LAB0 - Podstawy

Przedmiot: *Sztuczna Inteligencja w Systemach Informatycznych*

Autor: Tomasz Szydło

kontakt: tszydlo@agh.edu.pl

Wprowadzenie

Celem zajęć jest zaznajomienie się z podstawowymi technikami sotosowanymi w uczeniu maszynowym. W trakcie zajęć laboratoryjnych zostanie wykorzystany zbiór danych Irys, a następnie dla wybranych algorytmów uczenia maszynowego zostanie zidentyfikowany jeden, który cechuje się najwyższą jakością klasyfikacji.

Środowisko pracy: Anaconda, Jupyter Notebook

Narzędzia: Scikit Learn, Python

Zbiór *Iris*

Zbiór danych *iris* zawiera 150 instancji opisujących kwiaty Irysa. Kwiaty są określone przy pomocy 4 atrybutów numerycznych opisujących długości i szerokości płatków kwiatu *sepal* i *petal*. Ostatni atrybut jakościowy definiuje gatunek opisywanego Irysa (species). Jednym z problemów możliwych do rozwiązania przy użyciu tego zbioru uczącego jest określenie gatunku dla nowego kwiatu Irysa w zależności od wielkości jego płatków. Jest to przykład zadania klasyfikacji.

Zbiór uczący *iris* jest zbiorem etykietowanym ponieważ klasa, czyli konkretny gatunek irysa jest znany dla każdej instancji (obiektu) w zbiorze uczącym:

https://archive.ics.uci.edu/ml/datasets/iris

```
Source:

Creator:

R.A. Fisher

Donor:

Michael Marshall (MARSHALL%PLU '@' io.arc.nasa.gov)

Data Set Information:

This is perhaps the best known database to be found in the pattern
recognition literature. Fisher's paper is a classic in the field and
```

```
is referenced frequently to this day. (See Duda & Hart, for example.)
The data set contains 3 classes of 50 instances each, where each class
refers to a type of iris plant. One class is linearly separable from
the other 2; the latter are NOT linearly separable from each other.

Predicted attribute: class of iris plant.

This is an exceedingly simple domain.

This data differs from the data presented in Fishers article
(identified by Steve Chadwick, spchadwick '@' espeedaz.net ). The 35th
sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in
the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa"
where the errors are in the second and third features.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica
```

Konfiguracja wstępna

```
# Konfiguracja interakcji z wykresami pakietu matplotlib

# Standardowe rozwiązanie
%matplotlib inline

# Interaktywne wykresy - może powodować błędy
# %matplotlib notebook

# Załaduj biblioteki
from pandas import read_csv
from matplotlib import pyplot
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

import numpy as np


pyplot.rcParams["figure.figsize"] = (12, 6) # Changes the size of
pyplot graphs on Google Collab
```

Operacje na danych tablicowych - przykłady dla *NumPy*

```
# NumPy - tworzenie tablic

a = np.array([1,2,3])
b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]], dtype =
float)

print(a)
print(b)
print(c)
```

```
[1 2 3]
[[1.5 2.  3. ]
 [4.  5.  6. ]]
[[[1.5 2.  3. ]
  [4.  5.  6. ]]

 [[3.  2.  1. ]
  [4.  5.  6. ]]]
```

```
# NumPy - przeglądanie tablic

print(a.shape)
print(len(a))
print(b.ndim)
print(c.size)
print(b.dtype)
print(b.dtype.name)
print(b.astype(int))
```

```
(3,)
3
2
12
float64
float64
[[1 2 3]
 [4 5 6]]
```

```
# NumPy - subsetting

print(a[2]) #wybierz element z indeksem równym 2
print(b[1,2]) # wybierz element z pierwszego wiersza i drugiej kolumny

# NumPy - slicing
print(a[0:2])   #wybierz elementy o indeksach 0 i 1
print(b[0:2,1]) #wybierz elementy z wiersza 0 i 1 oraz kolumny 1

print(b[:1])    #wybierz elementy z wiersza 0
print(b[0:1,:]) #wybierz elementy z wiersza 0
```

```
3
6.0
[1 2]
[2. 5.]
[[1.5 2.  3. ]]
[[1.5 2.  3. ]]
```

Wczytywanie danych - biblioteka *Pandas*

Przeanalizuj w edytorze tekstowym format i zawartośc pliku wejściowego ze zbiorem - plik *iris.csv*.

UWAGA! W przypadku błędów z odczytem danych popraw pliki zawierające dane wejściowe.

```
filename = 'iris.csv'
dataset = read_csv(filename)
```

Wypisz rozmiar danych wejściowych.

```
print(dataset.shape)
```

```
(150, 5)
```

Sprawdź poprawność odczytania danych poprzez wypisanie pierwszych 20 wierszy.

```
dataset[0:20]
```

```
    sepal.length  sepal.width  petal.length  petal.width variety
0            5.1          3.5           1.4          0.2  Setosa
1            4.9          3.0           1.4          0.2  Setosa
2            4.7          3.2           1.3          0.2  Setosa
3            4.6          3.1           1.5          0.2  Setosa
4            5.0          3.6           1.4          0.2  Setosa
5            5.4          3.9           1.7          0.4  Setosa
6            4.6          3.4           1.4          0.3  Setosa
7            5.0          3.4           1.5          0.2  Setosa
8            4.4          2.9           1.4          0.2  Setosa
9            4.9          3.1           1.5          0.1  Setosa
10           5.4          3.7           1.5          0.2  Setosa
```

```
11           4.8          3.4           1.6          0.2  Setosa
12           4.8          3.0           1.4          0.1  Setosa
13           4.3          3.0           1.1          0.1  Setosa
14           5.8          4.0           1.2          0.2  Setosa
15           5.7          4.4           1.5          0.4  Setosa
16           5.4          3.9           1.3          0.4  Setosa
17           5.1          3.5           1.4          0.3  Setosa
18           5.7          3.8           1.7          0.3  Setosa
19           5.1          3.8           1.5          0.3  Setosa
```

Analiza statystyczna

Sprawdź jaki jest przedział wartości, czy nie ma elementów znacznie odbiegających od wartości oczekiwanej. Mogłoby to sugerować błędy w danych wejściowych.

```
# descriptions
print(dataset.describe())
```

```
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

Sprawdź czy liczność klas wynikowych jest zbliżona. Występowanie znacznych dysproporcji może skutkować błędnym wyuczeniem modelu.

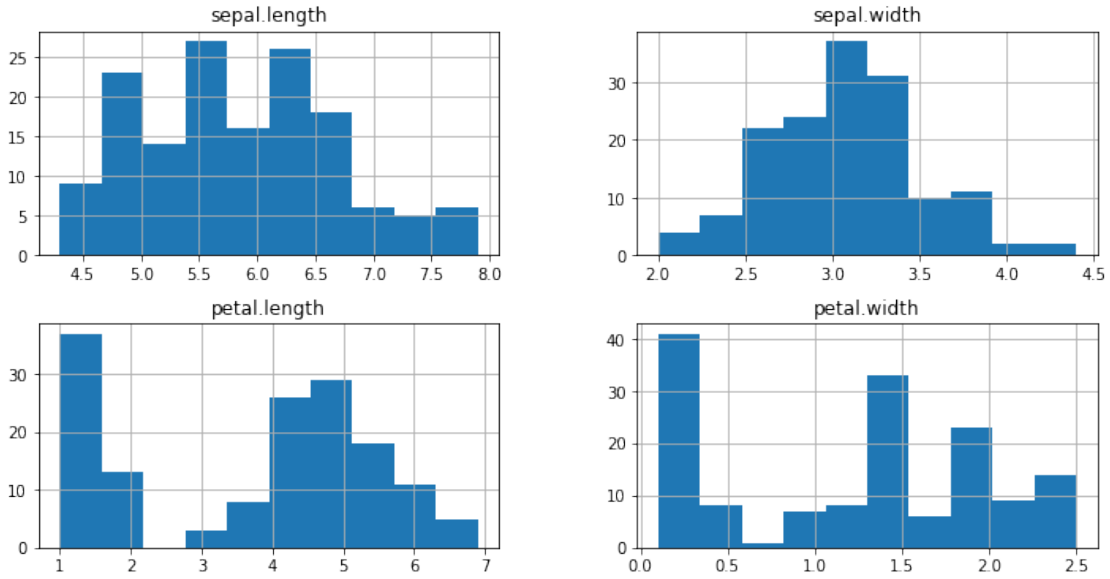
```
# class distribution
print(dataset.groupby(['variety']).size())
```

```
variety
Setosa        50
Versicolor    50
Virginica     50
dtype: int64
```

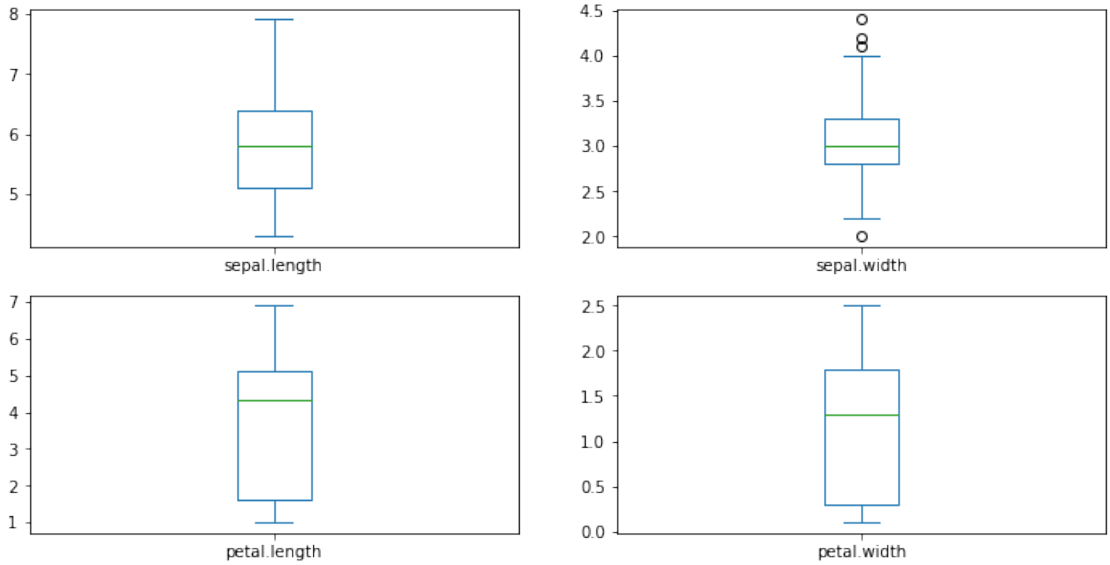
Wizualizacja danych

Narysuj wykresy przedstawiające dane wejściowe.

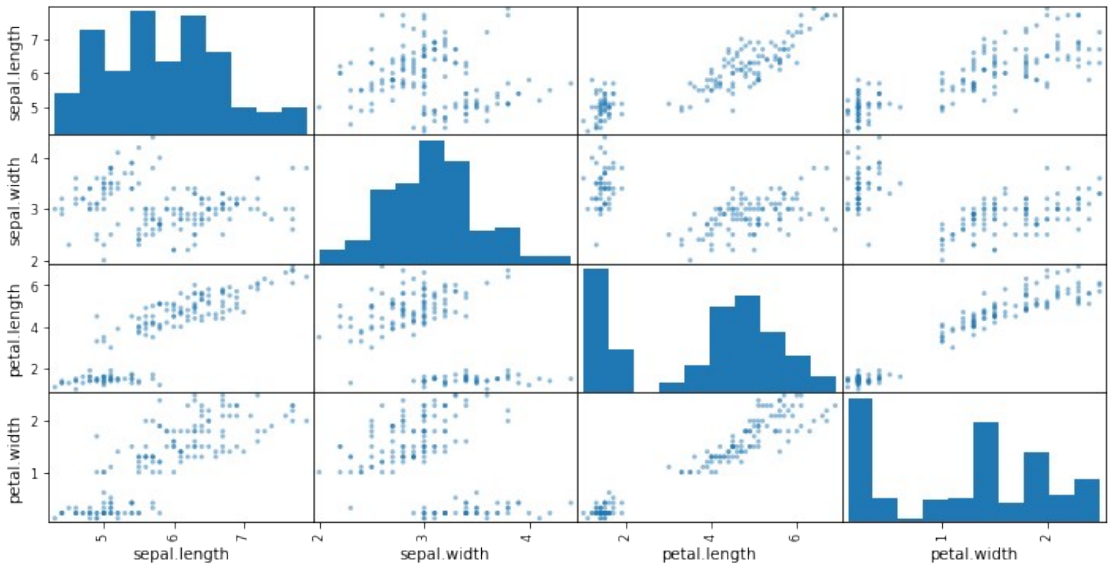
```
dataset.hist()
pyplot.show()
```




```
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
pyplot.show()
```




```
scatter_matrix(dataset)
pyplot.show()
```



Ewaluacja wybranego algorytmu ML

Na potrzeby klasyfikacji irysów zdecydowano się wykorzystać model ML bazujący ba regresji logistycznej. W procesie uczenia wykrozystamy dwa zbiory danych - treningowy oraz testowy z podziałem 20%/80%.

Stwórz dwie tablice:

- X składającą się z 4 kolumn - cechy wejściowe
- Y składającą się z 1 kolumny - etykiety klasy dla zbioru uczącego

```
array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]
```

Wypisz tablice aby sprawdzić ich poprawność.

```
print(X.shape)
print(X)

print(Y.shape)
print(Y)
```

```
(150, 4)
[[5.1 3.5 1.4 0.2]
 [4.9 3.0 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.0 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.0 3.4 1.5 0.2]
```

```
[4.4 2.9 1.4 0.2]
[4.9 3.1 1.5 0.1]
[5.4 3.7 1.5 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.0 1.4 0.1]
[4.3 3.0 1.1 0.1]
[5.8 4.0 1.2 0.2]
[5.7 4.4 1.5 0.4]
[5.4 3.9 1.3 0.4]
[5.1 3.5 1.4 0.3]
[5.7 3.8 1.7 0.3]
[5.1 3.8 1.5 0.3]
[5.4 3.4 1.7 0.2]
[5.1 3.7 1.5 0.4]
[4.6 3.6 1.0 0.2]
[5.1 3.3 1.7 0.5]
[4.8 3.4 1.9 0.2]
[5.0 3.0 1.6 0.2]
[5.0 3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.0 3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.0 1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.0 3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.0 3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.0 1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.0 3.3 1.4 0.2]
[7.0 3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.0 1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1.0]
```

[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.0 2.0 3.5 1.0]
[5.9 3.0 4.2 1.5]
[6.0 2.2 4.0 1.0]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.0 4.5 1.5]
[5.8 2.7 4.1 1.0]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.0 1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.0 4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.0 5.0 1.7]
[6.0 2.9 4.5 1.5]
[5.7 2.6 3.5 1.0]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1.0]
[5.8 2.7 3.9 1.2]
[6.0 2.7 5.1 1.6]
[5.4 3.0 4.5 1.5]
[6.0 3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.0 4.1 1.3]
[5.5 2.5 4.0 1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.0 4.6 1.4]
[5.8 2.6 4.0 1.2]
[5.0 2.3 3.3 1.0]
[5.6 2.7 4.2 1.3]
[5.7 3.0 4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.0 1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.0 2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.0 5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.0 5.8 2.2]
[7.6 3.0 6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]

```
 [6.7 2.5 5.8 1.8]
 [7.2 3.6 6.1 2.5]
 [6.5 3.2 5.1 2.0]
 [6.4 2.7 5.3 1.9]
 [6.8 3.0 5.5 2.1]
 [5.7 2.5 5.0 2.0]
 [5.8 2.8 5.1 2.4]
 [6.4 3.2 5.3 2.3]
 [6.5 3.0 5.5 1.8]
 [7.7 3.8 6.7 2.2]
 [7.7 2.6 6.9 2.3]
 [6.0 2.2 5.0 1.5]
 [6.9 3.2 5.7 2.3]
 [5.6 2.8 4.9 2.0]
 [7.7 2.8 6.7 2.0]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.0 1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.0 4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.0 5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2.0]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.0 6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.0 3.0 4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.0 5.2 2.3]
 [6.3 2.5 5.0 1.9]
 [6.5 3.0 5.2 2.0]
 [6.2 3.4 5.4 2.3]
 [5.9 3.0 5.1 1.8]]
(150,)
['Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
```

```
'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
'Setosa'
 'Setosa' 'Setosa' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
'Virginica'
 'Virginica' 'Virginica' 'Virginica']
```

Stwórz zbiór uczący i testowy z podziałem 80%/20%.

```
validation_size = 0.20
seed = 7

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=seed)
```

Naucz model, a następnie sprawdź jego skuteczność.

```
klr = LogisticRegression(max_iter=2000)
klr.fit(X_train, Y_train)
predictions = klr.predict(X_test)

print(accuracy_score(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test, predictions))
```

```
0.8666666666666667
[[ 7  0  0]
 [ 0 10  2]
 [ 0  2  9]]
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00         7
  Versicolor       0.83      0.83      0.83        12
   Virginica       0.82      0.82      0.82        11

    accuracy                           0.87        30
   macro avg       0.88      0.88      0.88        30
weighted avg       0.87      0.87      0.87        30
```

Wybór najlepszego algorytmu ML dla zadanego problemu

Sprawdź skuteczność innych modeli ML w problemie klasyfikacji Irysów. Jako miarę jakości klasyfikacji wybierz *accuracy_score*. Wyniki skuteczności ich działania przedstaw na wykresie.

Przeanalizuj metody:

- LogisticRegression()
- KNeighborsClassifier()
- DecisionTreeClassifier()
- SVC()
- GaussianNB()

Lista dostępnych algorytmów w bibliotece *scikit-learn*:
https://scikit-learn.org/stable/modules/multiclass.html#multiclass

Dla wybranej metody wylicz wskaźniki jakościowe dotyczące predykcji.

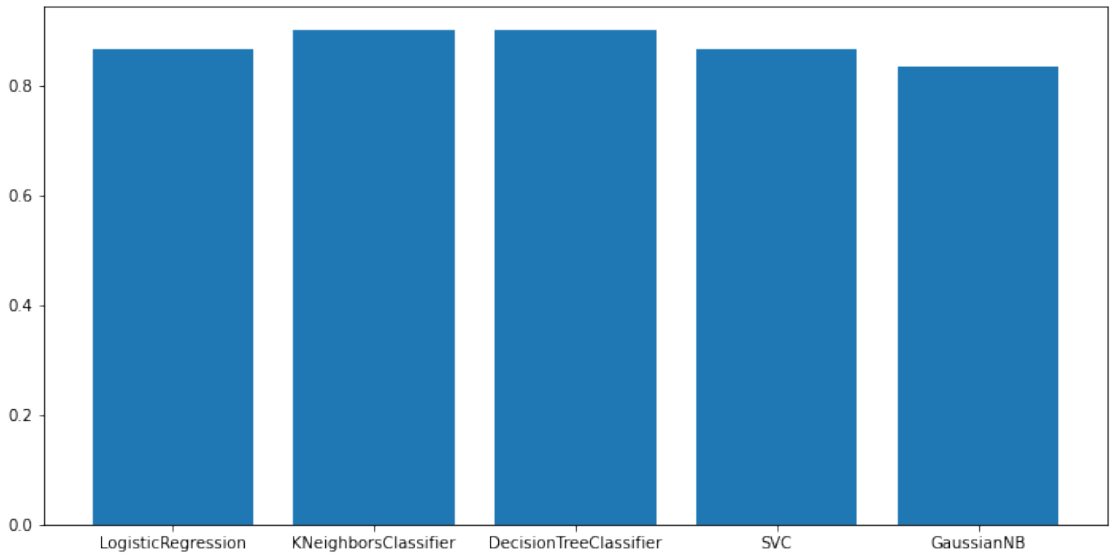
```
names = []
results = []


names.append("LogisticRegression")
temp_klr_1 = LogisticRegression(max_iter=2000)
temp_klr_1.fit(X_train, Y_train)
temp_predictions = temp_klr_1.predict(X_test)
results.append(accuracy_score(Y_test, temp_predictions))


names.append("KNeighborsClassifier")
temp_klr_2 = KNeighborsClassifier(n_neighbors=3)
temp_klr_2.fit(X_train, Y_train)
```

```
temp_predictions = temp_klr_2.predict(X_test)
results.append(accuracy_score(Y_test, temp_predictions))


names.append("DecisionTreeClassifier")
temp_klr_3 = DecisionTreeClassifier(random_state=0)
temp_klr_3.fit(X_train, Y_train)
temp_predictions = temp_klr_3.predict(X_test)
results.append(accuracy_score(Y_test, temp_predictions))


names.append("SVC")
temp_klr_4 = SVC(gamma=2, C=1)
temp_klr_4.fit(X_train, Y_train)
temp_predictions = temp_klr_4.predict(X_test)
results.append(accuracy_score(Y_test, temp_predictions))


names.append("GaussianNB")
temp_klr_5 = GaussianNB()
temp_klr_5.fit(X_train, Y_train)
temp_predictions = temp_klr_5.predict(X_test)
results.append(accuracy_score(Y_test, temp_predictions))

print(names)
print(results)
```

```
['LogisticRegression', 'KNeighborsClassifier',
'DecisionTreeClassifier', 'SVC', 'GaussianNB']
[0.8666666666666667, 0.9, 0.9, 0.8666666666666667, 0.8333333333333334]
```

```
pyplot.bar(names,results)
```

```
<BarContainer object of 5 artists>
```



Policz skuteczność najlepszego modelu.

```
best = temp_klr_3.predict(X_test) # DecisionTreeClassifier

print(accuracy_score(Y_test, best))
print(confusion_matrix(Y_test, best))
print(classification_report(Y_test, best))
```

```
0.9
[[ 7  0  0]
 [ 0 10  2]
 [ 0  1 10]]
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00         7
  Versicolor       0.91      0.83      0.87        12
   Virginica       0.83      0.91      0.87        11

    accuracy                           0.90        30
   macro avg       0.91      0.91      0.91        30
weighted avg       0.90      0.90      0.90        30
```

Podsumowanie

TODO jaki problem ML występował w zadaniu?

TODO jak się rozkłada liczność klas wynikowych

TODO jakie algorytmu ML były rozważane?

TODO który z algorytmów cechował się najwyższą skutecznością?