Projekt Jakub Gwiazda, Wojciech Windak

1. Przygotowanie i analiza danych

- **1.1 Wczytanie niezbędnych bibliotek**

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns;
import pandas as pd
```

- **1.2 Wczytanie danych**

In [3]:
```
import pandas as pd
df = pd.read_fwf('pop_failures.dat')

df.head()
```

Out[3]:

	Study	Run	vconst_corr	vconst_2	vconst_3	vconst_4	vconst_5	vconst_7	ah_corr	ah_bolus	...	efficiency_factor	tidal_mix_max	vertical_decay_sc
0	1	1	0.859036	0.927825	0.252866	0.298838	0.170521	0.735936	0.428325	0.567947	...	0.245675	0.104226	0.869(
1	1	2	0.606041	0.457728	0.359448	0.306957	0.843331	0.934851	0.444572	0.828015	...	0.616870	0.975786	0.914:
2	1	3	0.997600	0.373238	0.517399	0.504993	0.618903	0.605571	0.746225	0.195928	...	0.679355	0.803413	0.643
3	1	4	0.783408	0.104055	0.197533	0.421837	0.742056	0.490828	0.005525	0.392123	...	0.471463	0.597879	0.761(
4	1	5	0.406250	0.513199	0.061812	0.635837	0.844798	0.441502	0.191926	0.487546	...	0.551543	0.743877	0.312:

5 rows × 21 columns

- **1.3 Podstawowe dane o zbiorze**

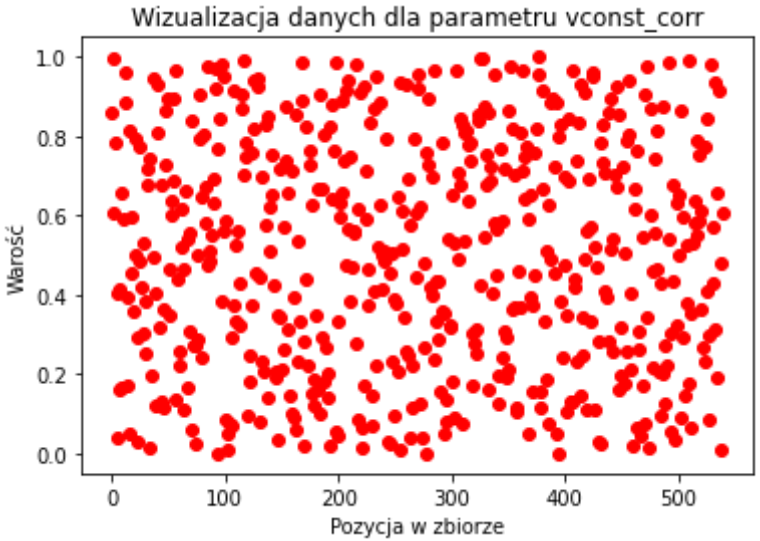
In [4]:
```
print (df.describe())
```

```
              Study         Run  vconst_corr    vconst_2    vconst_3  \
count  540.000000  540.000000   540.000000  540.000000  540.000000
mean     2.000000   90.500000     0.500026    0.500097    0.500027
std      0.817254   52.008901     0.288939    0.288922    0.289067
min      1.000000    1.000000     0.000414    0.001922    0.001181
25%      1.000000   45.750000     0.249650    0.251597    0.251540
50%      2.000000   90.500000     0.499998    0.499595    0.500104
75%      3.000000  135.250000     0.750042    0.750011    0.749180
max      3.000000  180.000000     0.999194    0.998815    0.998263

         vconst_4      vconst_5    vconst_7     ah_corr    ah_bolus  ...  \
count  540.000000  5.400000e+02  540.000000  540.000000  540.000000  ...
mean     0.500119  1.589598e+13    0.499913    0.500059    0.500076  ...
std      0.288993  3.693892e+14    0.288852    0.289010    0.288909  ...
min      0.001972  1.889182e-03    0.000476    0.004590    0.000296  ...
25%      0.250158  2.534061e-01    0.251325    0.253048    0.250402  ...
50%      0.500456  5.019133e-01    0.499174    0.499070    0.500074  ...
75%      0.750348  7.513031e-01    0.748166    0.750109    0.749091  ...
max      0.997673  8.583829e+15    0.997142    0.998930    0.998506  ...

       efficiency_factor  tidal_mix_max  vertical_decay_scale  convect_corr  \
count         540.000000     540.000000            540.000000    540.000000
mean            0.500111       0.499984              0.500032      0.499933
std             0.288966       0.289127              0.289014      0.288822
min             0.002015       0.000419              0.001188      0.001312
25%             0.250758       0.251676              0.249669      0.249988
50%             0.500393       0.500322              0.500151      0.500625
75%             0.749447       0.749346              0.749164      0.749569
max             0.999536       0.999942              0.997718      0.997518

       bckgrnd_vdc1  bckgrnd_vdc_ban  bckgrnd_vdc_eq  bckgrnd_vdc_psim  \
count    540.000000       540.000000    5.400000e+02        540.000000
mean       0.499944         0.499946    1.649366e+13          0.500020
std        0.288949         0.288923    3.832780e+14          0.288936
min        0.002509         0.000732    2.748962e-03          0.000219
25%        0.249586         0.249974    2.528092e-01          0.252739
50%        0.499080         0.499959    5.023113e-01          0.498955
75%        0.750012         0.747978    7.510987e-01          0.748539
max        0.999795         0.999155    8.906575e+15          0.999306

          Prandtl     outcome
count  540.000000  540.000000
mean     0.500021    0.914815
std      0.289013    0.279416
```

```
min      0.000263    0.000000
25%      0.249723    1.000000
50%      0.499431    1.000000
75%      0.749792    1.000000
max      0.999655    1.000000

[8 rows x 21 columns]
```

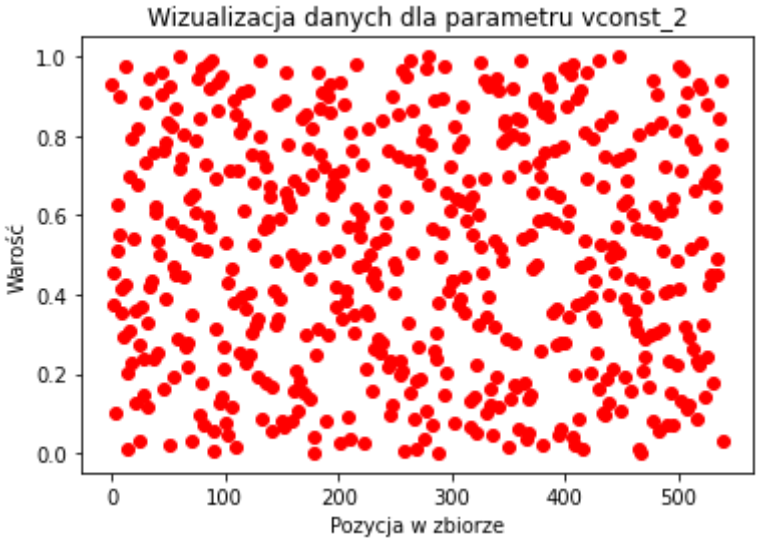
- **1.4.1 Wizualizacja danych dla parametru vconst_corr**

In [5]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_corr'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_corr')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



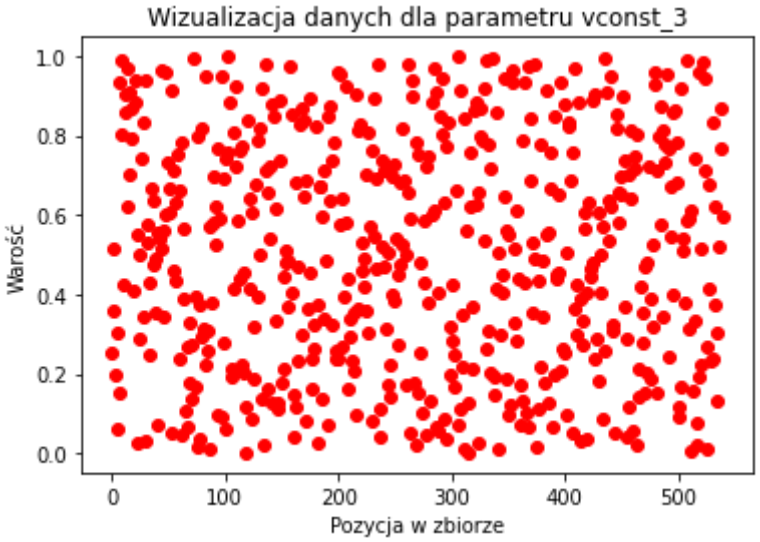
- **1.4.2 Wizualizacja danych dla parametru vconst_2**

In [6]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_2'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_2')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



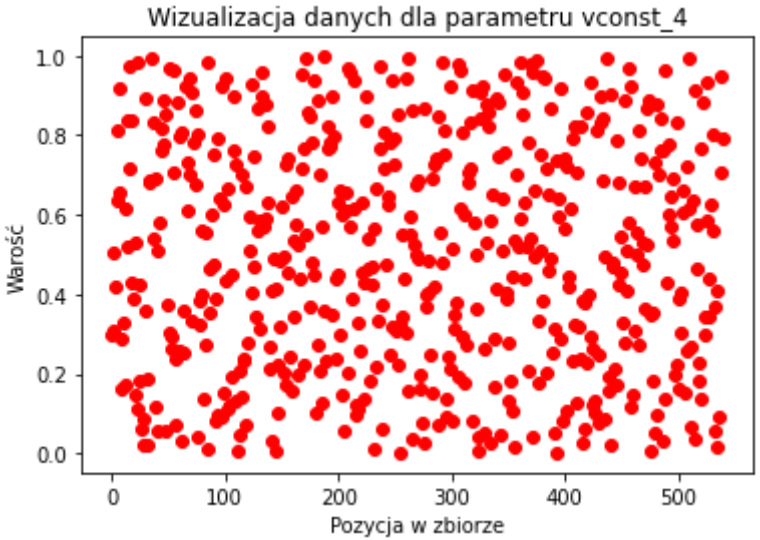
- **1.4.3 Wizualizacja danych dla parametru vconst_3**

In [7]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_3'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_3')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



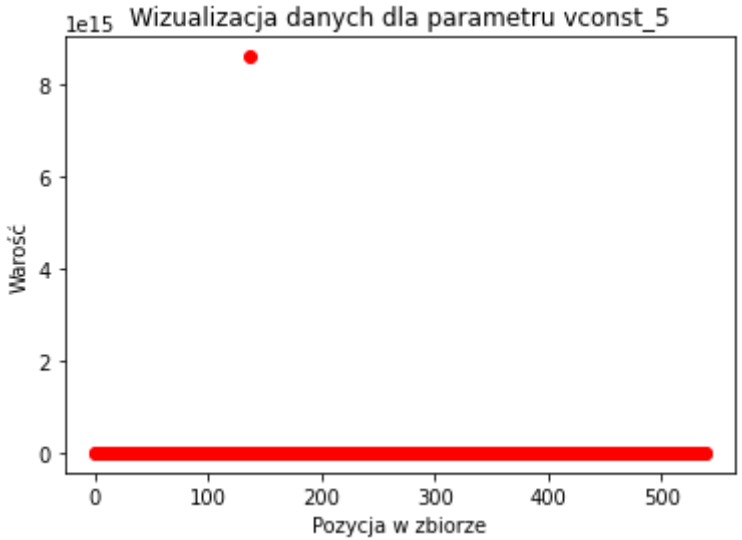
- **1.4.4 Wizualizacja danych dla parametru vconst_4**

In [8]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_4'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_4')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



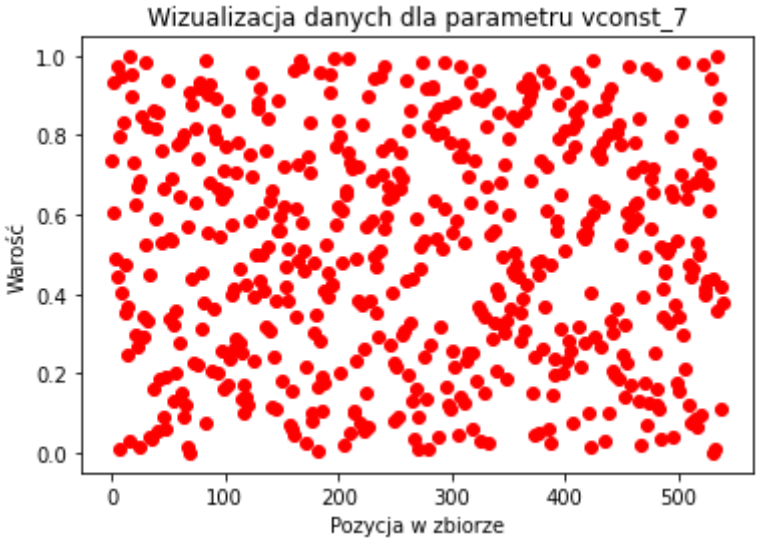
- **1.4.5 Wizualizacja danych dla parametru vconst_5**

In [9]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_5'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_5')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



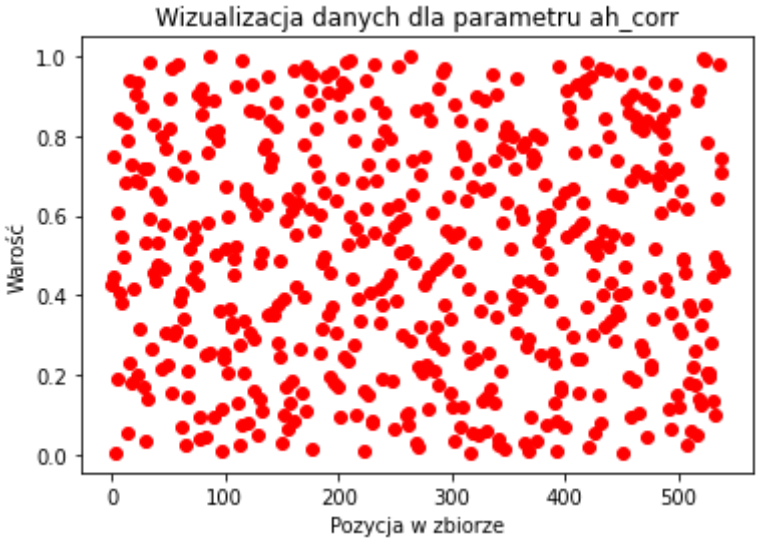
- **1.4.6 Wizualizacja danych dla parametru vconst_7**

In [10]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_7'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_7')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



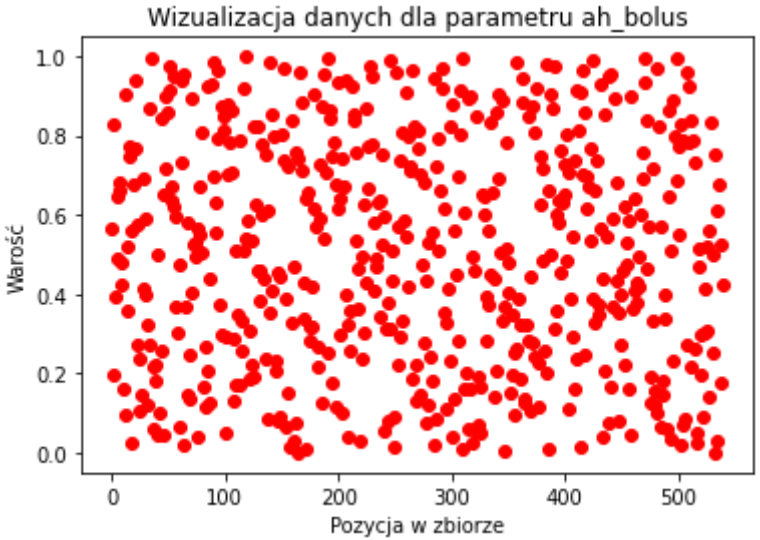
- **1.4.7 Wizualizacja danych dla parametru ah_corr**

In [11]:
```
import matplotlib.pyplot as plt
plt.plot(df['ah_corr'], 'ro')
plt.title('Wizualizacja danych dla parametru ah_corr')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



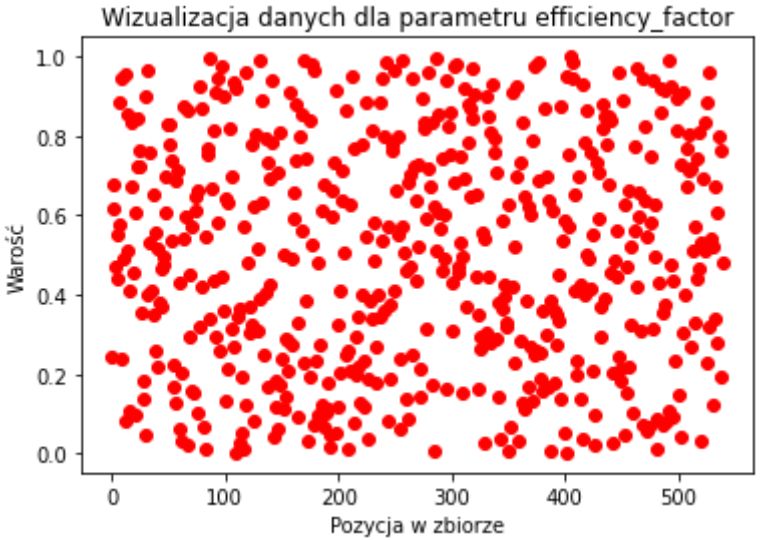
- **1.4.8 Wizualizacja danych dla parametru ah_bolus**

In [12]:
```
import matplotlib.pyplot as plt
plt.plot(df['ah_bolus'], 'ro')
plt.title('Wizualizacja danych dla parametru ah_bolus')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



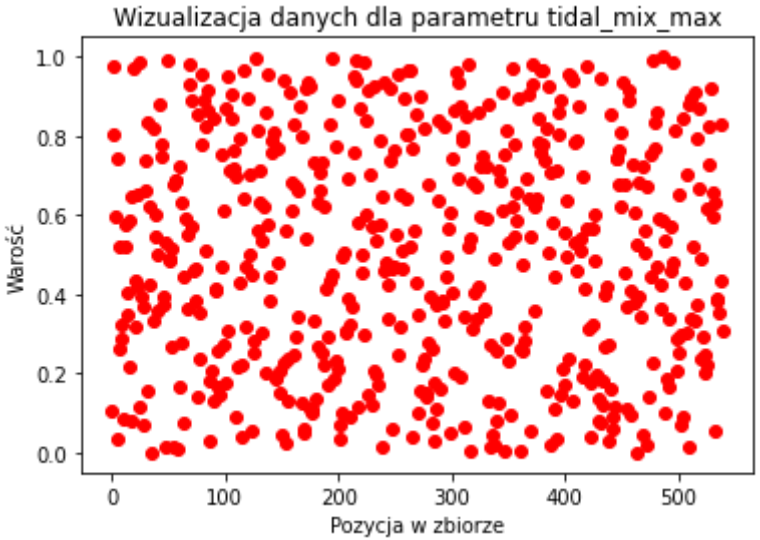
- **1.4.9 Wizualizacja danych dla parametru efficiency_factor**

In [13]:
```
import matplotlib.pyplot as plt
plt.plot(df['efficiency_factor'], 'ro')
plt.title('Wizualizacja danych dla parametru efficiency_factor')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



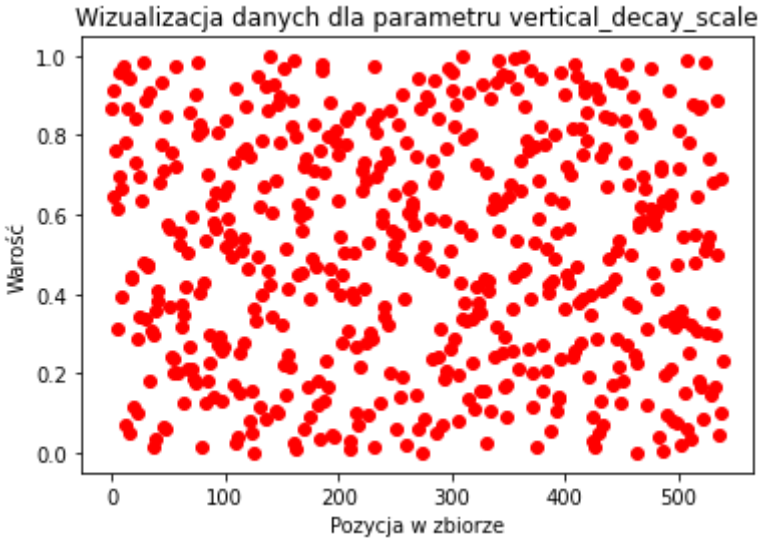
- **1.4.10 Wizualizacja danych dla parametru tidal_mix_max**

In [14]:
```
import matplotlib.pyplot as plt
plt.plot(df['tidal_mix_max'], 'ro')
plt.title('Wizualizacja danych dla parametru tidal_mix_max')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



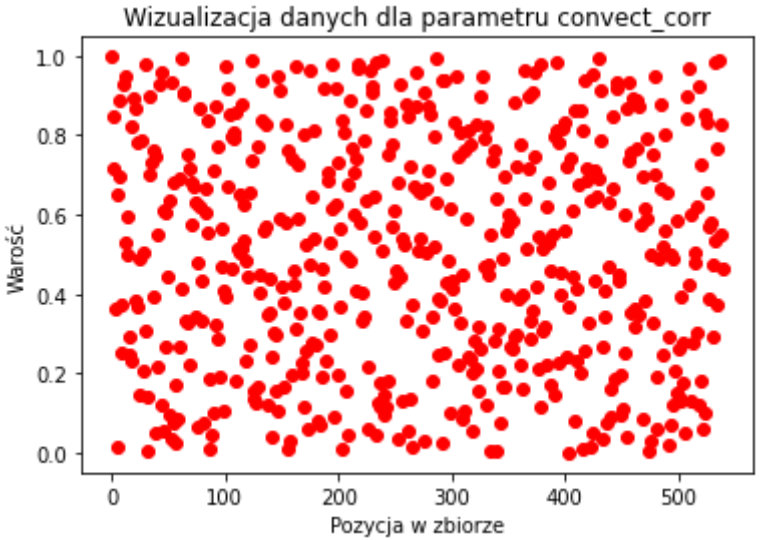
- **1.4.11 Wizualizacja danych dla parametru vertical_decay_scale**

In [15]:
```
import matplotlib.pyplot as plt
plt.plot(df['vertical_decay_scale'], 'ro')
plt.title('Wizualizacja danych dla parametru vertical_decay_scale')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



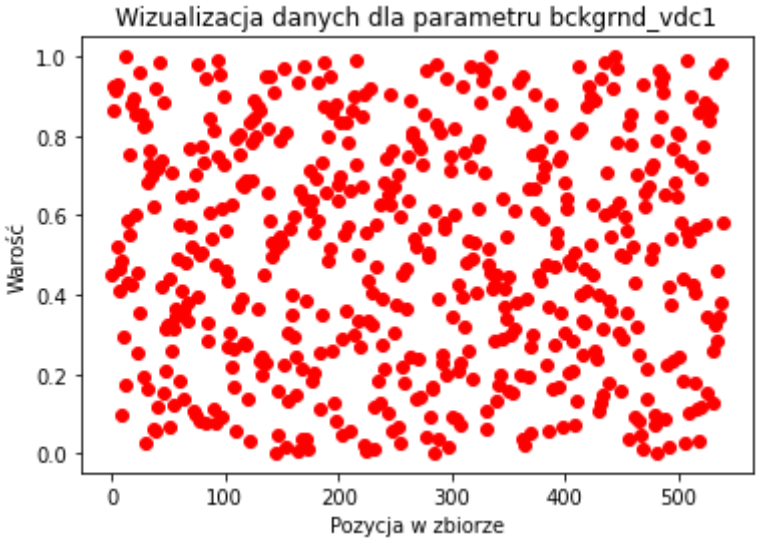
- **1.4.12 Wizualizacja danych dla parametru convect_corr**

In [16]:
```
import matplotlib.pyplot as plt
plt.plot(df['convect_corr'], 'ro')
plt.title('Wizualizacja danych dla parametru convect_corr')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



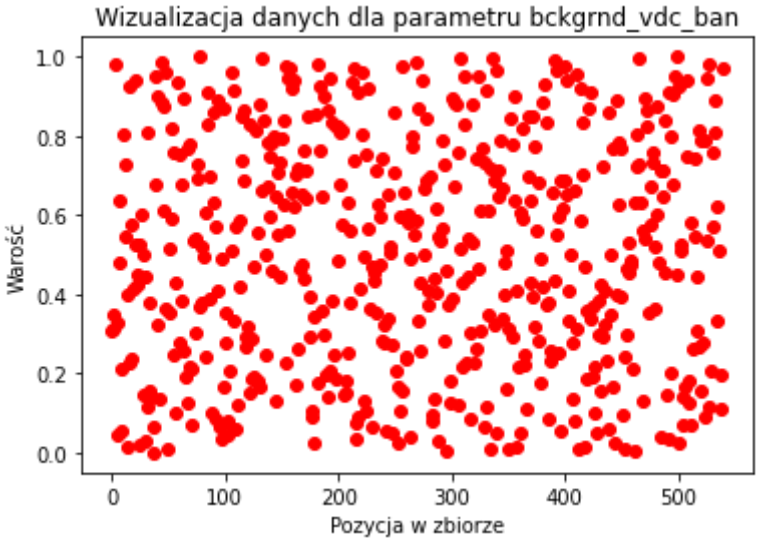
- **1.4.13 Wizualizacja danych dla parametru bckgrnd_vdc1**

In [17]:
```
import matplotlib.pyplot as plt
plt.plot(df['bckgrnd_vdc1'], 'ro')
plt.title('Wizualizacja danych dla parametru bckgrnd_vdc1')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



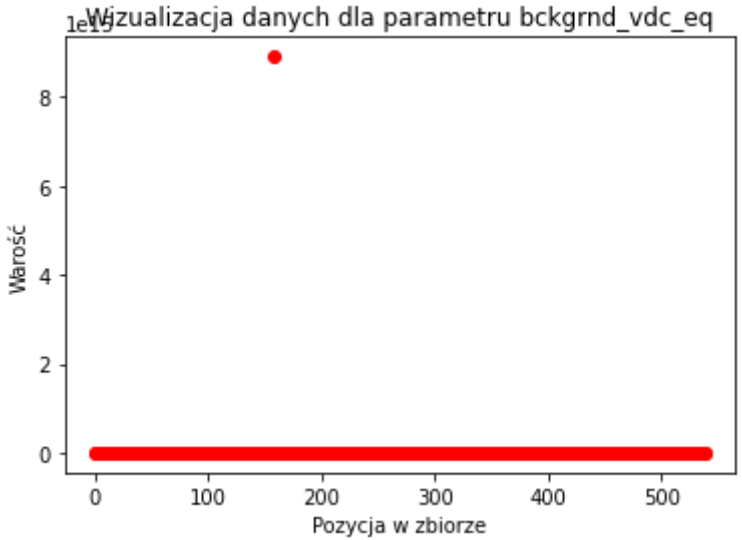
- **1.4.14 Wizualizacja danych dla parametru bckgrnd_vdc_ban**

In [18]:
```
import matplotlib.pyplot as plt
plt.plot(df['bckgrnd_vdc_ban'], 'ro')
plt.title('Wizualizacja danych dla parametru bckgrnd_vdc_ban')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



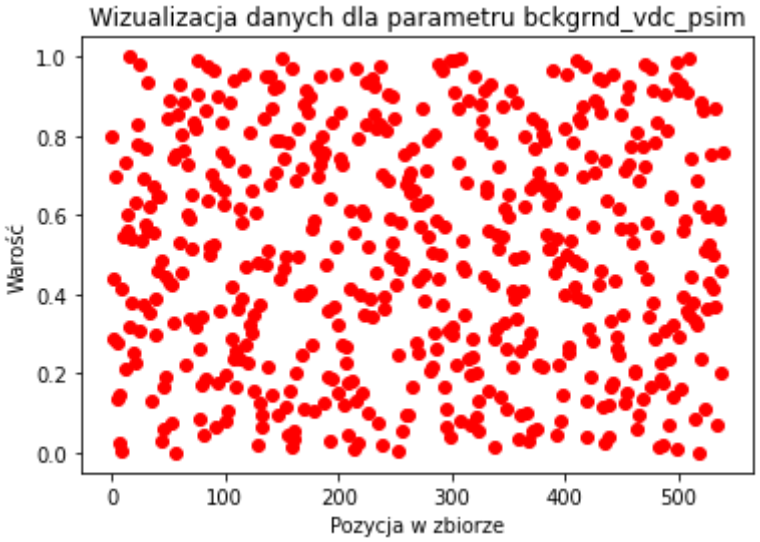
- **1.4.15 Wizualizacja danych dla parametru bckgrnd_vdc_eq**

In [19]:
```
import matplotlib.pyplot as plt
plt.plot(df['bckgrnd_vdc_eq'], 'ro')
plt.title('Wizualizacja danych dla parametru bckgrnd_vdc_eq')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



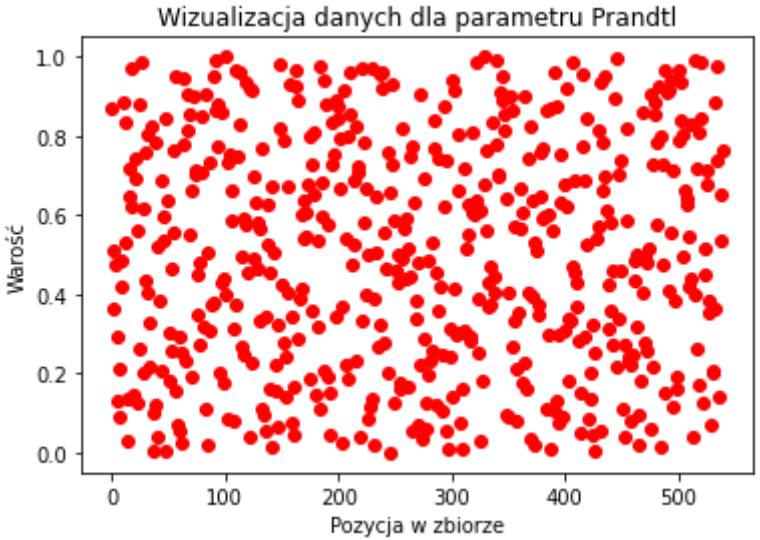
- **1.4.16 Wizualizacja danych dla parametru bckgrnd_vdc_psim**

In [20]:
```
import matplotlib.pyplot as plt
plt.plot(df['bckgrnd_vdc_psim'], 'ro')
plt.title('Wizualizacja danych dla parametru bckgrnd_vdc_psim')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



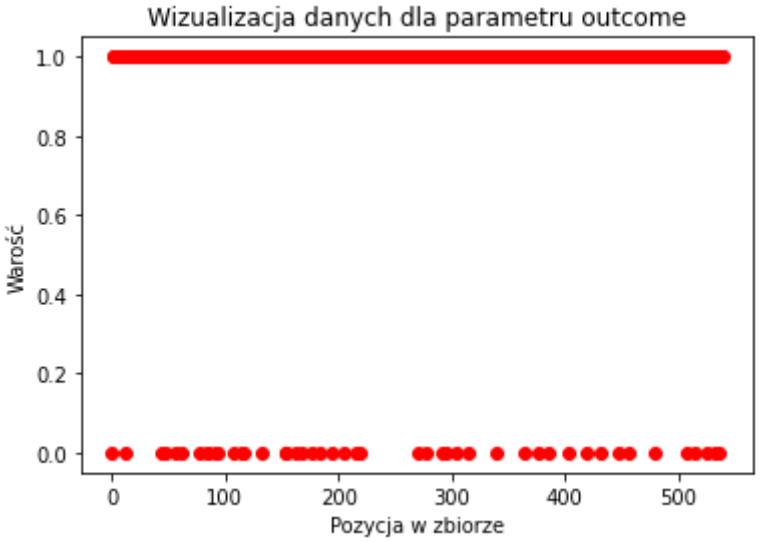
- **1.4.17 Wizualizacja danych dla parametru Prandtl**

In [21]:
```
import matplotlib.pyplot as plt
plt.plot(df['Prandtl'], 'ro')
plt.title('Wizualizacja danych dla parametru Prandtl')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



- **1.4.18 Wizualizacja danych dla parametru outcome**

In [22]:
```
import matplotlib.pyplot as plt
plt.plot(df['outcome'], 'ro')
plt.title('Wizualizacja danych dla parametru outcome')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



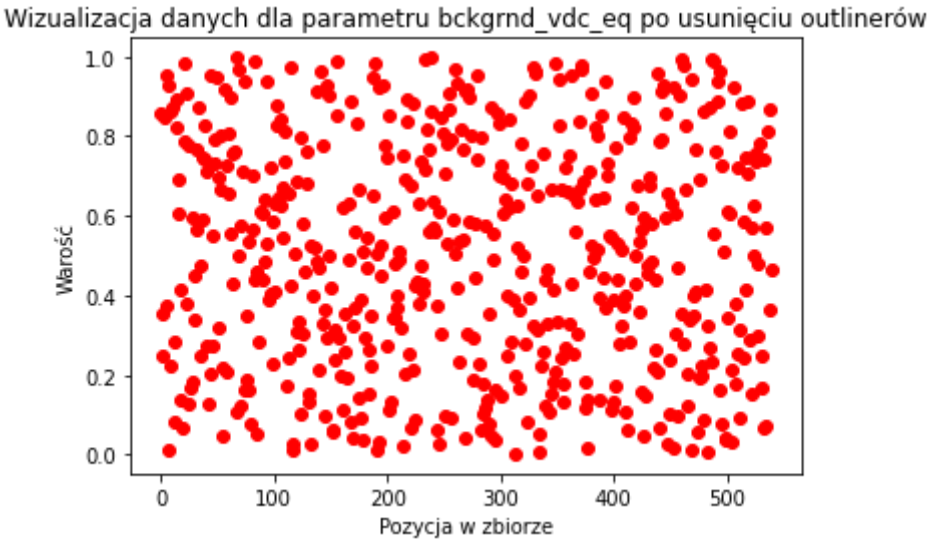
2. Usunięcie outlinerów z danych

- **2.1 Usunięcie outlinerów poprzez wyliczenie "z score" i odrzucenie przekraczających wyznaczony próg**

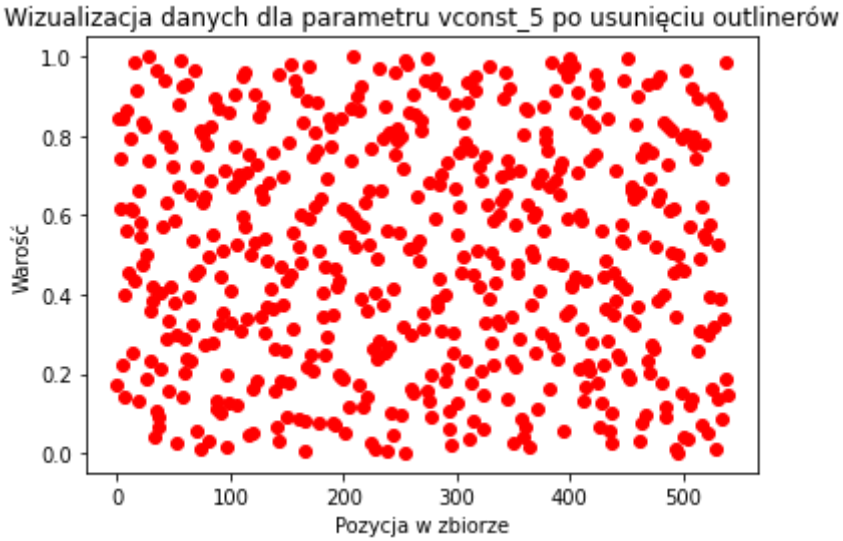
In [23]:
```
from scipy import stats
dfCopy = df

df = df[(np.abs(stats.zscore(df)) < 5).all(axis=1)] #zscore < 5 żeby nie wycinało poprawnych danych
```

- **2.2 Weryfikacja i wizualizacja wcześniej problematycznych kolumn**

In [24]:
```
import matplotlib.pyplot as plt
plt.plot(df['bckgrnd_vdc_eq'], 'ro')
plt.title('Wizualizacja danych dla parametru bckgrnd_vdc_eq po usunięciu outlinerów')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



In [25]:
```
import matplotlib.pyplot as plt
plt.plot(df['vconst_5'], 'ro')
plt.title('Wizualizacja danych dla parametru vconst_5 po usunięciu outlinerów')

plt.xlabel('Pozycja w zbiorze'); plt.ylabel('Warość');
```



- **2.3 Weryfikacja całości danych**

In [26]:
```
print (df.describe())
```

```
              Study         Run  vconst_corr    vconst_2    vconst_3  \
count  538.000000  538.000000   538.000000  538.000000  538.000000
mean     2.003717   90.286245     0.500628    0.499685    0.498483
std      0.816488   51.982591     0.288947    0.289300    0.288463
min      1.000000    1.000000     0.000414    0.001922    0.001181
25%      1.000000   45.250000     0.250620    0.250288    0.248889
50%      2.000000   90.000000     0.499998    0.498645    0.498887
75%      3.000000  135.000000     0.750795    0.750730    0.747663
max      3.000000  180.000000     0.999194    0.998815    0.998263

          vconst_4    vconst_5    vconst_7     ah_corr    ah_bolus  ...  \
count  538.000000  538.000000  538.000000  538.000000  538.000000  ...
mean     0.500054    0.501345    0.501065    0.500126    0.499939  ...
std      0.288701    0.288396    0.288665    0.288985    0.288861  ...
min      0.001972    0.001889    0.000476    0.004590    0.000296  ...
25%      0.251064    0.251983    0.252380    0.254232    0.251270  ...
50%      0.500456    0.501913    0.501757    0.499070    0.500074  ...
75%      0.749041    0.750208    0.750201    0.749003    0.747758  ...
max      0.997673    0.998944    0.997142    0.998930    0.998506  ...

       efficiency_factor  tidal_mix_max  vertical_decay_scale  convect_corr  \
count         538.000000     538.000000            538.000000    538.000000
mean            0.500709       0.499109              0.500082      0.500369
std             0.289308       0.289208              0.288520      0.289049
min             0.002015       0.000419              0.001188      0.001312
25%             0.250250       0.249863              0.250759      0.251371
50%             0.503133       0.498295              0.500151      0.500625
75%             0.751126       0.747833              0.747511      0.751128
max             0.999536       0.999942              0.997718      0.997518

       bckgrnd_vdc1  bckgrnd_vdc_ban  bckgrnd_vdc_eq  bckgrnd_vdc_psim  \
count    538.000000       538.000000      538.000000        538.000000
mean       0.499388         0.500122        0.501868          0.499197
std        0.288762         0.289170        0.287792          0.288761
min        0.002509         0.000732        0.002749          0.000219
25%        0.248744         0.250402        0.253663          0.250790
50%        0.499080         0.499959        0.502311          0.498955
75%        0.749161         0.749893        0.749923          0.747720
max        0.999795         0.999155        0.997265          0.999306

          Prandtl     outcome
count  538.000000  538.000000
mean     0.500602    0.914498
std      0.289393    0.279887
```

```
min      0.000263    0.000000
25%      0.248217    1.000000
50%      0.500931    1.000000
75%      0.751430    1.000000
max      0.999655    1.000000

[8 rows x 21 columns]
```

3. Podział danych na zestaw treningowy i testowy

In [27]:
```
from sklearn.model_selection import train_test_split

wether_train_data, wether_test_data, \
wether_train_target, wether_test_target = \
train_test_split(df.iloc[:, 2:20].values, df.outcome.values, test_size=0.35, random_state=41)
```

In [28]:
```
print("Zbiór treningowy:")
print("wether_train_data:", wether_train_data.shape)
print("wether_train_target:", wether_train_target.shape)
```

```
Zbiór treningowy:
wether_train_data: (349, 18)
wether_train_target: (349,)
```

In [29]:
```
print("Zbiór testowy:")
print("wether_test_data:", wether_test_data.shape)
print("wether_test_target:", wether_test_target.shape)
```

```
Zbiór testowy:
wether_test_data: (189, 18)
wether_test_target: (189,)
```

4. Klasyfikacja z wykorzystaniem regresji logistycznej

In [30]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression().fit(wether_train_data, wether_train_target)
```

In [49]:
```
from sklearn.metrics import accuracy_score
```

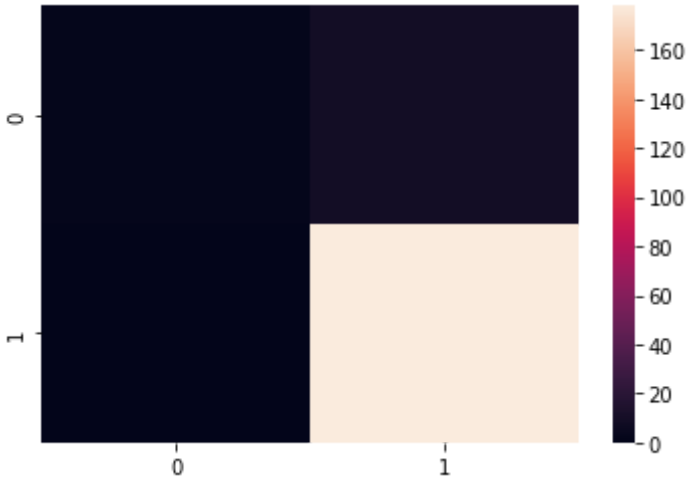
```
acc = accuracy_score(wether_test_target, clf.predict(wether_test_data))
print("Celność modelu: {0:0.2f}".format(acc))
```

```
Celność modelu: 0.95
```

In [32]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns;

conf_matrix = confusion_matrix(wether_test_target, clf.predict(wether_test_data))
print("Macierz konfuzji: ")
print(conf_matrix)
sns.heatmap(conf_matrix)
```

```
Macierz konfuzji:
[[  2   9]
 [  0 178]]
```

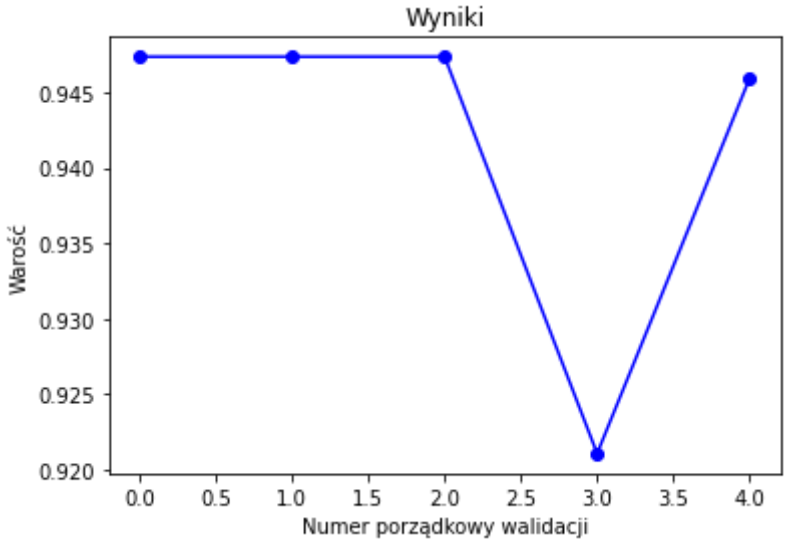
Out[32]:
```
<AxesSubplot:>
```



In [62]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

scores = cross_val_score(LogisticRegression(), wether_test_data, wether_test_target, cv=5)
print("\nWaldiacja krzyżowa: ")
print(scores)
```

```
Waldiacja krzyżowa:
[0.94736842 0.94736842 0.94736842 0.92105263 0.94594595]
```

In [34]:
```
import matplotlib.pyplot as plt

plt.plot(scores, 'bo-')
plt.title('Wyniki')

plt.xlabel('Numer porządkowy walidacji'); plt.ylabel('Warość');
```



5. Klasyfikacja z wykorzystaniem sieci neuronowych MLPClassifier

In [36]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

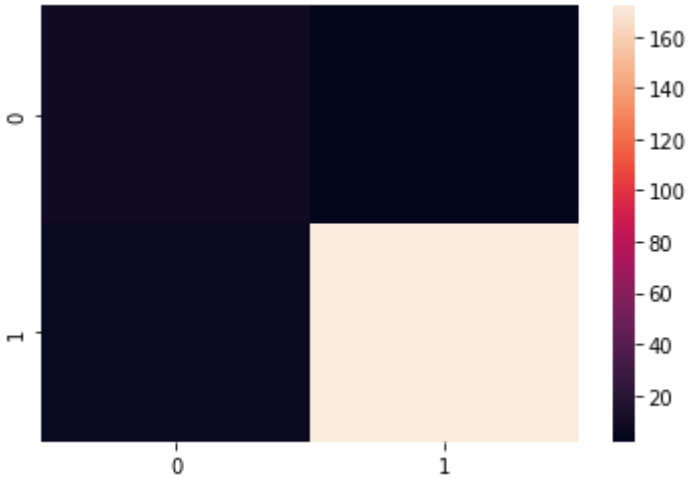
In [37]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier

parameters = {'hidden_layer_sizes': [(100), (50,25), (60,40,20,), (60, 40, 20, 10)], 'max_iter': [300,500,700],'solver': ['adam'

grid_searchMLP = GridSearchCV(MLPClassifier(), parameters, n_jobs=-1, verbose=True, cv=5, return_train_score=True)
grid_searchMLP.fit(wether_train_data, wether_train_target)

print("Najlepsze parametry: ")
print(grid_searchMLP.best_params_)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Najlepsze parametry:
{'activation': 'tanh', 'early_stopping': True, 'hidden_layer_sizes': (60, 40, 20), 'max_iter': 300, 'solver': 'lbfgs'}
```

In [38]:
```
import seaborn as sns;
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
classifier_score = grid_searchMLP.score(wether_test_data, wether_test_target)
print("Wynik najleszej kombinacji parametrów uczenia:")
print(classifier_score)

conf_matrix = confusion_matrix(wether_test_target, grid_searchMLP.predict(wether_test_data))
print("\n Macierz konfuzji:")
print(conf_matrix)
sns.heatmap(conf_matrix)
```

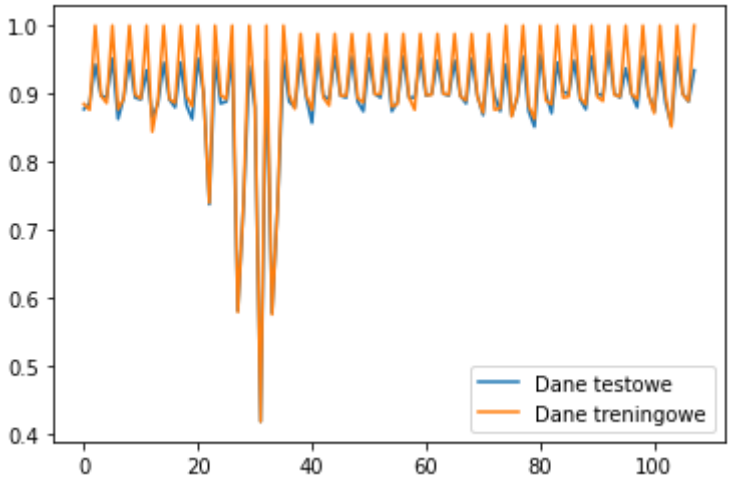
```
Wynik najleszej kombinacji parametrów uczenia:
0.9576719576719577

 Macierz konfuzji:
[[  9   2]
 [  6 172]]
```

Out[38]:
```
<AxesSubplot:>
```

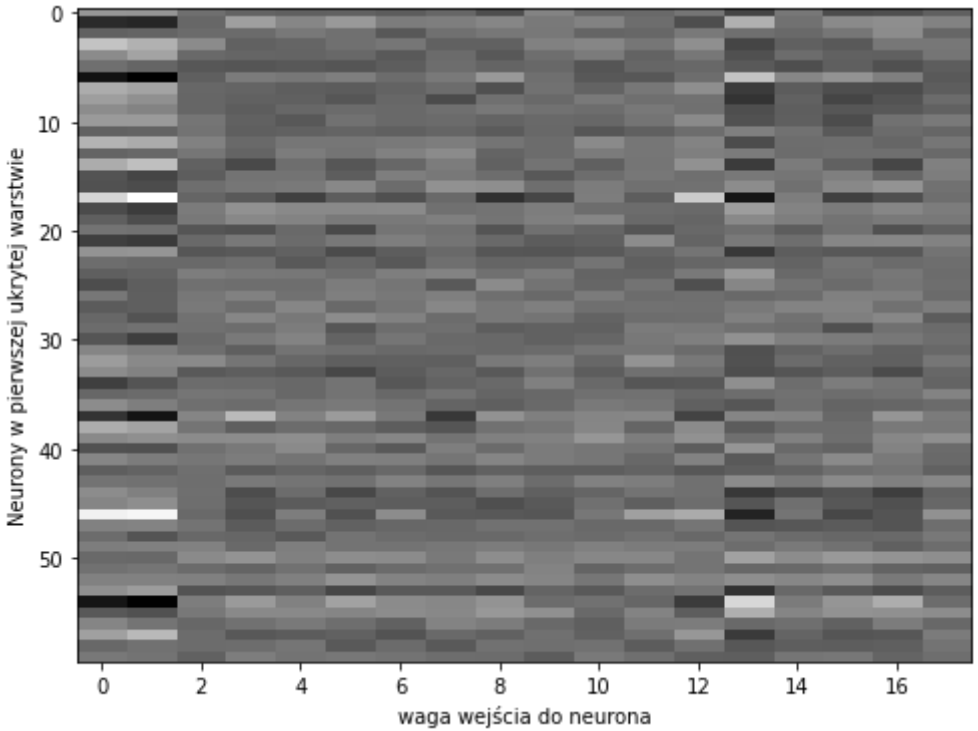


In [39]:
```
test_scores = grid_searchMLP.cv_results_['mean_test_score']
train_scores = grid_searchMLP.cv_results_['mean_train_score']

plt.plot(test_scores, label='Dane testowe')
```

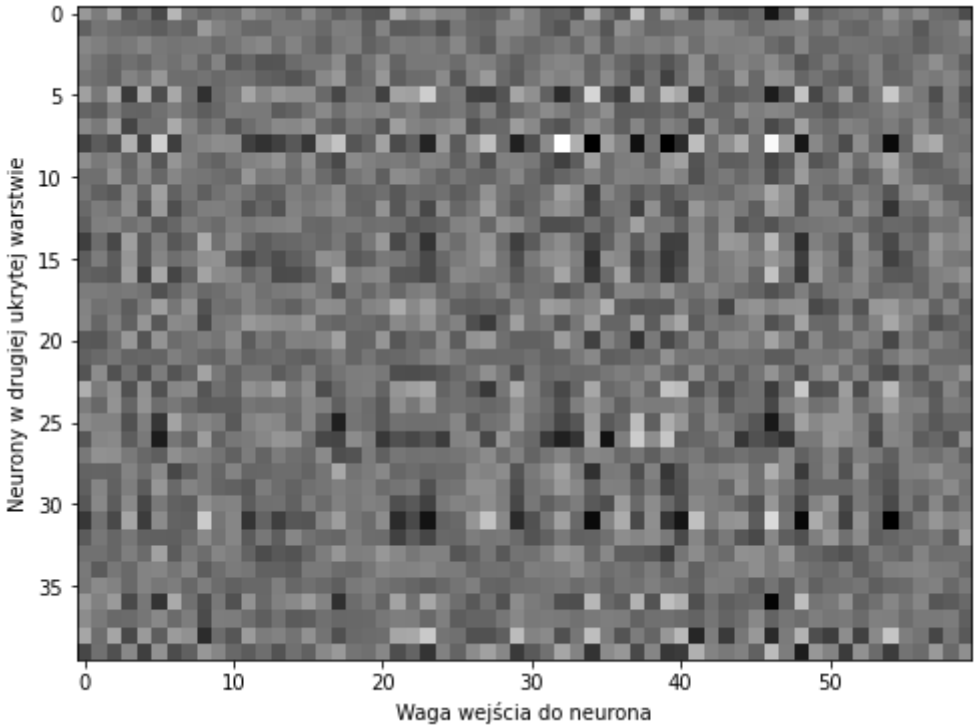
```
plt.plot(train_scores, label='Dane treningowe')
plt.legend(loc='best')
plt.show()
```



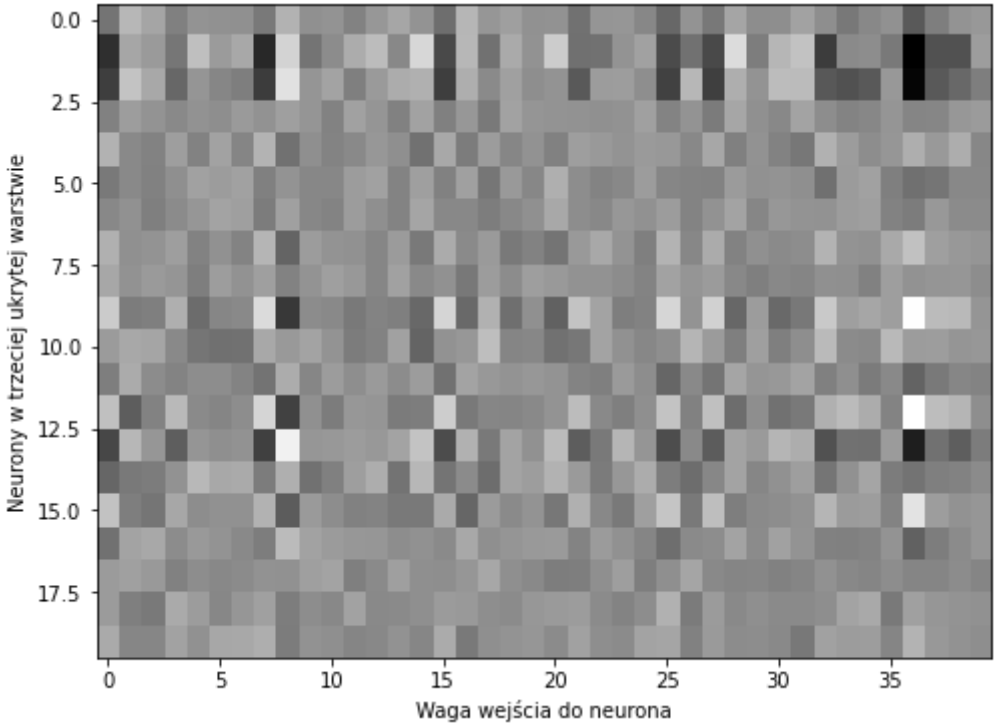
In [50]:
```
plt.rcParams['figure.figsize'] = (8.0, 6.0)
plt.imshow(np.transpose(grid_searchMLP.best_estimator_.coefs_[0]), cmap=plt.get_cmap("gray"), aspect="auto")
plt.ylabel('Neurony w pierwszej ukrytej warstwie'); plt.xlabel('Waga wejścia do neurona');
```



In [51]:
```
plt.rcParams['figure.figsize'] = (8.0, 6.0)
plt.imshow(np.transpose(grid_searchMLP.best_estimator_.coefs_[1]), cmap=plt.get_cmap("gray"), aspect="auto")
plt.ylabel('Neurony w drugiej ukrytej warstwie'); plt.xlabel('Waga wejścia do neurona');
```



In [52]:
```
plt.rcParams['figure.figsize'] = (8.0, 6.0)
plt.imshow(np.transpose(grid_searchMLP.best_estimator_.coefs_[2]), cmap=plt.get_cmap("gray"), aspect="auto")
plt.ylabel('Neurony w trzeciej ukrytej warstwie'); plt.xlabel('Waga wejścia do neurona');
```



6. Klasyfikacja z wykorzystaniem jeden przeciwko reszcie

In [44]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

multiclass_classifier = OneVsRestClassifier(LogisticRegression())
multiclass_classifier.fit(wether_train_data, wether_train_target);
```

In [61]:
```
from sklearn.metrics import accuracy_score
acc = accuracy_score(wether_test_target, clf.predict(wether_test_data))
print("Celność modelu: {0:0.2f}".format(acc))
```

```
Celność modelu: 0.95
```

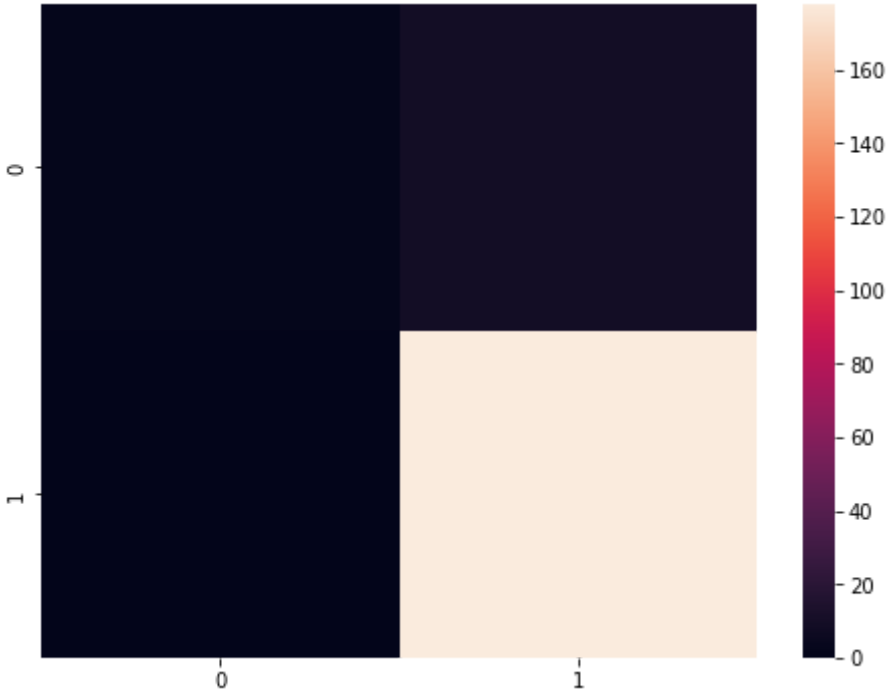
In [46]:
```
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(wether_test_target, multiclass_classifier.predict(wether_test_data))
print("Macierz konfuzji:")
```

```
print(conf_matrix)

import seaborn as sns;

conf_matrix = confusion_matrix(wether_test_target, clf.predict(wether_test_data))
sns.heatmap(conf_matrix)
```

```
Macierz konfuzji:
[[  2   9]
 [  0 178]]
```

Out[46]:

```
<AxesSubplot:>
```

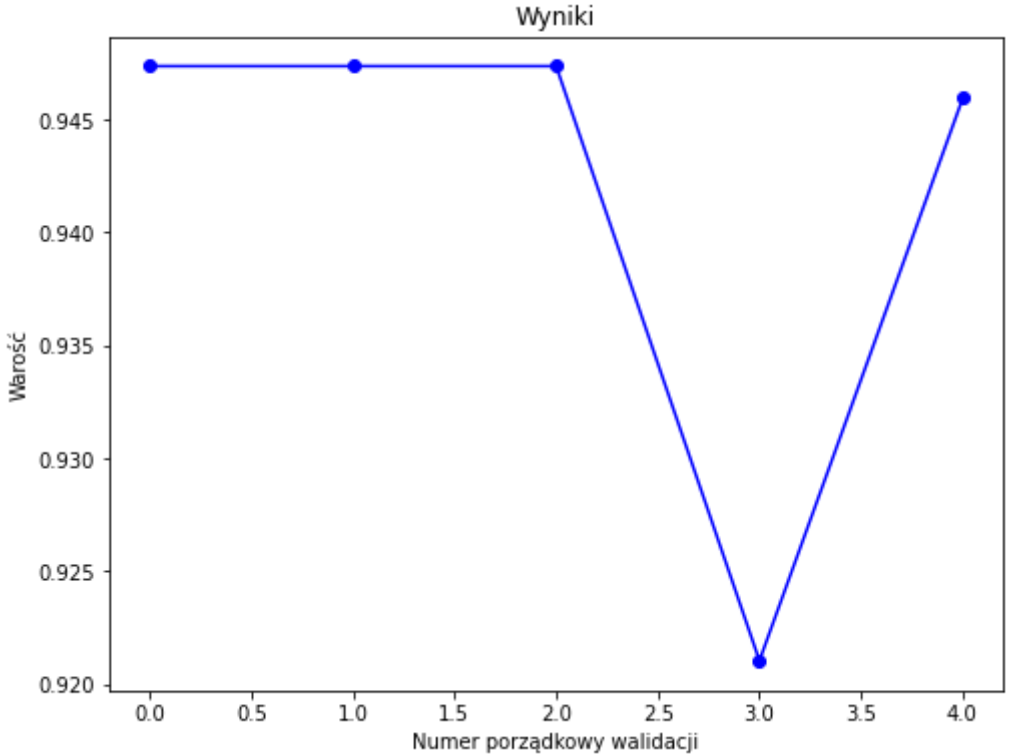


In [60]:

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

scores = cross_val_score(LogisticRegression(), wether_test_data, wether_test_target, cv=5)
print("Waldiacja krzyżowa: ")
print(scores)
```

```
Waldiacja krzyżowa:
[0.94736842 0.94736842 0.94736842 0.92105263 0.94594595]
```

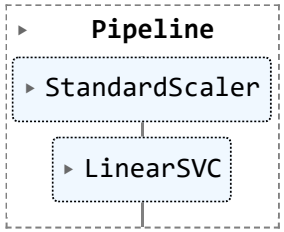
In [48]:
```
import matplotlib.pyplot as plt

plt.plot(scores, 'bo-')
plt.title('Wyniki')
plt.xlabel('Numer porządkowy walidacji'); plt.ylabel('Warość');
```



7. Klasyfikacja z wykorzystaniem jeden przeciwko reszcie

In [53]:
```
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

lSVC = make_pipeline(StandardScaler(),LinearSVC(random_state=0, tol=1e-5))
lSVC.fit(wether_train_data, wether_train_target)
```

Out[53]:



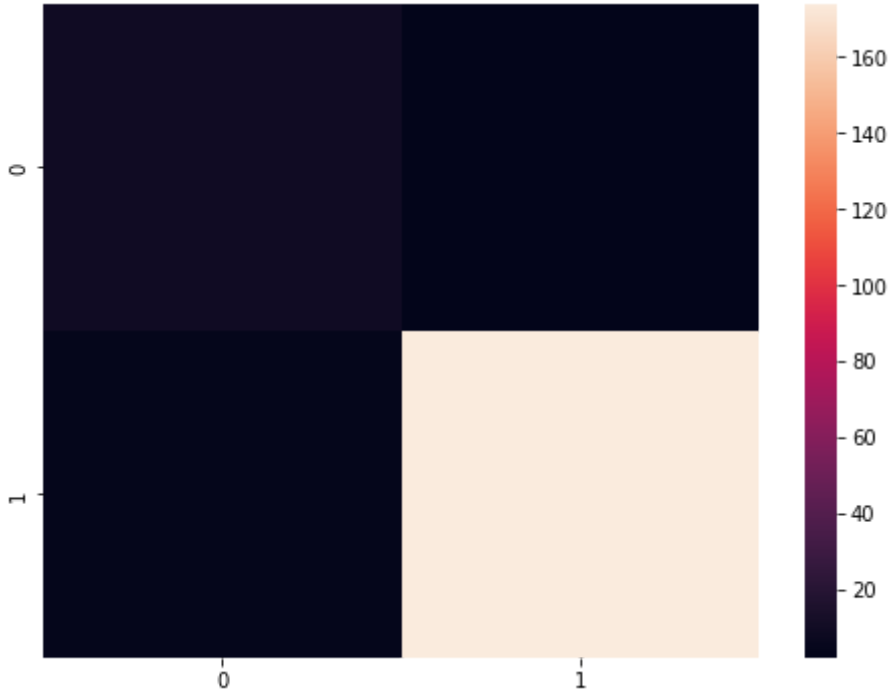
In [54]:
```
from sklearn.metrics import accuracy_score
acc = accuracy_score(wether_test_target, lSVC.predict(wether_test_data))
print("Celność modelu: {0:0.2f}".format(acc))
```

```
Celność modelu: 0.97
```

In [55]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns;

conf_matrix = confusion_matrix(wether_test_target, lSVC.predict(wether_test_data))
print("Macierz konfuzji: ")
print(conf_matrix)
sns.heatmap(conf_matrix)
```

```
Macierz konfuzji:
[[  9   2]
 [  4 174]]
```

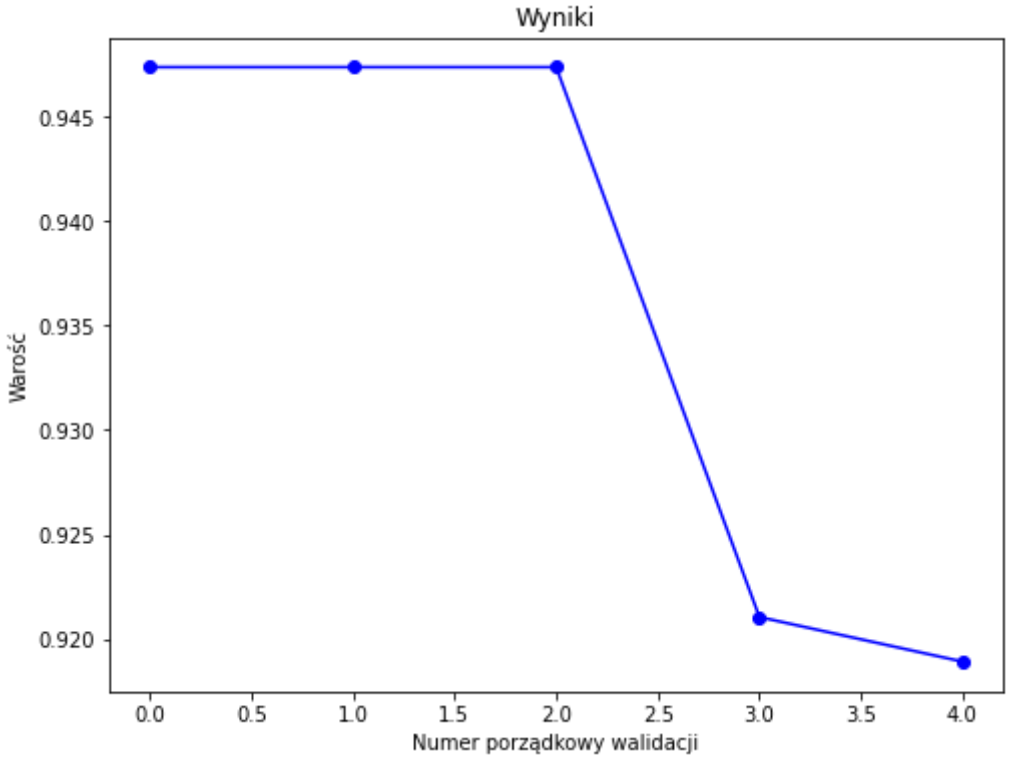
Out[55]:
```
<AxesSubplot:>
```



In [59]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

scores = cross_val_score(LinearSVC(), wether_test_data, wether_test_target, cv=5)
print("\nWaldiacja krzyżowa: ")
print(scores)
```

```
Waldiacja krzyżowa:
[0.94736842 0.94736842 0.94736842 0.92105263 0.91891892]
```

In [58]:
```
import matplotlib.pyplot as plt

plt.plot(scores, 'bo-')
plt.title('Wyniki')

plt.xlabel('Numer porządkowy walidacji'); plt.ylabel('Waroś');
```


Wyniki
Warość
0.945
0.940
0.935
0.930
0.925
0.920
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Numer porządkowy walidacji