<u>Zajęcia 21.04.2022</u>

###

CZĘŚĆ 2: Feature selection - DRZEWA DECYZYJNE

###

X_train, X_test, Y_train, Y_test = train_test_split(data3, data4, test_size=0.1)

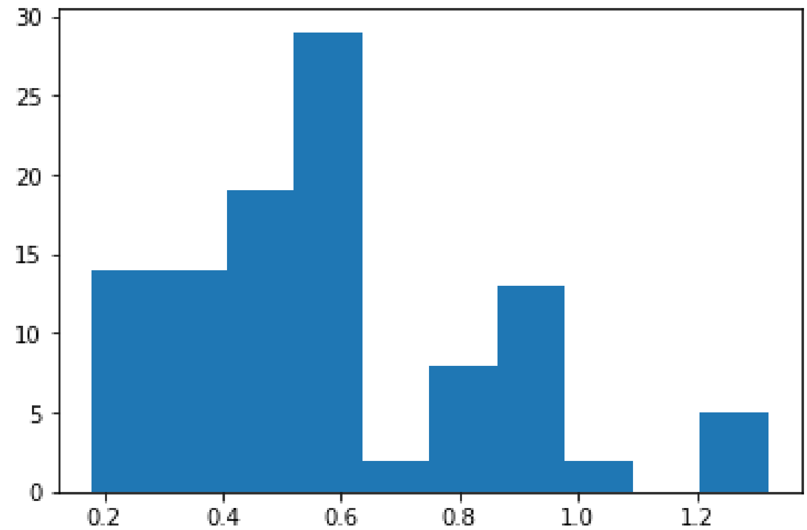
mi_score = MIC(X_train, Y_train.values.ravel())

print(mi_score)

```
[0.92403218 0.95123969 0.94306888 0.97765747 1.28502138 0.9226922
 0.22879732 0.92250271 0.49381046 1.28713197 1.31896165 1.29456677
 0.98617402 0.92101869 1.29654572 0.20777214 0.54913736 0.52320818
 0.28543165 0.62149566 0.62752931 0.61519629 0.37291342 0.40783803
 0.32958387 0.53770951 0.5684539  0.48385225 0.81200733 0.86314111
 0.83833096 0.58452376 0.6178193  0.35105181 0.32329391 0.5695011
 0.66515501 0.31418938 0.43756217 0.40451771 0.20097416 0.29823585
 0.56615183 0.54649127 0.53148449 0.3176608  0.52191221 0.77515371
 0.88593221 0.63970829 0.52702368 0.58015383 0.39129155 0.44779663
 0.44188477 0.5540886  0.17995615 0.2834558  0.87125072 0.841195
 0.90379885 0.51234104 0.87520837 0.47983681 0.62992435 0.57949626
 0.3895612  0.36293406 0.43681755 0.22049864 0.5461372  0.23225516
 0.18470539 0.86752468 0.82860514 0.85846519 0.49251693 0.51095388
 0.29059106 0.48526451 0.47261987 0.53412381 0.26711915 0.3181507
 0.23315623 0.5630301  0.57698081 0.49051373 0.62481873 0.41279126
 0.33491085 0.54578791 0.52173119 0.29018404 0.58387696 0.61629835
 0.35260353 0.52281828 0.77880738 0.88319226 0.50999203 0.87443756
 0.47768751 0.49406431 0.50923714 0.28732213]
```

np.histogram(mi_score)

plt.hist(mi_score) # widzimy, że najwięcej zmiennych wpada w przedział [0.4,0.6]



```
mi_score_selected_index = np.where(mi_score > 0.5)[0] # wybiorę zmienne, które mają mi_score > 0.5

X_2 = data3[data3.columns[mi_score_selected_index - 1]] # wybieram zmienne z odpowiednio dużym mi_score

X_train_2, X_test_2, Y_train2, Y_test2 = train_test_split(X_2, data4, test_size=0.1)

model_1 = DTC().fit(X_train,Y_train)
model_2 = DTC().fit(X_train_2,Y_train2)

score_1 = model_1.score(X_test,Y_test)
score_2 = model_2.score(X_test_2,Y_test2)

print(f"score_1:{score_1}\n score_2:{score_2}\n")
```

```
score_1:0.9954666666666667
 score_2:0.9969333333333333
```

```
# pozostałe kolumny w feature selection:

data3.columns[mi_score_selected_index - 1]

# liczba zmiennych objasniajacych które zostały:

len(data3.columns[mi_score_selected_index - 1]) # 63

# Czyli widzimy, że pomimo usunięcia 63 zmiennych, model praktycznie nie stracił na jakosci

###################################################
# Spróbujmy pójść dalej.
###################################################
```

```
mi_score_selected_index2 = np.where(mi_score > 0.8)[0] # wybiorę zmienne, które mają mi_score > 0.5

X_3 = data3[data3.columns[mi_score_selected_index2 - 1]] # wybieram zmienne z odpowiednio dużym mi_score

X_train_3, X_test_3, Y_train3, Y_test3 = train_test_split(X_3, data4, test_size=0.1)

model_3 = DTC().fit(X_train_3,Y_train3)
score_3 = model_3.score(X_test_3,Y_test3)

print(f"score_1:{score_1}\n score_3:{score_3}\n")
```

```
score_1:0.9954666666666667
 score_3:0.9926666666666667
```

```
# pozostałe kolumny w feature selection:

data3.columns[mi_score_selected_index2 - 1]

# liczba zmiennych objasniajacych które zostały:

len(data3.columns[mi_score_selected_index2 - 1]) # 26

##################################################
# Wciąż jest bardzo dobrze, idziemy dalej.
##################################################

mi_score_selected_index3 = np.where(mi_score > 0.95)[0] # wybiorę zmienne, które mają mi_score > 0.5

X_4 = data3[data3.columns[mi_score_selected_index2 - 1]] # wybieram zmienne z odpowiednio dużym mi_score
```

X_train_4, X_test_4, Y_train4, Y_test4 = train_test_split(X_4, data4, test_size=0.1)

model_4 = DTC().fit(X_train_4,Y_train4)

score_4 = model_4.score(X_test_4,Y_test4)

print(f"score_1:{score_1}\n score_4:{score_4}\n")

```
score_1:0.9954666666666667
 score_4:0.9916
```

pozostałe kolumny w feature selection:

data3.columns[mi_score_selected_index3 - 1]

```
Index(['AREA', 'MAJOR_AXIS', 'MINOR_AXIS', 'EXTENT', 'ASPECT_RATIO',
       'ROUNDNESS', 'COMPACTNESS', 'SHAPEFACTOR_2'],
      dtype='object')
```

liczba zmiennych objasniajacych które zostały:

len(data3.columns[mi_score_selected_index3 - 1]) # 8

Zostawiamy te 8 zmiennych.

##

############# CZĘŚĆ 3: LASY LOSOWE #############

##

model = RF()

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

n_scores = cross_val_score(model, X_train_4, Y_train4, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')

```
print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))

# Accuracy 0.995 - jestesmy bardzo zadowoleni
```

##

############# CZĘŚĆ 4: NAIWNY BAYES #############

##

```
gnb = GaussianNB()

Y_pred = gnb.fit(X_train_4, Y_train4.values.ravel()).predict(X_test_4)


print(classification_report(Y_test4, Y_pred))
```

```
              precision    recall  f1-score   support

     Arborio       0.49      0.33      0.39      1563
     Basmati       0.50      0.58      0.54      1452
      Ipsala       0.97      0.99      0.98      1472
     Jasmine       0.90      0.69      0.78      1499
   Karacadag       0.60      0.84      0.70      1514

    accuracy                           0.68      7500
   macro avg       0.69      0.69      0.68      7500
weighted avg       0.69      0.68      0.68      7500
```

```
# Sprawdzmy jeszcze dla wiekszej liczby zmiennych


Y_pred_wiecej = gnb.fit(X_train_3, Y_train3.values.ravel()).predict(X_test_3)


print(classification_report(Y_test3, Y_pred_wiecej))
```

```
              precision    recall  f1-score   support

     Arborio       0.47      0.33      0.38      1492
     Basmati       0.53      0.59      0.56      1507
      Ipsala       0.98      0.99      0.99      1471
     Jasmine       0.90      0.70      0.79      1512
   Karacadag       0.61      0.86      0.71      1518

    accuracy                           0.69      7500
   macro avg       0.70      0.69      0.69      7500
weighted avg       0.70      0.69      0.69      7500
```