464, 464.),  
( 57, 66, 0, 2.04569905e-03, 0.66974162, 462, 462.),  
( 58, 65, 0, 1.90519966e-03, 0.34246377, 47, 47.),  
( 59, 64, 0, 1.87576917e-03, 0.49123734, 28, 28.),  
( 60, 63, 0, 1.63682993e-03, 0.23519338, 26, 26.),  
( 61, 62, 0, 1.63299608e-03, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
( 67, 156, 0, 3.73439875e-03, 0.69716708, 415, 415.),  
( 68, 69, 0, 2.06035783e-03, 0.83989269, 119, 119.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 70, 79, 0, 2.38195062e-03, 0.82128094, 117, 117.),  
( 71, 78, 0, 2.12948013e-03, 0.41381685, 24, 24.),  
( 72, 73, 0, 2.10546202e-03, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 74, 75, 0, 2.10940861e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 76, 77, 0, 2.11561052e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
( 80, 81, 0, 2.40145810e-03, 0.88260133, 93, 93.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 82, 119, 0, 3.17116058e-03, 0.86312057, 91, 91.),  
( 83, 110, 0, 2.72947806e-03, 0.75221217, 51, 51.),  
( 84, 109, 0, 2.70016037e-03, 0.954434 , 24, 24.),  
( 85, 86, 0, 2.45152379e-03, 0.90239328, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
```

( 87, 108, 0, 2.63306801e-03, 0.96407876, 18, 18.),  
( 88, 89, 0, 2.46291247e-03, 1. , 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 90, 107, 0, 2.62573862e-03, 0.97986876, 12, 12.),  
( 91, 106, 0, 2.62156653e-03, 0.9456603 , 11, 11.),  
( 92, 105, 0, 2.60510342e-03, 0.99107606, 9, 9.),  
( 93, 104, 0, 2.58040894e-03, 0.954434 , 8, 8.),  
( 94, 95, 0, 2.46708468e-03, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 96, 97, 0, 2.47351208e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 98, 99, 0, 2.49572587e-03, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(100, 101, 0, 2.53631955e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(102, 103, 0, 2.56789243e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(111, 112, 0, 2.89658899e-03, 0.38094659, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(113, 114, 0, 2.90166319e-03, 0.54356444, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(115, 116, 0, 2.98702286e-03, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(117, 118, 0, 2.99863715e-03, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(120, 137, 0, 3.36702541e-03, 0.954434 , 40, 40.),  
(121, 136, 0, 3.36240220e-03, 0.99679163, 15, 15.),  
(122, 135, 0, 3.28504865e-03, 0.97986876, 12, 12.),  
(123, 134, 0, 3.24783765e-03, 0.99107606, 9, 9.),  
(124, 125, 0, 3.18288768e-03, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(126, 127, 0, 3.20859696e-03, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(128, 129, 0, 3.21874546e-03, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

(130, 131, 0, 3.23035987e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(132, 133, 0, 3.23938066e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(138, 139, 0, 3.52556654e-03, 0.85545081, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(140, 141, 0, 3.55488423e-03, 0.99679163, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(142, 143, 0, 3.57484282e-03, 0.9612366 , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(144, 145, 0, 3.60697950e-03, 1. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(146, 147, 0, 3.65749619e-03, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(148, 149, 0, 3.68557358e-03, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(150, 151, 0, 3.70023225e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(152, 153, 0, 3.71139555e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(154, 155, 0, 3.71635705e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(157, 202, 0, 6.08669338e-03, 0.62320363, 296, 296.),  
(158, 159, 0, 3.92327248e-03, 0.46613328, 111, 111.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(160, 161, 0, 3.98280984e-03, 0.49991596, 100, 100.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(162, 201, 0, 5.85801527e-03, 0.47218938, 99, 99.),  
(163, 200, 0, 5.84876887e-03, 0.49596907, 92, 92.),  
(164, 199, 0, 5.53732482e-03, 0.46550245, 91, 91.),  
(165, 194, 0, 5.44407195e-03, 0.51163978, 79, 79.),  
(166, 193, 0, 5.02437772e-03, 0.45583146, 73, 73.),  
(167, 180, 0, 4.77777072e-03, 0.56310282, 53, 53.),  
(168, 179, 0, 4.67775227e-03, 0.37764632, 41, 41.),  
(169, 178, 0, 4.65971068e-03, 0.43055187, 34, 34.),  
(170, 177, 0, 4.46655182e-03, 0.32984607, 33, 33.),  
(171, 176, 0, 4.43385146e-03, 0.46899559, 20, 20.),  
(172, 175, 0, 4.23077005e-03, 0.29747225, 19, 19.),

(173, 174, 0, 4.19265707e-03, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(181, 182, 0, 4.79750382e-03, 0.91829583, 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(183, 192, 0, 5.00272773e-03, 0.84535094, 11, 11.),  
(184, 191, 0, 4.89267358e-03, 0.72192809, 10, 10.),  
(185, 190, 0, 4.88714827e-03, 0.97095059, 5, 5.),  
(186, 189, 0, 4.85072657e-03, 0.81127812, 4, 4.),  
(187, 188, 0, 4.81498172e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
(195, 196, 0, 5.45004825e-03, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(197, 198, 0, 5.51849394e-03, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(203, 212, 0, 6.27207127e-03, 0.69977222, 185, 185.),  
(204, 207, 0, 6.14645635e-03, 0.954434 , 8, 8.),  
(205, 206, 0, 6.09988649e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(208, 209, 0, 6.24162611e-03, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(210, 211, 0, 6.26237388e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(213, 214, 0, 6.53435197e-03, 0.65654028, 177, 177.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
(215, 216, 0, 6.55295746e-03, 0.69128987, 162, 162.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(217, 218, 0, 6.75592641e-03, 0.68034873, 161, 161.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(219, 224, 0, 6.95844390e-03, 0.69028031, 157, 157.),  
(220, 221, 0, 6.82843127e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(222, 223, 0, 6.87556504e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(225, 226, 0, 7.15735322e-03, 0.66974162, 154, 154.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(227, 232, 0, 7.30236317e-03, 0.68804762, 147, 147.),  
(228, 231, 0, 7.26571609e-03, 0.97095059, 5, 5.),  
(229, 230, 0, 7.19095580e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(233, 234, 0, 7.47218030e-03, 0.65544444, 142, 142.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(235, 236, 0, 7.49044749e-03, 0.68106295, 133, 133.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(237, 238, 0, 7.64583144e-03, 0.66731824, 132, 132.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(239, 240, 0, 7.72397430e-03, 0.68560152, 126, 126.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(241, 242, 0, 7.97588099e-03, 0.67124789, 125, 125.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(243, 244, 0, 8.02910421e-03, 0.69734097, 117, 117.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(245, 246, 0, 8.19734251e-03, 0.68233489, 116, 116.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(247, 248, 0, 8.22632201e-03, 0.69977222, 111, 111.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(249, 250, 0, 8.48104758e-03, 0.68403844, 110, 110.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(251, 252, 0, 8.52682861e-03, 0.71012349, 103, 103.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(253, 254, 0, 8.80726334e-03, 0.69361264, 102, 102.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(255, 256, 0, 8.84785689e-03, 0.72192809, 95, 95.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(257, 258, 0, 8.92385747e-03, 0.70457671, 94, 94.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),

(259, 260, 0, 8.95464141e-03, 0.73092638, 88, 88.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(261, 320, 0, 1.10913380e-02, 0.71263684, 87, 87.),  
(262, 319, 0, 1.10536758e-02, 0.75319799, 74, 74.),  
(263, 318, 0, 1.09695564e-02, 0.73275255, 73, 73.),  
(264, 317, 0, 1.09316693e-02, 0.75537541, 69, 69.),  
(265, 316, 0, 1.04842358e-02, 0.70982589, 67, 67.),  
(266, 315, 0, 1.04726218e-02, 0.76765159, 58, 58.),  
(267, 272, 0, 9.18004941e-03, 0.74248757, 57, 57.),  
(268, 271, 0, 8.99602426e-03, 0.46899559, 10, 10.),  
(269, 270, 0, 8.97798268e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(273, 274, 0, 9.20293992e-03, 0.78499209, 47, 47.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(275, 282, 0, 9.28288698e-03, 0.75537541, 46, 46.),  
(276, 281, 0, 9.27059585e-03, 0.97095059, 5, 5.),  
(277, 280, 0, 9.23721865e-03, 0.81127812, 4, 4.),  
(278, 279, 0, 9.21421591e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(283, 284, 0, 9.53422952e-03, 0.71206405, 41, 41.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(285, 286, 0, 9.54787340e-03, 0.77551266, 35, 35.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(287, 292, 0, 9.81691992e-03, 0.73353793, 34, 34.),  
(288, 291, 0, 9.60075809e-03, 0.41381685, 12, 12.),  
(289, 290, 0, 9.56738135e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(293, 294, 0, 9.82075371e-03, 0.84535094, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(295, 296, 0, 9.83890845e-03, 0.79185835, 21, 21.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(297, 298, 0, 9.91806621e-03, 0.83147439, 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(299, 300, 0, 1.01730172e-02, 0.76420451, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(301, 302, 0, 1.01941032e-02, 0.86312057, 14, 14.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(303, 304, 0, 1.02463113e-02, 0.77934984, 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(305, 306, 0, 1.02537535e-02, 0.8812909 , 10, 10.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(307, 310, 0, 1.02768699e-02, 0.76420451, 9, 9.),
(308, 309, 0, 1.02657066e-02, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(311, 312, 0, 1.04576242e-02, 0.59167278, 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(313, 314, 0, 1.04616839e-02, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(321, 322, 0, 1.14768660e-02, 0.39124356, 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
(323, 324, 0, 1.15198274e-02, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),
(328, 425, 0, 2.71567618e-02, 0.87531983, 166, 166.),
(329, 424, 0, 2.62307739e-02, 0.9133526 , 137, 137.),
(330, 423, 0, 2.51084697e-02, 0.89116716, 133, 133.),
(331, 402, 0, 2.15384765e-02, 0.90472097, 128, 128.),
(332, 401, 0, 1.96654126e-02, 0.84409917, 103, 103.),
(333, 346, 0, 1.29495165e-02, 0.90637019, 87, 87.),
(334, 335, 0, 1.22788176e-02, 0.954434 , 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(336, 337, 0, 1.23969903e-02, 0.98522814, 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(338, 339, 0, 1.26260067e-02, 0.91829583, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(340, 341, 0, 1.27186961e-02, 1. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(342, 343, 0, 1.27834207e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(344, 345, 0, 1.28491600e-02, 1. , 2, 2.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(347, 352, 0, 1.39277130e-02, 0.87018834, 79, 79.),  
(348, 351, 0, 1.36281084e-02, 0.33729007, 16, 16.),  
(349, 350, 0, 1.35144461e-02, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(353, 354, 0, 1.39935650e-02, 0.93335726, 63, 63.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(355, 364, 0, 1.56900445e-02, 0.9235786 , 62, 62.),  
(356, 363, 0, 1.47634926e-02, 0.72192809, 20, 20.),  
(357, 362, 0, 1.46107026e-02, 0.97095059, 10, 10.),  
(358, 361, 0, 1.41215478e-02, 0.59167278, 7, 7.),  
(359, 360, 0, 1.40617848e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(365, 366, 0, 1.60116376e-02, 0.97366806, 42, 42.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(367, 400, 0, 1.96130918e-02, 0.94182854, 39, 39.),  
(368, 399, 0, 1.94890564e-02, 0.92681906, 38, 38.),  
(369, 398, 0, 1.94061771e-02, 0.94360163, 36, 36.),  
(370, 397, 0, 1.86381657e-02, 0.92752659, 35, 35.),  
(371, 392, 0, 1.83261577e-02, 0.954434 , 32, 32.),  
(372, 387, 0, 1.73859615e-02, 0.90592822, 28, 28.),  
(373, 386, 0, 1.70696685e-02, 0.99679163, 15, 15.),  
(374, 385, 0, 1.67839332e-02, 0.91829583, 12, 12.),  
(375, 380, 0, 1.62936514e-02, 1. , 8, 8.),  
(376, 379, 0, 1.62040070e-02, 0.81127812, 4, 4.),  
(377, 378, 0, 1.61225935e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(381, 382, 0, 1.65124061e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(383, 384, 0, 1.66676771e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),

(388, 389, 0, 1.77260460e-02, 0.61938219, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(390, 391, 0, 1.79073652e-02, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(393, 394, 0, 1.85172865e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(395, 396, 0, 1.85922720e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
(403, 404, 0, 2.20935950e-02, 0.99884554, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(405, 418, 0, 2.41666948e-02, 0.99403021, 22, 22.),  
(406, 417, 0, 2.36080801e-02, 0.91829583, 15, 15.),  
(407, 416, 0, 2.34519066e-02, 0.97986876, 12, 12.),  
(408, 415, 0, 2.31755311e-02, 0.8812909 , 10, 10.),  
(409, 410, 0, 2.24986309e-02, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(411, 414, 0, 2.29483191e-02, 0.81127812, 4, 4.),  
(412, 413, 0, 2.29054699e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(419, 420, 0, 2.47476362e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(421, 422, 0, 2.49782316e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(426, 427, 0, 2.84561003e-02, 0.57879462, 29, 29.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(428, 429, 0, 2.86799297e-02, 0.72192809, 20, 20.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(430, 431, 0, 3.09935473e-02, 0.62924922, 19, 19.),

( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(432, 433, 0, 3.13885463e-02, 0.8812909 , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(434, 441, 0, 3.32344342e-02, 0.76420451, 9, 9.),  
(435, 440, 0, 3.32057942e-02, 0.91829583, 6, 6.),  
(436, 439, 0, 3.24562751e-02, 0.72192809, 5, 5.),  
(437, 438, 0, 3.19780577e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(443, 566, 0, 1.47200130e-01, 0.92233162, 163, 163.),  
(444, 445, 0, 3.45518142e-02, 0.98982206, 118, 118.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(446, 447, 0, 3.46288290e-02, 0.99339005, 115, 115.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(448, 449, 0, 3.51146031e-02, 0.9919924 , 114, 114.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(450, 451, 0, 3.56965587e-02, 0.99525255, 111, 111.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(452, 461, 0, 3.82231809e-02, 0.99107606, 108, 108.),  
(453, 456, 0, 3.61851491e-02, 0.77934984, 13, 13.),  
(454, 455, 0, 3.59151997e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(457, 460, 0, 3.70543059e-02, 0.46899559, 10, 10.),  
(458, 459, 0, 3.68308146e-02, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(462, 467, 0, 4.04487308e-02, 0.99800088, 95, 95.),  
(463, 466, 0, 3.88409942e-02, 0.65002242, 6, 6.),  
(464, 465, 0, 3.85367665e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(468, 469, 0, 4.09484860e-02, 0.99261089, 89, 89.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(470, 471, 0, 4.09702491e-02, 0.99532511, 87, 87.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(472, 473, 0, 4.20656037e-02, 0.99374891, 86, 86.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),

(474, 475, 0, 4.26775552e-02, 0.99631652, 84, 84.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(476, 481, 0, 4.52499576e-02, 0.99312318, 82, 82.),  
(477, 480, 0, 4.32139561e-02, 0.59167278, 7, 7.),  
(478, 479, 0, 4.27579526e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(482, 483, 0, 4.71164789e-02, 0.99884554, 75, 75.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(484, 509, 0, 6.52139597e-02, 0.99297689, 71, 71.),  
(485, 504, 0, 5.59600517e-02, 0.93058613, 26, 26.),  
(486, 491, 0, 4.88154404e-02, 0.99750255, 17, 17.),  
(487, 490, 0, 4.78192028e-02, 0.86312057, 7, 7.),  
(488, 489, 0, 4.72999401e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(492, 493, 0, 5.03994990e-02, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(494, 495, 0, 5.36852255e-02, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(496, 497, 0, 5.48812747e-02, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(498, 499, 0, 5.49313389e-02, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(500, 501, 0, 5.51860649e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(502, 503, 0, 5.56194037e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(505, 506, 0, 6.18370119e-02, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(507, 508, 0, 6.32832777e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(510, 527, 0, 7.99387768e-02, 0.99964375, 45, 45.),  
(511, 526, 0, 7.80755207e-02, 0.83664074, 15, 15.),  
(512, 525, 0, 7.66224898e-02, 0.89049164, 13, 13.),  
(513, 516, 0, 6.62654489e-02, 0.81127812, 12, 12.),  
(514, 515, 0, 6.57700934e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

(517, 518, 0, 7.00675026e-02, 0.72192809, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(519, 520, 0, 7.06212707e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(521, 522, 0, 7.47089498e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(523, 524, 0, 7.55614154e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(528, 529, 0, 8.08838233e-02, 0.97095059, 30, 30.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(530, 539, 0, 9.14429314e-02, 0.98522814, 28, 28.),  
(531, 538, 0, 8.69909264e-02, 0.91829583, 6, 6.),  
(532, 533, 0, 8.13746676e-02, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(534, 535, 0, 8.31094868e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(536, 537, 0, 8.46663713e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(540, 549, 0, 1.16765082e-01, 0.9456603 , 22, 22.),  
(541, 548, 0, 1.11961037e-01, 0.68403844, 11, 11.),  
(542, 547, 0, 1.07546017e-01, 0.86312057, 7, 7.),  
(543, 544, 0, 9.92704183e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(545, 546, 0, 1.00508414e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(550, 551, 0, 1.18423786e-01, 0.99403021, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(552, 553, 0, 1.20031860e-01, 1. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(554, 555, 0, 1.21141873e-01, 0.99107606, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(556, 557, 0, 1.24251015e-01, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(558, 559, 0, 1.26612671e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
(560, 561, 0, 1.27006657e-01, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(562, 563, 0, 1.36103377e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(564, 565, 0, 1.43079750e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(567, 578, 0, 2.61911497e-01, 0.35335934, 45, 45.),  
(568, 577, 0, 2.36312412e-01, 0.49123734, 28, 28.),  
(569, 576, 0, 2.00959317e-01, 0.38094659, 27, 27.),  
(570, 575, 0, 1.97032437e-01, 0.56650951, 15, 15.),  
(571, 574, 0, 1.70623057e-01, 0.37123233, 14, 14.),  
(572, 573, 0, 1.69561639e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.)],  
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

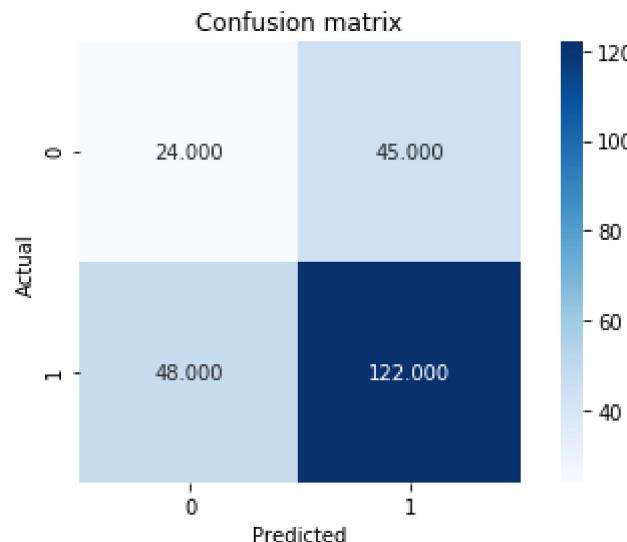
```
In [80]: # TASK 4  
# Predict class labels using decision tree  
x_test = x_val[["Total Population"]]  
y_pred = model.predict(x_test)
```

```
In [81]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 24 45]  
 [ 48 122]]
```

In [82]: # TASK 4

```
# 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()
```



In [83]: # TASK 4

```
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.6108786610878661
0.38912133891213385
[0.33333333 0.73053892]
[0.34782609 0.71764706]
[0.34042553 0.72403561]
```

```
In [84]: # Task 4  
# Build decision tree with two Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[84]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [85]: # TASK 4
# Show decision tree
model.tree_.__getstate__()['nodes']
```

```
Out[85]: array([( 1, 136, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 123, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 24, 1, 3.57535481e-01, 0.98654463, 176, 176.),
( 4, 9, 0, 3.94524951e-02, 0.63945713, 37, 37.),
( 5, 8, 1, 2.63141811e-01, 0.25801867, 23, 23.),
( 6, 7, 1, 2.58012146e-01, 0.46899559, 10, 10.),
(-1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 13, 13.),
( 10, 11, 0, 4.73814663e-02, 0.94028596, 14, 14.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 12, 23, 0, 1.02302209e-01, 0.81127812, 12, 12.),
( 13, 22, 0, 8.54340419e-02, 0.91829583, 9, 9.),
( 14, 21, 1, 3.33328024e-01, 0.81127812, 8, 8.),
( 15, 20, 1, 3.18483472e-01, 0.91829583, 6, 6.),
( 16, 19, 1, 1.24027018e-01, 0.72192809, 5, 5.),
( 17, 18, 0, 7.50223100e-02, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 25, 26, 0, 1.81961444e-03, 0.999996266, 139, 139.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 27, 28, 0, 2.42288259e-03, 0.9985091 , 132, 132.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 29, 30, 0, 3.05659592e-03, 0.99960984, 129, 129.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 31, 122, 0, 1.27006657e-01, 0.99836367, 126, 126.),
( 32, 55, 0, 1.60930511e-02, 0.99699543, 124, 124.),
( 33, 50, 1, 5.61291516e-01, 0.94743514, 41, 41.),
( 34, 49, 0, 1.16445404e-02, 0.74959526, 28, 28.),
( 35, 42, 0, 7.95840332e-03, 0.91829583, 18, 18.),
( 36, 41, 1, 4.97838512e-01, 0.65002242, 12, 12.),
( 37, 38, 0, 5.19385654e-03, 1. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 39, 40, 1, 4.87494513e-01, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( 43, 44, 0, 8.86014802e-03, 0.91829583, 6, 6.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 45, 46, 1, 4.58994254e-01, 1. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 47, 48, 0, 9.28581832e-03, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
( 51, 52, 0, 8.65503680e-03, 0.89049164, 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( 53, 54, 1, 5.78316599e-01, 0.98522814, 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 56, 75, 0, 3.37270852e-02, 0.9990574 , 83, 83.),
( 57, 70, 1, 5.30552149e-01, 0.8812909 , 30, 30.),
( 58, 67, 1, 4.86347914e-01, 0.99679163, 15, 15.),
( 59, 60, 0, 2.21794061e-02, 0.76420451, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 61, 62, 1, 4.27194670e-01, 1. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 63, 64, 0, 2.95596858e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 65, 66, 0, 3.18667637e-02, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 68, 69, 0, 2.74492623e-02, 0.65002242, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 71, 74, 0, 2.30477741e-02, 0.56650951, 15, 15.),
( 72, 73, 0, 1.90824419e-02, 0.97095059, 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
( 76, 117, 1, 5.83450347e-01, 0.97909817, 53, 53.),
( 77, 116, 1, 5.81172079e-01, 0.99679163, 45, 45.),
( 78, 115, 1, 5.79644561e-01, 0.99022469, 43, 43.),
( 79, 114, 0, 1.19533800e-01, 0.99819588, 40, 40.),
( 80, 89, 1, 4.75556821e-01, 1. , 38, 38.),
( 81, 88, 0, 5.03994990e-02, 0.91829583, 12, 12.),
( 82, 87, 1, 4.15925637e-01, 1. , 8, 8.),
( 83, 86, 0, 3.79716121e-02, 0.91829583, 6, 6.),
( 84, 85, 1, 4.03274387e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 90, 91, 0, 4.01309729e-02, 0.98285869, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 92, 99, 0, 6.18370119e-02, 0.99498483, 24, 24.),  
( 93, 98, 0, 5.64148147e-02, 0.91829583, 9, 9.),  
( 94, 97, 1, 5.39678007e-01, 0.97095059, 5, 5.),  
( 95, 96, 0, 5.11272531e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(100, 101, 0, 7.66224898e-02, 0.91829583, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(102, 103, 1, 5.12419969e-01, 1. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(104, 113, 1, 5.78821361e-01, 0.954434 , 8, 8.),  
(105, 112, 1, 5.64582467e-01, 0.86312057, 7, 7.),  
(106, 107, 1, 5.40877074e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(108, 109, 1, 5.54938257e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(110, 111, 1, 5.62174559e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(118, 119, 0, 6.82142898e-02, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(120, 121, 1, 5.96927643e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(124, 135, 1, 5.00625357e-01, 0.37123233, 42, 42.),  
(125, 134, 1, 4.90885124e-01, 0.5746357 , 22, 22.),  
(126, 133, 1, 3.42970073e-01, 0.45371634, 21, 21.),  
(127, 132, 1, 3.32374901e-01, 0.72192809, 10, 10.),  
(128, 131, 1, 2.10741274e-01, 0.50325833, 9, 9.),  
(129, 130, 1, 1.92532741e-01, 0.91829583, 3, 3.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
(137, 378, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
(138, 201, 1, 6.92266941e-01, 0.56310282, 689, 689.),  
(139, 148, 0, 2.43641390e-03, 0.79129911, 143, 143.),  
(140, 141, 1, 6.58703804e-01, 0.30137864, 56, 56.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
(142, 147, 1, 6.72340810e-01, 0.50325833, 27, 27.),  
(143, 146, 0, 1.43713126e-03, 0.8812909 , 10, 10.),  
(144, 145, 1, 6.58960432e-01, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
(149, 154, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
(150, 151, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(152, 153, 0, 2.02996898e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(155, 156, 0, 3.63336538e-03, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(157, 200, 0, 2.77971281e-02, 0.9612366 , 65, 65.),  
(158, 199, 0, 2.61669513e-02, 0.97641431, 61, 61.),  
(159, 198, 0, 2.51444401e-02, 0.95755348, 58, 58.),  
(160, 197, 0, 2.37771077e-02, 0.97095059, 55, 55.),  
(161, 196, 0, 1.90389156e-02, 0.95615502, 53, 53.),  
(162, 195, 1, 6.90091312e-01, 0.9839394 , 47, 47.),  
(163, 174, 0, 9.36373603e-03, 0.96241274, 44, 44.),  
(164, 173, 1, 6.75937116e-01, 0.78712659, 17, 17.),  
(165, 172, 0, 8.25981190e-03, 0.91829583, 12, 12.),  
(166, 171, 1, 6.68758631e-01, 0.99107606, 9, 9.),  
(167, 168, 0, 3.80194222e-03, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(169, 170, 1, 6.33273989e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(175, 178, 1, 6.38267547e-01, 0.99901027, 27, 27.),  
(176, 177, 0, 1.10741984e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(179, 184, 1, 6.72508746e-01, 0.98522814, 21, 21.),  
(180, 183, 1, 6.48556590e-01, 0.54356444, 8, 8.),  
(181, 182, 1, 6.41787350e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(185, 186, 0, 1.13350130e-02, 0.9612366 , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(187, 190, 1, 6.80730879e-01, 0.99107606, 9, 9.),  
(188, 189, 0, 1.82924420e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(191, 192, 1, 6.85705692e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(193, 194, 1, 6.87871605e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(202, 203, 0, 6.57956902e-04, 0.48277644, 546, 546.),  
( -1, -1, -2, -2.00000000e+00, 0. , 30, 30.),  
(204, 295, 0, 3.73439875e-03, 0.5013173 , 516, 516.),  
(205, 206, 1, 7.19135463e-01, 0.64230242, 202, 202.),  
( -1, -1, -2, -2.00000000e+00, 0. , 22, 22.),  
(207, 208, 0, 7.52224587e-04, 0.68731509, 180, 180.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(209, 268, 1, 8.11294794e-01, 0.66712714, 178, 178.),  
(210, 235, 1, 7.71310896e-01, 0.78499209, 94, 94.),  
(211, 212, 1, 7.19316483e-01, 0.56996138, 52, 52.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(213, 234, 1, 7.61038512e-01, 0.52255937, 51, 51.),  
(214, 227, 1, 7.56286770e-01, 0.6098403 , 40, 40.),  
(215, 226, 1, 7.40787297e-01, 0.43949699, 33, 33.),
```

(216, 221, 1, 7.38938808e-01, 0.62924922, 19, 19.),  
(217, 220, 0, 8.79305560e-04, 0.33729007, 16, 16.),  
(218, 219, 1, 7.32154161e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(222, 223, 1, 7.39733994e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(224, 225, 1, 7.40175098e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
(228, 233, 0, 3.23419366e-03, 0.98522814, 7, 7.),  
(229, 232, 0, 1.76413642e-03, 0.97095059, 5, 5.),  
(230, 231, 1, 7.59198755e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(236, 253, 1, 7.85911888e-01, 0.94028596, 42, 42.),  
(237, 248, 1, 7.79222667e-01, 0.99750255, 17, 17.),  
(238, 247, 1, 7.74449050e-01, 0.8812909 , 10, 10.),  
(239, 240, 0, 8.76148260e-04, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(241, 246, 0, 2.77695025e-03, 0.97095059, 5, 5.),  
(242, 243, 1, 7.72538334e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(244, 245, 1, 7.73585141e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(249, 250, 1, 7.83212394e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(251, 252, 1, 7.84530818e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(254, 255, 1, 7.95957208e-01, 0.79504028, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(256, 257, 1, 7.96819925e-01, 0.97095059, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(258, 259, 1, 8.00345033e-01, 0.89049164, 13, 13.),

( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(260, 261, 1, 8.01197886e-01, 0.99107606, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(262, 267, 1, 8.08480829e-01, 0.86312057, 7, 7.),  
(263, 266, 1, 8.03280801e-01, 0.65002242, 6, 6.),  
(264, 265, 1, 8.02897751e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(269, 270, 1, 8.33844781e-01, 0.49123734, 84, 84.),  
( -1, -1, -2, -2.00000000e+00, 0. , 22, 22.),  
(271, 272, 1, 8.35351348e-01, 0.59759778, 62, 62.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(273, 278, 0, 2.56789243e-03, 0.56057694, 61, 61.),  
(274, 277, 1, 8.41681749e-01, 0.2108423 , 30, 30.),  
(275, 276, 1, 8.36679488e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(279, 284, 0, 2.64851621e-03, 0.77062907, 31, 31.),  
(280, 281, 1, 8.59667778e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(282, 283, 1, 9.10862148e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(285, 286, 0, 3.18615767e-03, 0.61938219, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(287, 292, 0, 3.29463324e-03, 0.86312057, 14, 14.),  
(288, 289, 1, 8.65339071e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(290, 291, 1, 8.93833071e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(293, 294, 0, 3.67531227e-03, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(296, 297, 0, 4.65971068e-03, 0.38948465, 314, 314.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
(298, 313, 1, 7.12382376e-01, 0.42516284, 277, 277.),  
(299, 312, 1, 7.10477173e-01, 0.77934984, 26, 26.),  
(300, 301, 0, 7.21283141e-03, 0.65002242, 24, 24.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),

(302, 303, 0, 8.03282484e-03, 0.78712659, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(304, 305, 0, 1.33669553e-02, 0.69621226, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(306, 311, 0, 2.45137718e-02, 0.8812909 , 10, 10.),  
(307, 310, 1, 7.06671745e-01, 1. , 6, 6.),  
(308, 309, 0, 2.38023661e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(314, 315, 0, 4.67775227e-03, 0.3722845 , 251, 251.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(316, 359, 1, 8.27778816e-01, 0.35841532, 250, 250.),  
(317, 340, 1, 7.53632456e-01, 0.29405107, 193, 193.),  
(318, 323, 1, 7.39098072e-01, 0.43949699, 66, 66.),  
(319, 322, 0, 8.13081395e-03, 0.17203695, 39, 39.),  
(320, 321, 0, 7.95558421e-03, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 31, 31.),  
(324, 325, 1, 7.39496171e-01, 0.69128987, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(326, 339, 1, 7.52145201e-01, 0.61938219, 26, 26.),  
(327, 338, 1, 7.44749516e-01, 0.52936087, 25, 25.),  
(328, 337, 1, 7.44062692e-01, 0.74959526, 14, 14.),  
(329, 336, 0, 1.41009130e-02, 0.61938219, 13, 13.),  
(330, 335, 0, 1.23107289e-02, 0.81127812, 8, 8.),  
(331, 332, 0, 8.52040085e-03, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(333, 334, 0, 9.87476576e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(341, 358, 0, 7.28488527e-03, 0.20183993, 127, 127.),  
(342, 347, 0, 6.53435197e-03, 0.46121604, 41, 41.),  
(343, 346, 0, 5.07218810e-03, 0.20062232, 32, 32.),  
(344, 345, 0, 5.00272773e-03, 0.65002242, 6, 6.),

( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 26, 26.),  
(348, 349, 0, 6.68567652e-03, 0.91829583, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(350, 357, 1, 7.83345342e-01, 0.81127812, 8, 8.),  
(351, 352, 1, 7.63837278e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(353, 354, 1, 7.67989099e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(355, 356, 0, 7.15554901e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 86, 86.),  
(360, 375, 0, 1.62936514e-02, 0.53737609, 57, 57.),  
(361, 370, 1, 8.40767294e-01, 0.44506486, 54, 54.),  
(362, 369, 1, 8.38225424e-01, 0.76420451, 18, 18.),  
(363, 364, 1, 8.28199685e-01, 0.54356444, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(365, 366, 0, 7.98546593e-03, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(367, 368, 0, 9.42338631e-03, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(371, 372, 1, 8.96882266e-01, 0.18312207, 36, 36.),  
( -1, -1, -2, -2.00000000e+00, 0. , 33, 33.),  
(373, 374, 1, 9.08836722e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(376, 377, 0, 1.72288856e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(379, 406, 0, 1.65806837e-01, 0.99729438, 49, 49.),  
(380, 381, 0, 3.51146031e-02, 0.9985091 , 44, 44.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(382, 383, 0, 3.61851491e-02, 0.9892453 , 41, 41.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(384, 385, 0, 3.80437803e-02, 1. , 36, 36.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(386, 401, 1, 6.92403942e-01, 0.98869941, 32, 32.),  
(387, 400, 1, 6.54390246e-01, 0.90239328, 22, 22.),

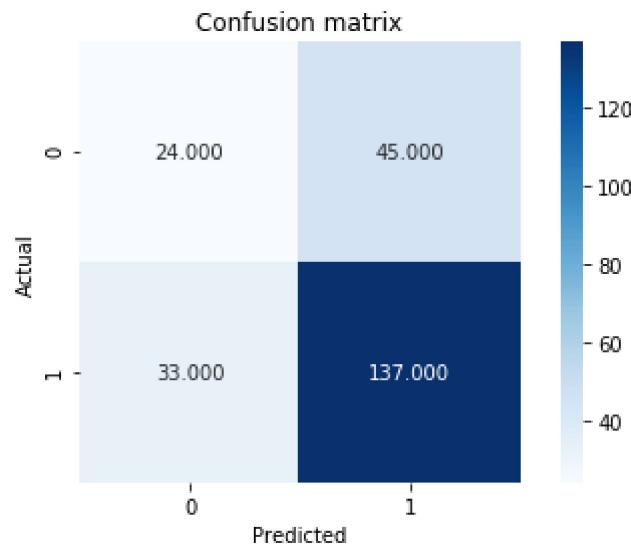
```
(388, 399, 1, 6.47211939e-01, 0.99572745, 13, 13.),
(389, 390, 1, 6.12718821e-01, 0.99403021, 11, 11.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(391, 392, 1, 6.23185068e-01, 0.97095059, 10, 10.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
(393, 394, 1, 6.27582967e-01, 0.98522814, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(395, 398, 0, 4.57745176e-02, 0.97095059, 5, 5.),
(396, 397, 1, 6.30692631e-01, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 9, 9.),
(402, 405, 1, 7.73642480e-01, 0.8812909, 10, 10.),
(403, 404, 0, 1.18469346e-01, 0.54356444, 8, 8.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

In [86]: # TASK 4  
*# Predict class labels using decision tree*  
`x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree"]]  
y_pred = model.predict(x_test)`

In [87]: # TASK 4  
*# Compute confusion matrix*  
`y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)`

```
[[ 24  45]
 [ 33 137]]
```

```
In [88]: # TASK 4  
# 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()
```



```
In [89]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.6736401673640168  
0.3263598326359832  
[0.42105263 0.75274725]  
[0.34782609 0.80588235]  
[0.38095238 0.77840909]

In [90]: # Task 4

```
# Build decision tree with three Parameters
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin
o"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

Out[90]: DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=None,
max\_features=None, max\_leaf\_nodes=None,
min\_impurity\_decrease=0.0, min\_impurity\_split=None,
min\_samples\_leaf=1, min\_samples\_split=2,
min\_weight\_fraction\_leaf=0.0, presort=False,
random\_state=0, splitter='best')

```
In [91]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[91]: array([( 1, 112, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 105, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 18, 1, 3.57535481e-01, 0.98654463, 176, 176.),
( 4, 9, 0, 3.94524951e-02, 0.63945713, 37, 37.),
( 5, 8, 2, 2.13248599e-02, 0.25801867, 23, 23.),
( 6, 7, 2, 2.06069369e-02, 0.65002242, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 17, 17.),
( 10, 17, 2, 8.57343748e-02, 0.94028596, 14, 14.),
( 11, 12, 1, 1.91999584e-01, 0.99107606, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 13, 14, 2, 3.15819420e-02, 0.91829583, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 15, 16, 2, 8.05552378e-02, 0.72192809, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 19, 20, 0, 1.81961444e-03, 0.99996266, 139, 139.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 21, 22, 0, 2.42288259e-03, 0.9985091 , 132, 132.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 23, 24, 0, 3.05659592e-03, 0.99960984, 129, 129.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 25, 104, 2, 3.73483226e-01, 0.99836367, 126, 126.),
( 26, 51, 0, 1.60930511e-02, 0.99995073, 121, 121.),
( 27, 46, 1, 5.61291516e-01, 0.94743514, 41, 41.),
( 28, 45, 0, 1.16445404e-02, 0.74959526, 28, 28.),
( 29, 38, 0, 7.95840332e-03, 0.91829583, 18, 18.),
( 30, 37, 1, 4.97838512e-01, 0.65002242, 12, 12.),
( 31, 32, 1, 4.17280599e-01, 1. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 33, 34, 1, 4.60516438e-01, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 35, 36, 1, 4.87494513e-01, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( 39, 40, 2, 1.02523337e-02, 0.91829583, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 41, 42, 0, 9.72468173e-03, 0.72192809, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 43, 44, 2, 1.15088682e-01, 1. , 2, 2.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
( 47, 48, 0, 8.65503680e-03, 0.89049164, 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( 49, 50, 1, 5.78316599e-01, 0.98522814, 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 52, 65, 0, 3.37270852e-02, 0.98869941, 80, 80.),
( 53, 64, 1, 5.30552149e-01, 0.76420451, 27, 27.),
( 54, 61, 1, 4.93271291e-01, 0.98522814, 14, 14.),
( 55, 56, 0, 2.21794061e-02, 0.76420451, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 57, 58, 1, 4.27194670e-01, 1. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 59, 60, 2, 2.63449112e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 62, 63, 0, 2.74492623e-02, 0.72192809, 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),
( 66, 103, 0, 1.27006657e-01, 0.99357048, 53, 53.),
( 67, 68, 2, 1.97092192e-02, 0.98636761, 51, 51.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 69, 70, 2, 2.31544580e-02, 0.99498483, 48, 48.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 71, 74, 2, 3.59187610e-02, 0.97602065, 44, 44.),
( 72, 73, 0, 1.05611384e-01, 0.54356444, 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 75, 98, 2, 2.21710473e-01, 0.99777247, 36, 36.),
( 76, 97, 2, 2.03721866e-01, 0.99679163, 30, 30.),
( 77, 96, 2, 1.67097434e-01, 0.99901027, 27, 27.),
( 78, 85, 0, 4.57241144e-02, 0.99498483, 24, 24.),
( 79, 84, 2, 8.42192620e-02, 0.8812909 , 10, 10.),
( 80, 83, 1, 4.09697846e-01, 0.54356444, 8, 8.),
( 81, 82, 0, 3.79716121e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 86, 95, 1, 5.73692113e-01, 0.86312057, 14, 14.),
```

( 87, 94, 1, 5.28889060e-01, 0.99107606, 9, 9.),  
( 88, 89, 2, 4.98318728e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 90, 91, 2, 1.05505794e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 92, 93, 0, 5.11040259e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 99, 100, 1, 5.40877074e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(101, 102, 1, 5.52039981e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(106, 111, 2, 1.46485843e-01, 0.37123233, 42, 42.),  
(107, 110, 2, 1.39678806e-01, 0.77934984, 13, 13.),  
(108, 109, 2, 2.99343411e-02, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
(113, 304, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
(114, 303, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
(115, 172, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
(116, 121, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
(117, 120, 0, 4.52394699e-04, 0.13303965, 54, 54.),  
(118, 119, 0, 4.25896011e-04, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
(122, 127, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
(123, 124, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(125, 126, 0, 2.02996898e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(128, 129, 0, 3.63336538e-03, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),

(130, 131, 2, 3.86667950e-03, 0.9612366 , 65, 65.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(132, 171, 0, 2.77971281e-02, 0.93831535, 62, 62.),  
(133, 170, 0, 2.61669513e-02, 0.95755348, 58, 58.),  
(134, 155, 2, 4.40534689e-02, 0.92994294, 55, 55.),  
(135, 154, 2, 2.31719753e-02, 0.82240423, 35, 35.),  
(136, 153, 2, 2.17614407e-02, 0.91829583, 27, 27.),  
(137, 150, 1, 6.75937116e-01, 0.85545081, 25, 25.),  
(138, 139, 1, 6.36993587e-01, 0.97095059, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(140, 141, 1, 6.41787350e-01, 0.99403021, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(142, 143, 2, 7.23552890e-03, 0.99107606, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(144, 145, 0, 5.65775274e-03, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(146, 147, 2, 1.85538232e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(148, 149, 2, 1.93917863e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(151, 152, 1, 6.89515024e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(156, 159, 2, 7.76724592e-02, 1. , 20, 20.),  
(157, 158, 0, 1.95331452e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(160, 169, 2, 3.48945007e-01, 0.94028596, 14, 14.),  
(161, 162, 0, 1.40508474e-02, 0.89049164, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(163, 164, 2, 9.17951874e-02, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(165, 168, 1, 6.57142520e-01, 0.91829583, 6, 6.),  
(166, 167, 2, 9.82754454e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),

(173, 272, 2, 3.17869987e-02, 0.45656871, 541, 541.),  
(174, 175, 0, 6.47921232e-04, 0.53722708, 334, 334.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(176, 225, 0, 3.68557358e-03, 0.56709563, 307, 307.),  
(177, 178, 1, 7.19135463e-01, 0.71886413, 131, 131.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(179, 212, 1, 8.11605304e-01, 0.78894066, 110, 110.),  
(180, 189, 2, 4.51100431e-03, 0.88797632, 72, 72.),  
(181, 184, 1, 7.96141684e-01, 0.50325833, 27, 27.),  
(182, 183, 1, 7.25176722e-01, 0.25801867, 23, 23.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 22, 22.),  
(185, 186, 0, 1.57909660e-03, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(187, 188, 0, 2.42874620e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(190, 197, 1, 7.55806565e-01, 0.98247409, 45, 45.),  
(191, 192, 0, 7.61358184e-04, 0.81127812, 20, 20.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(193, 194, 2, 1.34000210e-02, 0.65002242, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(195, 196, 1, 7.41338193e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(198, 201, 2, 8.14763736e-03, 0.98958752, 25, 25.),  
(199, 200, 1, 8.01848620e-01, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(202, 203, 2, 9.90183325e-03, 0.97741782, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(204, 205, 2, 1.39053124e-02, 0.99572745, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(206, 207, 1, 7.60225177e-01, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(208, 211, 1, 8.02247018e-01, 0.81127812, 8, 8.),  
(209, 210, 0, 3.44313867e-03, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(213, 216, 2, 7.08383421e-04, 0.48546076, 38, 38.),  
(214, 215, 0, 2.61784531e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(217, 222, 1, 8.97281468e-01, 0.31599713, 35, 35.),  
(218, 221, 0, 1.23686879e-03, 0.20062232, 32, 32.),  
(219, 220, 0, 1.18657766e-03, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 25, 25.),  
(223, 224, 1, 9.07615066e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(226, 271, 2, 3.14539718e-02, 0.42033516, 176, 176.),  
(227, 258, 1, 8.32422495e-01, 0.40217919, 175, 175.),  
(228, 257, 1, 7.78253704e-01, 0.30954343, 144, 144.),  
(229, 256, 1, 7.77404457e-01, 0.42304882, 93, 93.),  
(230, 243, 2, 6.63869525e-03, 0.38823898, 92, 92.),  
(231, 232, 2, 4.19529981e-03, 0.68403844, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(233, 236, 0, 6.81140437e-03, 0.91829583, 12, 12.),  
(234, 235, 2, 4.86540422e-03, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(237, 242, 1, 7.46956736e-01, 0.97095059, 5, 5.),  
(238, 241, 1, 7.42237657e-01, 0.81127812, 4, 4.),  
(239, 240, 2, 6.18952513e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(244, 245, 2, 1.77596882e-02, 0.25524211, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 43, 43.),  
(246, 247, 2, 1.80100072e-02, 0.50325833, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(248, 251, 1, 7.00808555e-01, 0.39124356, 26, 26.),  
(249, 250, 2, 2.79918984e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(252, 253, 1, 7.64922470e-01, 0.24988229, 24, 24.),  
( -1, -1, -2, -2.00000000e+00, 0. , 22, 22.),  
(254, 255, 0, 8.45736777e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 51, 51.),

(259, 270, 0, 1.52357332e-02, 0.70883567, 31, 31.),  
(260, 261, 1, 8.33037525e-01, 0.57879462, 29, 29.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(262, 269, 0, 6.23891968e-03, 0.49123734, 28, 28.),  
(263, 264, 2, 8.29666713e-03, 0.74959526, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(265, 266, 1, 8.38225424e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(267, 268, 2, 2.31178207e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(273, 282, 1, 7.19351888e-01, 0.299589 , 207, 207.),  
(274, 279, 2, 6.46481011e-02, 0.83664074, 15, 15.),  
(275, 276, 0, 6.75355853e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(277, 278, 1, 7.05695838e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(280, 281, 1, 7.00127363e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(283, 284, 2, 6.93185329e-02, 0.22581117, 192, 192.),  
( -1, -1, -2, -2.00000000e+00, 0. , 69, 69.),  
(285, 286, 2, 6.96306936e-02, 0.31505695, 123, 123.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(287, 302, 1, 8.28835458e-01, 0.28290479, 122, 122.),  
(288, 301, 1, 8.26132953e-01, 0.38431154, 80, 80.),  
(289, 298, 2, 6.20479047e-01, 0.34037329, 79, 79.),  
(290, 297, 2, 1.79032527e-01, 0.29461521, 77, 77.),  
(291, 296, 2, 1.68338291e-01, 0.42622866, 46, 46.),  
(292, 295, 2, 8.96823183e-02, 0.26676499, 44, 44.),  
(293, 294, 2, 8.40812661e-02, 0.65002242, 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 32, 32.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 31, 31.),  
(299, 300, 1, 8.10581416e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 42, 42.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(305, 334, 0, 1.65806837e-01, 0.99729438, 49, 49.),
(306, 307, 0, 3.51146031e-02, 0.9985091 , 44, 44.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(308, 309, 0, 3.61851491e-02, 0.9892453 , 41, 41.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(310, 311, 0, 3.80437803e-02, 1. , 36, 36.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(312, 329, 1, 6.92403942e-01, 0.98869941, 32, 32.),
(313, 328, 1, 6.54390246e-01, 0.90239328, 22, 22.),
(314, 327, 1, 6.47211939e-01, 0.99572745, 13, 13.),
(315, 326, 2, 4.86683130e-01, 0.99403021, 11, 11.),
(316, 317, 1, 6.12718821e-01, 0.97095059, 10, 10.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(318, 319, 1, 6.23185068e-01, 0.91829583, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(320, 321, 1, 6.27582967e-01, 1. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(322, 325, 2, 1.84076220e-01, 0.81127812, 4, 4.),
(323, 324, 0, 4.24444787e-02, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
(330, 333, 1, 7.73642480e-01, 0.8812909 , 10, 10.),
(331, 332, 0, 1.18469346e-01, 0.54356444, 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [92]: # TASK 4
# Predict class Labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino"]]
y_pred = model.predict(x_test)
```

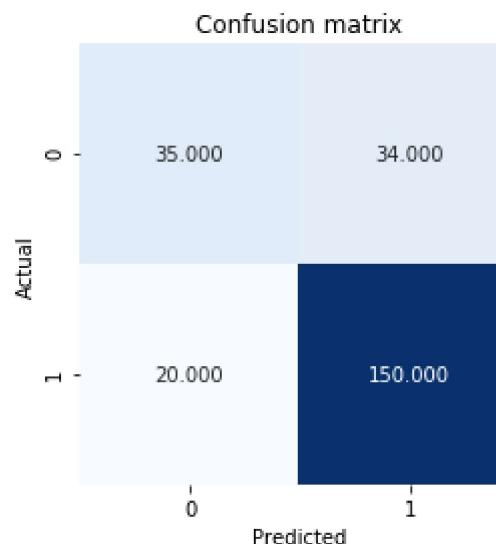
In [93]: # TASK 4

```
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 35  34]
 [ 20 150]]
```

In [94]: # TASK 4

```
# 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()
```



In [95]: # TASK 4

```
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7740585774058577
0.2259414225941423
[0.63636364 0.81521739]
[0.50724638 0.88235294]
[0.56451613 0.84745763]
```

In [96]: # Task 4

```
# Build decision tree with four Parameters
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin
o",
             "Percent White, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

```
Out[96]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                 max_features=None, max_leaf_nodes=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, presort=False,
                                 random_state=0, splitter='best')
```

```
In [97]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[97]: array([( 1,  90,  1,  6.00609809e-01,  0.83826453,  956,  956.),
( 2,  17,  3,  6.77590042e-01,  0.94464539,  218,  218.),
( 3,  16,  2,  2.09532775e-01,  0.49291578,  65,  65.),
( 4,  15,  2,  2.03020163e-01,  0.74551784,  33,  33.),
( 5,  6,  3,  5.34814984e-01,  0.6373875 ,  31,  31.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  13,  13.),
( 7,  14,  3,  6.10362053e-01,  0.85240518,  18,  18.),
( 8,  9,  1,  1.27966790e-01,  0.86312057,  7,  7.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
( 10, 13,  3,  5.53619564e-01,  0.65002242,  6,  6.),
( 11, 12,  1,  5.06573945e-01,  1.          ,  2,  2.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  4,  4.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  11, 11.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  2,  2.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  32, 32.),
( 18, 35,  1,  3.97715032e-01,  0.99750255,  153, 153.),
( 19, 34,  2,  8.57343748e-02,  0.69312742,  43,  43.),
( 20, 27,  0,  3.98362186e-02,  0.82381163,  31,  31.),
( 21, 22,  3,  7.55640894e-01,  0.48546076,  19, 19.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
( 23, 24,  3,  9.19624269e-01,  0.30954343,  18, 18.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  15, 15.),
( 25, 26,  2,  2.06069369e-02,  0.91829583,  3,  3.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  2,  2.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
( 28, 33,  3,  9.00250733e-01,  1.          ,  12, 12.),
( 29, 32,  3,  8.42892885e-01,  0.91829583,  9,  9.),
( 30, 31,  0,  5.26540298e-02,  0.81127812,  4,  4.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  3,  3.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  5,  5.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  3,  3.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  12, 12.),
( 36, 89,  2,  2.43311748e-01,  0.98059744,  110, 110.),
( 37, 38,  0,  1.84667693e-03,  0.99037484,  104, 104.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  6,  6.),
( 39, 58,  0,  1.60930511e-02,  0.99729438,  98,  98.),
( 40, 51,  1,  5.60719669e-01,  0.94360163,  36,  36.),
( 41, 50,  0,  1.19268922e-02,  0.68403844,  22,  22.),
( 42, 43,  0,  6.83418196e-03,  0.89049164,  13,  13.),
(-1, -1, -2, -2.00000000e+00,  0.          ,  5,  5.),
```

```
( 44, 49, 3, 9.32004809e-01, 1. , 8, 8.),  
( 45, 46, 2, 3.17055034e-02, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 47, 48, 2, 1.31500484e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( 52, 53, 1, 5.78316599e-01, 0.94028596, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 54, 57, 0, 8.33536126e-03, 0.99107606, 9, 9.),  
( 55, 56, 0, 3.40457470e-03, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 59, 66, 0, 3.55461352e-02, 0.95141225, 62, 62.),  
( 60, 61, 3, 9.05240089e-01, 0.54356444, 24, 24.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( 62, 65, 3, 9.20628935e-01, 0.954434 , 8, 8.),  
( 63, 64, 0, 2.98762042e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 67, 88, 3, 9.22450513e-01, 0.99800088, 38, 38.),  
( 68, 87, 3, 8.75249267e-01, 0.97741782, 34, 34.),  
( 69, 82, 3, 8.30632716e-01, 0.9991421 , 29, 29.),  
( 70, 79, 3, 7.54900783e-01, 0.91829583, 21, 21.),  
( 71, 76, 0, 1.47689842e-01, 0.99403021, 11, 11.),  
( 72, 73, 2, 1.80356719e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 74, 75, 2, 2.09753111e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 77, 78, 2, 3.29859294e-02, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 80, 81, 1, 4.35449436e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( 83, 86, 0, 4.97348886e-02, 0.54356444, 8, 8.),  
( 84, 85, 2, 9.50001925e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( 91, 94, 3, 2.09422484e-01, 0.63079708, 738, 738.),  
( 92, 93, 1, 9.16850775e-01, 0.37123233, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 95, 268, 0, 3.41651589e-02, 0.59370934, 724, 724.),  
( 96, 267, 2, 7.00087726e-01, 0.52407294, 677, 677.),  
( 97, 154, 1, 6.92266941e-01, 0.49675653, 670, 670.),  
( 98, 103, 0, 2.43641390e-03, 0.75247482, 139, 139.),  
( 99, 100, 3, 9.80961025e-01, 0.13303965, 54, 54.),  
( -1, -1, -2, -2.00000000e+00, 0. , 51, 51.),  
(101, 102, 1, 6.64987206e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(104, 109, 1, 6.23542339e-01, 0.92594006, 85, 85.),  
(105, 106, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(107, 108, 0, 2.02996898e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(110, 111, 0, 3.63336538e-03, 0.97741782, 68, 68.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(112, 117, 2, 7.23552890e-03, 0.954434 , 64, 64.),  
(113, 114, 0, 6.72356412e-03, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(115, 116, 3, 8.74205083e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(118, 125, 2, 1.45553285e-02, 0.92384223, 59, 59.),  
(119, 124, 3, 9.66520578e-01, 0.59167278, 14, 14.),  
(120, 121, 0, 2.07344936e-02, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(122, 123, 1, 6.38902605e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(126, 127, 1, 6.31248355e-01, 0.97095059, 45, 45.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(128, 129, 1, 6.38267547e-01, 0.94028596, 42, 42.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
```

(130, 147, 0, 1.90389156e-02, 0.98522814, 35, 35.),  
(131, 142, 1, 6.79557621e-01, 0.98522814, 21, 21.),  
(132, 141, 1, 6.74433678e-01, 0.98522814, 14, 14.),  
(133, 140, 0, 1.47164715e-02, 0.91829583, 9, 9.),  
(134, 135, 2, 1.97533704e-02, 1., , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 2, 2.),  
(136, 139, 3, 8.51981878e-01, 0.81127812, 4, 4.),  
(137, 138, 2, 1.39843173e-01, 1. , , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 5, 5.),  
(143, 144, 3, 9.26769912e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 4, 4.),  
(145, 146, 0, 1.10562695e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 2, 2.),  
(148, 153, 1, 6.58490986e-01, 0.74959526, 14, 14.),  
(149, 150, 3, 7.89511442e-01, 1. , , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 2, 2.),  
(151, 152, 2, 2.66822968e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 8, 8.),  
(155, 240, 1, 8.01030606e-01, 0.4056204 , 531, 531.),  
(156, 161, 2, 3.61989858e-03, 0.48748237, 321, 321.),  
(157, 160, 3, 4.15239334e-01, 0.12741851, 57, 57.),  
(158, 159, 3, 3.91409352e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 54, 54.),  
(162, 203, 0, 3.69854097e-03, 0.54356444, 264, 264.),  
(163, 164, 1, 7.18549222e-01, 0.78712659, 85, 85.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 15, 15.),  
(165, 174, 3, 8.92735869e-01, 0.86312057, 70, 70.),  
(166, 167, 1, 7.23192453e-01, 0.47983202, 29, 29.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 1, 1.),  
(168, 169, 3, 7.38200575e-01, 0.37123233, 28, 28.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 17, 17.),  
(170, 171, 1, 7.72475034e-01, 0.68403844, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , , 7, 7.),  
(172, 173, 0, 1.86404213e-03, 1. , , 4, 4.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(175, 180, 3, 9.17042375e-01, 0.97886985, 41, 41.),  
(176, 179, 1, 7.50855237e-01, 0.54356444, 8, 8.),  
(177, 178, 2, 6.86931051e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(181, 202, 3, 9.64229554e-01, 0.88496364, 33, 33.),  
(182, 183, 3, 9.29044455e-01, 0.95095605, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(184, 201, 0, 3.36499570e-03, 0.99403021, 22, 22.),  
(185, 186, 2, 5.92904724e-03, 0.97095059, 20, 20.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(187, 188, 2, 8.33067112e-03, 0.99107606, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(189, 194, 1, 7.44562596e-01, 0.954434 , 16, 16.),  
(190, 193, 2, 1.34000210e-02, 0.91829583, 6, 6.),  
(191, 192, 0, 9.64777922e-04, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(195, 196, 3, 9.32942390e-01, 0.72192809, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(197, 198, 3, 9.58364785e-01, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(199, 200, 0, 1.65092491e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(204, 217, 1, 7.12382376e-01, 0.37564672, 179, 179.),  
(205, 216, 1, 7.10477173e-01, 0.77934984, 26, 26.),  
(206, 215, 2, 2.98372276e-01, 0.65002242, 24, 24.),  
(207, 208, 0, 1.33669553e-02, 0.55862937, 23, 23.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(209, 214, 2, 6.04804475e-02, 0.8812909 , 10, 10.),  
(210, 211, 3, 9.29655761e-01, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(212, 213, 3, 9.55278039e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(218, 219, 3, 7.96662688e-01, 0.26806843, 153, 153.),  
( -1, -1, -2, -2.00000000e+00, 0. , 46, 46.),  
(220, 227, 3, 8.26656580e-01, 0.34859686, 107, 107.),  
(221, 226, 1, 7.71297932e-01, 0.91829583, 9, 9.),  
(222, 225, 2, 5.35712279e-02, 1. , 6, 6.),  
(223, 224, 0, 5.27075911e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(228, 235, 0, 8.13081395e-03, 0.24602258, 98, 98.),  
(229, 230, 2, 1.53482840e-02, 0.43055187, 34, 34.),  
( -1, -1, -2, -2.00000000e+00, 0. , 24, 24.),  
(231, 232, 2, 2.17032647e-02, 0.8812909 , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(233, 234, 0, 7.78159499e-03, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(236, 239, 2, 5.72627899e-03, 0.11611508, 64, 64.),  
(237, 238, 0, 1.42812165e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 62, 62.),  
(241, 260, 3, 9.72954571e-01, 0.25524211, 210, 210.),  
(242, 251, 3, 4.86974001e-01, 0.19823496, 195, 195.),  
(243, 250, 1, 8.40778619e-01, 0.56650951, 30, 30.),  
(244, 249, 1, 8.36999923e-01, 0.78712659, 17, 17.),  
(245, 248, 3, 4.72800955e-01, 0.56650951, 15, 15.),  
(246, 247, 2, 2.89359770e-03, 0.37123233, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(252, 253, 0, 6.53435197e-03, 0.09454834, 165, 165.),  
( -1, -1, -2, -2.00000000e+00, 0. , 115, 115.),  
(254, 255, 0, 6.61317166e-03, 0.24229219, 50, 50.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(256, 257, 2, 7.62819797e-02, 0.14372617, 49, 49.),  
( -1, -1, -2, -2.00000000e+00, 0. , 35, 35.),  
(258, 259, 3, 9.22998041e-01, 0.37123233, 14, 14.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  

(261, 262, 0, 3.24783765e-03, 0.72192809, 15, 15.),  

( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  

(263, 264, 3, 9.80361581e-01, 0.97095059, 5, 5.),  

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  

(265, 266, 3, 9.83373106e-01, 0.91829583, 3, 3.),  

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  

( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  

(269, 270, 3, 4.31685448e-01, 0.99967343, 47, 47.),  

( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  

(271, 272, 0, 3.51146031e-02, 0.99344724, 42, 42.),  

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  

(273, 274, 2, 4.09591515e-02, 0.97663491, 39, 39.),  

( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  

(275, 296, 0, 1.65806837e-01, 0.9993375 , 33, 33.),  

(276, 277, 3, 5.70170611e-01, 0.98713777, 30, 30.),  

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  

(278, 295, 0, 9.86937620e-02, 0.99901027, 27, 27.),  

(279, 294, 1, 7.07978636e-01, 0.99498483, 24, 24.),  

(280, 289, 1, 6.54390246e-01, 0.99836367, 21, 21.),  

(281, 282, 2, 1.28468506e-01, 0.91829583, 12, 12.),  

( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  

(283, 288, 0, 8.39187652e-02, 0.91829583, 6, 6.),  

(284, 285, 1, 6.37203485e-01, 0.72192809, 5, 5.),  

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  

(286, 287, 3, 7.32867718e-01, 1. , 2, 2.),  

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  

(290, 291, 1, 6.92071795e-01, 0.76420451, 9, 9.),  

( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  

(292, 293, 3, 8.41143191e-01, 0.91829583, 3, 3.),  

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.)],  

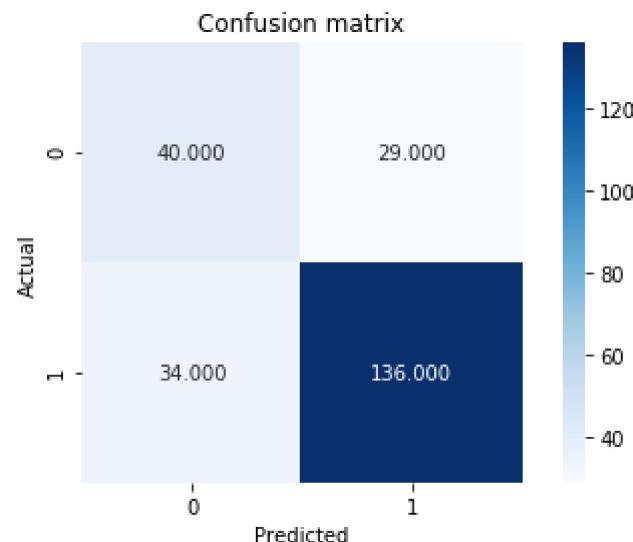
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [98]: # TASK 4  
# Predict class labels using decision tree  
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",  
                "Percent White, not Hispanic or Latino"]]  
y_pred = model.predict(x_test)
```

```
In [99]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 40  29]  
 [ 34 136]]
```

```
In [100]: # TASK 4  
# 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()
```



```
In [101]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7364016736401674  
0.2635983263598326  
[0.54054054 0.82424242]  
[0.57971014 0.8 ]  
[0.55944056 0.8119403 ]
```

```
In [102]: # Task 4  
# Build decision tree with four Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Hispanic or Latino"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[102]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [103]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[103]: array([( 1, 108, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 101, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 18, 1, 3.57535481e-01, 0.98654463, 176, 176.),
( 4, 9, 0, 3.94524951e-02, 0.63945713, 37, 37.),
( 5, 8, 3, 2.83492086e-02, 0.25801867, 23, 23.),
( 6, 7, 3, 2.72251442e-02, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),
( 10, 17, 2, 8.57343748e-02, 0.94028596, 14, 14.),
( 11, 12, 1, 1.91999584e-01, 0.99107606, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 13, 14, 3, 4.35278844e-02, 0.91829583, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 15, 16, 0, 1.02842331e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 19, 20, 0, 1.81961444e-03, 0.99996266, 139, 139.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 21, 100, 3, 3.68740737e-01, 0.9985091 , 132, 132.),
( 22, 99, 3, 2.46525131e-01, 0.99995528, 127, 127.),
( 23, 98, 2, 5.61466545e-01, 0.99825457, 122, 122.),
( 24, 97, 1, 5.97410738e-01, 0.9995415 , 119, 119.),
( 25, 42, 2, 1.53626800e-02, 0.99806926, 116, 116.),
( 26, 41, 3, 1.63817883e-01, 0.87086447, 24, 24.),
( 27, 40, 1, 5.72369903e-01, 0.77322667, 22, 22.),
( 28, 29, 3, 1.45865297e-02, 0.89603823, 16, 16.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 30, 31, 0, 7.13457563e-03, 0.97986876, 12, 12.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 32, 37, 2, 1.26421540e-02, 0.99107606, 9, 9.),
( 33, 36, 1, 4.70368236e-01, 0.72192809, 5, 5.),
( 34, 35, 1, 4.30281460e-01, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 38, 39, 1, 5.47550529e-01, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 43, 44, 3, 1.52967670e-02, 0.99863596, 92, 92.)],
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 45, 76, 1, 5.34610420e-01, 1. , 88, 88.),  
( 46, 71, 2, 1.26232438e-01, 0.96241274, 44, 44.),  
( 47, 52, 0, 1.73925012e-02, 1. , 32, 32.),  
( 48, 51, 2, 3.15953484e-02, 0.59167278, 7, 7.),  
( 49, 50, 1, 4.53624099e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 53, 70, 2, 8.42192620e-02, 0.97095059, 25, 25.),  
( 54, 69, 0, 6.11998029e-02, 0.99836367, 21, 21.),  
( 55, 56, 0, 2.17892546e-02, 0.99107606, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 57, 58, 0, 2.74492623e-02, 0.954434 , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 59, 68, 3, 9.44778658e-02, 0.99572745, 13, 13.),  
( 60, 61, 0, 3.34031247e-02, 0.99403021, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 62, 63, 0, 3.82438153e-02, 0.99107606, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 64, 65, 0, 5.03994990e-02, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 66, 67, 2, 5.02828229e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 72, 75, 3, 2.70198761e-02, 0.41381685, 12, 12.),  
( 73, 74, 2, 2.49131612e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( 77, 86, 0, 3.48667540e-02, 0.96241274, 44, 44.),  
( 78, 85, 0, 2.30477741e-02, 0.75537541, 23, 23.),  
( 79, 80, 0, 1.20450649e-02, 0.94028596, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 81, 84, 2, 6.47088140e-02, 0.99107606, 9, 9.),  
( 82, 83, 1, 5.58928132e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
```

( 87, 88, 1, 5.53604603e-01, 0.98522814, 21, 21.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 89, 96, 0, 1.05611384e-01, 0.91829583, 18, 18.),  
( 90, 91, 1, 5.73692113e-01, 0.72192809, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( 92, 93, 0, 4.52499576e-02, 0.91829583, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 94, 95, 0, 9.06764977e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(102, 107, 2, 1.46485843e-01, 0.37123233, 42, 42.),  
(103, 106, 2, 1.39678806e-01, 0.77934984, 13, 13.),  
(104, 105, 2, 2.99343411e-02, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
(109, 294, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
(110, 293, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
(111, 290, 3, 7.46519834e-01, 0.53839675, 682, 682.),  
(112, 169, 1, 6.92266941e-01, 0.51644905, 675, 675.),  
(113, 118, 0, 2.43641390e-03, 0.76276747, 140, 140.),  
(114, 117, 0, 4.52394699e-04, 0.13303965, 54, 54.),  
(115, 116, 0, 4.25896011e-04, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
(119, 124, 1, 6.23542339e-01, 0.9330253 , 86, 86.),  
(120, 121, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(122, 123, 0, 2.02996898e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(125, 126, 0, 3.63336538e-03, 0.98158862, 69, 69.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(127, 130, 2, 7.23552890e-03, 0.954434 , 64, 64.),  
(128, 129, 3, 1.05643423e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(131, 132, 3, 1.26214712e-02, 0.92384223, 59, 59.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(133, 138, 2, 1.45553285e-02, 0.89974376, 57, 57.),  
(134, 135, 0, 2.07344936e-02, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(136, 137, 3, 1.06180036e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(139, 140, 1, 6.31248355e-01, 0.96241274, 44, 44.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(141, 142, 1, 6.38267547e-01, 0.92621221, 41, 41.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(143, 162, 0, 1.90389156e-02, 0.97741782, 34, 34.),  
(144, 157, 0, 1.47164715e-02, 0.99277445, 20, 20.),  
(145, 146, 2, 1.94749255e-02, 0.98522814, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(147, 152, 1, 6.83628172e-01, 0.84535094, 11, 11.),  
(148, 149, 3, 1.10310268e-01, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(150, 151, 3, 1.40458483e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(153, 154, 0, 7.20042782e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(155, 156, 0, 1.10562695e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(158, 159, 0, 1.79855078e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(160, 161, 0, 1.82924420e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(163, 168, 1, 6.58490986e-01, 0.74959526, 14, 14.),  
(164, 167, 2, 1.06609281e-01, 1. , 6, 6.),  
(165, 166, 2, 2.66822968e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(170, 281, 3, 1.45061389e-01, 0.42925397, 535, 535.),  
(171, 172, 0, 6.47921232e-04, 0.48084469, 424, 424.),  
( -1, -1, -2, -2.00000000e+00, 0. , 23, 23.),

(173, 174, 0, 7.52224587e-04, 0.49908799, 401, 401.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(175, 250, 1, 8.01030606e-01, 0.48546076, 399, 399.),  
(176, 211, 0, 3.74499825e-03, 0.56204736, 243, 243.),  
(177, 188, 1, 7.56286770e-01, 0.74808811, 89, 89.),  
(178, 183, 2, 1.36191258e-02, 0.41381685, 48, 48.),  
(179, 182, 3, 2.53691869e-02, 0.16866093, 40, 40.),  
(180, 181, 3, 2.45874310e-02, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 25, 25.),  
(184, 185, 1, 7.28481054e-01, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(186, 187, 1, 7.41338193e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(189, 190, 3, 1.20565533e-02, 0.94743514, 41, 41.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(191, 196, 2, 4.37282701e-03, 0.98522814, 35, 35.),  
(192, 193, 1, 7.96141684e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(194, 195, 3, 3.05110142e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(197, 198, 2, 8.39653006e-03, 0.98958752, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(199, 210, 2, 1.97456181e-01, 0.96407876, 18, 18.),  
(200, 205, 0, 1.47671014e-03, 1. , 14, 14.),  
(201, 204, 3, 1.56743973e-02, 0.65002242, 6, 6.),  
(202, 203, 3, 1.31925303e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(206, 207, 0, 3.18841287e-03, 0.81127812, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(208, 209, 3, 1.37849711e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(212, 227, 1, 7.19351888e-01, 0.41754981, 154, 154.),  
(213, 214, 0, 7.15351943e-03, 0.72192809, 35, 35.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(215, 216, 0, 8.31698114e-03, 0.85545081, 25, 25.),

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(217, 226, 2, 6.46481011e-02, 0.68403844, 22, 22.),  
(218, 219, 2, 1.75233427e-02, 0.83664074, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(220, 223, 1, 7.05845952e-01, 0.99107606, 9, 9.),  
(221, 222, 0, 2.38023661e-02, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(224, 225, 0, 2.51523340e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(228, 249, 1, 7.77934521e-01, 0.28817913, 119, 119.),  
(229, 248, 1, 7.77404457e-01, 0.35910163, 88, 88.),  
(230, 239, 3, 6.27574828e-02, 0.3173239 , 87, 87.),  
(231, 232, 0, 1.12919388e-02, 0.18927843, 69, 69.),  
( -1, -1, -2, -2.00000000e+00, 0. , 46, 46.),  
(233, 238, 0, 1.29721817e-02, 0.42622866, 23, 23.),  
(234, 237, 1, 7.50546753e-01, 0.91829583, 6, 6.),  
(235, 236, 1, 7.35976428e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
(240, 241, 1, 7.52113760e-01, 0.65002242, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(242, 247, 1, 7.67549247e-01, 0.91829583, 9, 9.),  
(243, 244, 0, 8.23883805e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(245, 246, 2, 1.26900930e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 31, 31.),  
(251, 268, 2, 8.45105853e-03, 0.34351974, 156, 156.),  
(252, 267, 3, 5.63893206e-02, 0.61525389, 46, 46.),  
(253, 266, 2, 8.29666713e-03, 0.51078782, 44, 44.),  
(254, 257, 2, 1.60079834e-03, 0.44648135, 43, 43.),  
(255, 256, 0, 2.26772437e-03, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(258, 261, 0, 1.23686879e-03, 0.29181826, 39, 39.),

(259, 260, 0, 1.18657766e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(262, 265, 3, 6.93674362e-03, 0.17925607, 37, 37.),  
(263, 264, 1, 8.45317930e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 33, 33.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(269, 280, 1, 8.28199685e-01, 0.18052467, 110, 110.),  
(270, 279, 1, 8.27778816e-01, 0.33729007, 48, 48.),  
(271, 276, 2, 5.22239015e-01, 0.25387844, 47, 47.),  
(272, 273, 1, 8.16904843e-01, 0.15374218, 45, 45.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(274, 275, 1, 8.17857683e-01, 0.30954343, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
(277, 278, 2, 6.41748488e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 62, 62.),  
(282, 289, 2, 1.69041865e-02, 0.17925607, 111, 111.),  
(283, 284, 2, 9.84089263e-03, 0.52936087, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(285, 288, 1, 8.33475173e-01, 0.81127812, 12, 12.),  
(286, 287, 2, 1.02969408e-02, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 86, 86.),  
(291, 292, 0, 1.20067270e-03, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(295, 324, 0, 1.65806837e-01, 0.99729438, 49, 49.),  
(296, 323, 3, 5.12772128e-01, 0.9985091 , 44, 44.),  
(297, 312, 1, 6.54390246e-01, 0.9892453 , 41, 41.),  
(298, 311, 3, 1.39350265e-01, 0.94945202, 19, 19.),  
(299, 300, 0, 4.09484860e-02, 0.87398105, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(301, 310, 1, 6.47211939e-01, 0.99403021, 11, 11.),

```
(302, 303, 2, 4.93084919e-02, 0.99107606, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0., 1, 1.),
(304, 305, 1, 6.23185068e-01, 0.954434, 8, 8.),
(-1, -1, -2, -2.00000000e+00, 0., 2, 2.),
(306, 307, 1, 6.27582967e-01, 1., 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0., 2, 2.),
(308, 309, 3, 1.13365009e-01, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., 2, 2.),
(313, 314, 1, 6.94124609e-01, 0.84535094, 22, 22.),
(-1, -1, -2, -2.00000000e+00, 0., 10, 10.),
(315, 316, 2, 4.70557772e-02, 1., 12, 12.),
(-1, -1, -2, -2.00000000e+00, 0., 4, 4.),
(317, 322, 2, 2.61130571e-01, 0.81127812, 8, 8.),
(318, 319, 0, 3.61190718e-02, 1., 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., 1, 1.),
(320, 321, 3, 1.46320991e-01, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

In [104]:

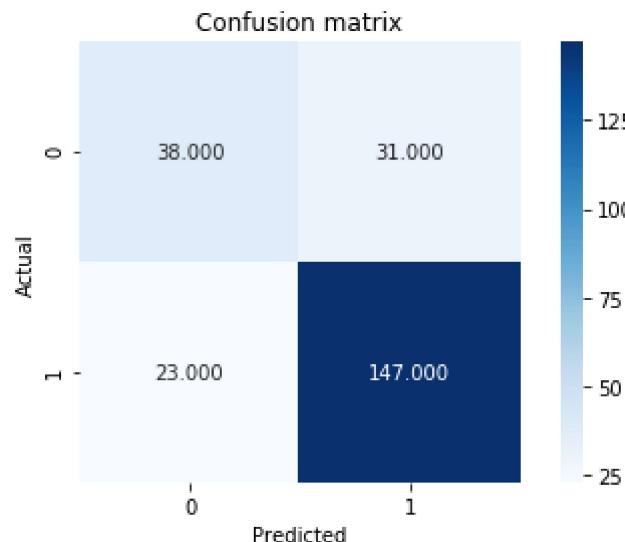
```
# TASK 4
# Predict class labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Hispanic or Latino"]]
y_pred = model.predict(x_test)
```

In [105]:

```
# TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 38  31]
 [ 23 147]]
```

```
In [106]: # TASK 4  
# 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()
```



```
In [107]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.7740585774058577  
0.2259414225941423  
[0.62295082 0.8258427 ]  
[0.55072464 0.86470588]  
[0.58461538 0.84482759]

```
In [108]: # Task 4
# Build decision tree with four Parameters
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin
o",
             "Percent Foreign Born"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

```
Out[108]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                   max_features=None, max_leaf_nodes=None,
                                   min_impurity_decrease=0.0, min_impurity_split=None,
                                   min_samples_leaf=1, min_samples_split=2,
                                   min_weight_fraction_leaf=0.0, presort=False,
                                   random_state=0, splitter='best')
```

```
In [109]: # TASK 4
# Show decision tree
model.tree_.__getstate__()['nodes']
```

```
Out[109]: array([( 1,  92,  1,  6.00609809e-01,  0.83826453,  956,  956.),
( 2,  85,  0,  1.36103377e-01,  0.94464539,  218,  218.),
( 3,  16,  3,  5.31811956e-02,  0.98654463,  176,  176.),
( 4,  11,  2,  1.66045949e-02,  0.79732651,  29,  29.),
( 5,  6,  0,  1.83359679e-03,  0.99572745,  13,  13.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  4,  4.),
( 7,  8,  1,  5.07332757e-01,  0.91829583,  9,  9.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  2,  2.),
( 9,  10,  1,  5.98042846e-01,  0.59167278,  7,  7.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  6,  6.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  1,  1.),
(12,  13,  0,  4.23690416e-02,  0.33729007,  16,  16.),
(-1, -1, -2, -2.00000000e+00,  0.,        , 12, 12.),
(14,  15,  3,  4.07223720e-02,  0.81127812,  4,  4.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  3,  3.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  1,  1.),
(17,  32,  1,  3.57535481e-01,  0.9486132 , 147, 147.),
(18,  31,  3,  1.46773122e-01,  0.63945713,  37, 37.),
(19,  20,  1,  2.00519867e-01,  0.89974376,  19, 19.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  3,  3.),
(21,  30,  0,  7.70637244e-02,  0.69621226,  16, 16.),
(22,  23,  0,  1.91903524e-02,  0.56650951,  15, 15.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  6,  6.),
(24,  25,  0,  2.08173729e-02,  0.76420451,  9,  9.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  1,  1.),
(26,  27,  2,  7.15300590e-02,  0.54356444,  8,  8.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  6,  6.),
(28,  29,  2,  2.86406785e-01,  1.        ,  2,  2.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  1,  1.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  1,  1.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  1,  1.),
(-1, -1, -2, -2.00000000e+00,  0.,        , 18, 18.),
(33,  84,  2,  3.20100173e-01,  0.98828361, 110, 110.),
(34,  83,  0,  1.24251015e-01,  0.99666573, 103, 103.),
(35,  80,  3,  2.68892780e-01,  0.99277445, 100, 100.),
(36,  79,  3,  2.28306502e-01,  0.99863596,  92,  92.),
(37,  78,  3,  1.97396681e-01,  0.99226664,  87,  87.),
(38,  61,  1,  5.34610420e-01,  0.99901027,  81,  81.),
(39,  40,  3,  6.52362593e-02,  0.9330253 ,  43,  43.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  7,  7.),
(41,  42,  3,  6.83506504e-02,  0.97986876,  36, 36.),
(-1, -1, -2, -2.00000000e+00,  0.,        ,  2,  2.),
(43,  44,  3,  8.68914314e-02,  0.95968689,  34, 34.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( 45, 60, 2, 1.26232438e-01, 0.99631652, 28, 28.),
( 46, 59, 2, 8.42192620e-02, 0.99884554, 25, 25.),
( 47, 58, 2, 5.96430153e-02, 0.98522814, 21, 21.),
( 48, 49, 0, 7.52630550e-03, 0.99750255, 17, 17.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 50, 51, 0, 1.28038297e-02, 0.97095059, 15, 15.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 52, 53, 3, 9.96176600e-02, 1. , 12, 12.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 54, 55, 3, 1.26400325e-01, 0.91829583, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 56, 57, 2, 3.80960573e-02, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 62, 77, 3, 1.63666159e-01, 0.94945202, 38, 38.),
( 63, 76, 3, 1.37628861e-01, 0.98337619, 33, 33.),
( 64, 75, 2, 2.09753111e-01, 0.94807824, 30, 30.),
( 65, 72, 0, 3.48667540e-02, 0.98958752, 25, 25.),
( 66, 71, 0, 1.66064491e-02, 0.87398105, 17, 17.),
( 67, 68, 1, 5.60719669e-01, 1. , 10, 10.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 69, 70, 0, 2.27708335e-03, 0.65002242, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 73, 74, 0, 6.33968264e-02, 0.81127812, 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 81, 82, 1, 4.18857068e-01, 0.54356444, 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 86, 91, 2, 1.46485843e-01, 0.37123233, 42, 42.),
```

( 87, 90, 2, 1.39678806e-01, 0.77934984, 13, 13.),  
( 88, 89, 2, 2.99343411e-02, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
( 93, 270, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 94, 269, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 95, 150, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 96, 101, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 97, 100, 0, 4.52394699e-04, 0.13303965, 54, 54.),  
( 98, 99, 0, 4.25896011e-04, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
(102, 107, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
(103, 104, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(105, 106, 3, 4.34439741e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(108, 109, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(110, 111, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(112, 149, 2, 3.48945007e-01, 0.92883915, 61, 61.),  
(113, 118, 0, 9.36373603e-03, 0.90657955, 59, 59.),  
(114, 115, 3, 6.09837826e-02, 0.56650951, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(116, 117, 3, 8.21332745e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(119, 148, 0, 2.77971281e-02, 0.96241274, 44, 44.),  
(120, 147, 0, 2.61669513e-02, 0.98370826, 40, 40.),  
(121, 122, 0, 9.96407261e-03, 0.95688867, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(123, 134, 2, 5.34386653e-02, 0.92752659, 35, 35.),  
(124, 129, 0, 1.19088506e-02, 0.77322667, 22, 22.),  
(125, 128, 3, 6.79037776e-02, 0.98522814, 7, 7.),  
(126, 127, 1, 6.29052788e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),

(130, 131, 3, 8.48130770e-02, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(132, 133, 0, 1.66286621e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(135, 136, 2, 7.76724592e-02, 0.99572745, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(137, 138, 2, 9.17951874e-02, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(139, 140, 2, 9.82754454e-02, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(141, 146, 1, 6.80520147e-01, 0.91829583, 6, 6.),  
(142, 143, 3, 1.66175619e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(144, 145, 3, 2.05570146e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(151, 238, 2, 3.17869987e-02, 0.45656871, 541, 541.),  
(152, 153, 0, 6.47921232e-04, 0.53722708, 334, 334.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(154, 201, 0, 3.68557358e-03, 0.56709563, 307, 307.),  
(155, 156, 1, 7.19135463e-01, 0.71886413, 131, 131.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(157, 158, 3, 8.78720824e-03, 0.78894066, 110, 110.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(159, 160, 0, 7.52224587e-04, 0.83863987, 97, 97.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(161, 162, 0, 8.57430045e-04, 0.8154225 , 95, 95.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(163, 200, 1, 8.97281468e-01, 0.84975114, 87, 87.),  
(164, 195, 1, 8.11605304e-01, 0.82496587, 85, 85.),  
(165, 180, 1, 7.70526022e-01, 0.91829583, 60, 60.),  
(166, 173, 2, 1.34000210e-02, 0.72192809, 35, 35.),  
(167, 168, 1, 7.19866216e-01, 0.38094659, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(169, 170, 1, 7.57786006e-01, 0.23519338, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
(171, 172, 1, 7.61501193e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(174, 175, 0, 2.22611579e-03, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(176, 177, 0, 2.66813650e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(178, 179, 3, 1.20002117e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(181, 194, 3, 6.82943799e-02, 0.99884554, 25, 25.),  
(182, 193, 0, 3.28301883e-03, 0.98522814, 21, 21.),  
(183, 192, 2, 1.39053124e-02, 0.91829583, 18, 18.),  
(184, 185, 1, 7.82922387e-01, 0.99572745, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(186, 191, 3, 3.41992024e-02, 0.9456603 , 11, 11.),  
(187, 188, 1, 7.97934055e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(189, 190, 1, 8.08480829e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(196, 199, 3, 1.50281880e-02, 0.40217919, 25, 25.),  
(197, 198, 1, 8.55425090e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(202, 237, 2, 3.14539718e-02, 0.42033516, 176, 176.),  
(203, 226, 1, 8.32422495e-01, 0.40217919, 175, 175.),  
(204, 225, 1, 7.78253704e-01, 0.30954343, 144, 144.),  
(205, 224, 1, 7.77404457e-01, 0.42304882, 93, 93.),  
(206, 213, 3, 5.66457137e-02, 0.38823898, 92, 92.),  
(207, 208, 0, 1.25664696e-02, 0.19590927, 66, 66.),  
( -1, -1, -2, -2.00000000e+00, 0. , 52, 52.),  
(209, 210, 0, 1.29721817e-02, 0.59167278, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(211, 212, 3, 1.98028758e-02, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(214, 217, 2, 6.63869525e-03, 0.70627409, 26, 26.),  
(215, 216, 2, 4.06386447e-03, 1. , 6, 6.),

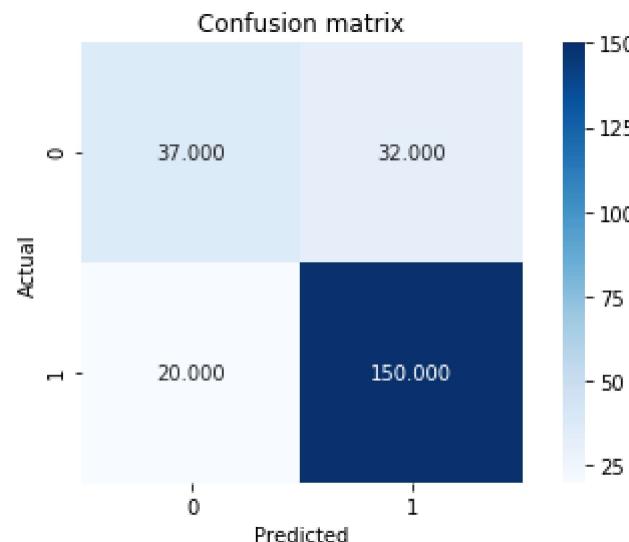
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(218, 223, 3, 7.81029537e-02, 0.46899559, 20, 20.),  
(219, 220, 2, 1.63192325e-02, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(221, 222, 0, 2.75331568e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 51, 51.),  
(227, 236, 0, 1.52357332e-02, 0.70883567, 31, 31.),  
(228, 233, 3, 9.12519805e-02, 0.57879462, 29, 29.),  
(229, 232, 3, 1.16126747e-02, 0.39124356, 26, 26.),  
(230, 231, 2, 8.79196217e-03, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
(234, 235, 2, 9.91394883e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(239, 248, 1, 7.19351888e-01, 0.299589 , 207, 207.),  
(240, 245, 2, 6.46481011e-02, 0.83664074, 15, 15.),  
(241, 242, 0, 6.75355853e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(243, 244, 1, 7.05695838e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(246, 247, 2, 2.98372276e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(249, 250, 2, 6.93185329e-02, 0.22581117, 192, 192.),  
( -1, -1, -2, -2.00000000e+00, 0. , 69, 69.),  
(251, 252, 2, 6.96306936e-02, 0.31505695, 123, 123.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(253, 268, 3, 8.77362937e-02, 0.28290479, 122, 122.),  
(254, 267, 3, 8.65705572e-02, 0.39481485, 77, 77.),  
(255, 266, 2, 1.79032527e-01, 0.35001059, 76, 76.),  
(256, 263, 2, 1.68338291e-01, 0.5349437 , 41, 41.),  
(257, 258, 3, 1.03464774e-02, 0.39845927, 38, 38.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
(259, 260, 0, 9.98797733e-03, 0.30337484, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 28, 28.),  
(261, 262, 0, 1.26664881e-02, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(264, 265, 3, 4.61300742e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 35, 35.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 45, 45.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(271, 296, 0, 1.65806837e-01, 0.99729438, 49, 49.),  
(272, 273, 0, 3.51146031e-02, 0.9985091 , 44, 44.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(274, 275, 0, 3.61851491e-02, 0.9892453 , 41, 41.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(276, 277, 0, 3.80437803e-02, 1. , 36, 36.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(278, 291, 1, 6.92403942e-01, 0.98869941, 32, 32.),  
(279, 290, 1, 6.54390246e-01, 0.90239328, 22, 22.),  
(280, 289, 3, 1.31794780e-01, 0.99572745, 13, 13.),  
(281, 288, 3, 1.10688057e-01, 0.99403021, 11, 11.),  
(282, 287, 3, 9.61573310e-02, 0.99107606, 9, 9.),  
(283, 286, 3, 8.16328749e-02, 0.98522814, 7, 7.),  
(284, 285, 0, 4.18539532e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(292, 295, 1, 7.73642480e-01, 0.8812909 , 10, 10.),  
(293, 294, 0, 1.18469346e-01, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],  
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [110]: # TASK 4  
# Predict class labels using decision tree  
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",  
                "Percent Foreign Born"]]  
y_pred = model.predict(x_test)
```

```
In [111]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)  
  
[[ 37  32]  
 [ 20 150]]
```

```
In [112]: # TASK 4  
# 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()
```



```
In [113]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7824267782426778  
0.2175732217573222  
[0.64912281 0.82417582]  
[0.53623188 0.88235294]  
[0.58730159 0.85227273]
```

```
In [114]: # Task 4  
# Build decision tree with five Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Foreign Born", "Percent Age 29 and Under"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[114]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [115]: # TASK 4
# Show decision tree
model.tree_.__getstate__()['nodes']
```

```
Out[115]: array([( 1, 88, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 81, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 16, 3, 5.31811956e-02, 0.98654463, 176, 176.),
( 4, 11, 2, 1.66045949e-02, 0.79732651, 29, 29.),
( 5, 6, 2, 6.58472651e-03, 0.99572745, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 7, 8, 1, 5.07332757e-01, 0.91829583, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 9, 10, 0, 1.45652602e-03, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 12, 13, 0, 4.23690416e-02, 0.33729007, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0., , 12, 12.),
( 14, 15, 0, 4.52499576e-02, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 17, 30, 1, 3.57535481e-01, 0.9486132 , 147, 147.),
( 18, 29, 3, 1.46773122e-01, 0.63945713, 37, 37.),
( 19, 20, 1, 2.00519867e-01, 0.89974376, 19, 19.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 21, 28, 4, 4.37841311e-01, 0.69621226, 16, 16.),
( 22, 23, 4, 4.04855132e-01, 0.91829583, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0., , 5, 5.),
( 24, 25, 3, 1.07218649e-01, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 26, 27, 1, 3.18483472e-01, 1. , , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 18, 18.),
( 31, 80, 2, 3.20100173e-01, 0.98828361, 110, 110.),
( 32, 79, 0, 1.24251015e-01, 0.99666573, 103, 103.),
( 33, 76, 3, 2.68892780e-01, 0.99277445, 100, 100.),
( 34, 75, 3, 2.28306502e-01, 0.99863596, 92, 92.),
( 35, 74, 3, 1.97396681e-01, 0.99226664, 87, 87.),
( 36, 57, 1, 5.34610420e-01, 0.99901027, 81, 81.),
( 37, 38, 3, 6.52362593e-02, 0.9330253 , 43, 43.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 39, 56, 4, 6.45427883e-01, 0.97986876, 36, 36.),
( 40, 41, 3, 6.83506504e-02, 0.99924925, 31, 31.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 42, 55, 2, 1.26232438e-01, 0.99226664, 29, 29.),
( 43, 54, 2, 8.42192620e-02, 0.99884554, 25, 25.),
```

( 44, 45, 4, 4.11624655e-01, 0.98522814, 21, 21.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 46, 47, 3, 8.45479891e-02, 0.98869941, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 48, 51, 3, 1.23242423e-01, 0.94028596, 14, 14.),  
( 49, 50, 4, 4.23987955e-01, 0.72192809, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( 52, 53, 2, 3.80960573e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 58, 73, 3, 1.63666159e-01, 0.94945202, 38, 38.),  
( 59, 72, 3, 1.37628861e-01, 0.98337619, 33, 33.),  
( 60, 71, 2, 2.09753111e-01, 0.94807824, 30, 30.),  
( 61, 68, 0, 3.48667540e-02, 0.98958752, 25, 25.),  
( 62, 67, 0, 1.66064491e-02, 0.87398105, 17, 17.),  
( 63, 64, 1, 5.60719669e-01, 1. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 65, 66, 0, 2.27708335e-03, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 69, 70, 0, 6.33968264e-02, 0.81127812, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 77, 78, 0, 2.01796001e-03, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 82, 87, 2, 1.46485843e-01, 0.37123233, 42, 42.),  
( 83, 84, 4, 5.19498080e-01, 0.77934984, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( 85, 86, 0, 3.32763255e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
( 89, 260, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 90, 259, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 91, 140, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 92, 97, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 93, 96, 4, 3.44815820e-01, 0.13303965, 54, 54.),  
( 94, 95, 4, 3.36660802e-01, 0.33729007, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 38, 38.),  
( 98, 103, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
( 99, 100, 4, 4.96087506e-01, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(101, 102, 3, 4.34439741e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(104, 105, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(106, 107, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(108, 139, 2, 3.48945007e-01, 0.92883915, 61, 61.),  
(109, 114, 0, 9.36373603e-03, 0.90657955, 59, 59.),  
(110, 111, 3, 6.09837826e-02, 0.56650951, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(112, 113, 3, 8.21332745e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(115, 138, 0, 2.77971281e-02, 0.96241274, 44, 44.),  
(116, 137, 0, 2.61669513e-02, 0.98370826, 40, 40.),  
(117, 118, 0, 9.96407261e-03, 0.95688867, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(119, 128, 2, 5.34386653e-02, 0.92752659, 35, 35.),  
(120, 125, 0, 1.19088506e-02, 0.77322667, 22, 22.),  
(121, 124, 3, 6.79037776e-02, 0.98522814, 7, 7.),  
(122, 123, 1, 6.29052788e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(126, 127, 4, 3.43762815e-01, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
(129, 130, 2, 7.76724592e-02, 0.99572745, 13, 13.),

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(131, 132, 0, 1.40508474e-02, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(133, 134, 2, 9.17951874e-02, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(135, 136, 4, 3.95526633e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(141, 252, 4, 6.66672230e-01, 0.45656871, 541, 541.),  
(142, 147, 2, 3.61989858e-03, 0.4326091 , 529, 529.),  
(143, 144, 1, 8.50911796e-01, 0.10559104, 72, 72.),  
( -1, -1, -2, -2.00000000e+00, 0. , 62, 62.),  
(145, 146, 1, 8.54703933e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(148, 211, 2, 2.82052122e-02, 0.47107306, 457, 457.),  
(149, 180, 0, 3.68557358e-03, 0.58705723, 241, 241.),  
(150, 179, 2, 2.66378755e-02, 0.76633504, 94, 94.),  
(151, 152, 1, 7.18549222e-01, 0.71751065, 91, 91.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
(153, 174, 1, 8.04456413e-01, 0.78451914, 77, 77.),  
(154, 173, 4, 5.90004027e-01, 0.93211157, 46, 46.),  
(155, 160, 1, 7.55806565e-01, 0.90239328, 44, 44.),  
(156, 159, 4, 3.55390757e-01, 0.66657836, 23, 23.),  
(157, 158, 2, 6.86931051e-03, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
(161, 164, 2, 8.14763736e-03, 0.99836367, 21, 21.),  
(162, 163, 1, 8.01848620e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(165, 166, 2, 9.90183325e-03, 0.86312057, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(167, 170, 2, 1.39053124e-02, 0.97095059, 10, 10.),  
(168, 169, 0, 9.69062821e-04, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(171, 172, 1, 7.59539276e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(175, 176, 4, 5.35265326e-01, 0.34511731, 31, 31.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(177, 178, 3, 1.94947638e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(181, 198, 3, 5.70345093e-02, 0.43122562, 147, 147.),  
(182, 193, 1, 8.60816181e-01, 0.29941133, 113, 113.),  
(183, 192, 4, 4.12703678e-01, 0.22693864, 109, 109.),  
(184, 185, 0, 7.13265873e-03, 0.45371634, 42, 42.),  
( -1, -1, -2, -2.00000000e+00, 0. , 25, 25.),  
(186, 187, 2, 5.66655444e-03, 0.78712659, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(188, 189, 2, 1.52147859e-02, 0.56650951, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(190, 191, 3, 2.32856590e-02, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 67, 67.),  
(194, 195, 2, 6.82448759e-03, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(196, 197, 4, 4.06992137e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(199, 208, 0, 9.00628557e-03, 0.73353793, 34, 34.),  
(200, 201, 0, 4.61133639e-03, 0.97095059, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(202, 203, 4, 4.20015097e-01, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(204, 205, 4, 4.79100510e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(206, 207, 2, 1.76622691e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(209, 210, 3, 6.34888560e-02, 0.29747225, 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 18, 18.),  
(212, 221, 1, 7.19351888e-01, 0.30954343, 216, 216.),  
(213, 220, 0, 1.68396370e-02, 0.76420451, 18, 18.),  
(214, 217, 2, 7.15110246e-02, 0.9456603 , 11, 11.),  
(215, 216, 4, 3.53202343e-01, 0.97095059, 5, 5.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(218, 219, 2, 2.85567433e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(222, 249, 3, 4.50359598e-01, 0.24414164, 198, 198.),  
(223, 248, 3, 8.77362937e-02, 0.22315499, 195, 195.),  
(224, 247, 3, 8.65705572e-02, 0.31319505, 124, 124.),  
(225, 246, 1, 8.28835458e-01, 0.2811938 , 123, 123.),  
(226, 245, 1, 8.27664375e-01, 0.37440885, 83, 83.),  
(227, 244, 4, 4.53428283e-01, 0.33130561, 82, 82.),  
(228, 241, 4, 4.51850533e-01, 0.42881096, 57, 57.),  
(229, 240, 0, 6.77960599e-03, 0.30954343, 54, 54.),  
(230, 239, 0, 6.53435197e-03, 0.50325833, 27, 27.),  
(231, 232, 2, 8.40812661e-02, 0.39124356, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(233, 234, 2, 9.80773047e-02, 0.56650951, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(235, 236, 4, 4.39260408e-01, 0.37123233, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(237, 238, 1, 7.88627088e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(242, 243, 0, 6.72018132e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 40, 40.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 71, 71.),  
(250, 251, 3, 4.96478334e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(253, 254, 2, 1.24893506e-03, 0.97986876, 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(255, 256, 0, 1.67020701e-03, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(257, 258, 4, 6.69304490e-01, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

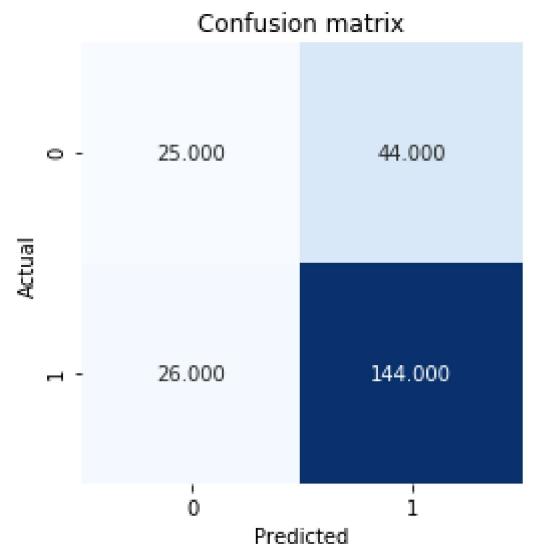
```
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(261, 262, 4, 3.91824558e-01, 0.99729438, 49, 49.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(263, 286, 3, 3.54934916e-01, 0.96816473, 43, 43.),
(264, 285, 4, 6.12080812e-01, 0.9919924 , 38, 38.),
(265, 280, 3, 1.17938362e-01, 0.9456603 , 33, 33.),
(266, 267, 1, 6.13993555e-01, 0.99863596, 23, 23.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(268, 277, 2, 1.70044117e-01, 0.99277445, 20, 20.),
(269, 272, 1, 6.54390246e-01, 0.86312057, 14, 14.),
(270, 271, 4, 4.83711615e-01, 0.97095059, 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(273, 274, 4, 4.79425743e-01, 0.50325833, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(275, 276, 1, 7.24720985e-01, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(278, 279, 2, 3.32290724e-01, 0.65002242, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(281, 282, 1, 7.18099475e-01, 0.46899559, 10, 10.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
(283, 284, 3, 1.96370035e-01, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [116]: # TASK 4
# Predict class labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Foreign Born", "Percent Age 29 and Under"]]
y_pred = model.predict(x_test)
```

```
In [117]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 25  44]  
 [ 26 144]]
```

```
In [118]: # TASK 4  
# 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()
```



```
In [119]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7071129707112971  
0.2928870292887029  
[0.49019608 0.76595745]  
[0.36231884 0.84705882]  
[0.41666667 0.80446927]
```

```
In [120]: # Task 4  
# Build decision tree with five Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Foreign Born", "Percent Age 65 and Older"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[120]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [121]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[121]: array([( 1, 86, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 81, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 16, 3, 5.31811956e-02, 0.98654463, 176, 176.),
( 4, 11, 2, 1.66045949e-02, 0.79732651, 29, 29.),
( 5, 6, 2, 6.58472651e-03, 0.99572745, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 7, 8, 1, 5.07332757e-01, 0.91829583, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 9, 10, 0, 1.45652602e-03, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 12, 13, 0, 4.23690416e-02, 0.33729007, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0., , 12, 12.),
( 14, 15, 0, 4.52499576e-02, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 17, 28, 1, 3.57535481e-01, 0.9486132 , 147, 147.),
( 18, 27, 3, 1.46773122e-01, 0.63945713, 37, 37.),
( 19, 20, 1, 2.00519867e-01, 0.89974376, 19, 19.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 21, 22, 4, 3.15014586e-01, 0.69621226, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 23, 26, 4, 3.86670068e-01, 0.91829583, 9, 9.),
( 24, 25, 3, 8.54628347e-02, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., , 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0., , 18, 18.),
( 29, 80, 2, 3.20100173e-01, 0.98828361, 110, 110.),
( 30, 77, 4, 6.01501554e-01, 0.99666573, 103, 103.),
( 31, 76, 3, 2.68892780e-01, 0.98644974, 95, 95.),
( 32, 75, 0, 1.24251015e-01, 0.99664403, 88, 88.),
( 33, 74, 3, 2.35165469e-01, 0.99189774, 85, 85.),
( 34, 73, 3, 1.95849225e-01, 0.98449613, 82, 82.),
( 35, 56, 1, 5.34610420e-01, 0.99549938, 76, 76.),
( 36, 37, 3, 6.52362593e-02, 0.92621221, 41, 41.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 38, 39, 3, 6.83506504e-02, 0.97741782, 34, 34.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 40, 41, 3, 8.68914314e-02, 0.954434 , 32, 32.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 42, 53, 3, 1.23242423e-01, 0.99572745, 26, 26.),
( 43, 52, 3, 1.12074453e-01, 0.954434 , 16, 16.)])
```

```
( 44, 45, 3, 9.16430429e-02, 0.99403021, 11, 11.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 46, 47, 3, 9.96176600e-02, 0.91829583, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 48, 51, 1, 4.57648963e-01, 0.97095059, 5, 5.),
( 49, 50, 4, 1.26354679e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 54, 55, 0, 3.34031247e-02, 0.72192809, 10, 10.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( 57, 64, 1, 5.70503086e-01, 0.97095059, 35, 35.),
( 58, 63, 1, 5.59382826e-01, 0.79185835, 21, 21.),
( 59, 60, 0, 1.93999745e-02, 0.9612366 , 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 61, 62, 1, 5.53604603e-01, 0.50325833, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( 65, 66, 4, 1.66006796e-01, 0.94028596, 14, 14.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 67, 72, 0, 3.21643390e-02, 0.81127812, 12, 12.),
( 68, 69, 3, 8.96081030e-02, 0.97095059, 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 70, 71, 1, 5.84798962e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 78, 79, 1, 3.97432268e-01, 0.54356444, 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 82, 85, 4, 1.07068963e-01, 0.37123233, 42, 42.),
( 83, 84, 3, 4.08240288e-01, 0.97095059, 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),
```

( 87, 260, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 88, 259, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 89, 136, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 90, 95, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 91, 94, 0, 4.52394699e-04, 0.13303965, 54, 54.),  
( 92, 93, 0, 4.25896011e-04, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
( 96, 101, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
( 97, 98, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
( 99, 100, 2, 6.22235965e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(102, 103, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(104, 105, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(106, 107, 4, 2.33378634e-01, 0.92883915, 61, 61.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(108, 111, 4, 2.78598204e-01, 0.954434 , 56, 56.),  
(109, 110, 3, 1.23745963e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(112, 123, 2, 4.51461952e-02, 0.90438146, 50, 50.),  
(113, 122, 2, 2.27405103e-02, 0.72192809, 30, 30.),  
(114, 115, 3, 3.08582708e-02, 0.84535094, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(116, 117, 3, 3.18981204e-02, 0.93666738, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(118, 119, 4, 3.76321495e-01, 0.83664074, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(120, 121, 3, 5.73114511e-02, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(124, 125, 1, 6.33328557e-01, 1. , 20, 20.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(126, 127, 1, 6.56217486e-01, 0.99107606, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(128, 133, 0, 1.92262111e-02, 0.9612366 , 13, 13.),  
(129, 132, 1, 6.79418623e-01, 0.76420451, 9, 9.),

(130, 131, 1, 6.74068540e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(134, 135, 1, 6.57565951e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(137, 228, 2, 3.17869987e-02, 0.45656871, 541, 541.),  
(138, 139, 0, 6.47921232e-04, 0.53722708, 334, 334.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(140, 189, 0, 3.68557358e-03, 0.56709563, 307, 307.),  
(141, 142, 1, 7.19135463e-01, 0.71886413, 131, 131.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(143, 144, 3, 8.78720824e-03, 0.78894066, 110, 110.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(145, 176, 4, 4.90405738e-01, 0.83863987, 97, 97.),  
(146, 175, 2, 2.66378755e-02, 0.93406806, 60, 60.),  
(147, 152, 1, 7.56286770e-01, 0.89974376, 57, 57.),  
(148, 151, 4, 4.87013131e-01, 0.52255937, 17, 17.),  
(149, 150, 1, 7.24217474e-01, 0.33729007, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(153, 166, 1, 8.14393073e-01, 0.97095059, 40, 40.),  
(154, 155, 0, 8.82462831e-04, 0.94945202, 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(156, 165, 3, 5.18329125e-02, 0.87398105, 17, 17.),  
(157, 164, 3, 3.84338573e-02, 0.97986876, 12, 12.),  
(158, 163, 3, 2.81428462e-02, 0.8812909 , 10, 10.),  
(159, 160, 1, 7.86800295e-01, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(161, 162, 4, 4.00892273e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(167, 168, 2, 1.50427764e-03, 0.70246655, 21, 21.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(169, 174, 4, 3.60941395e-01, 0.48546076, 19, 19.),  
(170, 171, 4, 3.15899879e-01, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(172, 173, 3, 1.25708785e-02, 0.91829583, 3, 3.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(177, 178, 0, 7.52224587e-04, 0.57135497, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(179, 188, 0, 3.38416500e-03, 0.42200052, 35, 35.),  
(180, 181, 2, 1.03808944e-02, 0.32275696, 34, 34.),  
( -1, -1, -2, -2.00000000e+00, 0. , 22, 22.),  
(182, 183, 2, 1.13613713e-02, 0.65002242, 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(184, 185, 4, 5.84357381e-01, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(186, 187, 4, 6.07921422e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(190, 227, 2, 3.14539718e-02, 0.42033516, 176, 176.),  
(191, 216, 1, 8.32422495e-01, 0.40217919, 175, 175.),  
(192, 215, 1, 7.78253704e-01, 0.30954343, 144, 144.),  
(193, 210, 4, 4.72428054e-01, 0.42304882, 93, 93.),  
(194, 209, 1, 7.77404457e-01, 0.32555171, 84, 84.),  
(195, 200, 0, 1.25794369e-02, 0.2786698 , 83, 83.),  
(196, 197, 3, 9.82759632e-02, 0.1132743 , 66, 66.),  
( -1, -1, -2, -2.00000000e+00, 0. , 56, 56.),  
(198, 199, 3, 1.01544295e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(201, 202, 0, 1.33290682e-02, 0.67229482, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(203, 208, 1, 7.11101323e-01, 0.54356444, 16, 16.),  
(204, 207, 2, 2.68256785e-02, 0.97095059, 5, 5.),  
(205, 206, 0, 1.80113306e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(211, 212, 3, 6.11308645e-02, 0.91829583, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(213, 214, 0, 9.57922102e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

( -1, -1, -2, -2.00000000e+00, 0. , 51, 51.),  
(217, 226, 0, 1.52357332e-02, 0.70883567, 31, 31.),  
(218, 223, 3, 9.12519805e-02, 0.57879462, 29, 29.),  
(219, 222, 3, 1.16126747e-02, 0.39124356, 26, 26.),  
(220, 221, 2, 8.79196217e-03, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
(224, 225, 4, 6.96732193e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(229, 238, 1, 7.19351888e-01, 0.299589 , 207, 207.),  
(230, 231, 4, 3.35660636e-01, 0.83664074, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(232, 235, 4, 4.20528665e-01, 0.99107606, 9, 9.),  
(233, 234, 3, 3.63720739e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(236, 237, 1, 7.16349602e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(239, 240, 2, 6.93185329e-02, 0.22581117, 192, 192.),  
( -1, -1, -2, -2.00000000e+00, 0. , 69, 69.),  
(241, 242, 2, 6.96306936e-02, 0.31505695, 123, 123.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(243, 258, 3, 8.77362937e-02, 0.28290479, 122, 122.),  
(244, 257, 3, 8.65705572e-02, 0.39481485, 77, 77.),  
(245, 256, 2, 1.79032527e-01, 0.35001059, 76, 76.),  
(246, 253, 2, 1.68338291e-01, 0.5349437 , 41, 41.),  
(247, 248, 3, 1.03464774e-02, 0.39845927, 38, 38.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(249, 250, 0, 9.98797733e-03, 0.30337484, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 28, 28.),  
(251, 252, 0, 1.26664881e-02, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(254, 255, 3, 4.61300742e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 35, 35.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 45, 45.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(261, 284, 4, 3.94287720e-01, 0.99729438, 49, 49.),
(262, 283, 3, 3.54934916e-01, 0.954434 , 40, 40.),
(263, 264, 4, 1.37885422e-01, 0.98522814, 35, 35.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(265, 274, 1, 6.54390246e-01, 0.954434 , 32, 32.),
(266, 273, 3, 9.39340778e-02, 0.69621226, 16, 16.),
(267, 272, 3, 8.16515274e-02, 0.954434 , 8, 8.),
(268, 271, 3, 5.12932688e-02, 0.65002242, 6, 6.),
(269, 270, 2, 1.12720162e-01, 1. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
(275, 282, 2, 2.57564157e-01, 0.98869941, 16, 16.),
(276, 277, 1, 6.86853975e-01, 0.81127812, 12, 12.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(278, 279, 0, 3.61851491e-02, 0.98522814, 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(280, 281, 4, 2.00991958e-01, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(285, 286, 3, 2.51908146e-01, 0.50325833, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

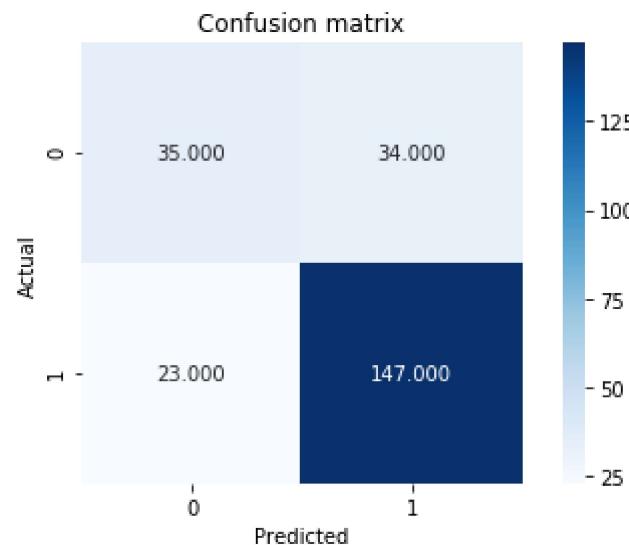
In [122]:

```
# TASK 4
# Predict class labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Foreign Born", "Percent Age 65 and Older"]]
y_pred = model.predict(x_test)
```

```
In [123]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 35  34]  
 [ 23 147]]
```

```
In [124]: # TASK 4  
# 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()
```



```
In [125]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7615062761506276  
0.2384937238493724  
[0.60344828 0.8121547 ]  
[0.50724638 0.86470588]  
[0.5511811 0.83760684]
```

```
In [126]: # Task 4  
# Build decision tree with five Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Foreign Born", "Percent Female"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[126]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [127]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[127]: array([( 1, 88, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 81, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 14, 3, 5.31811956e-02, 0.98654463, 176, 176.),
( 4, 11, 2, 1.66045949e-02, 0.79732651, 29, 29.),
( 5, 6, 2, 6.58472651e-03, 0.99572745, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 7, 8, 1, 5.07332757e-01, 0.91829583, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 9, 10, 0, 1.45652602e-03, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 12, 13, 4, 7.00245440e-01, 0.33729007, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 15, 15.),
( 15, 28, 1, 3.57535481e-01, 0.9486132 , 147, 147.),
( 16, 27, 3, 1.46773122e-01, 0.63945713, 37, 37.),
( 17, 18, 1, 2.00519867e-01, 0.89974376, 19, 19.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 19, 26, 0, 7.70637244e-02, 0.69621226, 16, 16.),
( 20, 21, 4, 7.52493203e-01, 0.56650951, 15, 15.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 22, 25, 4, 7.76990801e-01, 0.81127812, 8, 8.),
( 23, 24, 0, 5.71034420e-02, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 18, 18.),
( 29, 80, 2, 3.20100173e-01, 0.98828361, 110, 110.),
( 30, 79, 0, 1.24251015e-01, 0.99666573, 103, 103.),
( 31, 76, 3, 2.68892780e-01, 0.99277445, 100, 100.),
( 32, 75, 3, 2.28306502e-01, 0.99863596, 92, 92.),
( 33, 74, 3, 1.97396681e-01, 0.99226664, 87, 87.),
( 34, 57, 1, 5.34610420e-01, 0.99901027, 81, 81.),
( 35, 36, 3, 6.52362593e-02, 0.9330253 , 43, 43.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 37, 38, 3, 6.83506504e-02, 0.97986876, 36, 36.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 39, 40, 3, 8.68914314e-02, 0.95968689, 34, 34.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 41, 56, 2, 1.26232438e-01, 0.99631652, 28, 28.),
( 42, 55, 2, 8.42192620e-02, 0.99884554, 25, 25.),
( 43, 54, 2, 5.96430153e-02, 0.98522814, 21, 21.),
```

( 44, 45, 0, 7.52630550e-03, 0.99750255, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 46, 47, 0, 1.28038297e-02, 0.97095059, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 48, 49, 3, 9.96176600e-02, 1. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 50, 51, 3, 1.26400325e-01, 0.91829583, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 52, 53, 2, 3.80960573e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 58, 73, 3, 1.63666159e-01, 0.94945202, 38, 38.),  
( 59, 60, 4, 6.92934692e-01, 0.98337619, 33, 33.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 61, 68, 0, 3.48667540e-02, 0.94807824, 30, 30.),  
( 62, 67, 1, 5.59953392e-01, 0.74248757, 19, 19.),  
( 63, 66, 0, 1.98036572e-02, 0.99107606, 9, 9.),  
( 64, 65, 4, 7.56777704e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( 69, 70, 0, 6.33825064e-02, 0.9456603 , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( 71, 72, 4, 7.20173270e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 77, 78, 0, 2.01796001e-03, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 82, 87, 2, 1.46485843e-01, 0.37123233, 42, 42.),  
( 83, 86, 2, 1.39678806e-01, 0.77934984, 13, 13.),  
( 84, 85, 2, 2.99343411e-02, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
( 89, 258, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 90, 257, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 91, 140, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 92, 97, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 93, 96, 4, 6.19105488e-01, 0.13303965, 54, 54.),  
( 94, 95, 4, 6.05720460e-01, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 45, 45.),  
( 98, 103, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
( 99, 102, 4, 6.98251128e-01, 0.32275696, 17, 17.),  
(100, 101, 0, 1.61726596e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
(104, 105, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(106, 107, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(108, 139, 2, 3.48945007e-01, 0.92883915, 61, 61.),  
(109, 114, 0, 9.36373603e-03, 0.90657955, 59, 59.),  
(110, 111, 3, 6.09837826e-02, 0.56650951, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(112, 113, 3, 8.21332745e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(115, 138, 0, 2.77971281e-02, 0.96241274, 44, 44.),  
(116, 137, 0, 2.61669513e-02, 0.98370826, 40, 40.),  
(117, 118, 0, 9.96407261e-03, 0.95688867, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(119, 124, 4, 6.78834915e-01, 0.92752659, 35, 35.),  
(120, 123, 4, 6.41442597e-01, 0.954434 , 8, 8.),  
(121, 122, 1, 6.89204097e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(125, 132, 4, 7.46543765e-01, 0.82562653, 27, 27.),  
(126, 131, 2, 1.97533704e-02, 0.59167278, 21, 21.),  
(127, 128, 0, 1.10642756e-02, 1. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(129, 130, 3, 7.45505616e-02, 0.81127812, 4, 4.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
(133, 134, 2, 1.83991017e-02, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(135, 136, 3, 1.45137552e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(141, 228, 2, 3.17869987e-02, 0.45656871, 541, 541.),  
(142, 143, 0, 6.47921232e-04, 0.53722708, 334, 334.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(144, 189, 0, 3.68557358e-03, 0.56709563, 307, 307.),  
(145, 146, 1, 7.19135463e-01, 0.71886413, 131, 131.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(147, 148, 3, 8.78720824e-03, 0.78894066, 110, 110.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(149, 150, 0, 7.52224587e-04, 0.83863987, 97, 97.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(151, 188, 4, 7.47752547e-01, 0.8154225 , 95, 95.),  
(152, 187, 4, 7.44633824e-01, 0.84975114, 87, 87.),  
(153, 178, 1, 8.04456413e-01, 0.81127812, 84, 84.),  
(154, 159, 2, 4.51100431e-03, 0.90592822, 56, 56.),  
(155, 156, 3, 1.12789925e-02, 0.48546076, 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(157, 158, 1, 7.25176722e-01, 0.30954343, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
(160, 165, 1, 7.55806565e-01, 0.98678672, 37, 37.),  
(161, 162, 2, 1.34000210e-02, 0.72192809, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(163, 164, 1, 7.41338193e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(166, 169, 2, 8.14763736e-03, 0.97602065, 22, 22.),  
(167, 168, 1, 8.01848620e-01, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(170, 177, 4, 7.09334016e-01, 0.98522814, 14, 14.),  
(171, 176, 2, 1.87576124e-02, 0.97095059, 10, 10.),  
(172, 173, 4, 6.85288489e-01, 0.91829583, 6, 6.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(174, 175, 1, 7.92814821e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(179, 184, 1, 8.97281468e-01, 0.49123734, 28, 28.),  
(180, 183, 3, 1.50281880e-02, 0.24229219, 25, 25.),  
(181, 182, 3, 1.41094243e-02, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
(185, 186, 1, 9.16850775e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(190, 225, 4, 7.71884561e-01, 0.42033516, 176, 176.),  
(191, 208, 3, 5.70345093e-02, 0.36657801, 171, 171.),  
(192, 193, 4, 5.86997569e-01, 0.23939653, 127, 127.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(194, 195, 3, 5.70050674e-03, 0.20307393, 126, 126.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(196, 197, 4, 7.30844617e-01, 0.16334554, 125, 125.),  
( -1, -1, -2, -2.00000000e+00, 0. , 77, 77.),  
(198, 199, 4, 7.31151015e-01, 0.33729007, 48, 48.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(200, 205, 1, 8.58730733e-01, 0.25387844, 47, 47.),  
(201, 204, 3, 1.80760315e-02, 0.15649106, 44, 44.),  
(202, 203, 3, 1.78398192e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 38, 38.),  
(206, 207, 4, 7.44878232e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(209, 222, 0, 9.00628557e-03, 0.63213028, 44, 44.),  
(210, 211, 0, 4.61133639e-03, 0.84535094, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(212, 221, 1, 8.31766486e-01, 0.97095059, 15, 15.),  
(213, 214, 4, 6.97286874e-01, 0.89049164, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(215, 220, 3, 1.17309518e-01, 0.99107606, 9, 9.),

(216, 217, 3, 6.92887194e-02, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(218, 219, 1, 7.76076615e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(223, 224, 3, 6.34888560e-02, 0.26676499, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(226, 227, 0, 8.29544407e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(229, 256, 4, 8.57881814e-01, 0.299589 , 207, 207.),  
(230, 247, 1, 7.53632456e-01, 0.25990715, 205, 205.),  
(231, 246, 2, 4.10442799e-01, 0.62924922, 38, 38.),  
(232, 245, 1, 7.52113760e-01, 0.57135497, 37, 37.),  
(233, 244, 2, 8.88497867e-02, 0.50325833, 36, 36.),  
(234, 235, 2, 3.87805123e-02, 0.83664074, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(236, 243, 4, 7.63693273e-01, 0.9456603 , 11, 11.),  
(237, 238, 1, 7.05695838e-01, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(239, 242, 0, 8.68604612e-03, 0.91829583, 6, 6.),  
(240, 241, 3, 2.26112735e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(248, 251, 4, 2.73002967e-01, 0.1298515 , 167, 167.),  
(249, 250, 4, 2.29455657e-01, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(252, 255, 3, 1.37905986e-02, 0.05475066, 160, 160.),  
(253, 254, 0, 8.23060656e-03, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 147, 147.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),

```
(259, 288, 3, 3.54934916e-01, 0.99729438, 49, 49.),
(260, 287, 4, 7.84744412e-01, 0.9985091 , 44, 44.),
(261, 262, 0, 3.51146031e-02, 0.98370826, 40, 40.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(263, 264, 4, 7.18230367e-01, 0.95688867, 37, 37.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(265, 286, 4, 7.79635668e-01, 0.99679163, 30, 30.),
(266, 275, 2, 8.94560851e-02, 0.99901027, 27, 27.),
(267, 274, 4, 7.50391364e-01, 0.8812909 , 10, 10.),
(268, 269, 3, 7.23308101e-02, 1. , 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(270, 273, 2, 5.37061337e-02, 0.81127812, 4, 4.),
(271, 272, 3, 9.22344588e-02, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(276, 277, 2, 1.14497282e-01, 0.93666738, 17, 17.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(278, 279, 2, 1.68407619e-01, 0.99572745, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(280, 281, 0, 4.58115041e-02, 0.9456603 , 11, 11.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(282, 285, 3, 1.95761397e-01, 0.98522814, 7, 7.),
(283, 284, 3, 7.90550672e-02, 0.72192809, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

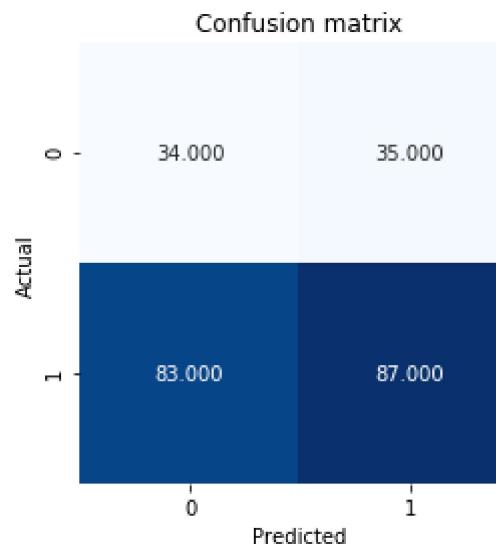
In [128]:

```
# TASK 4
# Predict class Labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Foreign Born", "Percent Female"]]
y_pred = model.predict(x_test)
```

```
In [129]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[34 35]  
 [83 87]]
```

```
In [130]: # TASK 4  
# 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()
```



```
In [131]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.5062761506276151  
0.4937238493723849  
[0.29059829 0.71311475]  
[0.49275362 0.51176471]  
[0.3655914 0.59589041]
```

```
In [132]: # Task 4  
# Build decision tree with five Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Foreign Born", "Median Household Income"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[132]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [133]: # TASK 4
# Show decision tree
model.tree_.__getstate__()['nodes']
```

```
Out[133]: array([( 1, 98, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 91, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 14, 3, 5.31811956e-02, 0.98654463, 176, 176.),
( 4, 11, 4, 2.95666456e-01, 0.79732651, 29, 29.),
( 5, 6, 0, 1.99247617e-03, 0.99572745, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 7, 10, 3, 4.04023752e-02, 0.91829583, 9, 9.),
( 8, 9, 3, 3.68007608e-02, 0.97095059, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 12, 13, 3, 2.90864659e-02, 0.33729007, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 15, 15.),
( 15, 28, 1, 3.57535481e-01, 0.9486132 , 147, 147.),
( 16, 17, 4, 4.27379221e-01, 0.63945713, 37, 37.),
(-1, -1, -2, -2.00000000e+00, 0., , 20, 20.),
( 18, 27, 3, 1.59773953e-01, 0.93666738, 17, 17.),
( 19, 22, 0, 3.55379051e-02, 0.99403021, 11, 11.),
( 20, 21, 2, 2.14165654e-02, 0.72192809, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 23, 24, 1, 3.33328024e-01, 0.65002242, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 25, 26, 3, 1.09335270e-01, 1. , , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 29, 90, 2, 3.20100173e-01, 0.98828361, 110, 110.),
( 30, 73, 4, 3.84038880e-01, 0.99666573, 103, 103.),
( 31, 32, 3, 6.01789672e-02, 0.96051871, 73, 73.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 33, 34, 4, 1.46136597e-01, 0.97424083, 69, 69.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 35, 54, 1, 5.34724742e-01, 0.98337619, 66, 66.),
( 36, 39, 4, 2.11021721e-01, 0.89805879, 35, 35.),
( 37, 38, 3, 2.85547540e-01, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
( 40, 41, 3, 8.68914314e-02, 0.82381163, 31, 31.),
(-1, -1, -2, -2.00000000e+00, 0., , 9, 9.),
( 42, 53, 1, 5.19176632e-01, 0.9456603 , 22, 22.),
( 43, 52, 1, 5.07110655e-01, 0.99750255, 17, 17.),
```

( 44, 51, 3, 2.20273942e-01, 0.97095059, 15, 15.),  
( 45, 50, 3, 1.20668042e-01, 0.89049164, 13, 13.),  
( 46, 49, 4, 3.37983906e-01, 1. , 8, 8.),  
( 47, 48, 3, 1.10852856e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 55, 56, 1, 5.50383002e-01, 0.99323382, 31, 31.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 57, 58, 3, 6.27731346e-02, 0.99901027, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 59, 66, 4, 2.74952784e-01, 0.97986876, 24, 24.),  
( 60, 61, 4, 2.21462145e-01, 0.91829583, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 62, 63, 1, 5.79878986e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 64, 65, 1, 5.81109166e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 67, 72, 2, 3.29556651e-02, 0.83664074, 15, 15.),  
( 68, 69, 0, 3.57339941e-02, 1. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 70, 71, 1, 5.65802842e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 74, 89, 3, 2.80708581e-01, 0.91829583, 30, 30.),  
( 75, 80, 1, 3.97046596e-01, 0.86312057, 28, 28.),  
( 76, 79, 4, 5.32039315e-01, 0.91829583, 6, 6.),  
( 77, 78, 3, 1.06370389e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 81, 82, 2, 7.46913068e-02, 0.68403844, 22, 22.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
( 83, 88, 0, 7.66224898e-02, 0.9456603 , 11, 11.),  
( 84, 87, 0, 1.61686004e-02, 0.76420451, 9, 9.),  
( 85, 86, 1, 4.83057573e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 92, 97, 2, 1.46485843e-01, 0.37123233, 42, 42.),  
( 93, 96, 2, 1.39678806e-01, 0.77934984, 13, 13.),  
( 94, 95, 2, 2.99343411e-02, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
( 99, 272, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
(100, 271, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
(101, 154, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
(102, 107, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
(103, 104, 4, 3.37329835e-01, 0.13303965, 54, 54.),  
( -1, -1, -2, -2.00000000e+00, 0. , 45, 45.),  
(105, 106, 4, 3.46029386e-01, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(108, 113, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
(109, 110, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(111, 112, 2, 6.22235965e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(114, 115, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(116, 117, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(118, 153, 4, 3.65410343e-01, 0.92883915, 61, 61.),  
(119, 136, 2, 4.51461952e-02, 0.954434 , 56, 56.),  
(120, 135, 4, 3.41082111e-01, 0.83376491, 34, 34.),  
(121, 134, 2, 2.31719753e-02, 0.70883567, 31, 31.),  
(122, 133, 2, 2.10911464e-02, 0.84535094, 22, 22.),  
(123, 128, 3, 7.71935321e-02, 0.72192809, 20, 20.),  
(124, 125, 2, 1.85538232e-02, 0.37123233, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(126, 127, 0, 1.10634863e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(129, 130, 1, 6.54426038e-01, 1. , 6, 6.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(131, 132, 3, 8.33722576e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(137, 152, 3, 2.02571407e-01, 0.99403021, 22, 22.),  
(138, 145, 3, 6.29115291e-02, 0.97095059, 20, 20.),  
(139, 144, 3, 4.78640925e-02, 0.97095059, 10, 10.),  
(140, 143, 3, 3.26485671e-02, 0.91829583, 6, 6.),  
(141, 142, 4, 2.17665628e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(146, 151, 0, 2.74713635e-02, 0.72192809, 10, 10.),  
(147, 148, 3, 1.49664454e-01, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(149, 150, 4, 2.79059142e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(155, 200, 4, 1.58435948e-01, 0.45656871, 541, 541.),  
(156, 187, 2, 2.20749900e-02, 0.64135102, 135, 135.),  
(157, 172, 3, 5.34684099e-02, 0.86312057, 63, 63.),  
(158, 171, 0, 9.14407894e-03, 0.67519144, 45, 45.),  
(159, 170, 2, 8.46201973e-03, 0.52661707, 42, 42.),  
(160, 165, 4, 1.26957275e-01, 0.72192809, 25, 25.),  
(161, 162, 4, 9.95455980e-02, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(163, 164, 3, 2.68860059e-02, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(166, 167, 2, 5.55548281e-03, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(168, 169, 2, 6.51189336e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),

(173, 186, 4, 1.50095403e-01, 0.99107606, 18, 18.),  
(174, 175, 1, 7.62518466e-01, 0.99679163, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(176, 185, 4, 1.31732434e-01, 0.9612366 , 13, 13.),  
(177, 178, 4, 2.93148551e-02, 0.99107606, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(179, 184, 0, 7.59102986e-03, 0.86312057, 7, 7.),  
(180, 181, 4, 1.20392043e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(182, 183, 0, 3.88876768e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(188, 189, 1, 7.20179349e-01, 0.30954343, 72, 72.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(190, 195, 3, 1.37905986e-02, 0.25253077, 71, 71.),  
(191, 192, 2, 7.77588077e-02, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(193, 194, 4, 1.05520695e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(196, 199, 1, 7.80238420e-01, 0.11911603, 62, 62.),  
(197, 198, 1, 7.78470993e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 55, 55.),  
(201, 206, 2, 4.69704624e-03, 0.38028142, 406, 406.),  
(202, 205, 3, 9.57869599e-03, 0.10215804, 75, 75.),  
(203, 204, 3, 8.62109568e-03, 0.37123233, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 61, 61.),  
(207, 234, 0, 3.69854097e-03, 0.42845272, 331, 331.),  
(208, 229, 2, 3.02194478e-02, 0.65770478, 100, 100.),  
(209, 210, 1, 7.21233070e-01, 0.81750729, 63, 63.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(211, 216, 4, 2.00286217e-01, 0.90438146, 50, 50.),  
(212, 215, 0, 1.23686879e-03, 0.29747225, 19, 19.),  
(213, 214, 2, 7.58682168e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),

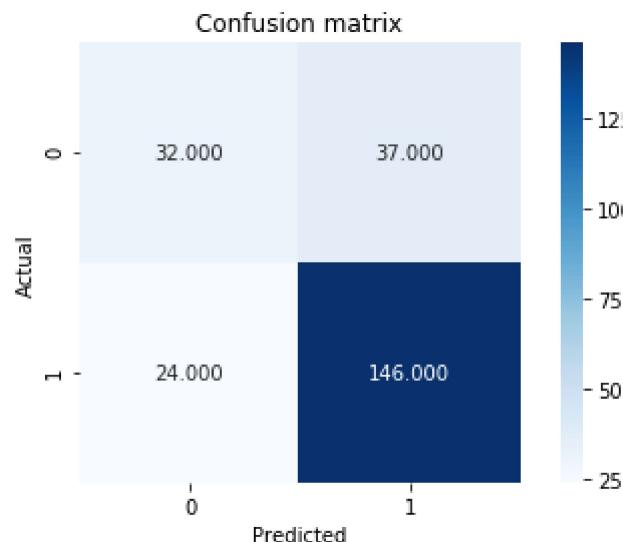
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
(217, 228, 2, 1.37113445e-02, 0.99924925, 31, 31.),  
(218, 227, 1, 8.09032619e-01, 0.94268319, 25, 25.),  
(219, 226, 1, 7.72879958e-01, 0.99277445, 20, 20.),  
(220, 221, 2, 6.68896036e-03, 0.83664074, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(222, 223, 2, 8.34391546e-03, 0.9456603 , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(224, 225, 3, 1.66143645e-02, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(230, 231, 3, 4.34002250e-01, 0.17925607, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 35, 35.),  
(232, 233, 3, 5.85608944e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(235, 246, 1, 7.12382376e-01, 0.29461521, 231, 231.),  
(236, 237, 2, 1.75233427e-02, 0.72192809, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(238, 245, 0, 2.45137718e-02, 0.87398105, 17, 17.),  
(239, 244, 1, 7.07583576e-01, 0.9612366 , 13, 13.),  
(240, 241, 3, 5.66457137e-02, 0.72192809, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(242, 243, 4, 2.54450589e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(247, 270, 2, 5.91170639e-01, 0.2139773 , 206, 206.),  
(248, 269, 4, 2.53816202e-01, 0.190709 , 205, 205.),  
(249, 268, 4, 2.53299832e-01, 0.26086291, 136, 136.),  
(250, 253, 2, 5.48125640e-03, 0.22853814, 135, 135.),  
(251, 252, 0, 8.59764195e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(254, 255, 4, 2.00817339e-01, 0.19476878, 133, 133.),  
( -1, -1, -2, -2.00000000e+00, 0. , 57, 57.),  
(256, 257, 4, 2.01555006e-01, 0.29747225, 76, 76.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(258, 265, 0, 2.44631423e-02, 0.24229219, 75, 75.),

```
(259, 264, 0, 6.95844390e-03, 0.1811664 , 73, 73.),
(260, 261, 0, 6.42993581e-03, 0.39124356, 26, 26.),
(-1, -1, -2, -2.00000000e+00, 0. , 23, 23.),
(262, 263, 2, 1.54202851e-02, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 47, 47.),
(266, 267, 3, 4.79277764e-02, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 69, 69.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(273, 296, 3, 3.54934916e-01, 0.99729438, 49, 49.),
(274, 275, 0, 3.51146031e-02, 0.9985091 , 44, 44.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(276, 277, 2, 4.09591515e-02, 0.9892453 , 41, 41.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(278, 281, 4, 1.98117480e-01, 0.99941106, 35, 35.),
(279, 280, 2, 1.79708004e-01, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(282, 291, 1, 6.54390246e-01, 0.98522814, 28, 28.),
(283, 288, 2, 1.84076220e-01, 0.91829583, 15, 15.),
(284, 287, 1, 6.18632317e-01, 0.68403844, 11, 11.),
(285, 286, 2, 5.42513076e-02, 1. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(289, 290, 3, 1.31794780e-01, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(292, 293, 1, 6.92071795e-01, 0.61938219, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
(294, 295, 4, 2.49218076e-01, 1. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [134]: # TASK 4  
# Predict class labels using decision tree  
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",  
                "Percent Foreign Born", "Median Household Income"]]  
y_pred = model.predict(x_test)
```

```
In [135]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)  
  
[[ 32  37]  
 [ 24 146]]
```

```
In [136]: # TASK 4  
# 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()
```



```
In [137]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7447698744769874  
0.2552301255230126  
[0.57142857 0.79781421]  
[0.46376812 0.85882353]  
[0.512      0.82719547]
```

```
In [138]: # Task 4  
# Build decision tree with five Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Foreign Born", "Percent Unemployed"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[138]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [139]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[139]: array([( 1, 88, 1, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 85, 0, 1.36103377e-01, 0.94464539, 218, 218.),
( 3, 16, 3, 5.31811956e-02, 0.98654463, 176, 176.),
( 4, 11, 2, 1.66045949e-02, 0.79732651, 29, 29.),
( 5, 6, 2, 6.58472651e-03, 0.99572745, 13, 13.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 7, 8, 1, 5.07332757e-01, 0.91829583, 9, 9.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 9, 10, 0, 1.45652602e-03, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 12, 13, 0, 4.23690416e-02, 0.33729007, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0., , 12, 12.),
( 14, 15, 0, 4.52499576e-02, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
( 17, 28, 1, 3.57535481e-01, 0.9486132 , 147, 147.),
( 18, 23, 4, 2.09830336e-01, 0.63945713, 37, 37.),
( 19, 22, 4, 1.91108666e-01, 0.954434 , 8, 8.),
( 20, 21, 0, 3.30535676e-02, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0., , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 4, 4.),
( 24, 25, 1, 3.18483472e-01, 0.21639693, 29, 29.),
(-1, -1, -2, -2.00000000e+00, 0., , 21, 21.),
( 26, 27, 1, 3.32529366e-01, 0.54356444, 8, 8.),
(-1, -1, -2, -2.00000000e+00, 0., , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 29, 84, 2, 3.20100173e-01, 0.98828361, 110, 110.),
( 30, 83, 0, 1.24251015e-01, 0.99666573, 103, 103.),
( 31, 80, 3, 2.68892780e-01, 0.99277445, 100, 100.),
( 32, 79, 3, 2.28306502e-01, 0.99863596, 92, 92.),
( 33, 78, 3, 1.97396681e-01, 0.99226664, 87, 87.),
( 34, 57, 1, 5.34610420e-01, 0.99901027, 81, 81.),
( 35, 36, 3, 6.52362593e-02, 0.9330253 , 43, 43.),
(-1, -1, -2, -2.00000000e+00, 0., , 7, 7.),
( 37, 38, 3, 6.83506504e-02, 0.97986876, 36, 36.),
(-1, -1, -2, -2.00000000e+00, 0., , 2, 2.),
( 39, 40, 3, 8.68914314e-02, 0.95968689, 34, 34.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
( 41, 44, 4, 2.84229696e-01, 0.99631652, 28, 28.),
( 42, 43, 3, 1.23242423e-01, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0., , 6, 6.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 45, 46, 1, 4.01377693e-01, 0.91829583, 21, 21.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( 47, 54, 0, 3.88660263e-02, 0.99679163, 15, 15.),
( 48, 53, 2, 2.63659135e-02, 0.91829583, 9, 9.),
( 49, 50, 1, 4.53624099e-01, 0.97095059, 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( 51, 52, 4, 4.30317611e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 55, 56, 2, 3.35072447e-02, 0.65002242, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 58, 77, 3, 1.63666159e-01, 0.94945202, 38, 38.),
( 59, 76, 3, 1.37628861e-01, 0.98337619, 33, 33.),
( 60, 71, 4, 4.00537908e-01, 0.94807824, 30, 30.),
( 61, 70, 2, 8.56052414e-02, 0.77322667, 22, 22.),
( 62, 67, 0, 3.57339941e-02, 0.91829583, 15, 15.),
( 63, 66, 1, 5.51199228e-01, 0.68403844, 11, 11.),
( 64, 65, 4, 2.47119538e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( 68, 69, 1, 5.68420947e-01, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 72, 75, 0, 1.20450649e-02, 0.81127812, 8, 8.),
( 73, 74, 3, 9.06827450e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 81, 82, 2, 9.33686970e-03, 0.54356444, 8, 8.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
( 86, 87, 4, 2.80387834e-01, 0.37123233, 42, 42.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 39, 39.),  
( 89, 256, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 90, 255, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 91, 136, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 92, 97, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 93, 96, 0, 4.52394699e-04, 0.13303965, 54, 54.),  
( 94, 95, 0, 4.25896011e-04, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
( 98, 103, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
( 99, 100, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(101, 102, 0, 2.02996898e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(104, 105, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(106, 135, 4, 4.97614488e-01, 0.954434 , 64, 64.),  
(107, 110, 2, 7.23552890e-03, 0.91829583, 60, 60.),  
(108, 109, 3, 1.22745018e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(111, 114, 2, 1.85538232e-02, 0.86989269, 55, 55.),  
(112, 113, 3, 1.48829598e-01, 0.35335934, 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(115, 116, 1, 6.31248355e-01, 0.954434 , 40, 40.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(117, 118, 1, 6.38267547e-01, 0.90902216, 37, 37.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(119, 130, 0, 1.90389156e-02, 0.96290041, 31, 31.),  
(120, 127, 1, 6.79557621e-01, 0.99107606, 18, 18.),  
(121, 122, 2, 1.94749255e-02, 0.9456603 , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(123, 124, 3, 7.21841212e-02, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(125, 126, 3, 1.41124912e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(128, 129, 4, 4.38949838e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(131, 132, 4, 4.36525166e-01, 0.61938219, 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
(133, 134, 1, 6.58490986e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(137, 250, 4, 7.77149290e-01, 0.45656871, 541, 541.),
(138, 195, 0, 3.68557358e-03, 0.43204588, 530, 530.),
(139, 140, 1, 7.21233070e-01, 0.54671754, 222, 222.),
( -1, -1, -2, -2.00000000e+00, 0. , 30, 30.),
(141, 184, 2, 2.96944054e-02, 0.59931424, 192, 192.),
(142, 183, 2, 2.66378755e-02, 0.72192809, 125, 125.),
(143, 144, 0, 6.47921232e-04, 0.68079378, 122, 122.),
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),
(145, 146, 0, 7.52224587e-04, 0.75221217, 102, 102.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(147, 148, 3, 8.78720824e-03, 0.72192809, 100, 100.),
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),
(149, 150, 1, 7.38505304e-01, 0.77322667, 88, 88.),
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
(151, 174, 3, 5.18329125e-02, 0.82128094, 78, 78.),
(152, 173, 3, 3.60547658e-02, 0.69396597, 59, 59.),
(153, 172, 1, 8.55425090e-01, 0.7601675 , 50, 50.),
(154, 155, 3, 1.12664807e-02, 0.83900406, 41, 41.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(156, 161, 2, 5.47810318e-03, 0.77934984, 39, 39.),
(157, 158, 0, 2.43393320e-03, 0.32275696, 17, 17.),
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),
(159, 160, 2, 1.24893506e-03, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(162, 167, 2, 8.08177842e-03, 0.9456603 , 22, 22.),
(163, 164, 3, 2.91048158e-02, 0.86312057, 7, 7.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(165, 166, 0, 1.84622593e-03, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(168, 171, 4, 2.02076368e-01, 0.72192809, 15, 15.),
(169, 170, 3, 2.87141800e-02, 1. , 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(175, 176, 0, 9.42225859e-04, 0.99800088, 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(177, 178, 2, 1.22560994e-02, 0.94028596, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(179, 180, 4, 3.48079696e-01, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(181, 182, 4, 6.74127936e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(185, 186, 4, 4.21003237e-01, 0.26377744, 67, 67.),  
( -1, -1, -2, -2.00000000e+00, 0. , 42, 42.),  
(187, 188, 4, 4.29214492e-01, 0.52936087, 25, 25.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(189, 190, 3, 9.06392699e-03, 0.41381685, 24, 24.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(191, 194, 0, 1.90576346e-03, 0.25801867, 23, 23.),  
(192, 193, 0, 1.66840281e-03, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
(196, 209, 1, 7.12382376e-01, 0.33411075, 308, 308.),  
(197, 208, 1, 7.10477173e-01, 0.74959526, 28, 28.),  
(198, 199, 2, 1.75233427e-02, 0.61938219, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(200, 203, 1, 7.00666279e-01, 0.81127812, 16, 16.),  
(201, 202, 1, 6.96625143e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(204, 205, 1, 7.07583576e-01, 0.61938219, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(206, 207, 2, 6.89322036e-02, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(210, 211, 3, 5.39405714e-03, 0.27102755, 280, 280.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(212, 249, 3, 1.17283653e-01, 0.25593004, 279, 279.),  
(213, 248, 3, 1.17194992e-01, 0.2965131 , 229, 229.),  
(214, 247, 1, 8.33037525e-01, 0.27889718, 228, 228.),  
(215, 246, 1, 8.32422495e-01, 0.32275696, 187, 187.),

(216, 233, 4, 4.65944514e-01, 0.30217362, 186, 186.),  
(217, 218, 0, 6.53435197e-03, 0.21639693, 145, 145.),  
( -1, -1, -2, -2.00000000e+00, 0. , 41, 41.),  
(219, 228, 0, 8.04590527e-03, 0.27817101, 104, 104.),  
(220, 227, 2, 4.13624961e-02, 0.61938219, 26, 26.),  
(221, 222, 2, 1.47319660e-02, 0.81127812, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(223, 224, 4, 3.83113325e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(225, 226, 3, 2.20937012e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(229, 232, 2, 5.52802766e-03, 0.0989591 , 78, 78.),  
(230, 231, 1, 7.50016689e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 73, 73.),  
(234, 239, 2, 8.09433535e-02, 0.5349437 , 41, 41.),  
(235, 238, 2, 6.90776296e-03, 0.22228483, 28, 28.),  
(236, 237, 2, 6.07098150e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 25, 25.),  
(240, 245, 3, 7.09828027e-02, 0.89049164, 13, 13.),  
(241, 244, 4, 4.87461746e-01, 0.68403844, 11, 11.),  
(242, 243, 0, 8.28304025e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 41, 41.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 50, 50.),  
(251, 254, 2, 2.29016375e-02, 0.99403021, 11, 11.),  
(252, 253, 2, 2.98666710e-03, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(257, 282, 0, 1.65806837e-01, 0.99729438, 49, 49.),  
(258, 281, 3, 3.54934916e-01, 0.9985091 , 44, 44.),

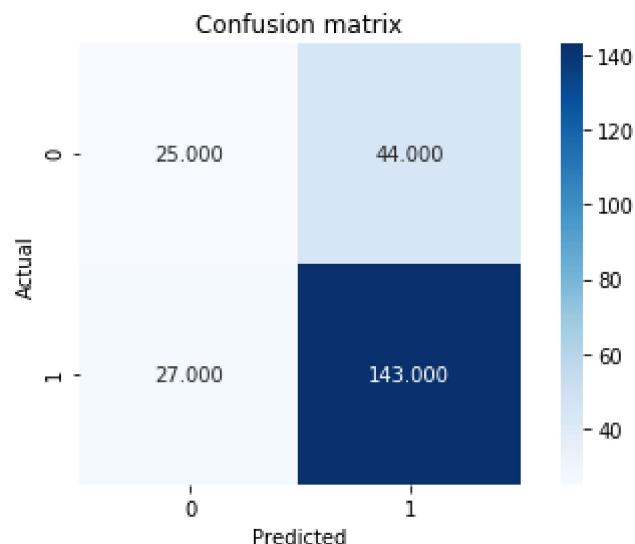
```
(259, 260, 0, 3.51146031e-02, 0.9892453 , 41, 41.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(261, 262, 2, 4.09591515e-02, 0.96778846, 38, 38.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(263, 264, 4, 3.26585621e-01, 0.9971804 , 32, 32.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(265, 274, 1, 6.54390246e-01, 0.99631652, 28, 28.),
(266, 267, 2, 1.84076220e-01, 0.81127812, 12, 12.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(268, 269, 2, 2.98215017e-01, 0.97095059, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(270, 271, 3, 7.90550672e-02, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(272, 273, 2, 4.86683130e-01, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(275, 276, 1, 6.92071795e-01, 0.954434 , 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
(277, 278, 0, 3.61190718e-02, 0.81127812, 8, 8.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(279, 280, 0, 1.18469346e-01, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [140]: # TASK 4
# Predict class labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Foreign Born", "Percent Unemployed"]]
y_pred = model.predict(x_test)
```

```
In [141]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

  
[[ 25 44]  
[ 27 143]]

```
In [142]: # TASK 4  
# 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()
```



```
In [143]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.702928870292887  
0.297071129707113  
[0.48076923 0.76470588]  
[0.36231884 0.84117647]  
[0.41322314 0.80112045]
```

```
In [144]: # Task 4  
# Build decision tree with five Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin  
o",  
           "Percent Foreign Born", "Percent Less than High School Degree"]]  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[144]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
                                 max_features=None, max_leaf_nodes=None,  
                                 min_impurity_decrease=0.0, min_impurity_split=None,  
                                 min_samples_leaf=1, min_samples_split=2,  
                                 min_weight_fraction_leaf=0.0, presort=False,  
                                 random_state=0, splitter='best')
```

```
In [145]: # TASK 4
# Show decision tree
model.tree_.__getstate__()['nodes']
```

```
Out[145]: array([( 1,  94,  1,  6.00609809e-01,  0.83826453,  956,  956.),
   ( 2,  87,  0,  1.36103377e-01,  0.94464539,  218,  218.),
   ( 3,  16,  3,  5.31811956e-02,  0.98654463,  176,  176.),
   ( 4,  11,  2,  1.66045949e-02,  0.79732651,  29,  29.),
   ( 5,  6,  2,  6.58472651e-03,  0.99572745,  13,  13.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  4,  4.),
   ( 7,  8,  1,  5.07332757e-01,  0.91829583,  9,  9.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  2,  2.),
   ( 9,  10,  0,  1.45652602e-03,  0.59167278,  7,  7.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  6,  6.),
   (12, 13,  0,  4.23690416e-02,  0.33729007,  16,  16.),
   (-1, -1, -2, -2.00000000e+00,  0.          , 12, 12.),
   (14, 15,  4,  8.76736976e-02,  0.81127812,  4,  4.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  3,  3.),
   (17, 30,  1,  3.57535481e-01,  0.9486132 , 147, 147.),
   (18, 25,  4,  4.04738747e-02,  0.63945713,  37, 37.),
   (19, 24,  3,  1.24822907e-01,  0.99107606,  9,  9.),
   (20, 23,  3,  9.50216018e-02,  0.91829583,  6,  6.),
   (21, 22,  2,  1.34591726e-02,  0.91829583,  3,  3.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  2,  2.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  3,  3.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  3,  3.),
   (26, 29,  3,  1.32442802e-01,  0.22228483,  28, 28.),
   (27, 28,  3,  1.30613677e-01,  0.59167278,  7,  7.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  6,  6.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
   (-1, -1, -2, -2.00000000e+00,  0.          , 21, 21.),
   (31, 86,  2,  3.20100173e-01,  0.98828361, 110, 110.),
   (32, 41,  4,  9.17319208e-02,  0.99666573, 103, 103.),
   (33, 34,  0,  2.78865471e-02,  0.72192809,  20,  20.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  9,  9.),
   (35, 36,  0,  3.34031247e-02,  0.9456603 , 11,  11.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  2,  2.),
   (37, 38,  4,  5.25505580e-02,  0.76420451,  9,  9.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
   (39, 40,  0,  1.02079052e-01,  0.54356444,  8,  8.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  7,  7.),
   (-1, -1, -2, -2.00000000e+00,  0.          ,  1,  1.),
   (42, 83,  3,  2.68892780e-01,  0.99738066,  83,  83.),
   (43, 82,  3,  2.28306502e-01,  0.9844269 ,  75,  75.)],
```

( 44, 81, 3, 1.97396681e-01, 0.9946938 , 70, 70.),  
( 45, 78, 3, 1.52200811e-01, 0.9744894 , 64, 64.),  
( 46, 77, 3, 1.37628861e-01, 0.99603836, 54, 54.),  
( 47, 48, 0, 2.27708335e-03, 0.97552595, 49, 49.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 49, 50, 3, 5.88882565e-02, 0.96011866, 47, 47.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 51, 56, 0, 9.72468173e-03, 0.93893201, 45, 45.),  
( 52, 53, 4, 1.91367626e-01, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 54, 55, 1, 5.66962361e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 57, 62, 0, 1.72619252e-02, 0.97986876, 36, 36.),  
( 58, 61, 0, 1.20450649e-02, 0.81127812, 8, 8.),  
( 59, 60, 1, 4.98925418e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 63, 72, 1, 5.68405598e-01, 0.90592822, 28, 28.),  
( 64, 71, 3, 1.04607537e-01, 0.72192809, 20, 20.),  
( 65, 70, 1, 5.33278704e-01, 0.9456603 , 11, 11.),  
( 66, 69, 3, 9.15925130e-02, 0.91829583, 6, 6.),  
( 67, 68, 2, 4.94297948e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( 73, 74, 0, 3.21643390e-02, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 75, 76, 0, 6.33968264e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 79, 80, 4, 1.09592441e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 84, 85, 0, 2.01796001e-03, 0.54356444, 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),

( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( 88, 93, 2, 1.46485843e-01, 0.37123233, 42, 42.),  
( 89, 92, 2, 1.39678806e-01, 0.77934984, 13, 13.),  
( 90, 91, 2, 2.99343411e-02, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 29, 29.),  
( 95, 262, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 96, 261, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 97, 148, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 98, 103, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 99, 102, 4, 7.11084008e-02, 0.13303965, 54, 54.),  
(100, 101, 4, 7.03762509e-02, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 41, 41.),  
(104, 109, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
(105, 106, 0, 1.77675420e-02, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(107, 108, 0, 2.02996898e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(110, 111, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(112, 113, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(114, 147, 2, 3.48945007e-01, 0.92883915, 61, 61.),  
(115, 146, 4, 2.46285446e-01, 0.90657955, 59, 59.),  
(116, 135, 2, 4.51461952e-02, 0.95260921, 51, 51.),  
(117, 126, 1, 6.66937649e-01, 0.83376491, 34, 34.),  
(118, 119, 0, 9.06762760e-03, 0.98869941, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(120, 121, 0, 1.10741984e-02, 0.99572745, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(122, 123, 3, 8.52802806e-02, 0.91829583, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(124, 125, 1, 6.48851544e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(127, 132, 1, 6.89204097e-01, 0.50325833, 18, 18.),  
(128, 131, 0, 6.59197266e-03, 0.33729007, 16, 16.),  
(129, 130, 1, 6.75937116e-01, 1. , 2, 2.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
(133, 134, 2, 2.47595077e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(136, 137, 0, 1.72357643e-02, 0.97741782, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(138, 145, 3, 9.39139649e-02, 0.97986876, 12, 12.),  
(139, 144, 4, 1.86793864e-01, 0.99107606, 9, 9.),  
(140, 141, 0, 2.25820737e-02, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(142, 143, 1, 6.71001315e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(149, 232, 2, 3.17869987e-02, 0.45656871, 541, 541.),  
(150, 151, 0, 6.47921232e-04, 0.53722708, 334, 334.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(152, 193, 0, 3.68557358e-03, 0.56709563, 307, 307.),  
(153, 154, 1, 7.19135463e-01, 0.71886413, 131, 131.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(155, 156, 3, 8.78720824e-03, 0.78894066, 110, 110.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(157, 158, 0, 7.52224587e-04, 0.83863987, 97, 97.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(159, 160, 0, 8.57430045e-04, 0.8154225 , 95, 95.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(161, 192, 4, 6.07850760e-01, 0.84975114, 87, 87.),  
(162, 187, 1, 8.11605304e-01, 0.81127812, 84, 84.),  
(163, 176, 1, 7.70526022e-01, 0.90657955, 59, 59.),  
(164, 171, 2, 1.34000210e-02, 0.72192809, 35, 35.),  
(165, 166, 1, 7.19866216e-01, 0.38094659, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(167, 168, 1, 7.57786006e-01, 0.23519338, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
(169, 170, 1, 7.61501193e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(172, 175, 4, 2.78884001e-01, 0.954434 , 8, 8.),

(173, 174, 4, 8.90559107e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(177, 180, 4, 1.71659440e-01, 1. , 24, 24.),  
(178, 179, 2, 3.49423394e-03, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(181, 186, 2, 1.34528433e-02, 0.91829583, 15, 15.),  
(182, 185, 4, 2.47928798e-01, 1. , 10, 10.),  
(183, 184, 3, 4.46308386e-02, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(188, 191, 3, 1.50281880e-02, 0.40217919, 25, 25.),  
(189, 190, 1, 8.55425090e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(194, 221, 4, 3.64172906e-01, 0.42033516, 176, 176.),  
(195, 220, 2, 3.14539718e-02, 0.33228663, 147, 147.),  
(196, 219, 1, 7.78253704e-01, 0.30642473, 146, 146.),  
(197, 218, 1, 7.77404457e-01, 0.42622866, 92, 92.),  
(198, 205, 3, 5.66457137e-02, 0.39124356, 91, 91.),  
(199, 200, 0, 1.25664696e-02, 0.19590927, 66, 66.),  
( -1, -1, -2, -2.00000000e+00, 0. , 52, 52.),  
(201, 202, 0, 1.29721817e-02, 0.59167278, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(203, 204, 3, 1.98028758e-02, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(206, 213, 4, 1.99971087e-01, 0.72192809, 25, 25.),  
(207, 208, 4, 1.66832671e-01, 0.99107606, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(209, 212, 3, 1.23558510e-01, 0.91829583, 6, 6.),  
(210, 211, 0, 4.45087836e-03, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(214, 217, 2, 6.63869525e-03, 0.33729007, 16, 16.),  
(215, 216, 4, 3.13238040e-01, 1. , 2, 2.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 54, 54.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(222, 223, 1, 8.32422495e-01, 0.73550858, 29, 29.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(224, 225, 2, 4.42261482e-03, 0.954434 , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(226, 229, 2, 8.79196217e-03, 1. , 12, 12.),  
(227, 228, 3, 1.29751762e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(230, 231, 3, 9.12519805e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(233, 240, 1, 7.19351888e-01, 0.299589 , 207, 207.),  
(234, 237, 2, 6.46481011e-02, 0.83664074, 15, 15.),  
(235, 236, 4, 1.74003914e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(238, 239, 1, 7.00127363e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(241, 242, 2, 6.93185329e-02, 0.22581117, 192, 192.),  
( -1, -1, -2, -2.00000000e+00, 0. , 69, 69.),  
(243, 260, 4, 7.53534973e-01, 0.31505695, 123, 123.),  
(244, 259, 4, 3.57937157e-01, 0.28290479, 122, 122.),  
(245, 258, 2, 5.91170639e-01, 0.45868582, 62, 62.),  
(246, 257, 4, 3.57680067e-01, 0.40907314, 61, 61.),  
(247, 252, 2, 8.96823183e-02, 0.35335934, 60, 60.),  
(248, 251, 2, 8.40812661e-02, 0.84535094, 11, 11.),  
(249, 250, 3, 1.42369270e-02, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(253, 254, 0, 2.30396548e-02, 0.14372617, 49, 49.),  
( -1, -1, -2, -2.00000000e+00, 0. , 45, 45.),  
(255, 256, 0, 2.51084697e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 60, 60.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(263, 288, 3, 3.54934916e-01, 0.99729438, 49, 49.),
(264, 265, 0, 3.51146031e-02, 0.9985091 , 44, 44.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(266, 267, 2, 4.09591515e-02, 0.9892453 , 41, 41.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(268, 287, 0, 1.65806837e-01, 0.99941106, 35, 35.),
(269, 286, 0, 9.86937620e-02, 0.9971804 , 32, 32.),
(270, 285, 1, 7.03338951e-01, 0.9991421 , 29, 29.),
(271, 284, 3, 1.50136843e-01, 0.98958752, 25, 25.),
(272, 277, 2, 1.14497282e-01, 0.99277445, 20, 20.),
(273, 274, 1, 6.54390246e-01, 0.76420451, 9, 9.),
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(275, 276, 0, 3.72781362e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(278, 279, 4, 1.67947575e-01, 0.9456603 , 11, 11.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(280, 281, 3, 1.00945447e-01, 0.91829583, 6, 6.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(282, 283, 3, 1.31794780e-01, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

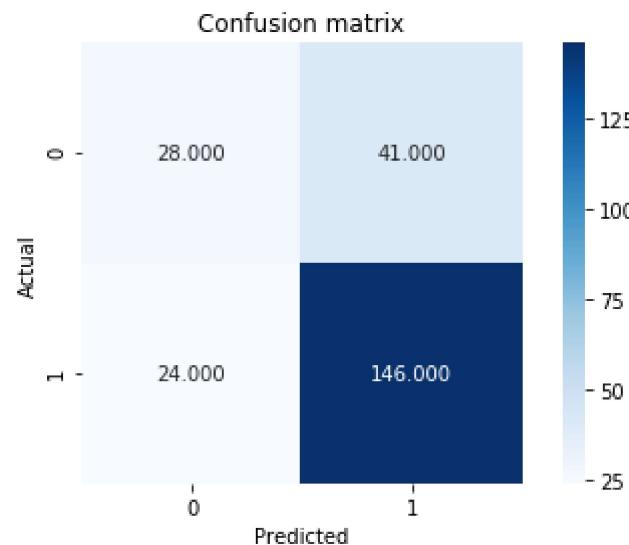
In [146]:

```
# TASK 4
# Predict class Labels using decision tree
x_test = x_val[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Foreign Born","Percent Less than High School Degree"]]
y_pred = model.predict(x_test)
```

```
In [147]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 28  41]  
 [ 24 146]]
```

```
In [148]: # TASK 4  
# 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()
```



In [149]: # TASK 4

```
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7280334728033473
0.2719665271966527
[0.53846154 0.78074866]
[0.4057971 0.85882353]
[0.46280992 0.81792717]
```

In [150]: # Task 4

```
# Build decision tree with five Parameters
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)
x = x_train[["Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latin
o",
            "Percent Foreign Born", "Percent Rural"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

Out[150]: DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=None,  
max\_features=None, max\_leaf\_nodes=None,  
min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
min\_samples\_leaf=1, min\_samples\_split=2,  
min\_weight\_fraction\_leaf=0.0, presort=False,  
random\_state=0, splitter='best')

```
In [151]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[151]: array([( 1,  94,  1,  6.00609809e-01,  0.83826453,  956,  956.),
   ( 2,   9,  4,  4.29369155e-02,  0.94464539,  218,  218.),
   ( 3,   8,  2,  1.28731821e-01,  0.39662777,   51,   51.),
   ( 4,   7,  3,  2.60152347e-01,  0.91829583,   12,   12.),
   ( 5,   6,  1,  3.35271969e-01,  0.72192809,    5,    5.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    1,    1.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    4,    4.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    7,    7.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,   39,   39.),
   ( 10,  25,  3,  5.31811956e-02,  0.99251207,  167,  167.),
   ( 11,  20,  2,  1.66045949e-02,  0.79732651,   29,   29.),
   ( 12,  15,  4,  7.19376117e-01,  0.99572745,   13,   13.),
   ( 13,  14,  4,  5.34084231e-01,  0.65002242,    6,    6.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    1,    1.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    5,    5.),
   ( 16,  19,  3,  3.28101981e-02,  0.59167278,    7,    7.),
   ( 17,  18,  3,  1.85428376e-02,  1.          ,    2,    2.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    1,    1.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    1,    1.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    5,    5.),
   ( 21,  22,  0,  4.23690416e-02,  0.33729007,   16,   16.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,   12,   12.),
   ( 23,  24,  0,  4.52499576e-02,  0.81127812,    4,    4.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    1,    1.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    3,    3.),
   ( 26,  67,  1,  5.34610420e-01,  0.96085769,  138,  138.),
   ( 27,  28,  3,  6.63515702e-02,  0.88946639,   88,   88.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    8,    8.),
   ( 29,  64,  2,  1.65527709e-01,  0.92240626,   80,   80.),
   ( 30,  31,  4,  7.19443858e-02,  0.96353598,   67,   67.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    5,    5.),
   ( 32,  55,  3,  1.32442802e-01,  0.9235786 ,   62,   62.),
   ( 33,  54,  2,  6.48660623e-02,  0.98999279,   34,   34.),
   ( 34,  53,  1,  5.15293539e-01,  0.94807824,   30,   30.),
   ( 35,  36,  3,  8.54628347e-02,  0.87671629,   27,   27.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    6,    6.),
   ( 37,  38,  0,  8.64804583e-03,  0.95871188,   21,   21.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    4,    4.),
   ( 39,  52,  2,  5.77327460e-02,  0.99750255,   17,   17.),
   ( 40,  51,  2,  4.94624116e-02,  0.98522814,   14,   14.),
   ( 41,  42,  0,  1.28038297e-02,  0.99403021,   11,   11.),
   (-1,  -1, -2, -2.00000000e+00,  0.          ,    2,    2.),
   ( 43,  50,  0,  7.70637244e-02,  0.91829583,    9,    9.),
])
```

```
( 44, 49, 0, 2.95596858e-02, 0.81127812, 8, 8.),  
( 45, 46, 4, 2.18572155e-01, 1., , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( 47, 48, 2, 1.89147200e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0., , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0., , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0., , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0., , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0., , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0., , 4, 4.),  
( 56, 57, 1, 3.67759630e-01, 0.74959526, 28, 28.),  
( -1, -1, -2, -2.00000000e+00, 0., , 14, 14.),  
( 58, 63, 1, 5.20866960e-01, 0.98522814, 14, 14.),  
( 59, 60, 0, 3.82399820e-02, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0., , 4, 4.),  
( 61, 62, 3, 2.26727799e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0., , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0., , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0., , 4, 4.),  
( 65, 66, 4, 8.47890884e-01, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0., , 12, 12.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( 68, 81, 1, 5.70503086e-01, 0.99884554, 50, 50.),  
( 69, 76, 4, 4.08189476e-01, 0.81127812, 24, 24.),  
( 70, 71, 0, 5.17941192e-02, 0.54356444, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0., , 9, 9.),  
( 72, 73, 0, 6.33825064e-02, 0.86312057, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( 74, 75, 1, 5.64737916e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0., , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( 77, 80, 1, 5.60255438e-01, 1., , 8, 8.),  
( 78, 79, 0, 1.98036572e-02, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0., , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0., , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0., , 3, 3.),  
( 82, 83, 3, 6.27483130e-02, 0.89049164, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0., , 2, 2.),  
( 84, 89, 4, 1.20978422e-01, 0.81127812, 24, 24.),  
( 85, 88, 0, 1.34396292e-01, 0.98522814, 7, 7.),  
( 86, 87, 1, 5.74865937e-01, 0.72192809, 5, 5.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 90, 91, 4, 3.35052252e-01, 0.52255937, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( 92, 93, 3, 1.63713828e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 95, 272, 0, 3.41651589e-02, 0.63079708, 738, 738.),  
( 96, 271, 2, 7.00087726e-01, 0.56310282, 689, 689.),  
( 97, 146, 1, 6.92266941e-01, 0.53839675, 682, 682.),  
( 98, 103, 0, 2.43641390e-03, 0.77265709, 141, 141.),  
( 99, 102, 0, 4.52394699e-04, 0.13303965, 54, 54.),  
(100, 101, 0, 4.25896011e-04, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
(104, 107, 1, 6.23542339e-01, 0.93958762, 87, 87.),  
(105, 106, 4, 2.28239097e-01, 0.32275696, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
(108, 109, 3, 2.12041056e-02, 0.98522814, 70, 70.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(110, 111, 2, 3.86667950e-03, 0.954434 , 64, 64.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(112, 113, 4, 2.26481423e-01, 0.92883915, 61, 61.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(114, 133, 2, 4.51461952e-02, 0.95931603, 55, 55.),  
(115, 116, 3, 3.08582708e-02, 0.83376491, 34, 34.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(117, 118, 3, 3.17892041e-02, 0.90592822, 28, 28.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(119, 132, 1, 6.78198606e-01, 0.84035867, 26, 26.),  
(120, 131, 2, 3.30871120e-02, 0.94945202, 19, 19.),  
(121, 126, 3, 4.95797917e-02, 1. , 14, 14.),  
(122, 125, 4, 5.23223504e-01, 0.65002242, 6, 6.),  
(123, 124, 0, 7.85590382e-03, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(127, 130, 4, 4.50167596e-01, 0.81127812, 8, 8.),  
(128, 129, 1, 6.42785132e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(134, 145, 4, 6.30392313e-01, 0.98522814, 21, 21.),  
(135, 140, 4, 4.21490431e-01, 0.99750255, 17, 17.),  
(136, 139, 2, 9.15906057e-02, 0.81127812, 8, 8.),  
(137, 138, 2, 5.87546527e-02, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(141, 144, 2, 7.78770410e-02, 0.76420451, 9, 9.),  
(142, 143, 4, 5.82395464e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(147, 236, 2, 3.17869987e-02, 0.45656871, 541, 541.),  
(148, 149, 0, 6.47921232e-04, 0.53722708, 334, 334.),  
( -1, -1, -2, -2.00000000e+00, 0. , 27, 27.),  
(150, 197, 0, 3.68557358e-03, 0.56709563, 307, 307.),  
(151, 152, 1, 7.19135463e-01, 0.71886413, 131, 131.),  
( -1, -1, -2, -2.00000000e+00, 0. , 21, 21.),  
(153, 154, 3, 8.78720824e-03, 0.78894066, 110, 110.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(155, 156, 0, 7.52224587e-04, 0.83863987, 97, 97.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(157, 158, 0, 8.57430045e-04, 0.8154225 , 95, 95.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
(159, 196, 1, 8.97281468e-01, 0.84975114, 87, 87.),  
(160, 191, 1, 8.11605304e-01, 0.82496587, 85, 85.),  
(161, 176, 1, 7.70526022e-01, 0.91829583, 60, 60.),  
(162, 169, 2, 1.34000210e-02, 0.72192809, 35, 35.),  
(163, 164, 1, 7.19866216e-01, 0.38094659, 27, 27.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(165, 166, 1, 7.57786006e-01, 0.23519338, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
(167, 168, 1, 7.61501193e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(170, 171, 0, 2.22611579e-03, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(172, 173, 0, 2.66813650e-03, 0.97095059, 5, 5.),

( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(174, 175, 1, 7.59539276e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(177, 190, 3, 6.82943799e-02, 0.99884554, 25, 25.),  
(178, 189, 0, 3.28301883e-03, 0.98522814, 21, 21.),  
(179, 188, 2, 1.39053124e-02, 0.91829583, 18, 18.),  
(180, 181, 1, 7.82922387e-01, 0.99572745, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(182, 187, 3, 3.41992024e-02, 0.9456603 , 11, 11.),  
(183, 184, 0, 1.57909660e-03, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(185, 186, 0, 2.42874620e-03, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(192, 195, 3, 1.50281880e-02, 0.40217919, 25, 25.),  
(193, 194, 1, 8.55425090e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 19, 19.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(198, 235, 2, 3.14539718e-02, 0.42033516, 176, 176.),  
(199, 224, 1, 8.32422495e-01, 0.40217919, 175, 175.),  
(200, 223, 1, 7.78253704e-01, 0.30954343, 144, 144.),  
(201, 222, 1, 7.77404457e-01, 0.42304882, 93, 93.),  
(202, 209, 3, 5.66457137e-02, 0.38823898, 92, 92.),  
(203, 204, 4, 2.85429776e-01, 0.19590927, 66, 66.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(205, 206, 4, 8.30364853e-01, 0.11467551, 65, 65.),  
( -1, -1, -2, -2.00000000e+00, 0. , 56, 56.),  
(207, 208, 0, 1.20014274e-02, 0.50325833, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(210, 211, 4, 4.19467732e-01, 0.70627409, 26, 26.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
(212, 213, 1, 7.13595062e-01, 0.89603823, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(214, 215, 1, 7.43275404e-01, 0.74959526, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),

(216, 217, 3, 7.30122402e-02, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(218, 219, 2, 7.54642719e-03, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(220, 221, 4, 4.32819232e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 51, 51.),  
(225, 234, 0, 1.52357332e-02, 0.70883567, 31, 31.),  
(226, 231, 3, 9.12519805e-02, 0.57879462, 29, 29.),  
(227, 230, 3, 1.16126747e-02, 0.39124356, 26, 26.),  
(228, 229, 2, 8.79196217e-03, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
(232, 233, 4, 5.58363289e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(237, 270, 4, 7.23074585e-01, 0.299589 , 207, 207.),  
(238, 247, 1, 7.19351888e-01, 0.41164557, 133, 133.),  
(239, 246, 4, 6.45300746e-01, 0.91829583, 12, 12.),  
(240, 245, 2, 2.98372276e-01, 0.72192809, 10, 10.),  
(241, 244, 2, 6.46481011e-02, 0.50325833, 9, 9.),  
(242, 243, 4, 4.85312179e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(248, 251, 3, 1.44244060e-02, 0.31885542, 121, 121.),  
(249, 250, 1, 8.48830074e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(252, 257, 4, 1.45603254e-01, 0.25306794, 118, 118.),  
(253, 256, 4, 1.41069062e-01, 0.86312057, 7, 7.),  
(254, 255, 2, 4.26484033e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(258, 269, 1, 7.80238420e-01, 0.17925607, 111, 111.),

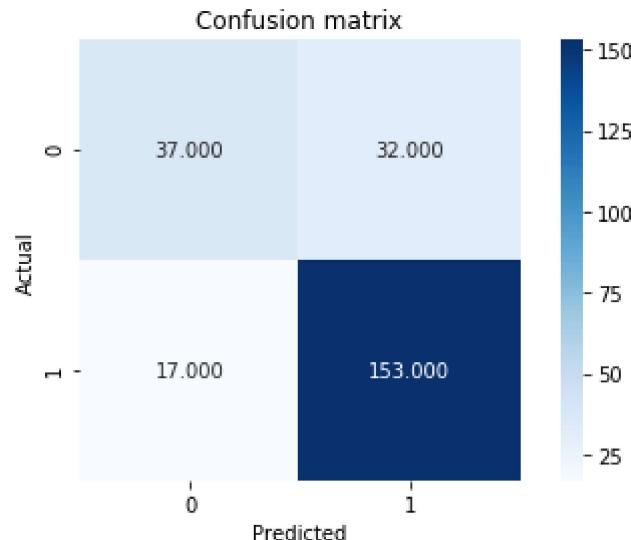
```
(259, 268, 1, 7.79382974e-01, 0.38431154, 40, 40.),
(260, 267, 3, 7.75203481e-02, 0.29181826, 39, 39.),
(261, 262, 3, 5.73914312e-02, 0.45371634, 21, 21.),
(-1, -1, -2, -2.00000000e+00, 0. , 17, 17.),
(263, 264, 1, 7.41444528e-01, 1. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(265, 266, 3, 6.08133189e-02, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 18, 18.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 71, 71.),
(-1, -1, -2, -2.00000000e+00, 0. , 74, 74.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(273, 274, 4, 8.06163996e-02, 0.99729438, 49, 49.),
(-1, -1, -2, -2.00000000e+00, 0. , 11, 11.),
(275, 290, 0, 5.36852255e-02, 0.96778846, 38, 38.),
(276, 277, 0, 3.51146031e-02, 0.99901027, 27, 27.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(278, 289, 4, 3.30905542e-01, 0.99498483, 24, 24.),
(279, 280, 0, 3.57989464e-02, 0.96407876, 18, 18.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(281, 282, 4, 9.94521640e-02, 0.89603823, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(283, 284, 3, 9.15447399e-02, 0.74959526, 14, 14.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(285, 288, 1, 6.58641517e-01, 0.954434 , 8, 8.),
(286, 287, 2, 2.34103978e-01, 0.65002242, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(291, 294, 3, 5.75758908e-02, 0.43949699, 11, 11.),
(292, 293, 4, 1.51426330e-01, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 9, 9.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [152]: # TASK 4  
# Plot decision tree  
dot_data = StringIO()  
tree.export_graphviz(model, out_file = dot_data, feature_names = x.columns)  
figure = pydot.graph_from_dot_data(dot_data.getvalue())  
figure[0].write_pdf("Project2tree.pdf")
```

```
In [153]: # TASK 4  
# Predict class labels using decision tree  
x_test = x_val[['Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or La  
tino",  
                 "Percent Foreign Born", "Percent Rural"]]  
y_pred = model.predict(x_test)
```

```
In [154]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)  
  
[[ 37  32]  
 [ 17 153]]
```

```
In [155]: # TASK 4  
# 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()
```



```
In [156]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7949790794979079  
0.20502092050209209  
[0.68518519 0.82702703]  
[0.53623188 0.9 ]  
[0.60162602 0.86197183]
```

In [157]: # Task 4

```
# Build decision tree with four Parameters
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)
x = x_train[["Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
             "Percent Foreign Born","Percent Rural"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

Out[157]: DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=None,
 max\_features=None, max\_leaf\_nodes=None,
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,
 min\_samples\_leaf=1, min\_samples\_split=2,
 min\_weight\_fraction\_leaf=0.0, presort=False,
 random\_state=0, splitter='best')

```
In [158]: # TASK 4
# Show decision tree
model.tree_.__getstate__()['nodes']
```

```
Out[158]: array([( 1, 98, 0, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 9, 3, 4.29369155e-02, 0.94464539, 218, 218.),
( 3, 8, 1, 1.28731821e-01, 0.39662777, 51, 51.),
( 4, 7, 2, 2.60152347e-01, 0.91829583, 12, 12.),
( 5, 6, 0, 3.35271969e-01, 0.72192809, 5, 5.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0. , 39, 39.),
( 10, 25, 2, 5.31811956e-02, 0.99251207, 167, 167.),
( 11, 20, 1, 1.66045949e-02, 0.79732651, 29, 29.),
( 12, 15, 3, 7.19376117e-01, 0.99572745, 13, 13.),
( 13, 14, 3, 5.34084231e-01, 0.65002242, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 16, 19, 2, 3.28101981e-02, 0.59167278, 7, 7.),
( 17, 18, 1, 4.42281365e-03, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 21, 22, 0, 5.79278052e-01, 0.33729007, 16, 16.),
(-1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
( 23, 24, 0, 5.82103640e-01, 0.65002242, 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 26, 67, 0, 5.34610420e-01, 0.96085769, 138, 138.),
( 27, 28, 2, 6.63515702e-02, 0.88946639, 88, 88.),
(-1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
( 29, 64, 1, 1.65527709e-01, 0.92240626, 80, 80.),
( 30, 31, 3, 7.19443858e-02, 0.96353598, 67, 67.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
( 32, 53, 2, 1.32442802e-01, 0.9235786 , 62, 62.),
( 33, 52, 1, 6.48660623e-02, 0.98999279, 34, 34.),
( 34, 51, 0, 5.15293539e-01, 0.94807824, 30, 30.),
( 35, 36, 2, 8.54628347e-02, 0.87671629, 27, 27.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
( 37, 50, 1, 5.77327460e-02, 0.95871188, 21, 21.),
( 38, 49, 1, 4.94624116e-02, 0.99107606, 18, 18.),
( 39, 40, 2, 9.02317390e-02, 0.91829583, 15, 15.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( 41, 42, 0, 3.25684026e-01, 0.86312057, 14, 14.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
( 43, 46, 2, 1.13319114e-01, 0.97095059, 10, 10.)])
```

( 44, 45, 0, 3.62960249e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 47, 48, 2, 1.26400325e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 54, 55, 0, 3.67759630e-01, 0.74959526, 28, 28.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( 56, 63, 0, 5.20866960e-01, 0.98522814, 14, 14.),  
( 57, 62, 2, 2.26727799e-01, 0.97095059, 10, 10.),  
( 58, 59, 3, 1.85122855e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 60, 61, 3, 3.57810378e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 65, 66, 2, 7.49860108e-02, 0.39124356, 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( 68, 83, 0, 5.70503086e-01, 0.99884554, 50, 50.),  
( 69, 78, 3, 4.08189476e-01, 0.81127812, 24, 24.),  
( 70, 71, 2, 1.37628861e-01, 0.54356444, 16, 16.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( 72, 77, 1, 2.41790533e-01, 0.86312057, 7, 7.),  
( 73, 74, 2, 1.99734472e-01, 0.65002242, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 75, 76, 2, 2.31149949e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 79, 82, 0, 5.60255438e-01, 1. , 8, 8.),  
( 80, 81, 1, 1.64485224e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 84, 97, 3, 4.15849045e-01, 0.89049164, 26, 26.),  
( 85, 90, 3, 1.20978422e-01, 0.81127812, 24, 24.),  
( 86, 89, 2, 2.43148088e-01, 0.98522814, 7, 7.),

```
( 87,  88,  0,  5.74865937e-01,  0.72192809,   5,   5.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   4,   4.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   2,   2.),  
( 91,  92,   3,  3.10056701e-01,  0.52255937,  17,  17.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,  13,  13.),  
( 93,  94,   0,  5.81760198e-01,  1.          ,   4,   4.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( 95,  96,   0,  5.93336910e-01,  0.91829583,   3,   3.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   2,   2.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   2,   2.),  
( 99, 106,   3,  8.06163996e-02,  0.63079708,  738, 738.),  
(100, 101,   2,  3.56555246e-02,  0.76420451,  18,  18.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   3,   3.),  
(102, 105,   1,  3.17645189e-02,  0.35335934,  15,  15.),  
(103, 104,   1,  9.62324743e-03,  1.          ,   2,   2.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,  13,  13.),  
(107, 192,   0,  7.00566828e-01,  0.59218544,  720, 720.),  
(108, 123,   1,  6.54346612e-03,  0.81977896,  184, 184.),  
(109, 110,   3,  2.70538598e-01,  0.3860189 ,  53,  53.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
(111, 122,   2,  3.05352015e-02,  0.3182153 ,  52,  52.),  
(112, 113,   2,  2.12864587e-02,  0.50325833,  27,  27.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,  16,  16.),  
(114, 115,   3,  4.76039827e-01,  0.84535094,  11, 11.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
(116, 121,   2,  2.99604014e-02,  0.72192809,  10,  10.),  
(117, 118,   1,  2.06988020e-03,  0.50325833,   9,   9.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   6,   6.),  
(119, 120,   1,  2.54247384e-03,  0.91829583,   3,   3.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   2,   2.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   1,   1.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,  25,  25.),  
(124, 125,   0,  6.07824951e-01,  0.91312249,  131, 131.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   7,   7.),  
(126, 127,   0,  6.11639887e-01,  0.9311541 ,  124, 124.),  
( -1,  -1,  -2, -2.00000000e+00,  0.          ,   4,   4.),  
(128, 131,   0,  6.23542339e-01,  0.90973612,  120, 120.),  
(129, 130,   2,  2.70351037e-01,  0.35335934,   15,  15.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(132, 191, 1, 3.86196002e-01, 0.94425293, 105, 105.),  
(133, 138, 2, 1.89808952e-02, 0.92156087, 101, 101.),  
(134, 137, 0, 6.60109490e-01, 0.72192809, 5, 5.),  
(135, 136, 1, 1.35322078e-02, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(139, 190, 3, 8.71422321e-01, 0.89603823, 96, 96.),  
(140, 143, 1, 7.83072785e-03, 0.9219957 , 89, 89.),  
(141, 142, 0, 6.51350588e-01, 0.72192809, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(144, 149, 1, 1.45553285e-02, 0.89262301, 84, 84.),  
(145, 148, 0, 6.38902605e-01, 0.48546076, 19, 19.),  
(146, 147, 0, 6.29487514e-01, 1. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
(150, 189, 0, 6.98249251e-01, 0.95007963, 65, 65.),  
(151, 188, 1, 1.84076220e-01, 0.9235786 , 62, 62.),  
(152, 175, 0, 6.74433678e-01, 0.954434 , 56, 56.),  
(153, 154, 0, 6.26806319e-01, 0.99777247, 36, 36.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(155, 164, 2, 6.44632764e-02, 0.98999279, 34, 34.),  
(156, 163, 3, 4.75948066e-01, 0.83664074, 15, 15.),  
(157, 162, 3, 3.71416211e-01, 0.97095059, 10, 10.),  
(158, 159, 1, 9.28375907e-02, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(160, 161, 2, 3.90717573e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(165, 174, 2, 2.02571407e-01, 0.98194079, 19, 19.),  
(166, 173, 2, 1.15186553e-01, 0.93666738, 17, 17.),  
(167, 172, 2, 1.02909397e-01, 0.99572745, 13, 13.),  
(168, 169, 1, 7.42100738e-02, 0.76420451, 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(170, 171, 3, 2.42394403e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),

( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(176, 187, 2, 4.78640925e-02, 0.72192809, 20, 20.),  
(177, 186, 2, 3.99038568e-02, 0.9456603 , 11, 11.),  
(178, 185, 2, 3.13996645e-02, 0.76420451, 9, 9.),  
(179, 184, 2, 3.05371033e-02, 0.91829583, 6, 6.),  
(180, 183, 2, 2.42617354e-02, 0.72192809, 5, 5.),  
(181, 182, 0, 6.89344764e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(193, 324, 1, 7.06888705e-01, 0.48303068, 536, 536.),  
(194, 287, 1, 3.17869987e-02, 0.46180433, 532, 532.),  
(195, 210, 1, 4.54190909e-03, 0.54247024, 321, 321.),  
(196, 209, 2, 5.19529685e-01, 0.32275696, 85, 85.),  
(197, 208, 2, 2.58555235e-02, 0.27619543, 84, 84.),  
(198, 207, 2, 2.53207870e-02, 0.43949699, 44, 44.),  
(199, 200, 0, 7.94335842e-01, 0.36505519, 43, 43.),  
( -1, -1, -2, -2.00000000e+00, 0. , 28, 28.),  
(201, 206, 1, 1.54128450e-03, 0.72192809, 15, 15.),  
(202, 205, 1, 5.95394813e-04, 0.97095059, 5, 5.),  
(203, 204, 0, 7.97934055e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 40, 40.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(211, 254, 3, 8.90801877e-01, 0.60558248, 236, 236.),  
(212, 223, 2, 3.29185687e-02, 0.49958441, 182, 182.),  
(213, 222, 2, 1.92432692e-02, 0.27376917, 85, 85.),  
(214, 221, 3, 6.57445103e-01, 0.49418293, 37, 37.),  
(215, 216, 1, 5.62833413e-03, 0.74248757, 19, 19.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(217, 218, 1, 1.53482840e-02, 0.65002242, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 11, 11.),  
(219, 220, 2, 1.64827146e-02, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 48, 48.),  
(224, 225, 0, 7.39092171e-01, 0.64601748, 97, 97.),  
( -1, -1, -2, -2.00000000e+00, 0. , 24, 24.),  
(226, 227, 2, 3.30698583e-02, 0.75866386, 73, 73.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(228, 239, 2, 7.30122402e-02, 0.71625839, 71, 71.),  
(229, 232, 1, 5.67361014e-03, 0.48546076, 38, 38.),  
(230, 231, 2, 3.69869638e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(233, 238, 0, 8.54643106e-01, 0.31599713, 35, 35.),  
(234, 235, 1, 2.64190491e-02, 0.19143325, 34, 34.),  
( -1, -1, -2, -2.00000000e+00, 0. , 31, 31.),  
(236, 237, 1, 2.89261770e-02, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(240, 241, 3, 4.03863952e-01, 0.88496364, 33, 33.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(242, 253, 3, 6.75047547e-01, 0.99836367, 21, 21.),  
(243, 246, 0, 7.75460303e-01, 0.99107606, 18, 18.),  
(244, 245, 3, 4.19467732e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
(247, 248, 1, 1.35784517e-02, 0.9456603 , 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(249, 252, 2, 1.08335506e-01, 0.91829583, 6, 6.),  
(250, 251, 3, 6.42112494e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(255, 278, 0, 8.17651898e-01, 0.85240518, 54, 54.),  
(256, 277, 2, 7.92986006e-02, 0.96729478, 33, 33.),  
(257, 258, 0, 7.09055752e-01, 0.99226664, 29, 29.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),

(259, 268, 2, 2.75962753e-02, 1. , 26, 26.),  
(260, 267, 1, 1.10812467e-02, 0.84535094, 11, 11.),  
(261, 266, 0, 7.99633890e-01, 0.98522814, 7, 7.),  
(262, 265, 0, 7.75794744e-01, 0.97095059, 5, 5.),  
(263, 264, 1, 9.64698987e-03, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(269, 276, 0, 7.65780747e-01, 0.91829583, 15, 15.),  
(270, 271, 1, 5.90227498e-03, 1. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(272, 273, 2, 5.58517370e-02, 0.954434 , 8, 8.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(274, 275, 2, 7.50163123e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(279, 286, 2, 1.62287718e-02, 0.45371634, 21, 21.),  
(280, 285, 2, 1.41094243e-02, 0.81127812, 8, 8.),  
(281, 282, 0, 8.86305660e-01, 0.59167278, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(283, 284, 0, 9.06445205e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(288, 323, 3, 7.23074585e-01, 0.31490085, 211, 211.),  
(289, 292, 2, 1.442444060e-02, 0.42837817, 137, 137.),  
(290, 291, 0, 8.48830074e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(293, 310, 0, 7.55131125e-01, 0.38295767, 134, 134.),  
(294, 305, 2, 7.75203481e-02, 0.68920199, 38, 38.),  
(295, 302, 1, 1.54103719e-01, 0.86312057, 21, 21.),  
(296, 297, 3, 6.19184166e-01, 0.67229482, 17, 17.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
(298, 299, 1, 5.00671677e-02, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(300, 301, 3, 6.58602625e-01, 0.91829583, 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),

```
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(303, 304, 3, 4.96454626e-01, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(306, 309, 0, 7.14342445e-01, 0.32275696, 17, 17.),
(307, 308, 0, 7.11310685e-01, 0.81127812, 4, 4.),
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),
(311, 318, 3, 2.30634443e-01, 0.20062232, 96, 96.),
(312, 317, 3, 2.26805329e-01, 0.59167278, 14, 14.),
(313, 314, 2, 4.50359598e-01, 0.39124356, 13, 13.),
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),
(315, 316, 1, 6.18490763e-02, 0.91829583, 3, 3.),
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(319, 322, 0, 7.81281948e-01, 0.09501725, 82, 82.),
(320, 321, 0, 7.79382974e-01, 0.29747225, 19, 19.),
( -1, -1, -2, -2.00000000e+00, 0. , 18, 18.),
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
( -1, -1, -2, -2.00000000e+00, 0. , 63, 63.),
( -1, -1, -2, -2.00000000e+00, 0. , 74, 74.),
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.)],
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

In [159]:

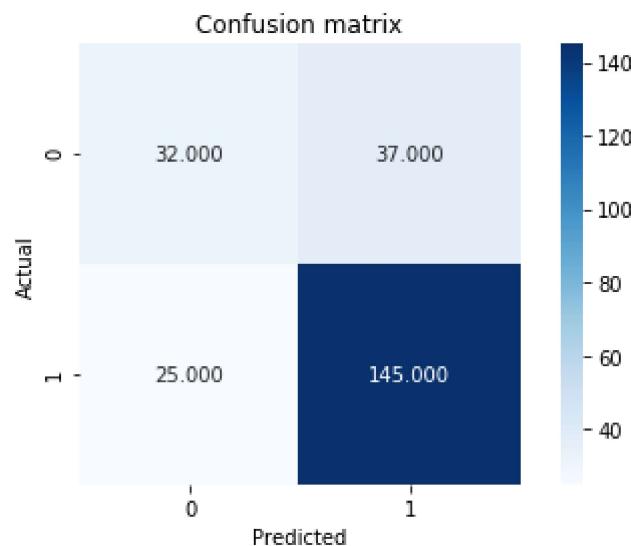
```
# TASK 4
# Predict class labels using decision tree
x_test = x_val[["Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino",
                 "Percent Foreign Born", "Percent Rural"]]
y_pred = model.predict(x_test)
```

In [160]:

```
# TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 32  37]
 [ 25 145]]
```

```
In [161]: # TASK 4  
# 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()
```



```
In [162]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.7405857740585774  
0.2594142259414226  
[0.56140351 0.7967033 ]  
[0.46376812 0.85294118]  
[0.50793651 0.82386364]

```
In [163]: # Task 4  
# Build decision tree with all Parameters  
model = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 0)  
x = x_train  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

```
Out[163]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,  
max_features=None, max_leaf_nodes=None,  
min_impurity_decrease=0.0, min_impurity_split=None,  
min_samples_leaf=1, min_samples_split=2,  
min_weight_fraction_leaf=0.0, presort=False,  
random_state=0, splitter='best')
```

```
In [164]: # TASK 4  
# Show decision tree  
model.tree_.__getstate__()['nodes']
```

```
Out[164]: array([( 1, 78, 11, 6.00609809e-01, 0.83826453, 956, 956.),
( 2, 15, 1, 6.77590042e-01, 0.94464539, 218, 218.),
( 3, 4, 12, 4.91339378e-02, 0.49291578, 65, 65.),
(-1, -1, -2, -2.00000000e+00, 0. , 38, 38.),
( 5, 10, 9, 3.21665928e-01, 0.82562653, 27, 27.),
( 6, 7, 3, 1.48114860e-01, 0.97095059, 10, 10.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
( 8, 9, 1, 6.41990334e-01, 0.59167278, 7, 7.),
(-1, -1, -2, -2.00000000e+00, 0. , 6, 6.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(11, 14, 12, 6.34181444e-02, 0.32275696, 17, 17.),
(12, 13, 5, 7.58136928e-01, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 13, 13.),
(16, 33, 11, 3.97715032e-01, 0.99750255, 153, 153.),
(17, 22, 9, 2.11251423e-01, 0.69312742, 43, 43.),
(18, 19, 10, 3.36324815e-02, 0.954434 , 8, 8.),
(-1, -1, -2, -2.00000000e+00, 0. , 4, 4.),
(20, 21, 9, 1.97683074e-01, 0.81127812, 4, 4.),
(-1, -1, -2, -2.00000000e+00, 0. , 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(23, 24, 11, 3.18483472e-01, 0.42200052, 35, 35.),
(-1, -1, -2, -2.00000000e+00, 0. , 21, 21.),
(25, 30, 3, 6.16376624e-02, 0.74959526, 14, 14.),
(26, 27, 10, 9.13845077e-02, 0.43949699, 11, 11.),
(-1, -1, -2, -2.00000000e+00, 0. , 9, 9.),
(28, 29, 5, 6.65342689e-01, 1. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(31, 32, 4, 1.60660572e-01, 0.91829583, 3, 3.),
(-1, -1, -2, -2.00000000e+00, 0. , 2, 2.),
(-1, -1, -2, -2.00000000e+00, 0. , 1, 1.),
(34, 63, 8, 3.61500695e-01, 0.98059744, 110, 110.),
(35, 60, 7, 6.00444168e-01, 0.99403021, 66, 66.),
(36, 37, 0, 1.84667693e-03, 0.954434 , 56, 56.),
(-1, -1, -2, -2.00000000e+00, 0. , 5, 5.),
(38, 59, 3, 1.08709380e-01, 0.89742719, 51, 51.),
(39, 40, 6, 3.55239794e-01, 0.84265788, 48, 48.),
(-1, -1, -2, -2.00000000e+00, 0. , 8, 8.),
(41, 46, 6, 4.03946847e-01, 0.90973612, 40, 40.),
(42, 45, 2, 2.20890976e-02, 0.91829583, 9, 9.),
(43, 44, 10, 8.54855105e-02, 0.81127812, 4, 4.),
```

```
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( 47, 48, 9, 2.76295856e-01, 0.77062907, 31, 31.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.),  
( 49, 58, 9, 4.07899201e-01, 0.88654089, 23, 23.),  
( 50, 57, 6, 5.60184479e-01, 0.98869941, 16, 16.),  
( 51, 56, 8, 2.80780345e-01, 0.89049164, 13, 13.),  
( 52, 55, 10, 1.52072743e-01, 0.98522814, 7, 7.),  
( 53, 54, 4, 9.94872451e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 61, 62, 10, 9.12366472e-02, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( 64, 77, 4, 2.67057054e-01, 0.77322667, 44, 44.),  
( 65, 66, 11, 4.86225545e-01, 0.70246655, 42, 42.),  
( -1, -1, -2, -2.00000000e+00, 0. , 12, 12.),  
( 67, 72, 0, 1.54390410e-02, 0.83664074, 30, 30.),  
( 68, 71, 6, 5.08045778e-01, 0.99107606, 9, 9.),  
( 69, 70, 11, 5.54864705e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( 73, 74, 0, 5.26874084e-02, 0.59167278, 21, 21.),  
( -1, -1, -2, -2.00000000e+00, 0. , 14, 14.),  
( 75, 76, 3, 9.18121561e-02, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( 79, 82, 1, 2.09422484e-01, 0.63079708, 738, 738.),  
( 80, 81, 11, 9.16850775e-01, 0.37123233, 14, 14.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( 83, 210, 0, 3.41651589e-02, 0.59370934, 724, 724.),  
( 84, 209, 2, 7.00087726e-01, 0.52407294, 677, 677.),  
( 85, 182, 10, 2.66254365e-01, 0.49675653, 670, 670.),  
( 86, 87, 1, 4.74234059e-01, 0.60742881, 416, 416.),
```

( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( 88, 99, 2, 4.54258593e-03, 0.58626931, 412, 412.),  
( 89, 98, 11, 6.91277385e-01, 0.19902377, 97, 97.),  
( 90, 97, 11, 6.90150946e-01, 0.43949699, 33, 33.),  
( 91, 96, 5, 6.88500553e-01, 0.33729007, 32, 32.),  
( 92, 95, 1, 9.79626954e-01, 0.65002242, 12, 12.),  
( 93, 94, 0, 2.97788926e-03, 0.43949699, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 20, 20.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 64, 64.),  
(100, 177, 11, 8.01030606e-01, 0.66812733, 315, 315.),  
(101, 176, 8, 3.61569554e-01, 0.72941136, 265, 265.),  
(102, 137, 11, 6.92940772e-01, 0.75764604, 247, 247.),  
(103, 104, 0, 2.43641390e-03, 0.91829583, 78, 78.),  
( -1, -1, -2, -2.00000000e+00, 0. , 13, 13.),  
(105, 106, 4, 2.12041056e-02, 0.97095059, 65, 65.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(107, 136, 9, 4.97614488e-01, 0.93406806, 60, 60.),  
(108, 135, 5, 7.67381281e-01, 0.89974376, 57, 57.),  
(109, 134, 8, 3.52348730e-01, 0.93666738, 51, 51.),  
(110, 133, 5, 7.63258338e-01, 0.91134238, 49, 49.),  
(111, 112, 2, 7.23552890e-03, 0.87867449, 47, 47.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(113, 124, 2, 4.40534689e-02, 0.83664074, 45, 45.),  
(114, 115, 5, 6.40000820e-01, 0.59167278, 28, 28.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(116, 123, 12, 3.93836856e-01, 0.50325833, 27, 27.),  
(117, 118, 9, 2.71754757e-01, 0.84535094, 11, 11.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(119, 120, 7, 3.10442835e-01, 0.98522814, 7, 7.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(121, 122, 5, 6.87297910e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
(125, 132, 0, 2.50942614e-02, 0.99750255, 17, 17.),  
(126, 131, 6, 4.76447165e-01, 0.98522814, 14, 14.),  
(127, 128, 11, 6.78615510e-01, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(129, 130, 2, 5.34386653e-02, 0.72192809, 5, 5.),

( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(138, 149, 9, 2.18830310e-01, 0.64772749, 169, 169.),  
(139, 146, 11, 7.56240815e-01, 0.97095059, 30, 30.),  
(140, 141, 9, 6.42391481e-02, 0.6098403 , 20, 20.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
(142, 143, 7, 5.70671797e-01, 0.30954343, 18, 18.),  
( -1, -1, -2, -2.00000000e+00, 0. , 16, 16.),  
(144, 145, 9, 1.38802387e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(147, 148, 7, 5.45738980e-01, 0.46899559, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(150, 175, 8, 2.74933115e-01, 0.51513327, 139, 139.),  
(151, 154, 2, 7.13361707e-03, 0.60866968, 107, 107.),  
(152, 153, 6, 3.62128571e-01, 1. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(155, 156, 2, 1.44751538e-02, 0.51009308, 97, 97.),  
( -1, -1, -2, -2.00000000e+00, 0. , 37, 37.),  
(157, 158, 8, 1.95865132e-01, 0.68731509, 60, 60.),  
( -1, -1, -2, -2.00000000e+00, 0. , 17, 17.),  
(159, 174, 12, 6.87577665e-01, 0.82036364, 43, 43.),  
(160, 165, 0, 8.86657555e-03, 0.91829583, 33, 33.),  
(161, 164, 7, 3.88882667e-01, 0.91829583, 9, 9.),  
(162, 163, 11, 7.24570930e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(166, 169, 5, 6.38583511e-01, 0.73828487, 24, 24.),  
(167, 168, 4, 2.94545302e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(170, 173, 11, 7.08289564e-01, 0.46899559, 20, 20.),  
(171, 172, 10, 1.75122499e-01, 0.97095059, 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),

( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 15, 15.),  
( -1, -1, -2, -2.00000000e+00, 0. , 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 32, 32.),  
( -1, -1, -2, -2.00000000e+00, 0. , 18, 18.),  
(178, 181, 10, 1.78502090e-01, 0.14144054, 50, 50.),  
(179, 180, 4, 3.09000220e-02, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 46, 46.),  
(183, 208, 8, 1.65414274e-01, 0.25725482, 254, 254.),  
(184, 185, 9, 3.82322073e-01, 0.45238162, 116, 116.),  
( -1, -1, -2, -2.00000000e+00, 0. , 34, 34.),  
(186, 205, 5, 8.15051436e-01, 0.56870087, 82, 82.),  
(187, 198, 3, 1.58847636e-02, 0.47707131, 78, 78.),  
(188, 197, 6, 4.18999135e-01, 0.81127812, 24, 24.),  
(189, 196, 10, 4.21925426e-01, 0.97095059, 15, 15.),  
(190, 191, 2, 8.87389528e-03, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
(192, 195, 10, 3.51518914e-01, 0.98522814, 7, 7.),  
(193, 194, 5, 6.64997518e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(199, 200, 8, 4.30403650e-02, 0.22853814, 54, 54.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
(201, 204, 5, 3.02078500e-01, 0.1350362 , 53, 53.),  
(202, 203, 10, 4.23595771e-01, 1. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 51, 51.),  
(206, 207, 10, 3.01944286e-01, 0.81127812, 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 1, 1.),  
( -1, -1, -2, -2.00000000e+00, 0. , 3, 3.),  
( -1, -1, -2, -2.00000000e+00, 0. , 138, 138.),  
( -1, -1, -2, -2.00000000e+00, 0. , 7, 7.),  
(211, 212, 12, 8.06163996e-02, 0.99967343, 47, 47.),  
( -1, -1, -2, -2.00000000e+00, 0. , 9, 9.),  
(213, 226, 7, 3.94287720e-01, 0.96778846, 38, 38.),  
(214, 225, 6, 6.07856989e-01, 1. , 30, 30.),  
(215, 216, 10, 1.43117152e-01, 0.97095059, 25, 25.),

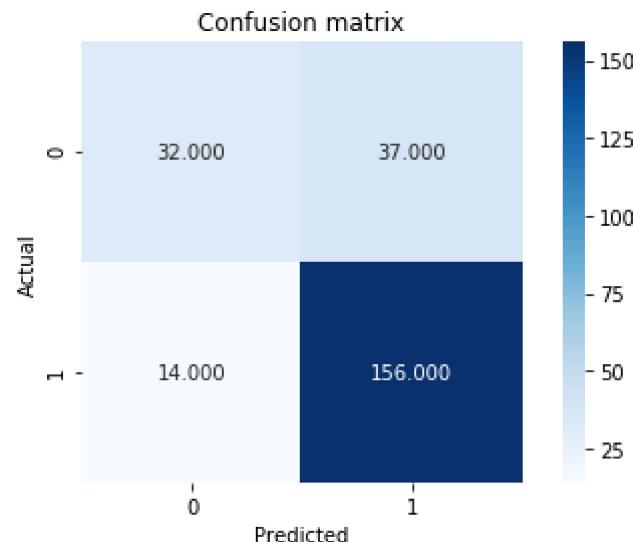
```
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
(217, 224, 8, 3.24228883e-01, 1. , 20, 20.),  
(218, 223, 3, 6.30933437e-02, 0.954434 , 16, 16.),  
(219, 220, 12, 1.51426330e-01, 0.97095059, 10, 10.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
(221, 222, 12, 3.58148322e-01, 0.91829583, 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 2, 2.),  
( -1, -1, -2, -2.00000000e+00, 0. , 6, 6.),  
( -1, -1, -2, -2.00000000e+00, 0. , 4, 4.),  
( -1, -1, -2, -2.00000000e+00, 0. , 5, 5.),  
( -1, -1, -2, -2.00000000e+00, 0. , 8, 8.)],  
dtype=[('left_child', '<i8'), ('right_child', '<i8'), ('feature', '<i8'), ('threshold', '<f8'), ('impurity', '<f8'), ('n_node_samples', '<i8'), ('weighted_n_node_samples', '<f8')])
```

```
In [165]: # TASK 4  
# Predict class labels using decision tree  
x_test = x_val  
y_pred = model.predict(x_test)
```

```
In [166]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 32 37]  
 [ 14 156]]
```

```
In [167]: # TASK 4  
# 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()
```



```
In [168]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7866108786610879  
0.21338912133891208  
[0.69565217 0.80829016]  
[0.46376812 0.91764706]  
[0.55652174 0.85950413]
```

```
In [169]: # TASK 4  
# The Decision tree with paramaters:  
# "Total Population", "Percent Less than Bachelor's Degree", "Percent Black, not Hispanic or Latino", and  
# "Percent Foreign Born", "Percent Rural"  
# is the best classifier in case of decision tree classifiers in terms of F1 Score [0.60162602 0.86197183]
```

```
In [ ]:
```

```
In [ ]:
```

```
In [170]: x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 956 entries, 1002 to 1065  
Data columns (total 13 columns):  
Total Population           956 non-null float64  
Percent White, not Hispanic or Latino 956 non-null float64  
Percent Black, not Hispanic or Latino 956 non-null float64  
Percent Hispanic or Latino    956 non-null float64  
Percent Foreign Born        956 non-null float64  
Percent Female              956 non-null float64  
Percent Age 29 and Under    956 non-null float64  
Percent Age 65 and Older    956 non-null float64  
Median Household Income     956 non-null float64  
Percent Unemployed          956 non-null float64  
Percent Less than High School Degree 956 non-null float64  
Percent Less than Bachelor's Degree 956 non-null float64  
Percent Rural               956 non-null float64  
dtypes: float64(13)  
memory usage: 104.6 KB
```

```
In [171]: # TASK 4  
# K nearest neighbour with k = 3 and 1 parameter  
model = KNeighborsClassifier(n_neighbors = 3)  
x = x_train[["Total Population"]]  
y = y_train.iloc[:,2:]  
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[171]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,  
                                weights='uniform')
```

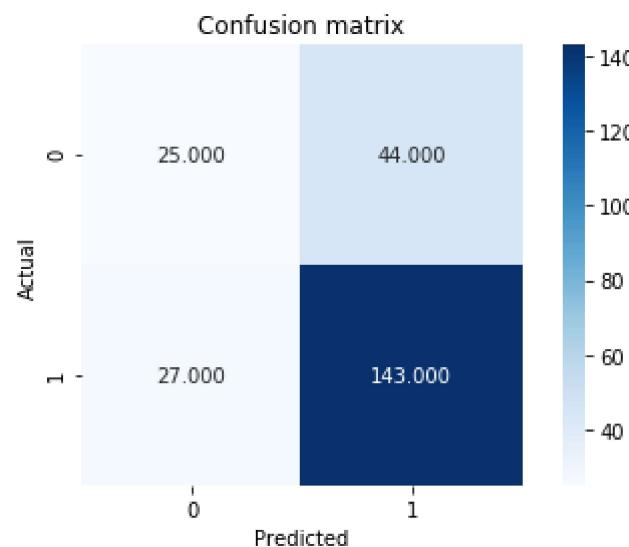
```
In [172]: # TASK 4  
# Predict class labels using KNearest Neighbour  
x_test = x_val[["Total Population"]]  
y_pred = model.predict(x_test)
```

```
In [173]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

[[ 25 44]  
 [ 27 143]]

```
In [174]: # TASK 4  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
In [175]: # TASK 4  
# 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()
```



```
In [176]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.702928870292887  
0.297071129707113  
[0.48076923 0.76470588]  
[0.36231884 0.84117647]  
[0.41322314 0.80112045]

```
In [177]: # TASK 4  
# K nearest neighbour with k = 3 and 2 parameter  
model = KNeighborsClassifier(n_neighbors = 3)  
x = x_train[["Total Population", "Percent White, not Hispanic or Latino"]]  
y = y_train.iloc[:,2:]  
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

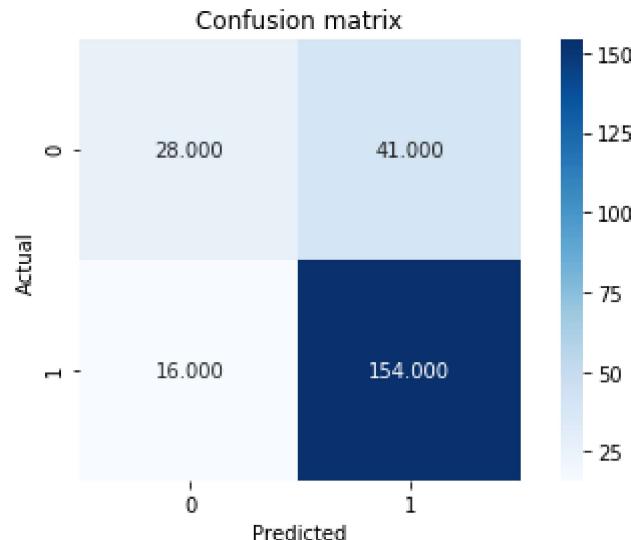
```
Out[177]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,  
                                weights='uniform')
```

```
In [178]: # TASK 4  
# Predict class labels using KNearest Neighbour  
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino"]]  
y_pred = model.predict(x_test)
```

```
In [179]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 28  41]  
 [ 16 154]]
```

```
In [180]: # TASK 4  
# 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()
```



```
In [181]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.7615062761506276  
0.2384937238493724  
[0.63636364 0.78974359]  
[0.4057971 0.90588235]  
[0.49557522 0.84383562]

```
In [182]: # TASK 4
# K nearest neighbour with k = 3 and 3 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

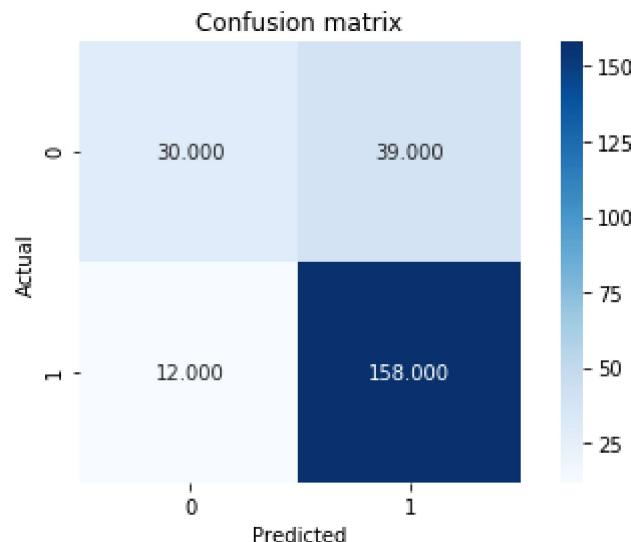
```
Out[182]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                weights='uniform')
```

```
In [183]: # TASK 4
# Predict class Labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino"]]
y_pred = model.predict(x_test)
```

```
In [184]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 30  39]
 [ 12 158]]
```

```
In [185]: # TASK 4  
# 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()
```



```
In [186]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.7866108786610879  
0.21338912133891208  
[0.71428571 0.80203046]  
[0.43478261 0.92941176]  
[0.54054054 0.86103542]

```
In [187]: # TASK 4
# K nearest neighbour with k = 3 and 4 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

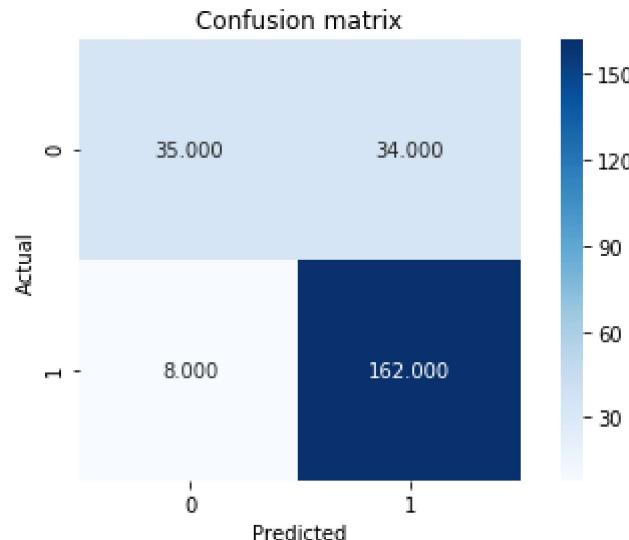
```
Out[187]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                weights='uniform')
```

```
In [188]: # TASK 4
# Predict class Labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino"]]
y_pred = model.predict(x_test)
```

```
In [189]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 35  34]
 [  8 162]]
```

```
In [190]: # TASK 4  
# 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()
```



```
In [191]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.8242677824267782  
0.17573221757322177  
[0.81395349 0.82653061]  
[0.50724638 0.95294118]  
[0.625      0.8852459]
```

```
In [ ]:
```

```
In [192]: # TASK 4
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Foreign Born"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

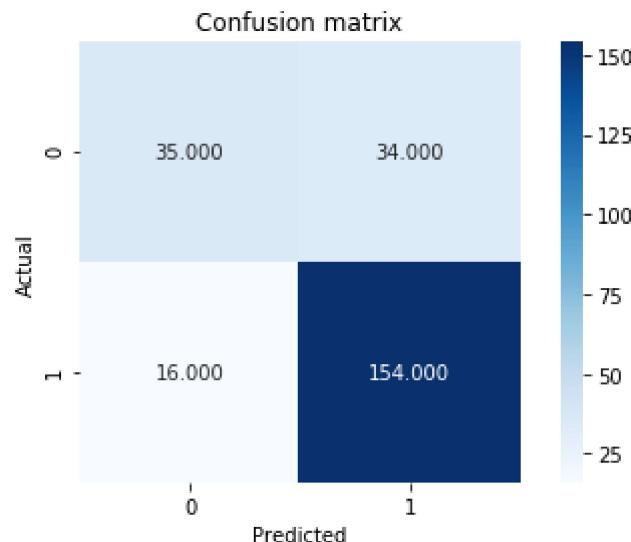
```
Out[192]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                weights='uniform')
```

```
In [193]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Foreign Born"]]
y_pred = model.predict(x_test)
```

```
In [194]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 35  34]
 [ 16 154]]
```

```
In [195]: # TASK 4  
# 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()
```



```
In [196]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

0.7907949790794979  
0.20920502092050208  
[0.68627451 0.81914894]  
[0.50724638 0.90588235]  
[0.58333333 0.8603352 ]

In [197]: # TASK 4

```
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Female"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[197]: KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
metric\_params=None, n\_jobs=None, n\_neighbors=3, p=2,  
weights='uniform')

In [198]: # TASK 4

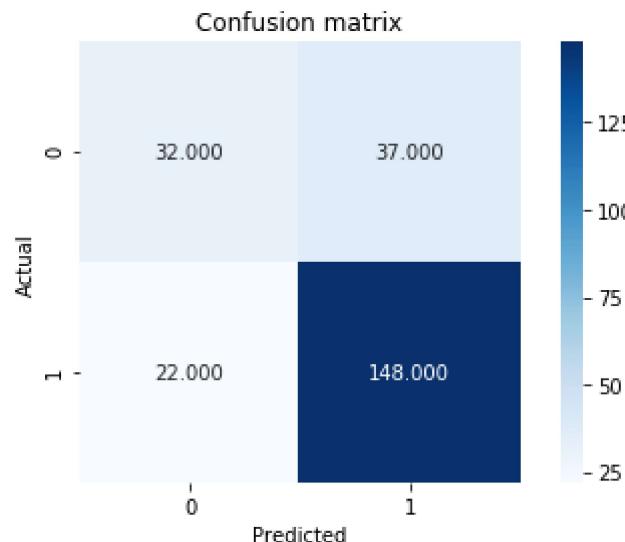
```
# Predict class Labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Female"]]
y_pred = model.predict(x_test)
```

In [199]: # TASK 4

```
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 32  37]
 [ 22 148]]
```

```
In [200]: # TASK 4  
# 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()
```



```
In [201]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7531380753138075  
0.2468619246861925  
[0.59259259 0.8 ]  
[0.46376812 0.87058824]  
[0.5203252 0.83380282]
```

```
In [ ]:
```

In [202]: # TASK 4

```
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Age 29 and Under"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[202]: KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
metric\_params=None, n\_jobs=None, n\_neighbors=3, p=2,  
weights='uniform')

In [203]: # TASK 4

```
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Age 29 and Under"]]
y_pred = model.predict(x_test)
```

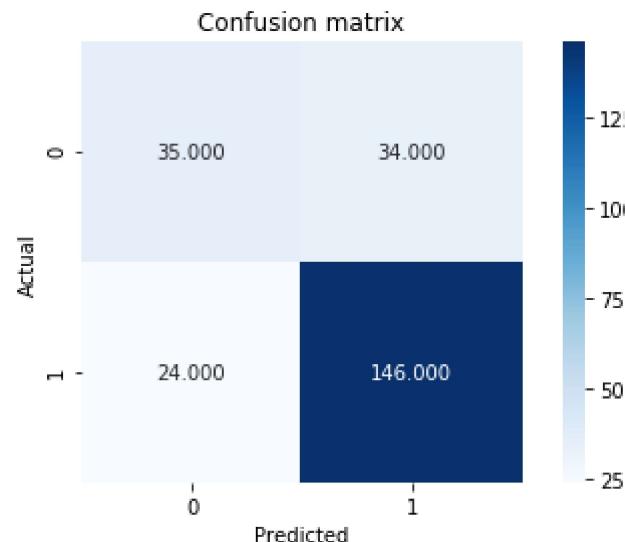
In [204]: # TASK 4

```
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 35  34]
 [ 24 146]]
```

In [205]: # TASK 4

```
# 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()
```



In [206]: # TASK 4

```
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7573221757322176
0.2426778242677824
[0.59322034 0.81111111]
[0.50724638 0.85882353]
[0.546875 0.83428571]
```

In [ ]:

In [207]: # TASK 4

```
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Age 65 and Older"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[207]: KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
metric\_params=None, n\_jobs=None, n\_neighbors=3, p=2,  
weights='uniform')

In [208]: # TASK 4

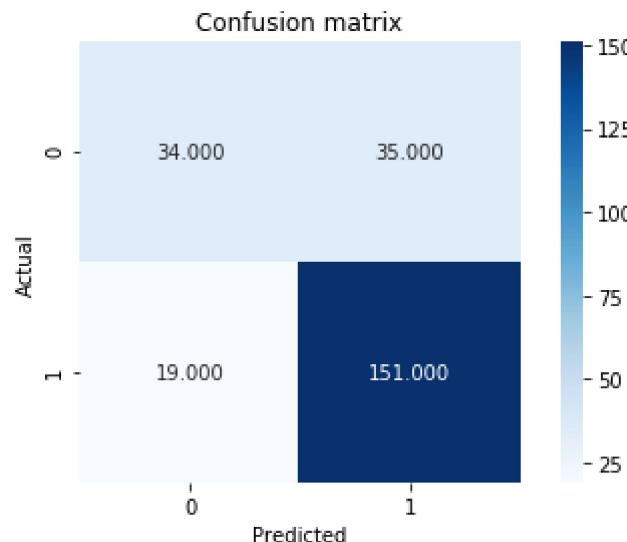
```
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Age 65 and Older"]]
y_pred = model.predict(x_test)
```

In [209]: # TASK 4

```
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 34  35]
 [ 19 151]]
```

```
In [210]: # TASK 4  
# 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()
```



```
In [211]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7740585774058577  
0.2259414225941423  
[0.64150943 0.81182796]  
[0.49275362 0.88823529]  
[0.55737705 0.84831461]
```

```
In [ ]:
```

```
In [212]: # TASK 4
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Median Household Income"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

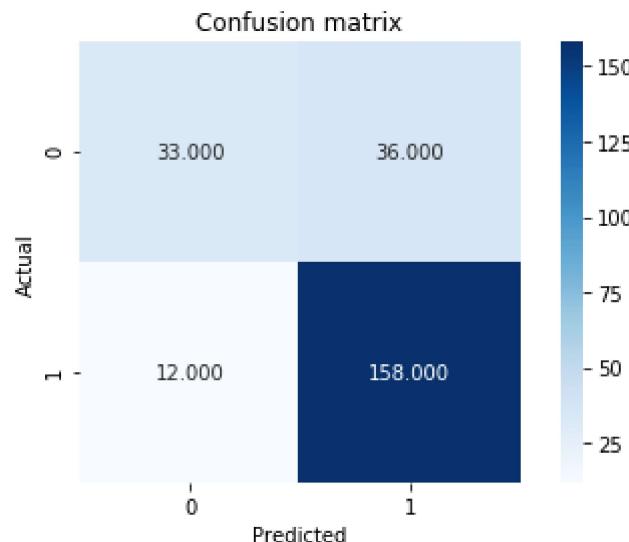
```
Out[212]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                weights='uniform')
```

```
In [213]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Median Household Income"]]
y_pred = model.predict(x_test)
```

```
In [214]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 33  36]
 [ 12 158]]
```

```
In [215]: # TASK 4  
# 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()
```



```
In [216]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.799163179916318  
0.20083682008368198  
[0.73333333 0.81443299]  
[0.47826087 0.92941176]  
[0.57894737 0.86813187]
```

```
In [ ]:
```

```
In [217]: # TASK 4
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Unemployed"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

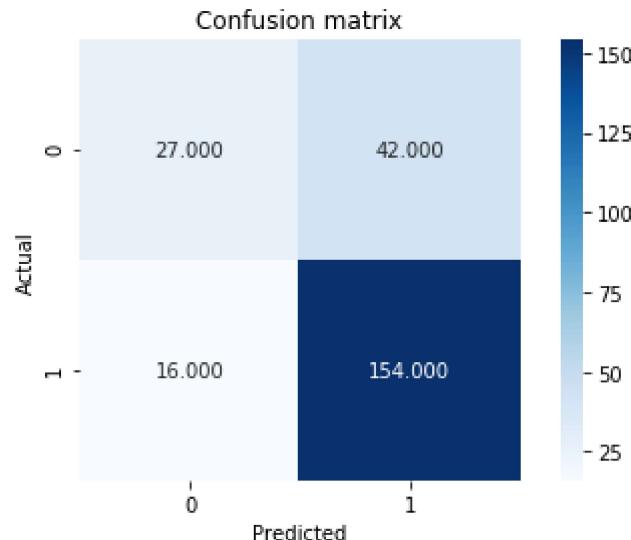
```
Out[217]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                weights='uniform')
```

```
In [218]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino", "Percent Unemployed"]]
y_pred = model.predict(x_test)
```

```
In [219]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 27  42]
 [ 16 154]]
```

```
In [220]: # TASK 4  
# 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()
```



```
In [221]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7573221757322176  
0.2426778242677824  
[0.62790698 0.78571429]  
[0.39130435 0.90588235]  
[0.48214286 0.84153005]
```

```
In [ ]:
```

```
In [222]: # TASK 4
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Less than High School Degree"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

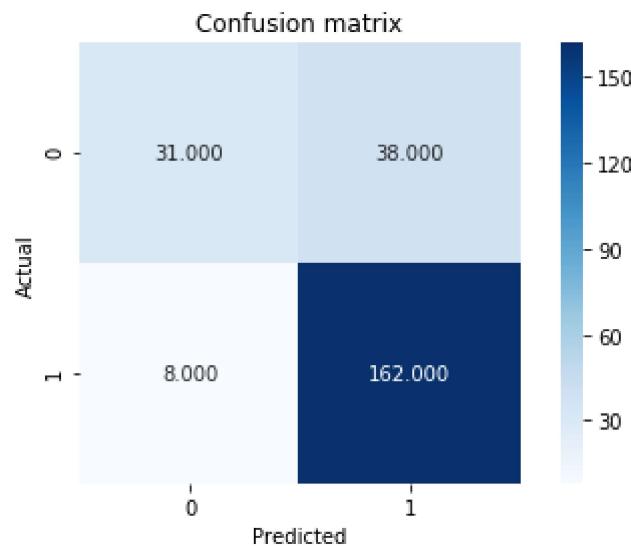
```
Out[222]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                 metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                 weights='uniform')
```

```
In [223]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                 "Percent Less than High School Degree"]]
y_pred = model.predict(x_test)
```

```
In [224]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

[[ 31  38]
 [  8 162]]
```

```
In [225]: # TASK 4  
# 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()
```



```
In [226]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.8075313807531381  
0.19246861924686187  
[0.79487179 0.81      ]  
[0.44927536 0.95294118]  
[0.57407407 0.87567568]
```

In [ ]:

```
In [227]: # TASK 4
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Less than Bachelor's Degree"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[227]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                 metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                 weights='uniform')
```

In [228]: # TASK 4

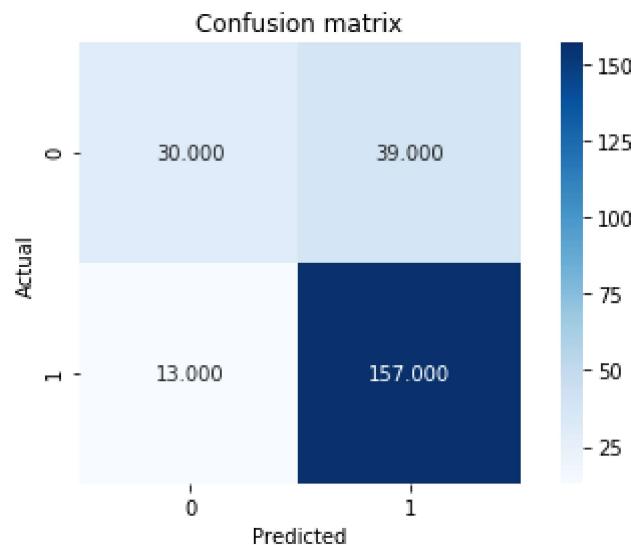
```
# Predict class Labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                 "Percent Less than Bachelor's Degree"]]
y_pred = model.predict(x_test)
```

In [229]: # TASK 4

```
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 30  39]
 [ 13 157]]
```

```
In [230]: # TASK 4  
# 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()
```



```
In [231]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7824267782426778  
0.2175732217573222  
[0.69767442 0.80102041]  
[0.43478261 0.92352941]  
[0.53571429 0.8579235 ]
```

```
In [ ]:
```

```
In [232]: # TASK 4
# K nearest neighbour with k = 3 and 5 parameters
model = KNeighborsClassifier(n_neighbors = 3)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
             "Percent Rural"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

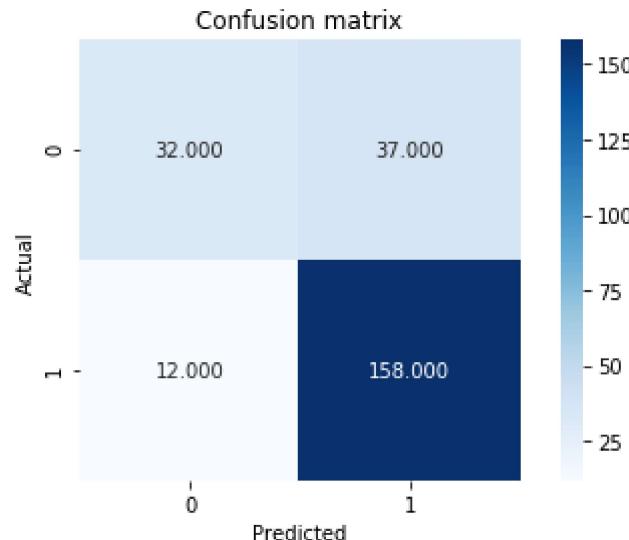
```
Out[232]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                 metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                                 weights='uniform')
```

```
In [233]: # TASK 4
# Predict class Labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                 "Percent Rural"]]
y_pred = model.predict(x_test)
```

```
In [234]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 32  37]
 [ 12 158]]
```

```
In [235]: # TASK 4  
# 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()
```



```
In [236]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.7949790794979079  
0.20502092050209209  
[0.72727273 0.81025641]  
[0.46376812 0.92941176]  
[0.56637168 0.86575342]
```

```
In [ ]:
```

```
In [237]: # TASK 4  
# K nearest neighbour k = 3 and with ALL parameters  
  
model = KNeighborsClassifier(n_neighbors = 3)  
x = x_train  
y = y_train.iloc[:,2:]  
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

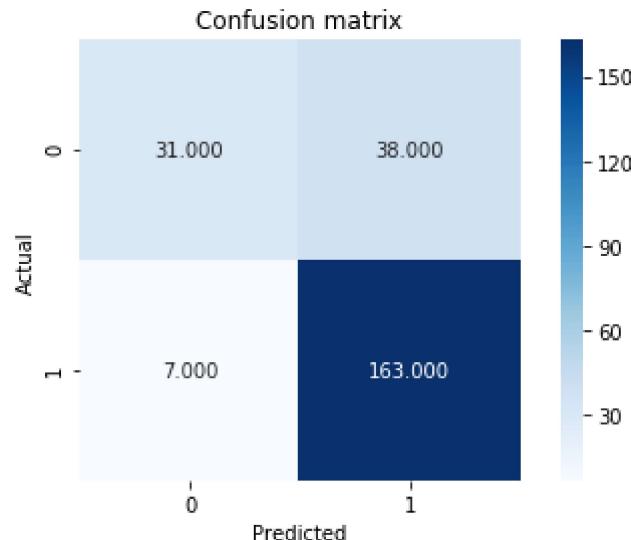
```
Out[237]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                                metric_params=None, n_jobs=None, n_neighbors=3, p=2,  
                                weights='uniform')
```

```
In [238]: # TASK 4  
# Predict class labels using KNearest Neighbour  
x_test = x_val  
y_pred = model.predict(x_test)
```

```
In [239]: # TASK 4  
# Compute confusion matrix  
y_test = y_val.iloc[:, 2:]  
conf_matrix = metrics.confusion_matrix(y_test, y_pred)  
print(conf_matrix)
```

```
[[ 31  38]  
 [ 7 163]]
```

```
In [240]: # TASK 4  
# 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()
```



```
In [241]: # TASK 4  
# Compute evaluation metrics  
print(metrics.accuracy_score(y_test, y_pred)) # accuracy  
print(1 - metrics.accuracy_score(y_test, y_pred)) # error  
print(metrics.precision_score(y_test, y_pred, average = None)) # precision  
print(metrics.recall_score(y_test, y_pred, average = None)) # recall  
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
0.8117154811715481  
0.18828451882845187  
[0.81578947 0.81094527]  
[0.44927536 0.95882353]  
[0.57943925 0.8787062 ]
```

```
In [ ]:
```

```
In [242]: # In case of K Nearest Neighbour classifier with k=3, the classifier with four parameters:  
# "Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino"  
# has the best performance in terms of F1 Scores ([0.625      0.8852459])
```

```
In [ ]:
```

```
In [ ]:
```

```
In [243]: # TASK 4  
# K nearest neighbour with k =4 and with 1 parameter  
model = KNeighborsClassifier(n_neighbors = 4)  
x = x_train[["Total Population"]]  
y = y_train.iloc[:,2:]  
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[243]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,  
                                weights='uniform')
```

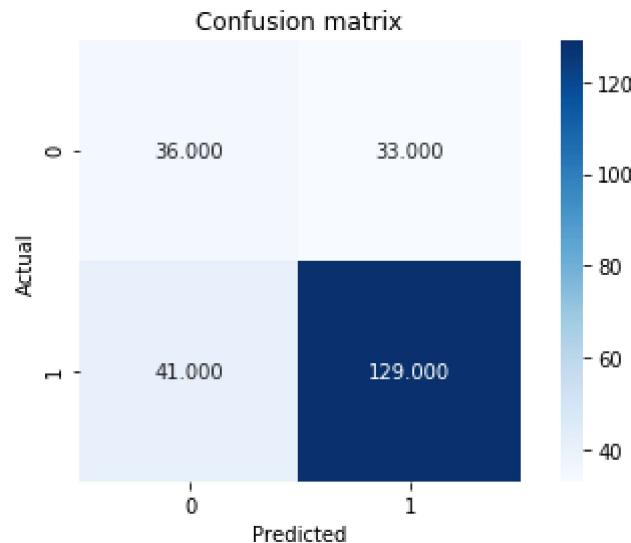
```
In [244]: # TASK 4  
# Predict class labels using KNearest Neighbour [0.49315068 0.77710843]  
x_test = x_val[["Total Population"]]  
y_pred = model.predict(x_test)
```

```
In [245]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 36  33]
 [ 41 129]]
0.6903765690376569
0.3096234309623431
[0.46753247 0.7962963 ]
[0.52173913 0.75882353]
[0.49315068 0.77710843]
```



```
In [246]: # TASK 4
# K nearest neighbour k = 4 and with 2 parameters [0.515625  0.82285714]
model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[['Total Population", "Percent White, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[246]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

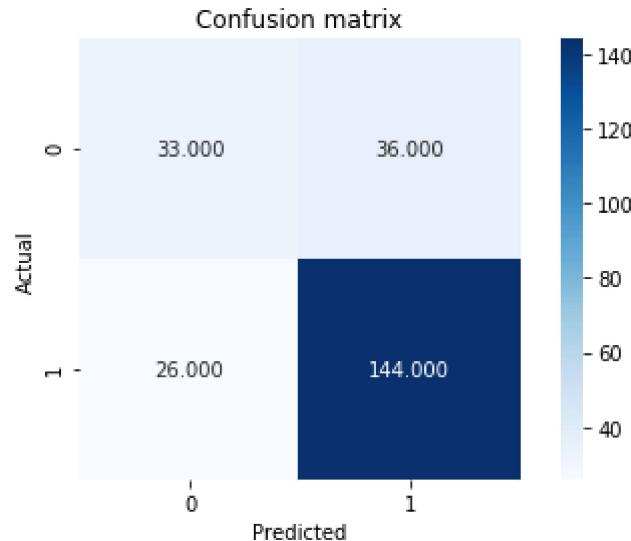
```
In [247]: # TASK 4  
# Predict class labels using KNearest Neighbour  
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino"]]  
y_pred = model.predict(x_test)
```

```
In [248]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 33  36]
 [ 26 144]]
0.7405857740585774
0.2594142259414226
[0.55932203  0.8        ]
[0.47826087  0.84705882]
[0.515625    0.82285714]
```



```
In [249]: # TASK 4
# K nearest neighbour k = 4 and with 3 parameters 0.52307692 / 0.82183908
model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[['Total Population", "Percent White, not Hispanic or Latino","Percent Black, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[249]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

In [250]: # TASK 4

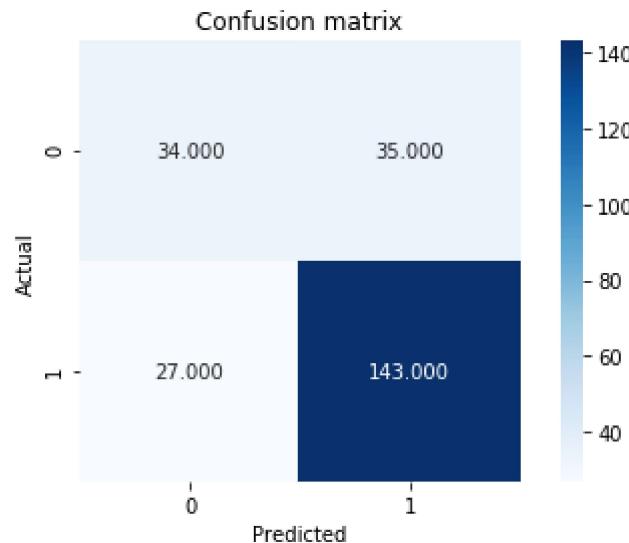
```
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino"]]
y_pred = model.predict(x_test)
```

```
In [251]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 34  35]
 [ 27 143]]
0.7405857740585774
0.2594142259414226
[0.55737705 0.80337079]
[0.49275362 0.84117647]
[0.52307692 0.82183908]
```



```
In [252]: # TASK 4
# K nearest neighbour k = 4 and with 3 parameters [0.55813953 0.83667622]
model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[['Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[252]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

In [253]: # TASK 4

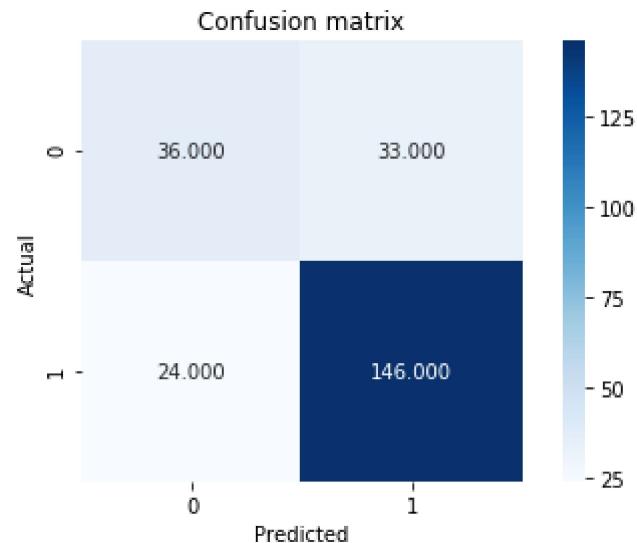
```
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino"]]
y_pred = model.predict(x_test)
```

```
In [254]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 36  33]
 [ 24 146]]
0.7615062761506276
0.2384937238493724
[0.6      0.81564246]
[0.52173913 0.85882353]
[0.55813953 0.83667622]
```



```
In [255]: # TASK 4
# K nearest neighbour k = 4 and with 4 parameters [0.45945946 0.75757576]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[255]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

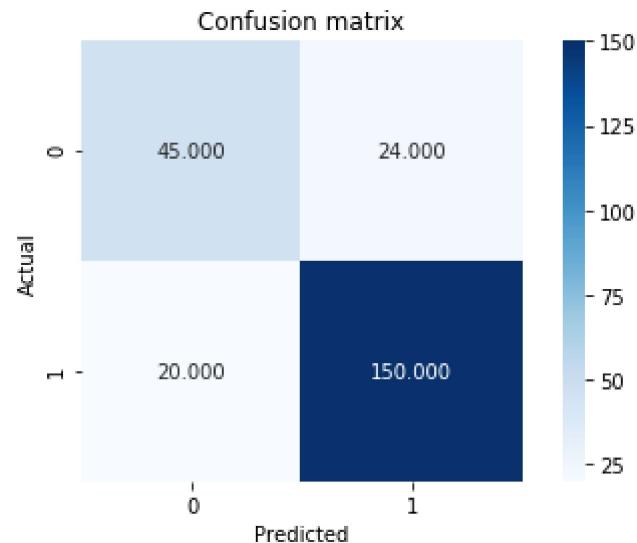
```
In [256]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born"]]
y_pred = model.predict(x_test)
```

```
In [257]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 45  24]
 [ 20 150]]
0.8158995815899581
0.18410041841004188
[0.69230769 0.86206897]
[0.65217391 0.88235294]
[0.67164179 0.87209302]
```



```
In [258]: # TASK 4
# K nearest neighbour k = 4 and with 5 parameters

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Female"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[258]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

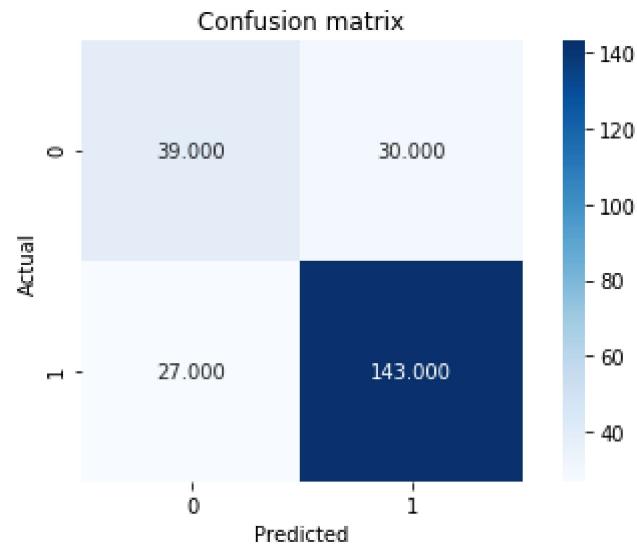
```
In [259]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Female"]]
y_pred = model.predict(x_test)
```

```
In [260]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 39  30]
 [ 27 143]]
0.7615062761506276
0.2384937238493724
[0.59090909 0.8265896 ]
[0.56521739 0.84117647]
[0.57777778 0.83381924]
```



```
In [261]: # TASK 4
# K nearest neighbour k = 4 and with 5 parameters [0.42758621 0.75075075]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Age 29 and Under"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[261]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

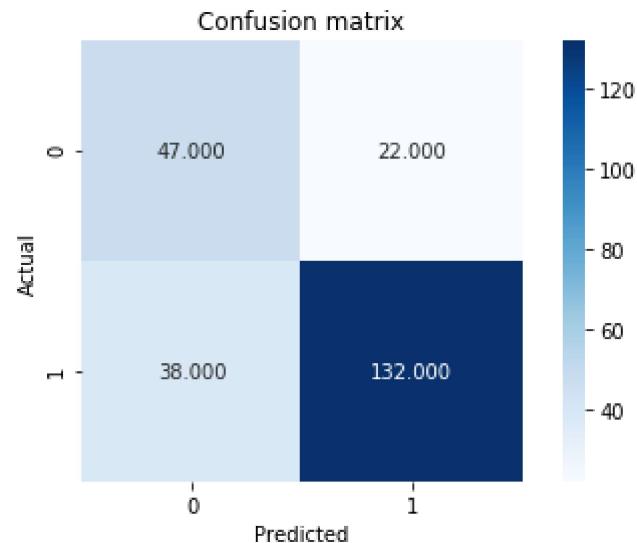
```
In [262]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Age 29 and Under"]]
y_pred = model.predict(x_test)
```

```
In [263]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 47  22]
 [ 38 132]]
0.7489539748953975
0.2510460251046025
[0.55294118 0.85714286]
[0.68115942 0.77647059]
[0.61038961 0.81481481]
```



```
In [264]: # TASK 4
# K nearest neighbour k = 4 and with 5 parameters [[0.45112782 0.7884058 ]]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Age 65 and Older"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[264]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

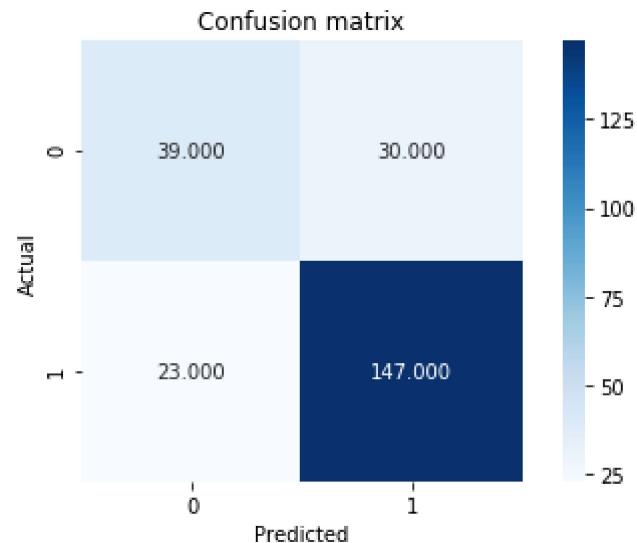
```
In [265]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Age 65 and Older"]]
y_pred = model.predict(x_test)
```

```
In [266]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 39  30]
 [ 23 147]]
0.7782426778242678
0.22175732217573219
[0.62903226 0.83050847]
[0.56521739 0.86470588]
[0.59541985 0.84726225]
```



```
In [267]: # TASK 4
# K nearest neighbour k = 4 and with 5 parameters [0.48920863 0.79056047]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Median Household Income"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[267]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

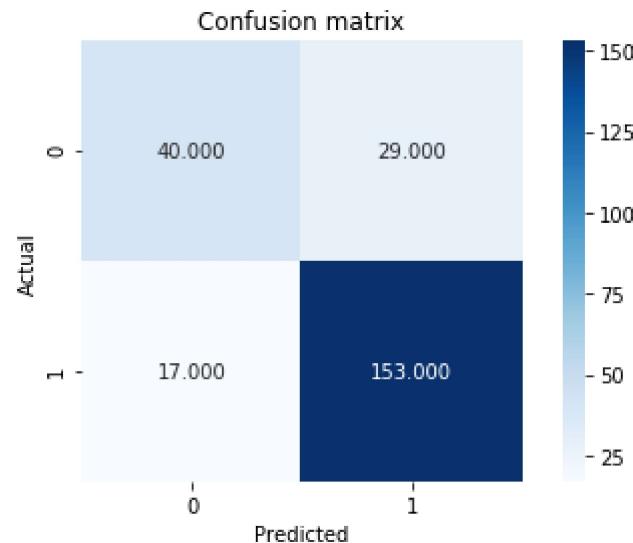
```
In [268]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born", "Median Household Income"]]
y_pred = model.predict(x_test)
```

```
In [269]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 40  29]
 [ 17 153]]
0.8075313807531381
0.19246861924686187
[0.70175439 0.84065934]
[0.57971014 0.9      ]
[0.63492063 0.86931818]
```



```
In [270]: # TASK 4
# K nearest neighbour k = 4 and with 5 parameters [0.44285714 0.76923077]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Unemployed"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[270]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

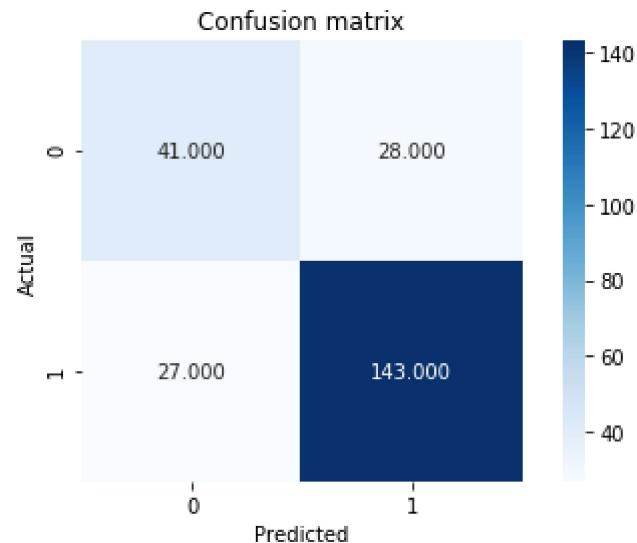
```
In [271]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Unemployed"]]
y_pred = model.predict(x_test)
```

```
In [272]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 41  28]
 [ 27 143]]
0.7698744769874477
0.2301255230125523
[0.60294118 0.83625731]
[0.5942029 0.84117647]
[0.59854015 0.83870968]
```



```
In [273]: # TASK 4
# K nearest neighbour k = 4 and with 5 parameters [0.58267717 0.84900285]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Less than High School Degree"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[273]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

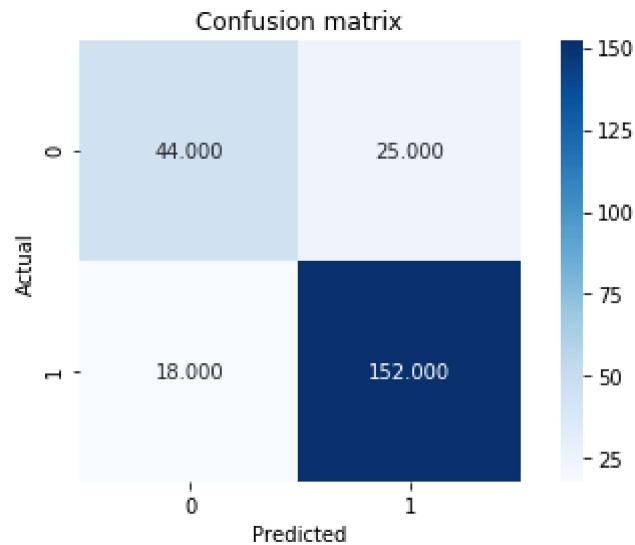
```
In [274]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Less than High School Degree"]]
y_pred = model.predict(x_test)
```

```
In [275]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 44  25]
 [ 18 152]]
0.8200836820083682
0.17991631799163177
[0.70967742 0.85875706]
[0.63768116 0.89411765]
[0.67175573 0.87608069]
```



```
In [276]: # TASK 4
# K nearest neighbour k = 4 and with 6 parameters [0.57575758 0.83815029]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Less than High School Degree","Percent Less than Bachelor's Degree"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[276]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

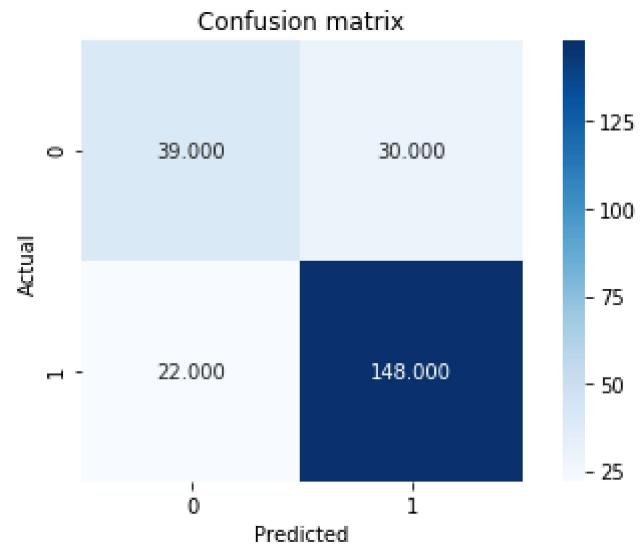
```
In [277]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born",
                 "Percent Less than High School Degree","Percent Less than Bachelor's Degree"]]
y_pred = model.predict(x_test)
```

```
In [278]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 39  30]
 [ 22 148]]
0.7824267782426778
0.2175732217573222
[0.63934426 0.83146067]
[0.56521739 0.87058824]
[0.6          0.85057471]
```



```
In [279]: # TASK 4
# K nearest neighbour k = 4 and with 6 parameters [0.53030303 0.82080925]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born",
             "Percent Less than High School Degree","Percent Rural"]]
y = y_train.iloc[:,2:]
model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:8: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[279]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

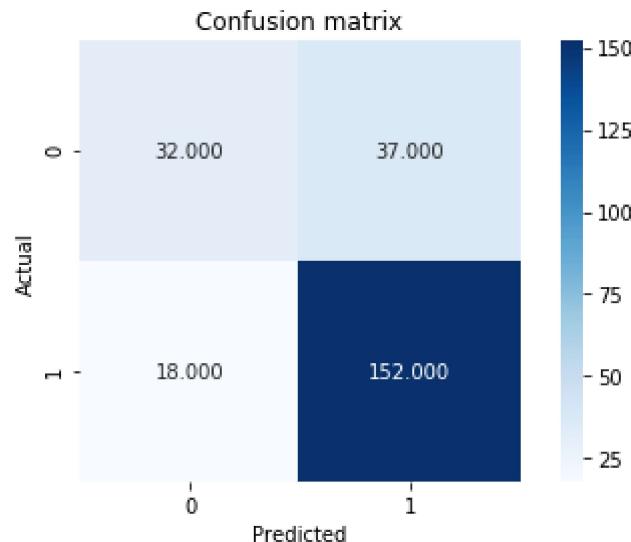
```
In [280]: # TASK 4
# Predict class labels using KNearest Neighbour
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino","Percent Hispanic or Latino","Percent Foreign Born","Percent Less than High School Degree","Percent Rural"]]
y_pred = model.predict(x_test)
```

```
In [281]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 32  37]
 [ 18 152]]
0.7698744769874477
0.2301255230125523
[0.64      0.8042328]
[0.46376812 0.89411765]
[0.53781513 0.84679666]
```



In [ ]:

```
# TASK 4
# K nearest neighbour k = 4 and with ALL parameters [0.53030303 0.82080925]

model = KNeighborsClassifier(n_neighbors = 4)
x = x_train
y = y_train.iloc[:,2:]
m = model.fit(x, y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:7: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

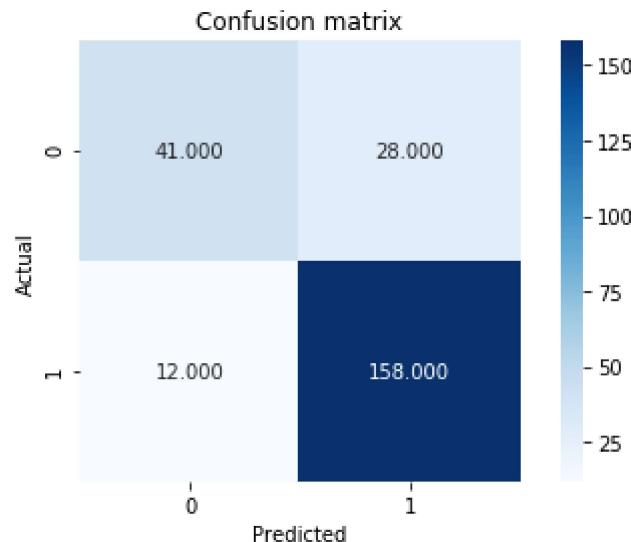
```
In [283]: # TASK 4  
# Predict class labels using KNearest Neighbour  
x_test = x_val  
y_pred = m.predict(x_test)
```

```
In [284]: # TASK 4
# Compute confusion matrix
y_test = y_val.iloc[:, 2:]
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 41  28]
 [ 12 158]]
0.8326359832635983
0.16736401673640167
[0.77358491 0.84946237]
[0.5942029  0.92941176]
[0.67213115 0.88764045]
```



```
In [285]: best_classifier = m
best_classifier
```

```
Out[285]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                                weights='uniform')
```

```
In [286]: #in case of k nearest neighbor classifier with k=4, the model with all the variables:
# has the best performance.
# [0.67213115 0.88764045]
```

```
In [ ]:
```

```
In [ ]:
```

```
In [287]: # TASK 4  
# Naive bayes  
from sklearn.naive_bayes import GaussianNB
```

```
In [288]: # CLASSIFIER: Naive Bayes  
# Initialize Naive Bayes classifier with all parameters  
model = GaussianNB()  
x = x_train  
y = y_train.iloc[:,2:]  
model.fit(x,y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[288]: GaussianNB(priors=None, var\_smoothing=1e-09)

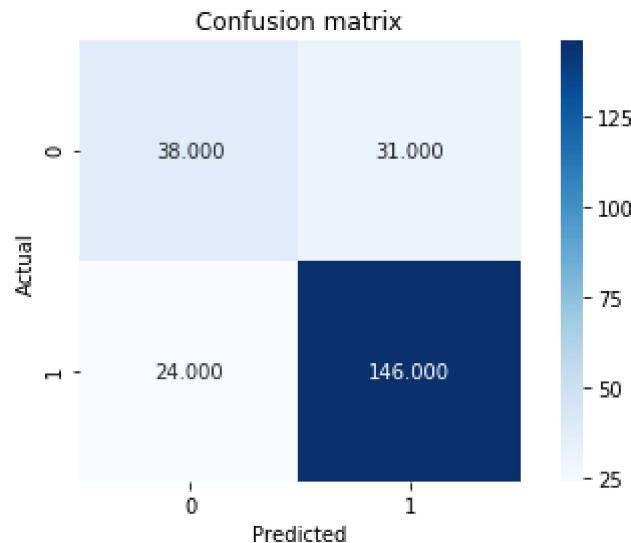
```
In [289]: # Predict class Labels using Naive Bayes classifier  
x_test = x_val  
y_pred = model.predict(x_test)
```

```
In [290]: # Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 38  31]
 [ 24 146]]
0.7698744769874477
0.2301255230125523
[0.61290323 0.82485876]
[0.55072464 0.85882353]
[0.58015267 0.84149856]
```



```
In [291]: # Using 3 parameters
model = GaussianNB()
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Rural"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

```
Out[291]: GaussianNB(priors=None, var_smoothing=1e-09)
```

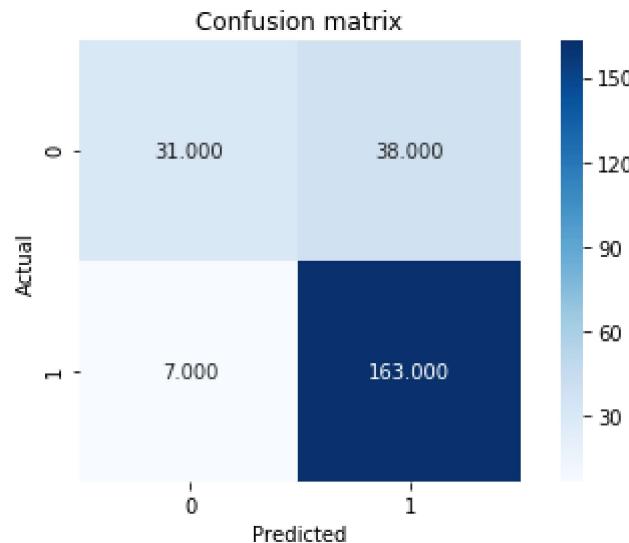
```
In [292]: # Predict class labels using Naive Bayes classifier
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Rural"]]
y_pred = model.predict(x_test)
```

```
In [293]: # Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 31  38]
 [ 7 163]]
0.8117154811715481
0.18828451882845187
[0.81578947 0.81094527]
[0.44927536 0.95882353]
[0.57943925 0.8787062 ]
```



In [294]: # *["Total Population", "Percent White, not Hispanic or Latino", "Percent Rural"]* gives us best F1 score

In [295]: # Using 4 parameters

```
model = GaussianNB()
x = x_train[['Total Population", "Percent White, not Hispanic or Latino", "Percent Rural", "Percent Less than High School Degree"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[295]: GaussianNB(priors=None, var\_smoothing=1e-09)

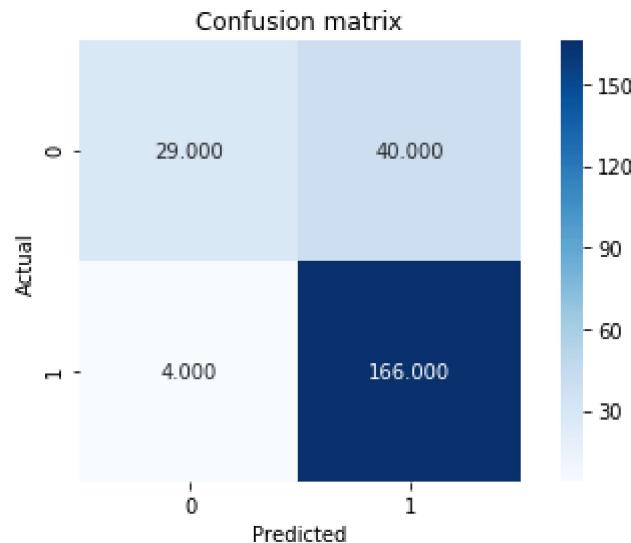
```
In [296]: # Predict class labels using Naive Bayes classifier
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Rural", "Percent Less than High School Degree"]]
y_pred = model.predict(x_test)
```

```
In [297]: # Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 29  40]
 [  4 166]]
0.8158995815899581
0.18410041841004188
[0.87878788 0.80582524]
[0.42028986 0.97647059]
[0.56862745 0.88297872]
```



```
In [298]: # Using 4 parameters
model = GaussianNB()
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Rural", "Percent Unemployed"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[298]: GaussianNB(priors=None, var\_smoothing=1e-09)

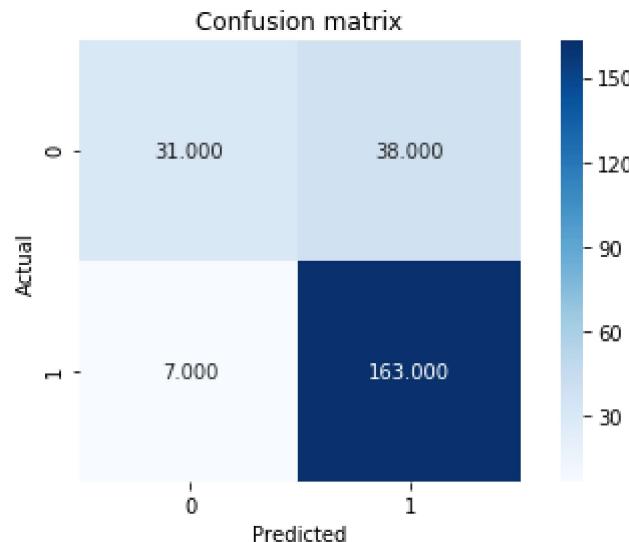
```
In [299]: # Predict class labels using Naive Bayes classifier
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Rural", "Percent Unemployed"]]
y_pred = model.predict(x_test)
```

```
In [300]: # Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 31  38]
 [  7 163]]
0.8117154811715481
0.18828451882845187
[0.81578947 0.81094527]
[0.44927536 0.95882353]
[0.57943925 0.8787062 ]
```



```
In [301]: # Using 5 parameters
model = GaussianNB()
x = x_train[["Total Population", "Percent White, not Hispanic or Latino", "Percent Unemployed", "Percent Rural", "Percent Less than High School Degree"]]
y = y_train.iloc[:,2:]
model.fit(x,y)
```

C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

Out[301]: GaussianNB(priors=None, var\_smoothing=1e-09)

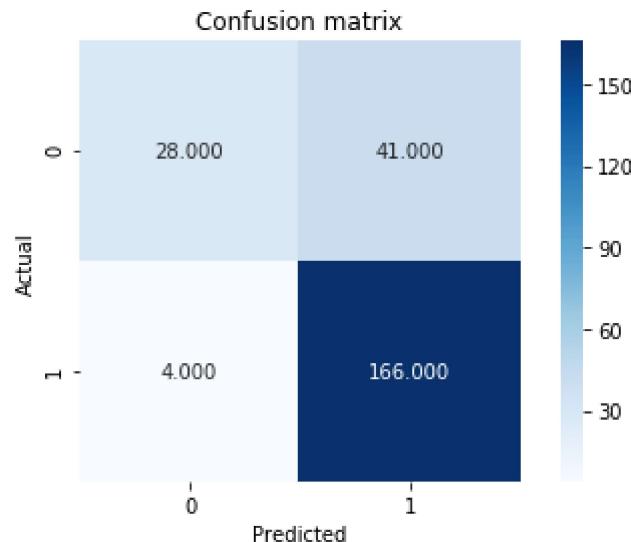
```
In [302]: # Predict class labels using Naive Bayes classifier
x_test = x_val[["Total Population", "Percent White, not Hispanic or Latino", "Percent Unemployed", "Percent Rural", "Percent Less than High School Degree"]]
y_pred = model.predict(x_test)
```

```
In [303]: # Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[ 28  41]
 [  4 166]]
0.8117154811715481
0.18828451882845187
[0.875      0.80193237]
[0.4057971  0.97647059]
[0.55445545 0.8806366 ]
```



```
In [304]: # In case of Naive Bayes Classifier, the classifier with the variables:
# "Total Population", "Percent White, not Hispanic or Latino", "Percent Rural"
# has the best performance [0.57943925 0.8787062 ]
```

```
In [305]: # The K-nearest classifier with K=4 the model with all the variables
# has the best performance in terms of F1 Score [0.67213115 0.88764045].
```

```
In [ ]:
```

```
In [ ]:
```

```
In [306]: # x_train.info()
```

```
In [307]: # CLUSTERING  
#merged.head()
```

```
In [308]: x_train = merged.iloc[:,3:16]  
y_train = merged.iloc[:,16:19]  
y = y_train.iloc[:,2:]  
#x_train.head()
```

```
In [309]: scaler = MinMaxScaler(feature_range=(0, 1))  
x_train[["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"]] = scaler.fit_transform(x_train[["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"]])  
  
# x_train.head()
```

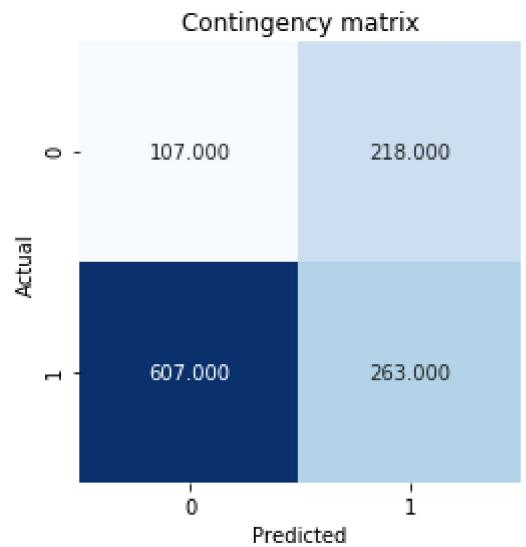
```
In [310]: silhouette_coefficient = metrics.silhouette_score(x_train, y["Party"], metric = "euclidean")  
print(silhouette_coefficient)
```

```
0.1451942421282395
```

```
In [311]: # K-Means with random initial centroids and with all variables
```

```
x = x_train
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



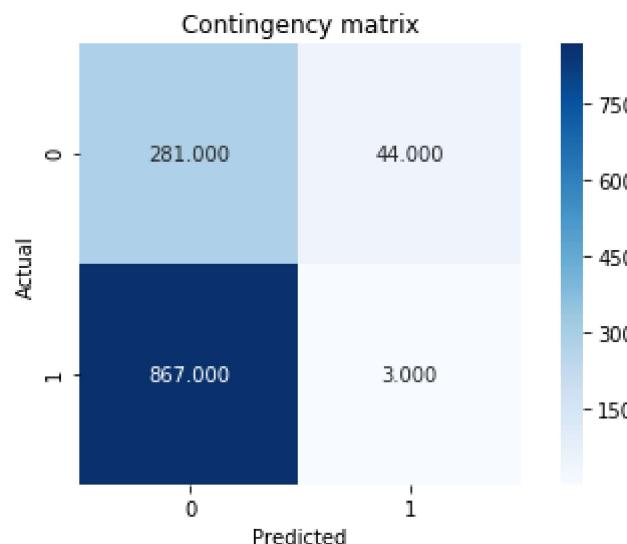
```
In [312]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.13760334763372073, 0.3055385783249025]
```

In [313]: # K-Means with random initial centroids and with one variable

```
x = x_train[["Total Population"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [314]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)

silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")

print([adjusted\_rand\_index, silhouette\_coefficient])

[0.11979747814620154, 0.902954408093495]

In [315]: # This K-Means clustering model has best performance in terms of silhouette coefficient (better cluster quality i.e. high cohesion and separation) [0.903]

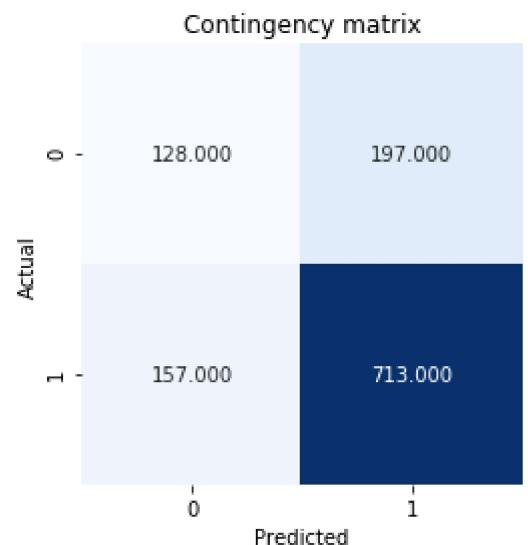
# (with adjusted\_rand\_index value of [0.120])

In [ ]:

In [316]: # K-Means with random initial centroids and with two variables

```
x = x_train[["Total Population", "Percent White, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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

In [317]:  
adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)  
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")  
print([adjusted\_rand\_index, silhouette\_coefficient])

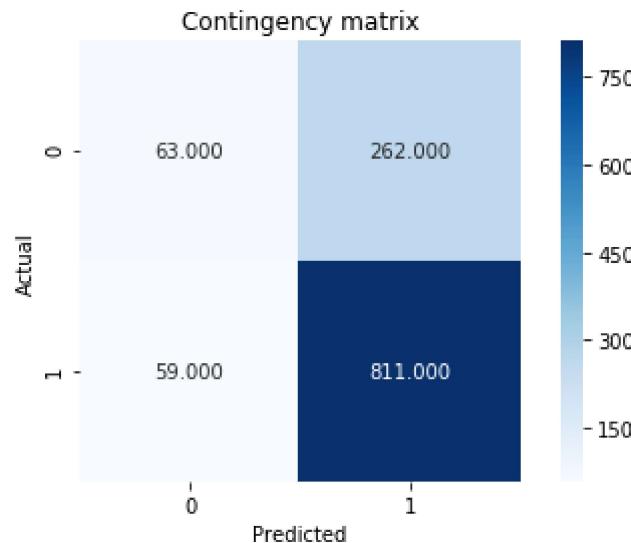
[0.11532085509726435, 0.6763123517609197]

In [ ]:

In [318]: # K-Means with random initial centroids and with 2 variables

```
x = x_train[["Total Population", "Percent Black, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [319]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

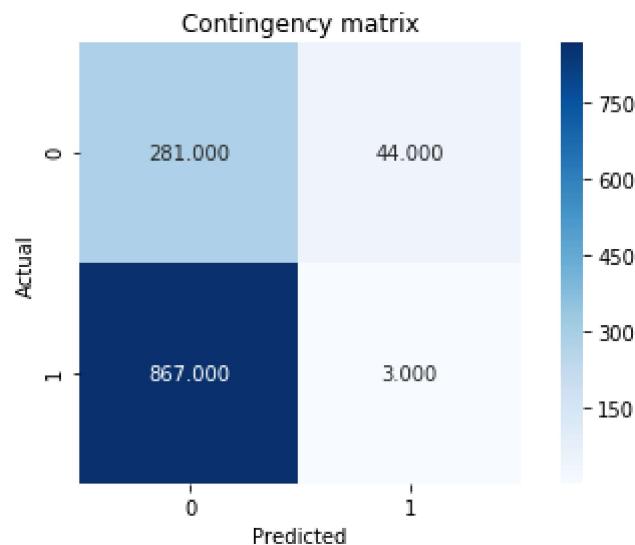
```
[0.09466944416059365, 0.7645110103727762]
```

In [ ]:

In [320]: # K-Means with random initial centroids and with 2 variables

```
x = x_train[["Total Population", "Percent Female"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [321]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

[0.11979747814620154, 0.7534661088840465]

In [322]: # This K-Means clustering model has the best performance

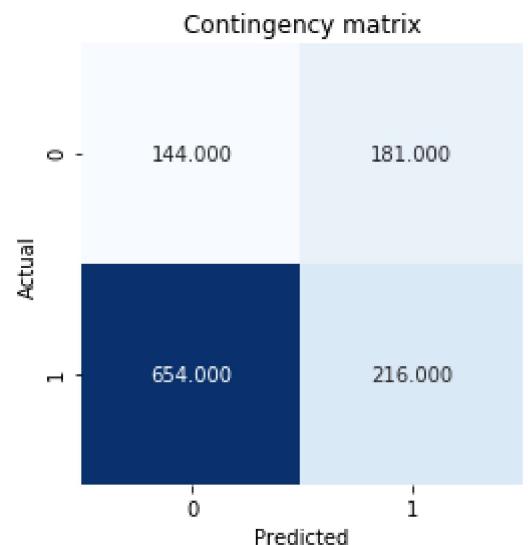
# in terms of both adjusted\_rand\_index (true clusters) and silhouette\_coefficient (cluster quality i.e. cohesion and separation) [0.122, 0.752]

In [ ]:

In [323]: # K-Means with random initial centroids and with 3 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [324]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

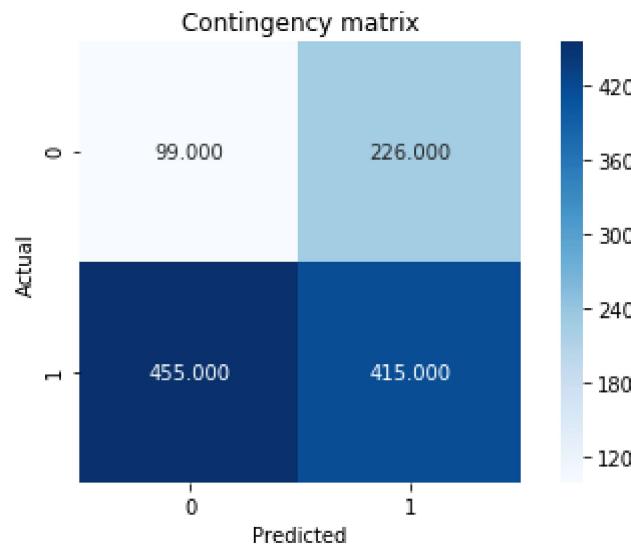
[0.13728423215451566, 0.382499108466535]

In [ ]:

In [325]: # K-Means with random initial centroids and with 4 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under", "Percent Age 65 and Older"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [326]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

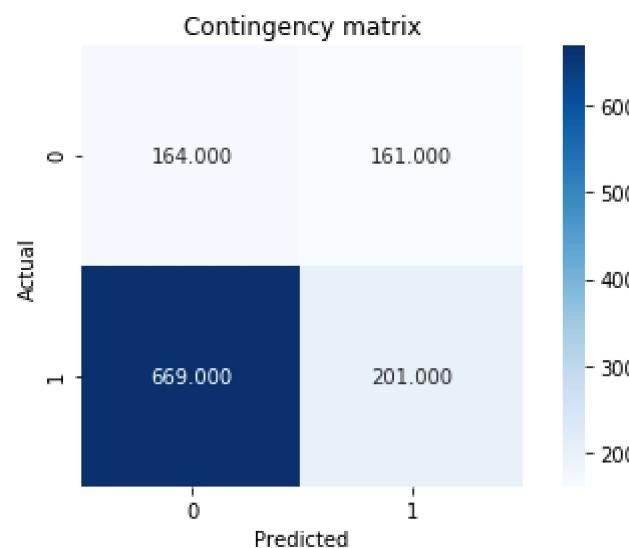
[0.01779888057340007, 0.3903856440381637]

In [ ]:

In [327]: # K-Means with random initial centroids and with 4 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under", "Median Household Income"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [328]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

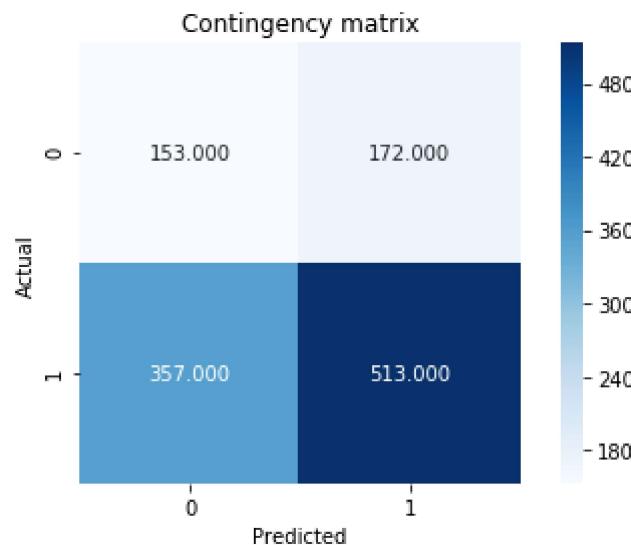
```
[0.12257241875521525, 0.338608328121946]
```

In [ ]:

In [329]: # K-Means with random initial centroids and with 4 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under", "Percent Unemployed"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [330]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

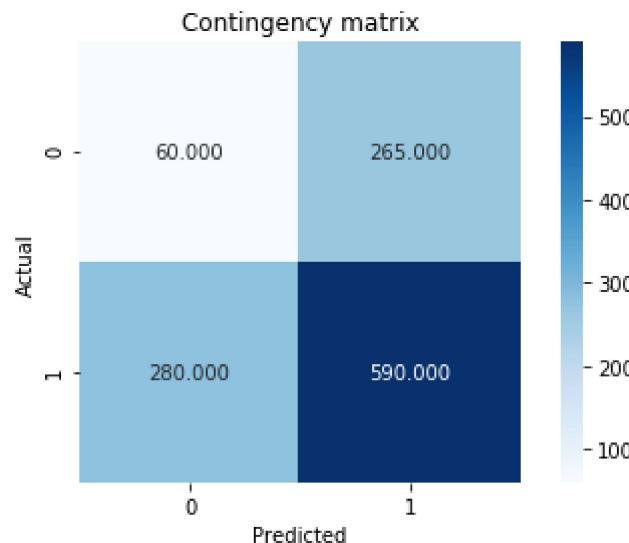
```
[0.008074850837010678, 0.30371309058269136]
```

In [ ]:

In [331]: # K-Means with random initial centroids and with 4 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under", "Percent Less than High School Degree"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [332]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")
print([adjusted\_rand\_index, silhouette\_coefficient])

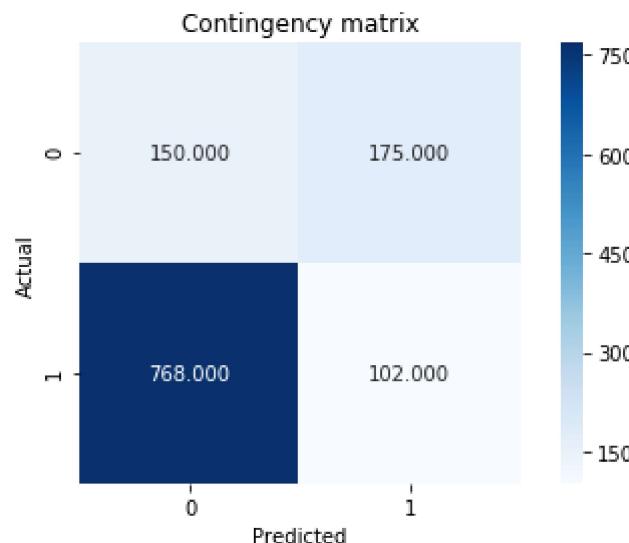
```
[-0.0327328407808789, 0.38246208744863036]
```

In [ ]:

In [333]: # K-Means with random initial centroids and with 4 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under", "Percent Less than Bachelor's Degree"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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



In [334]: adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)

```
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

[0.2916370001659343, 0.43138306521096526]

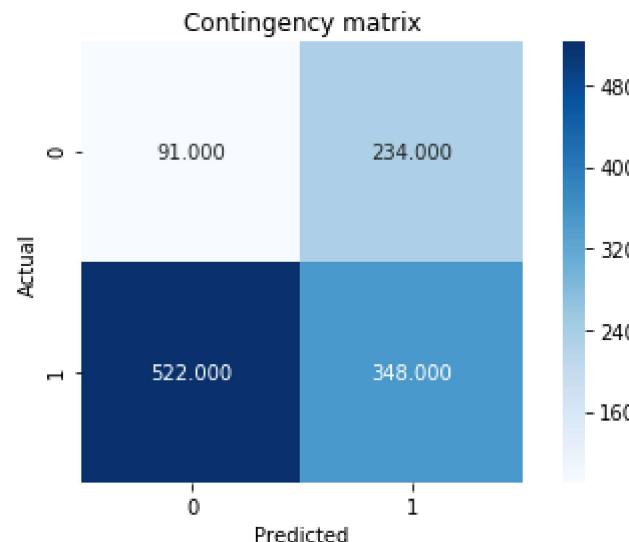
In [335]: # This K-Means clustering model has best performance in terms of adjusted\_rand\_index (true clusters) [0.289]
# (with silhouette coefficient (cluster quality i.e. cohesion and separation) value of [0.432])

In [ ]:

In [336]: # K-Means with random initial centroids and with 4 variables

```
x = x_train[["Total Population", "Percent Female", "Percent Age 29 and Under", "Percent Rural"]]
y = y_train.iloc[:,2:]
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 10).fit(x)
clusters = clustering.labels_

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

In [337]:  
adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)  
silhouette\_coefficient = metrics.silhouette\_score(x, clusters, metric = "euclidean")  
print([adjusted\_rand\_index, silhouette\_coefficient])

[0.06962226634891566, 0.5150160361694801]

In [ ]:

In [338]: # In case of K-Means clustering,

```
# i) the clustering model with 1 variable: "Total Population" has the best performance
#      in terms of silhuette coefficient (better cluster quality i.e. high cohesion and seperation) [0.903]
# (with adjusted_rand_index value of [0.120])

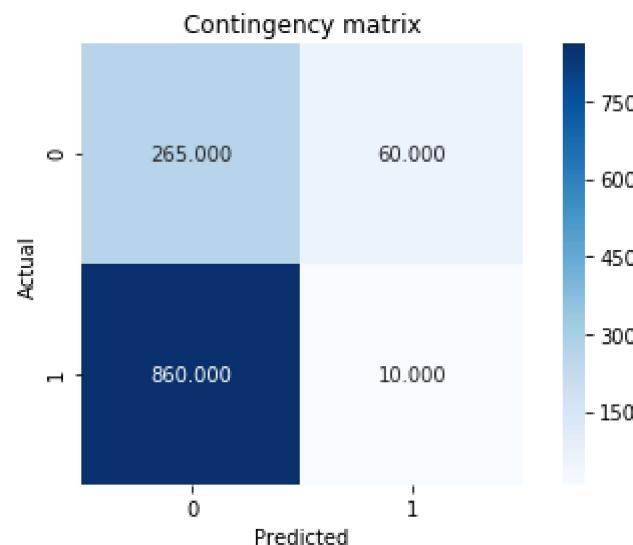
# ii) the clustering model with 4 variables: "Total Population", "Percent Female", "Percent Age 29 and Under",
#      "Percent Less than Bachelor's Degree" has the best performance
#      in terms of adjusted_rand_index (true clusters) [0.289] (with silhuette coefficient (cluster quality
# i.e. cohesion and seperation) value of [0.432])

# iii) the clustering model with 2 variables: "Total Population", "Percent Female" has the best performance
#      in terms of both adjusted_rand_index (true clusters) and silhouette_coefficient (cluster quality i.e.
# cohesion and seperation) [0.122, 0.752]
```

In [ ]:

In [ ]:

```
In [339]: # Task 5  
# Complete Linkage Method with all the variables  
  
x = x_train  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

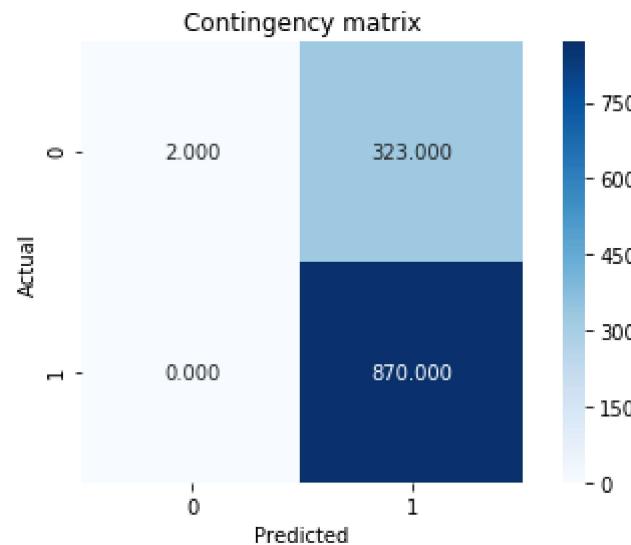


```
In [340]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.15406550837040792, 0.41135505363261715]
```

```
In [ ]:
```

```
In [341]: # Task 5  
# Complete Linkage Method with one variable  
  
x = x_train[["Total Population"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



```
In [342]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

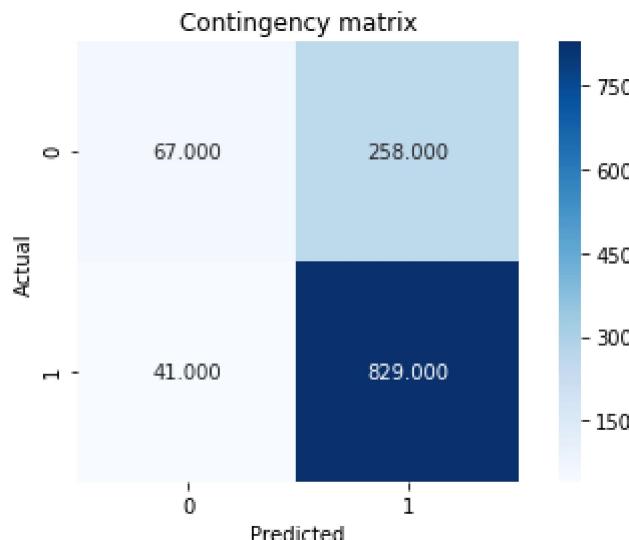
```
[0.005608925119335567, 0.9531008389502824]
```

```
In [343]: # This complete Linkage hierarichal clustering model has best performance  
# in terms of silhuoette coefficient (better cluster quality i.e. high cohesion and seperation) [0.953] (with  
adjusted_rand_index value of [0.0056])
```

```
In [ ]:
```

```
In [344]: # Task 5  
# Complete Linkage Method with 1 variable
```

```
x = x_train[["Percent White, not Hispanic or Latino"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



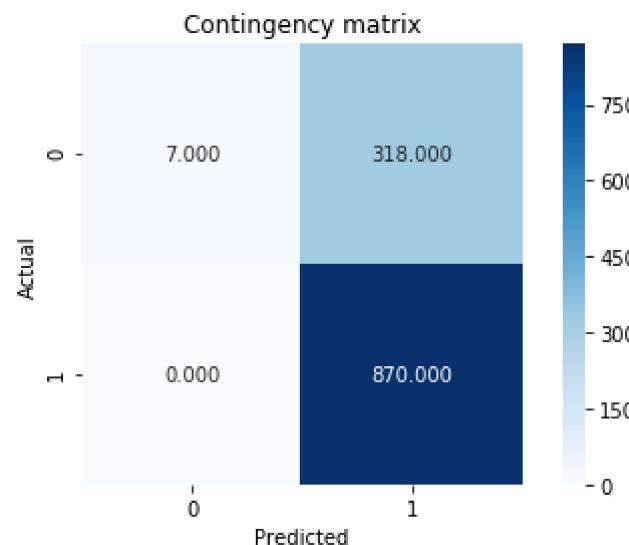
```
In [345]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

[0.1276047164044394, 0.6688371001371756]

```
In [346]: # This complete linkage hierarichal clustering model has the best performance
# in terms of both adjusted_rand_index (true clusters) and silhouette_coefficient (cluster quality i.e. cohesion and seperation) [0.128, 0.669]
```

```
In [ ]:
```

```
In [347]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Total Population"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

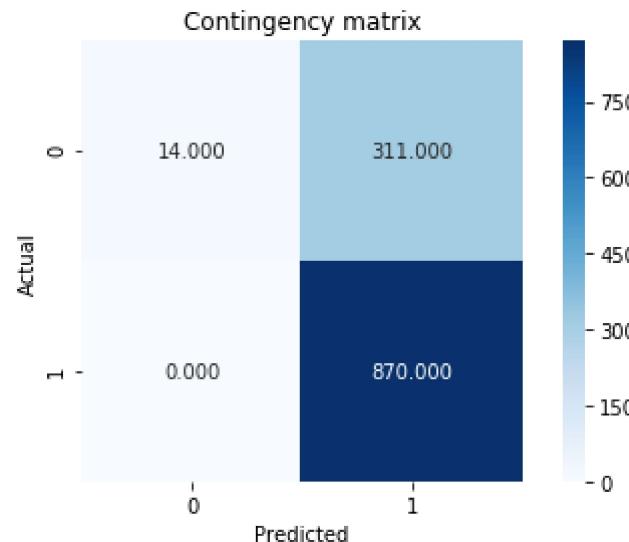


```
In [348]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.019643917192894232, 0.7191298947353636]
```

```
In [ ]:
```

```
In [349]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

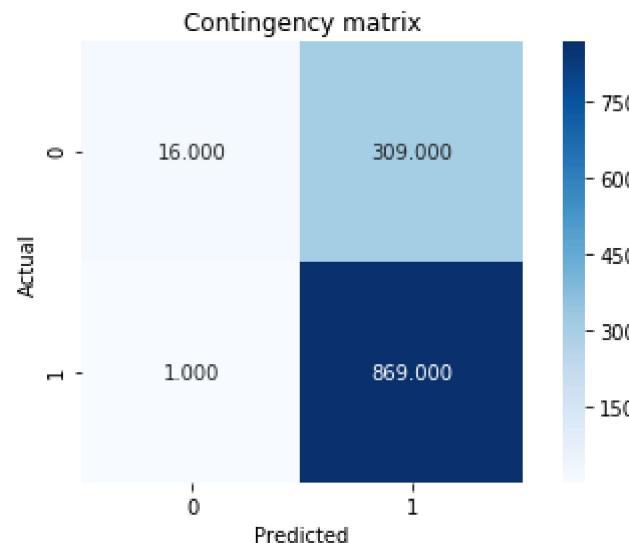


```
In [350]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.03932536341009194, 0.6975454950141516]
```

```
In [ ]:
```

```
In [351]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Hispanic or Latino"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

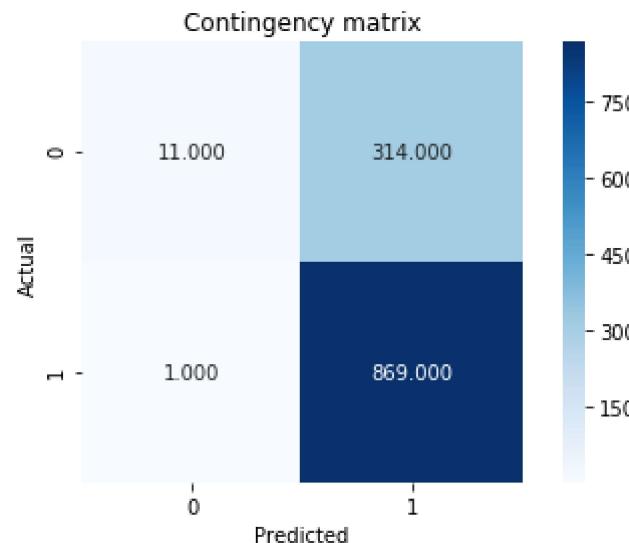


```
In [352]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.04375034893207605, 0.720293331167655]
```

```
In [ ]:
```

```
In [353]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Foreign Born"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

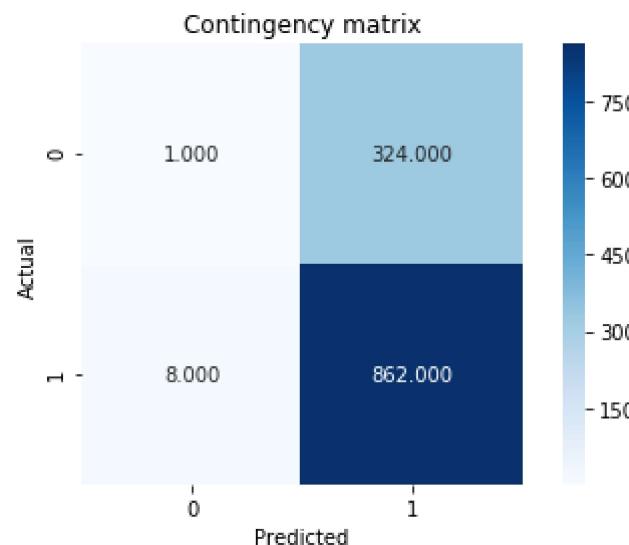


```
In [354]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.02972876675841617, 0.6876378378422308]
```

```
In [ ]:
```

```
In [355]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Female"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

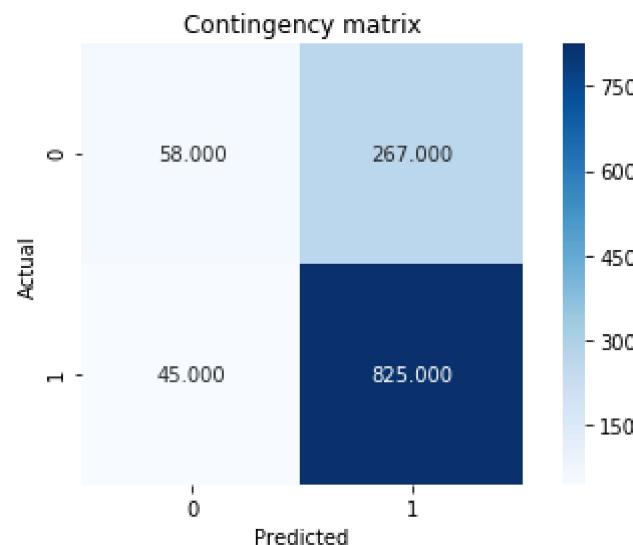


```
In [356]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[ -0.005449546864877415, 0.4902880192506524]
```

```
In [ ]:
```

```
In [357]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Age 29 and Under"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

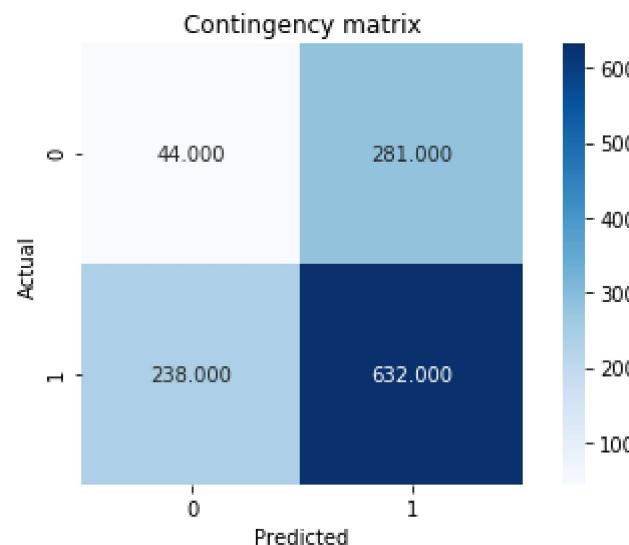


```
In [358]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.09989238656645157, 0.5942783409331475]
```

```
In [ ]:
```

```
In [359]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[['Percent White, not Hispanic or Latino", "Percent Age 65 and Older"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

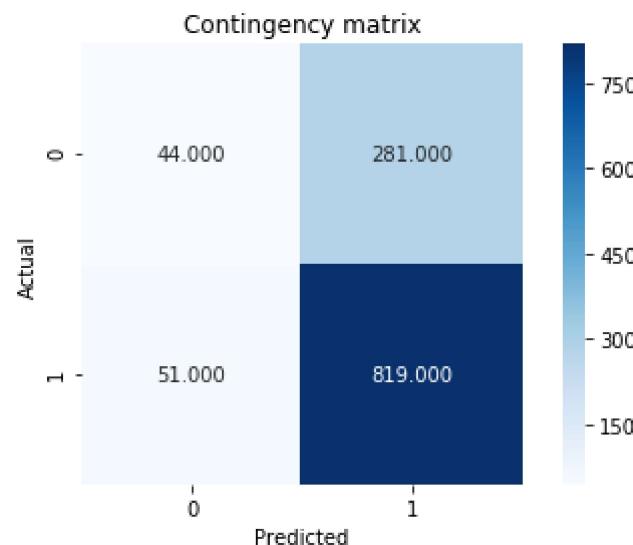


```
In [360]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.0437704178712109, 0.25628802871841994]
```

```
In [ ]:
```

```
In [361]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Median Household Income"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

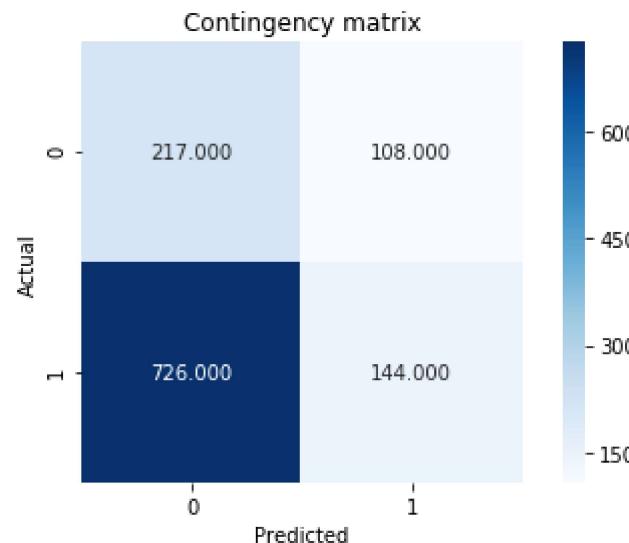


```
In [362]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.058802276205939016, 0.3204418833656496]
```

```
In [ ]:
```

```
In [363]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Unemployed"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

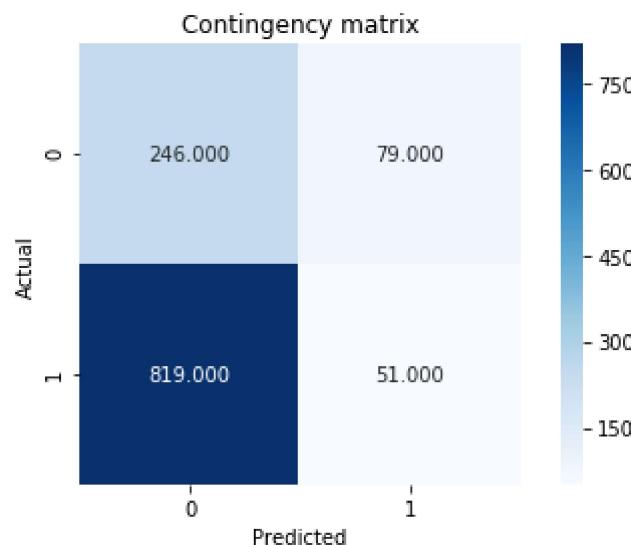


```
In [364]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.09320512098801267, 0.47472403860752804]
```

```
In [ ]:
```

```
In [365]: # Task 5  
# Complete Linkage Method with 2 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Less than High School Degree"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



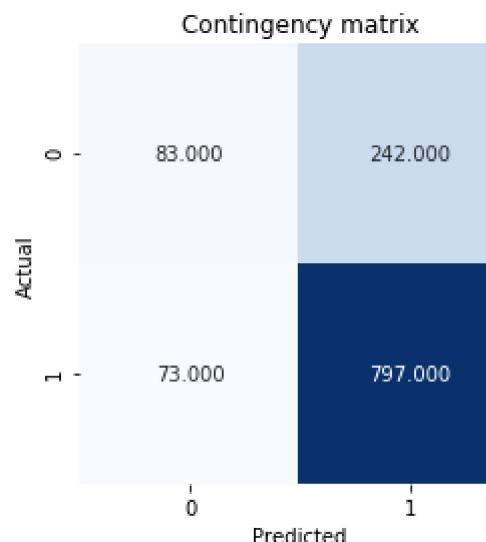
```
In [366]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.14367823630933002, 0.5882770688419815]
```

```
In [367]: # This complete Linkage hierarichal model has has the best performance  
# in terms of adjusted_rand_index (true clusters) [0.144] (with silhuette coefficient (cluster quality i.e.  
cohesion and seperation) value of [0.588])
```

```
In [ ]:
```

```
In [368]: # Task 5  
# Complete Linkage Method with 3 variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Less than High School Degree", "Percent Less than Bachelor's Degree"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



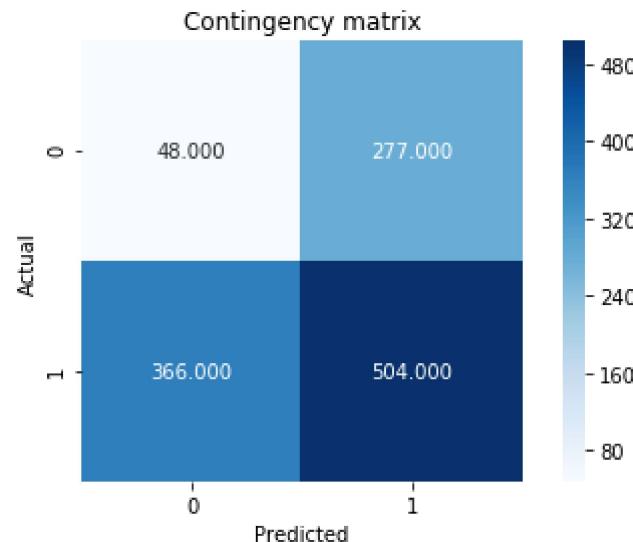
```
In [369]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])

[0.12377051806700631, 0.5070966792612928]
```

```
In [ ]:
```

```
In [370]: # Task 5  
# Complete Linkage Method with 4 variables
```

```
x = x_train[["Percent White, not Hispanic or Latino", "Percent Less than High School Degree", "Percent Rural"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "complete", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



```
In [371]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.014717348416978523, 0.3862068249470741]
```

In [ ]:

In [372]: # In case of complete Linkage clustering,

```
# i) the clustering model with 1 variable: "Total Population" has the best performance
#      in terms of silhuoette coefficient (better cluster quality i.e. high cohesion and seperation) [0.953]
# (with adjusted_rand_index value of [0.0056])

# ii) the clustering model with 2 variables: "Percent White, not Hispanic or Latino", "Percent Less than High School Degree" has the best performance
#      in terms of adjusted_rand_index (true clusters) [0.144] (with silhuoette coefficient (cluster quality i.e. cohesion and seperation) value of [0.588])

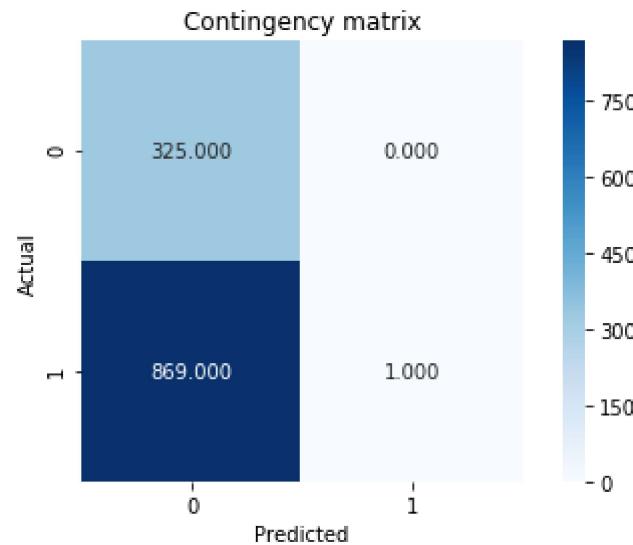
# iii) the clustering model with 1 variable: "Percent White, not Hispanic or Latino" has the best performance
#      in terms of both adjusted_rand_index (true clusters) and silhouette_coefficient (cluster quality i.e. cohesion and seperation) [0.128, 0.669]
```

In [ ]:

In [ ]:

In [373]: #x\_train.head()

```
In [374]: # Task 5  
# Single Linkage Method with all the variables  
  
x = x_train  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



```
In [375]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.001047512629882871, 0.3899390366856754]
```

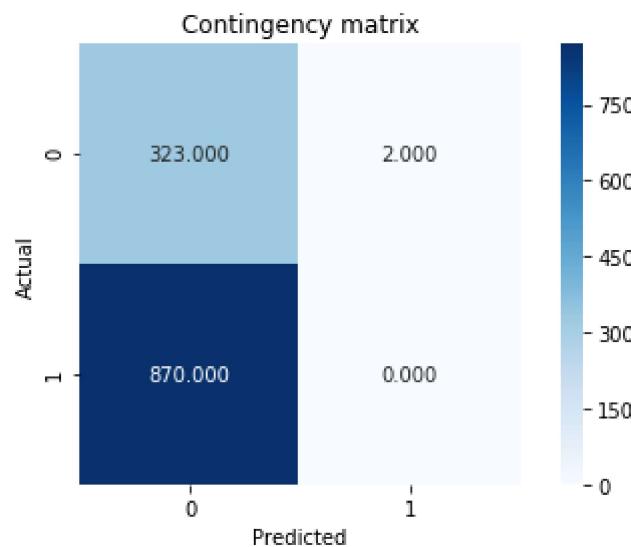
```
In [ ]:
```

```
In [376]: # Task 5
# Single Linkage Method with one variable

x = x_train[["Total Population"]]
y = y_train.iloc[:,2:]

clustering = linkage(x, method = "single", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
#print(clusters)

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



```
In [377]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

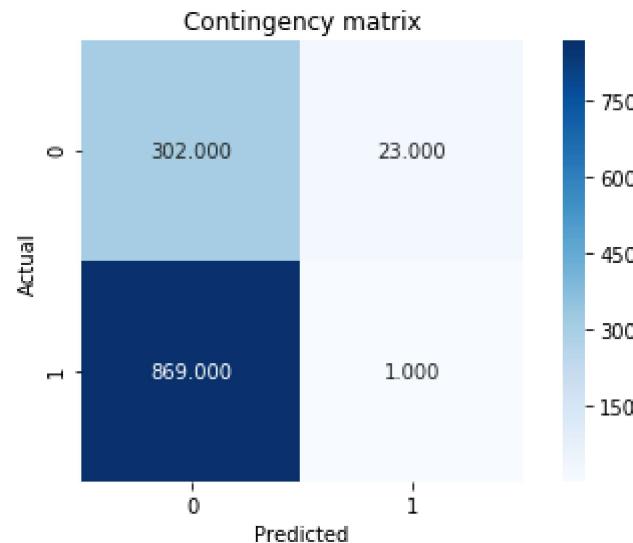
```
[0.005608925119335567, 0.9531008389502824]
```

```
In [378]: # Task 5
# Single Linkage Method with one variable

x = x_train[["Percent White, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]

clustering = linkage(x, method = "single", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
#print(clusters)

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



```
In [379]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

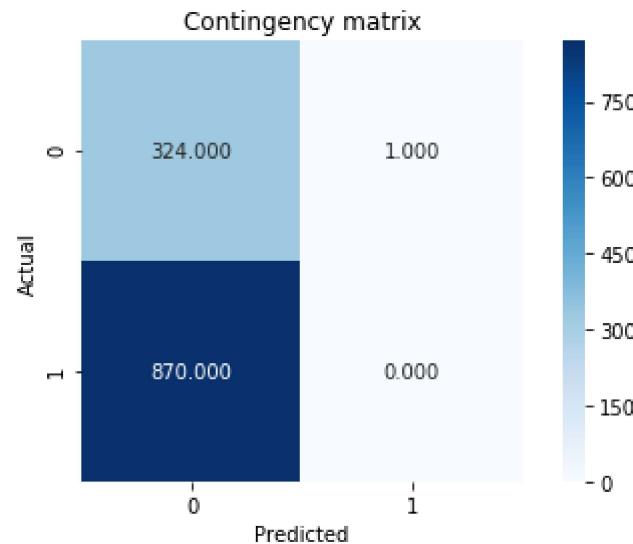
```
[0.06341778206546299, 0.6589687284593577]
```

```
In [380]: # Task 5
# Single Linkage Method with two variables

x = x_train[["Percent White, not Hispanic or Latino", "Percent Black, not Hispanic or Latino"]]
y = y_train.iloc[:,2:]

clustering = linkage(x, method = "single", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
#print(clusters)

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



```
In [381]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

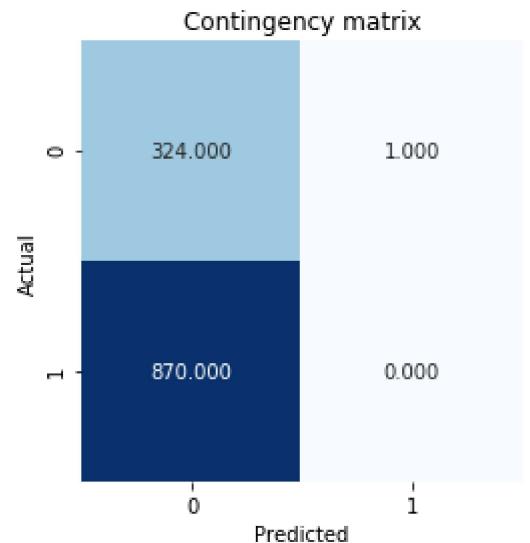
```
[0.0028041107323011935, 0.7352730414279328]
```

```
In [382]: # Task 5
# Single Linkage Method with two variables

x = x_train[["Percent White, not Hispanic or Latino", "Percent Hispanic or Latino"]]
y = y_train.iloc[:,2:]

clustering = linkage(x, method = "single", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
#print(clusters)

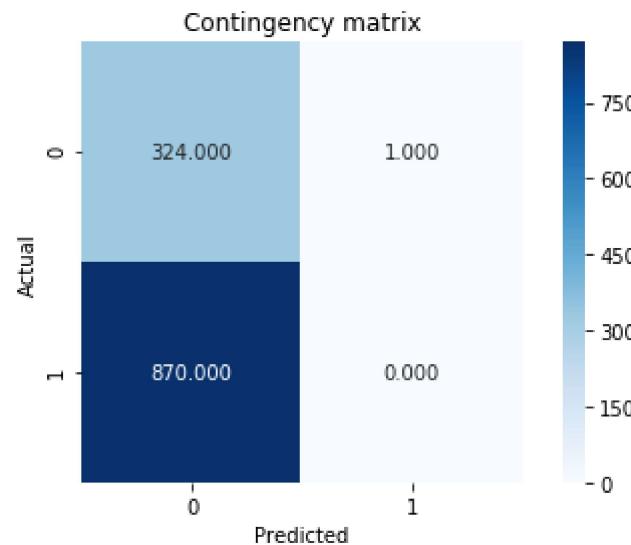
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



```
In [383]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0028041107323011935, 0.5476643584343294]
```

```
In [384]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Foreign Born"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

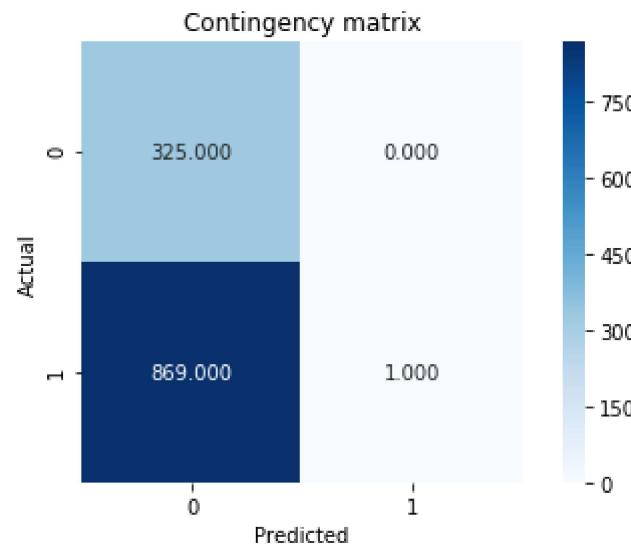


```
In [385]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0028041107323011935, 0.7534561225713804]
```

```
In [ ]:
```

```
In [386]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Female"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



```
In [387]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.001047512629882871, 0.7245163639894573]
```

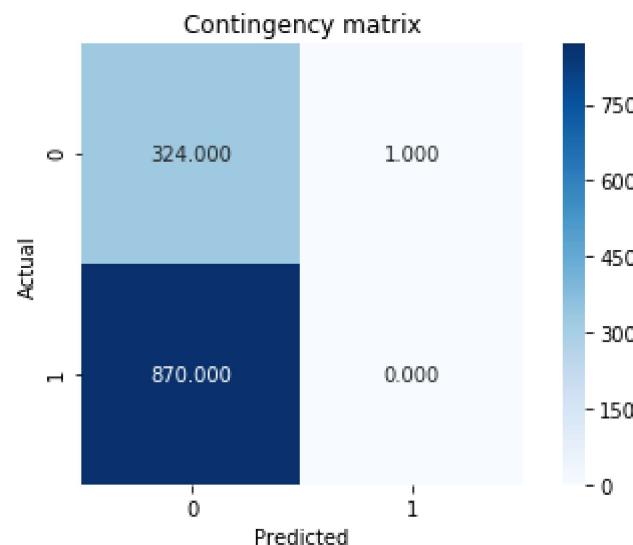
```
In [ ]:
```

```
In [388]: # Task 5
# Single Linkage Method with two variables

x = x_train[["Percent White, not Hispanic or Latino", "Percent Age 29 and Under"]]
y = y_train.iloc[:,2:]

clustering = linkage(x, method = "single", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
#print(clusters)

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

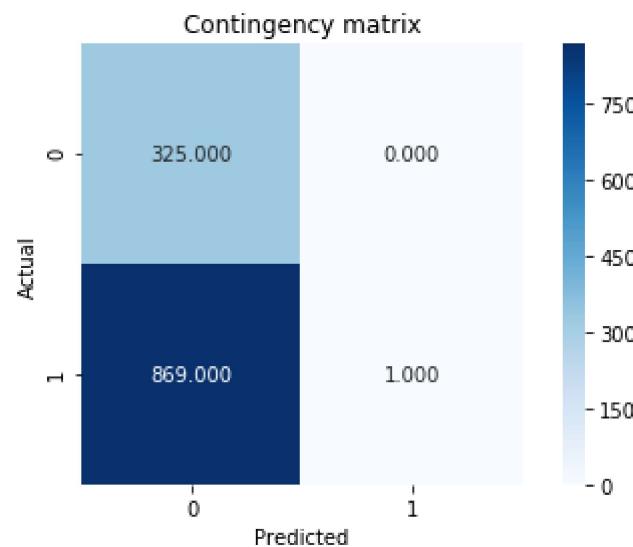


```
In [389]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0028041107323011935, 0.5546876308329813]
```

```
In [ ]:
```

```
In [390]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[['Percent White, not Hispanic or Latino", "Percent Age 65 and Older"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

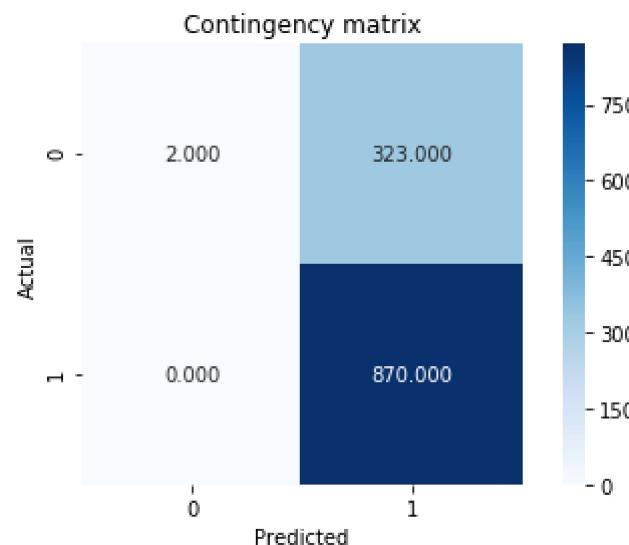


```
In [391]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.001047512629882871, 0.5187819968601595]
```

```
In [ ]:
```

```
In [392]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Median Household Income"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

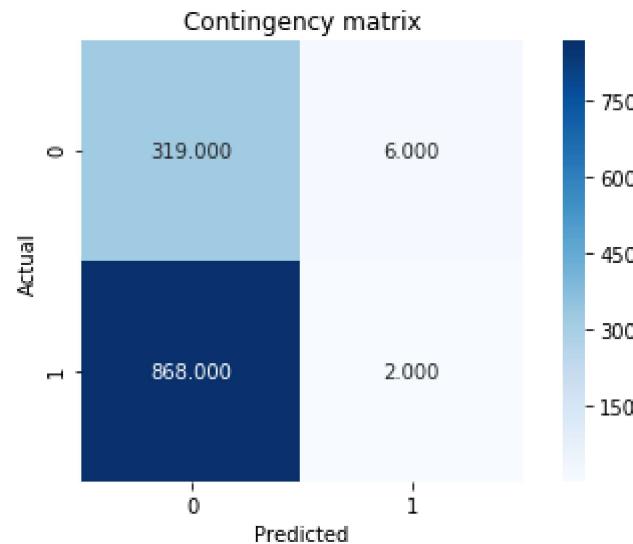


```
In [393]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.005608925119335567, 0.5693870855397117]
```

```
In [ ]:
```

```
In [394]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Unemployed"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



```
In [395]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.014627908627183214, 0.6173885535713797]
```

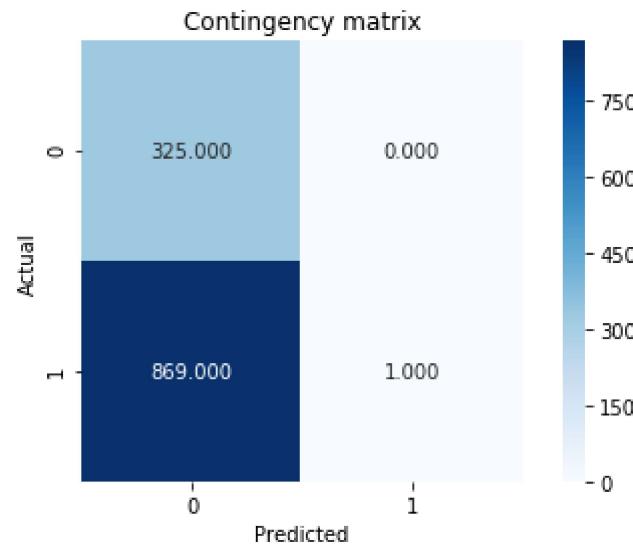
```
In [ ]:
```

```
In [396]: # Task 5
# Single Linkage Method with two variables

x = x_train[["Percent White, not Hispanic or Latino", "Percent Less than High School Degree"]]
y = y_train.iloc[:,2:]

clustering = linkage(x, method = "single", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
#print(clusters)

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

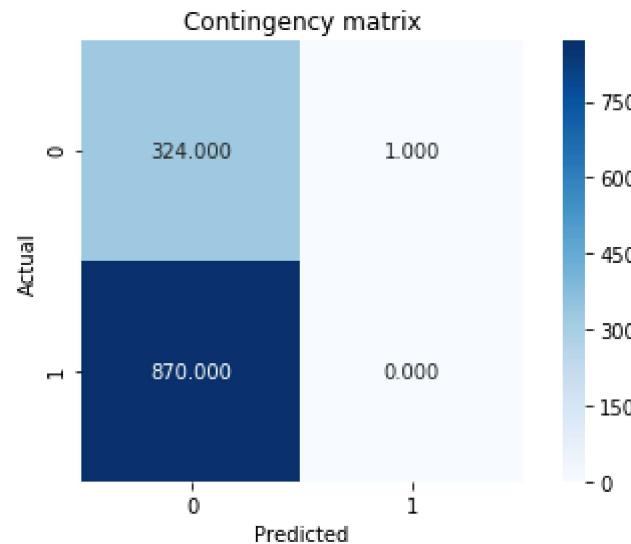


```
In [397]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.001047512629882871, 0.5214922405966899]
```

```
In [ ]:
```

```
In [398]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```

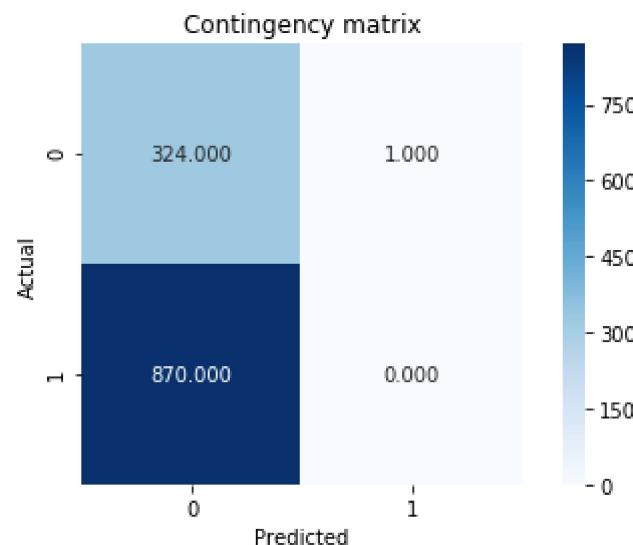


```
In [399]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0028041107323011935, 0.6211242566893129]
```

```
In [ ]:
```

```
In [400]: # Task 5  
# Single Linkage Method with two variables  
  
x = x_train[["Percent White, not Hispanic or Latino", "Percent Rural"]]  
y = y_train.iloc[:,2:]  
  
clustering = linkage(x, method = "single", metric = "euclidean")  
clusters = fcluster(clustering, 2, criterion = 'maxclust')  
#print(clusters)  
  
cont_matrix = metrics.cluster.contingency_matrix(y, clusters)  
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.cm.Blues)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.title('Contingency matrix')  
plt.tight_layout()
```



```
In [401]: adjusted_rand_index = metrics.adjusted_rand_score(y["Party"], clusters)  
silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")  
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0028041107323011935, 0.2345489030121581]
```

```
In [ ]:
```

In [402]: # In case of single Linkage clustering,

```
# i) the clustering model with 1 variable: "Total Population" has the best performance
#      in terms of silhuoette coefficient (better cluster quality i.e. high cohesion and seperation) [0.953]
#      (with adjusted_rand_index value of [0.0056])

# ii) the clustering model with 1 variable: "Percent White, not Hispanic or Latino" has the best performance
#      [0.06341778206546299, 0.6589687284593577]
#      in terms of adjusted_rand_index (true clusters) [0.063] (with silhuoette coefficient (cluster quality
#      i.e. cohesion and seperation) value of [0.659])

# iii) the clustering model with 1 variable: "Percent White, not Hispanic or Latino" also has the best perfor-
#      mance
#      in terms of both adjusted_rand_index (true clusters) and silhouette_coefficient (cluster quality i.e.
#      cohesion and seperation) [0.063, 0.659]
```

In [ ]:

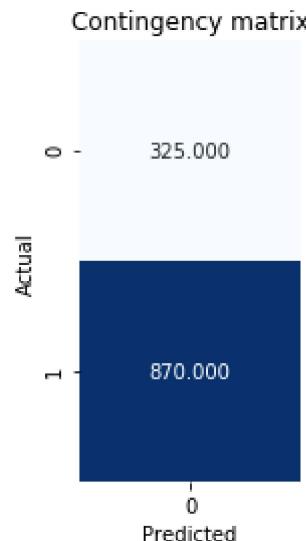
In [ ]:

In [403]: # DBSCAN with random initial centroids and with all variables

```
x = x_train
y = y_train.iloc[:,2:]

clustering = DBSCAN(eps = 0.0375, min_samples = 3, metric = "euclidean").fit(x)
clusters = clustering.labels_

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



In [404]: #adjusted\_rand\_index = metrics.adjusted\_rand\_score(y["Party"], clusters)

```
#silhouette_coefficient = metrics.silhouette_score(x, clusters, metric = "euclidean")
#print([adjusted_rand_index, silhouette_coefficient])
```

```
In [405]: # i) The K-means clustering model with 1 variable: "Total Population" has the best performance  
#       in terms of silhuoette coefficient (better cluster quality i.e. high cohesion and seperation) [0.903]  
(with adjusted_rand_index value of [0.120])  
  
# ii) The K-means clustering model with 3 variables: "Total Population", "Percent Female", "Percent Age 29 a  
nd Under", "Percent Less than Bachelor's Degree" has the best performance  
#       in terms of adjusted_rand_index (true clusters) [0.289] (with silhuoette coefficient (cluster quality  
i.e. cohesion and seperation) value of [0.432])  
  
# iii) The K-means clustering model with 2 variables: "Total Population", "Percent Female" has the best perfo  
rmance  
#       in terms of both adjusted_rand_index (true clusters) and silhouette_coefficient (cluster quality i.e.  
cohesion and seperation) [0.122, 0.752]
```

```
In [ ]:
```

```
In [ ]:
```

```
In [406]: # Task 6
```

```
p = best_classifier.predict(x_train)  
#p.tolist()
```

```
In [ ]:
```

In [407]: # TASK 6

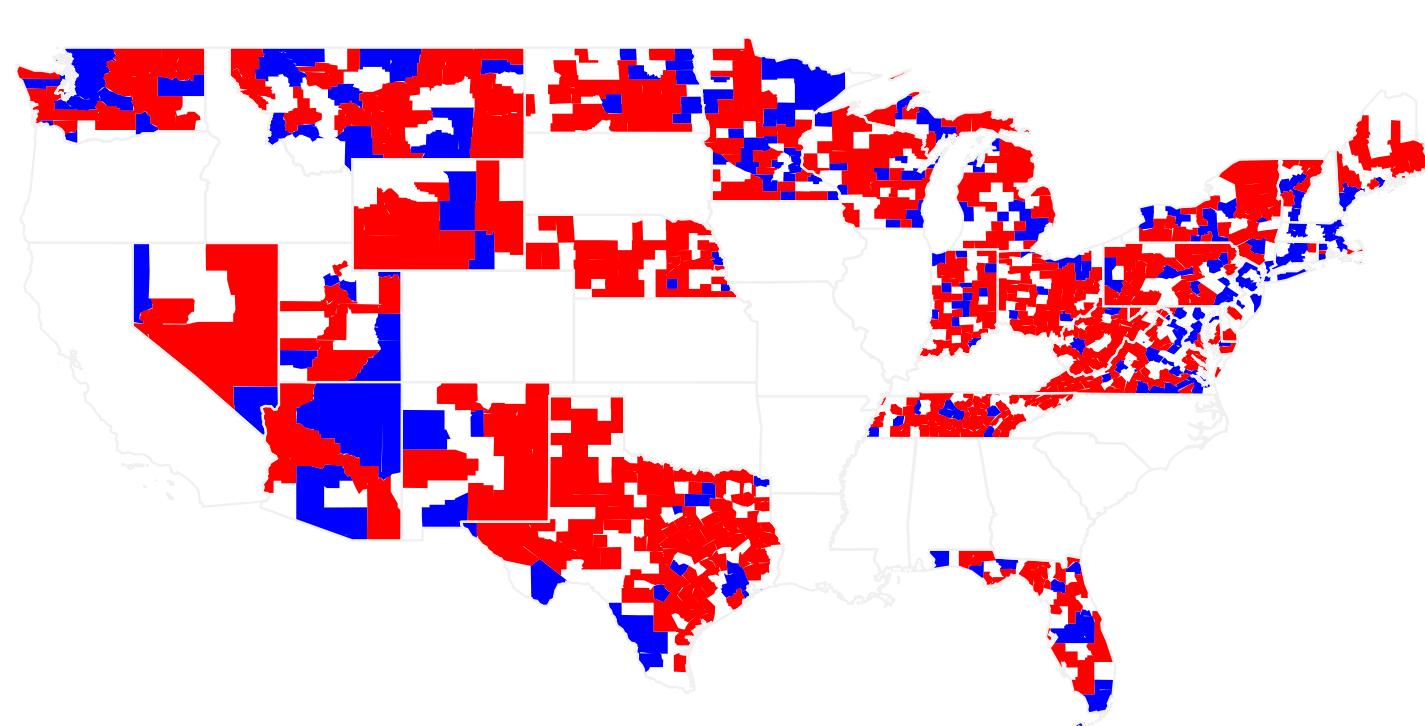
```
fips = merged['FIPS'].astype(str).tolist()
values = p.tolist()
colorscale = ['rgb(0,0,255)', 'rgb(255,0,0)'
              ]
fig = ff.create_choropleth(fips=fips, values=values, colorscale = colorscale)
fig.layout.template = None
fig.show()
```

C:\Users\kalya\Anaconda3\lib\site-packages\pandas\core\frame.py:6692: FutureWarning:

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

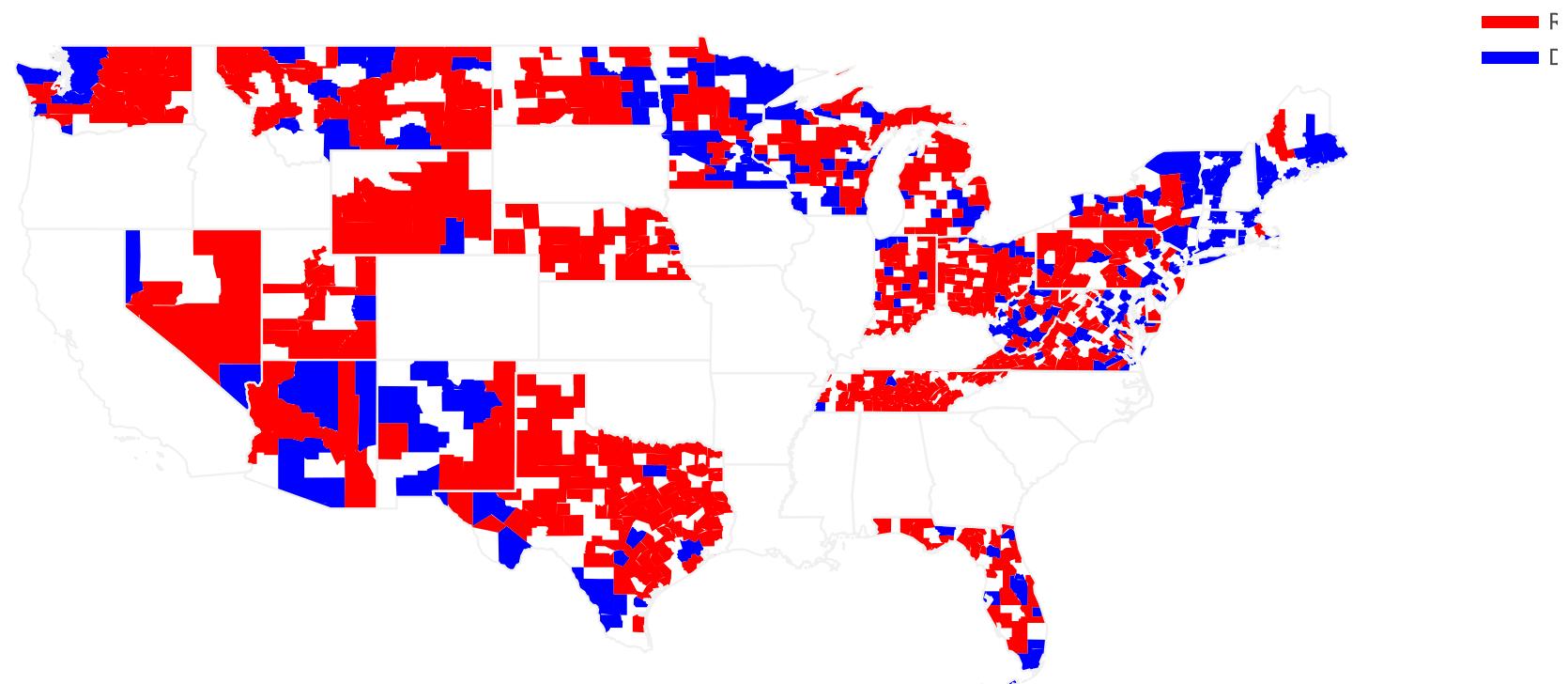
To accept the future behavior, pass 'sort=False'.

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



In [408]: # Task 6

```
fips = merged['FIPS'].astype(str).tolist()
values = merged["Party"].tolist()
colorscale = ['rgb(0,0,255)', 'rgb(255,0,0)'
              ]
fig = ff.create_choropleth(fips=fips, values=values, colorscale = colorscale)
fig.layout.template = None
fig.show()
```

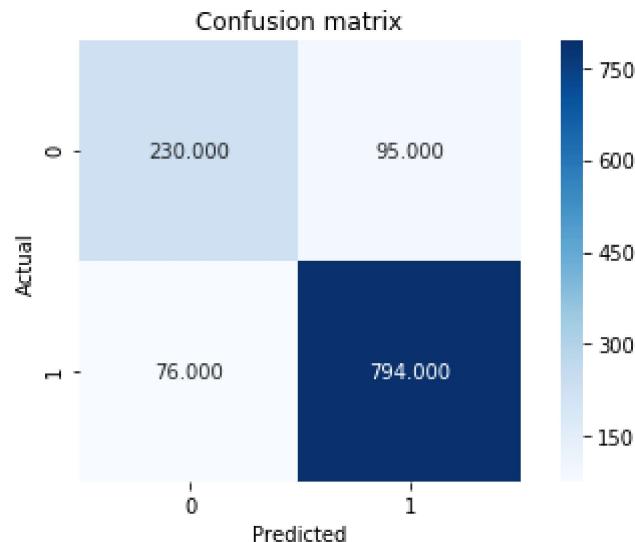


```
In [409]: # TASK 6
# Compute confusion matrix
y_test = y
y_pred = p
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(conf_matrix)

# TASK 4
# 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()

# TASK 4
# Compute evaluation metrics
print(metrics.accuracy_score(y_test, y_pred)) # accuracy
print(1 - metrics.accuracy_score(y_test, y_pred)) # error
print(metrics.precision_score(y_test, y_pred, average = None)) # precision
print(metrics.recall_score(y_test, y_pred, average = None)) # recall
print(metrics.f1_score(y_test, y_pred, average = None)) # F1 score
```

```
[[230 95]
 [ 76 794]]
0.8569037656903765
0.14309623430962348
[0.75163399 0.89313836]
[0.70769231 0.91264368]
[0.72900158 0.90278567]
```



```
In [410]: # From the above two plots, the best classifier we used to predict the counties had wrongly predicted some counties as Republican
# and some as Democratic.
# And the accuracy obtained is 85.7 percent and F1 Score is [0.72900158 0.90278567]
```

```
In [ ]:
```

```
In [411]: # Task 7
data = pd.read_csv("demographics_test.csv")
test = data[["Total Population", "Percent Black, not Hispanic or Latino",
             "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Perce
nt Unemployed",
             "Percent Less than High School Degree"]]
#test.head()

test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 7 columns):
Total Population           400 non-null int64
Percent Black, not Hispanic or Latino 400 non-null float64
Percent Less than Bachelor's Degree    400 non-null float64
Percent Foreign Born            400 non-null float64
Median Household Income        400 non-null int64
Percent Unemployed            400 non-null float64
Percent Less than High School Degree 400 non-null float64
dtypes: float64(5), int64(2)
memory usage: 22.0 KB
```

```
In [412]: # task 7
scaler = MinMaxScaler(feature_range=(0, 1))
test[["Total Population", "Percent Black, not Hispanic or Latino",
       "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Perce
nt Unemployed",
       "Percent Less than High School Degree"]] = scaler.fit_transform(test[["Total Population", "Perce
nt Black, not Hispanic or Latino",
       "Percent Less than Bachelor's Degree", "Percent Foreign Born", "Median Household Income", "Perce
nt Unemployed",
       "Percent Less than High School Degree"]])
test= sm.add_constant(test)
test_scaled = test
#test_scaled.head()
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:

Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

In [413]: `test_scaled.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
const                400 non-null float64
Total Population      400 non-null float64
Percent Black, not Hispanic or Latino 400 non-null float64
Percent Less than Bachelor's Degree    400 non-null float64
Percent Foreign Born      400 non-null float64
Median Household Income   400 non-null float64
Percent Unemployed       400 non-null float64
Percent Less than High School Degree 400 non-null float64
dtypes: float64(8)
memory usage: 25.1 KB
```

In [414]: `Demo = best_regression_Demo.predict(test_scaled).round()  
Demo.head()`

Out[414]:

```
0    -9074.0
1    -5776.0
2    20148.0
3    453771.0
4    11205.0
dtype: float64
```

```
In [415]: test2 = data[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                  "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                  "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
                  "Percent Less than High School Degree", "Percent Rural"]]

scaler = MinMaxScaler(feature_range=(0, 1))
test2[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
        "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
        "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
        "Percent Less than High School Degree", "Percent Rural"]] = scaler.fit_transform(test2[["Total Population", "Percent White, not Hispanic or Latino", "Percent Less than Bachelor's Degree",
                                                                 "Percent Black, not Hispanic or Latino", "Percent Hispanic or Latino",
                                                                 "Percent Foreign Born", "Median Household Income", "Percent Unemployed",
                                                                 "Percent Less than High School Degree", "Percent Rural"]])
test2= sm.add_constant(test2)
test_scaled2 = test2
```

C:\Users\kalya\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning:  
Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```
In [416]: Rep = best_regression_Rep.predict(test_scaled2).round()
Rep.head()
```

```
Out[416]: 0      8888.0
1      4464.0
2     23612.0
3    276454.0
4     9091.0
dtype: float64
```

```
In [417]: data["Democratic"] = Demo
data["Republican"] = Rep
```

```
In [418]: data["Democratic"][data["Democratic"] < 0] = 0
data["Republican"][data["Republican"] < 0] = 0
```

In [419]: `data.head()`

Out[419]:

	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	Median Household Income	Per Unempl
0	NV	eureka	32011	1730	98.265896	0.057803	0.462428	0.346821	51.156069	27.109827	15.606936	70000	3.75%
1	TX	zavala	48507	12107	5.798299	0.594697	93.326175	9.193029	49.723301	49.302057	12.480383	26639	11.95%
2	VA	king george	51099	25260	73.804434	16.722090	4.441805	2.505938	50.166271	40.186065	11.868567	84342	6.47%
3	OH	hamilton	39061	805965	66.354867	25.654340	2.890944	5.086945	51.870615	40.779686	14.161657	50399	7.86%
4	TX	austin	48015	29107	63.809393	8.479060	25.502456	9.946061	50.671660	37.351840	17.799842	56681	5.78%



In [420]: `data["Party"] = data["Democratic"] > data["Republican"]`

In [421]: `data.head()`

Out[421]:

	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	Median Household Income	Per Unempl
0	NV	eureka	32011	1730	98.265896	0.057803	0.462428	0.346821	51.156069	27.109827	15.606936	70000	3.75%
1	TX	zavala	48507	12107	5.798299	0.594697	93.326175	9.193029	49.723301	49.302057	12.480383	26639	11.95%
2	VA	king george	51099	25260	73.804434	16.722090	4.441805	2.505938	50.166271	40.186065	11.868567	84342	6.47%
3	OH	hamilton	39061	805965	66.354867	25.654340	2.890944	5.086945	51.870615	40.779686	14.161657	50399	7.86%
4	TX	austin	48015	29107	63.809393	8.479060	25.502456	9.946061	50.671660	37.351840	17.799842	56681	5.78%



```
In [422]: change_values = pd.Series({True : 1, False : 0})
data["Party"] = data["Party"].map(change_values)
data["Party"][data["Democratic"] == data["Republican"]] = "NaN"
data["Party"].replace("NaN", np.nan, inplace= True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 19 columns):
State                      400 non-null object
County                     400 non-null object
FIPS                       400 non-null int64
Total Population           400 non-null int64
Percent White, not Hispanic or Latino 400 non-null float64
Percent Black, not Hispanic or Latino 400 non-null float64
Percent Hispanic or Latino    400 non-null float64
Percent Foreign Born        400 non-null float64
Percent Female              400 non-null float64
Percent Age 29 and Under   400 non-null float64
Percent Age 65 and Older   400 non-null float64
Median Household Income     400 non-null int64
Percent Unemployed          400 non-null float64
Percent Less than High School Degree 400 non-null float64
Percent Less than Bachelor's Degree 400 non-null float64
Percent Rural               400 non-null float64
Democratic                 400 non-null float64
Republican                 400 non-null float64
Party                      356 non-null float64
dtypes: float64(14), int64(3), object(2)
memory usage: 59.5+ KB
```

```
In [423]: out = data[["State", "County", "Democratic", "Republican", "Party"]]
out.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
State          400 non-null object
County         400 non-null object
Democratic    400 non-null float64
Republican    400 non-null float64
Party          356 non-null float64
dtypes: float64(3), object(2)
memory usage: 15.7+ KB
```

```
In [425]: out.to_csv("Out.csv", index=False, na_rep='NaN')
```

```
In [426]: out.head()
```

Out[426]:

	State	County	Democratic	Republican	Party
0	NV	eureka	0.0	8888.0	0.0
1	TX	zavala	0.0	4464.0	0.0
2	VA	king george	20148.0	23612.0	0.0
3	OH	hamilton	453771.0	276454.0	1.0
4	TX	austin	11205.0	9091.0	1.0

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```