Aleksandra Wasik Assignment 1

```
# import modules for this project
from sklearn import datasets
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
#help(KNeighborsClassifier)
```

Use the following code to generate an artificial dataset which contain three classes. Conduct a similar KNN analysis to the dataset and report your accuracy.

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import numpy as np

centers = [[2, 4], [6, 6], [1, 9]]
n_classes = len(centers)
data, labels = make_blobs(n_samples=150,
                          centers=np.array(centers),
                          random_state=1)
# do a 80-20 split of the data
res = train_test_split(data, labels,
                       train_size=0.8,
                       test_size=0.2,
                       random_state=None)
train_data, test_data, train_labels, test_labels = res
# perform a KNN analysis of the simulated data
from sklearn.neighbors import KNeighborsClassifier
# knn, no parameters
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)
print("Predictions from the classifier:")
learn_data_predicted = knn.predict(train_data)
print(learn_data_predicted)
print("Target values:")
print(train_labels)
print(accuracy_score(learn_data_predicted, train_labels))

# re-do KNN using some specific parameters.
knn = KNeighborsClassifier(algorithm = 'auto',
                           leaf_size = 30,
                           metric = 'minkowski',
                           p = 2,         # p=2 is equivalent to euclidian distance
                           metric_params = None,
                           n_jobs = None,
                           n_neighbors = 5,
                           weights = 'uniform')

knn.fit(train_data, train_labels)
test_data_predicted = knn.predict(test_data)

# output accuracy score
accuracy_score(test_data_predicted, test_labels)

# plot your different results
#I cannot show the different results, because the accuracy is 100%.
#It means that the model can perfectly classify each dataset.
#Not very common in real life
```

```
Predictions from the classifier:
[1 0 1 1 0 1 1 1 2 1 2 0 2 0 2 0 2 1 0 0 2 1 0 1 2 0 0 1 0 0 0 0 0 1 0 2 0
 1 2 1 1 1 2 0 0 1 0 1 1 0 0 0 1 1 1 2 0 0 0 0 1 0 2 2 1 2 0 1 1 2 2 1 2 0
 2 1 2 2 2 0 2 2 1 0 1 0 2 0 1 2 2 2 0 2 1 0 0 0 0 0 1 0 2 1 2 1 2 1 2 2 1
 2 1 2 0 1 2 2 0 2]
Target values:
[1 0 1 1 0 1 1 1 2 1 2 0 2 0 2 0 2 1 0 0 2 1 0 1 2 0 0 1 0 0 0 0 0 1 0 2 0
 1 2 1 1 1 2 0 0 1 0 1 1 0 0 0 1 1 1 2 0 0 0 0 1 0 2 2 1 2 0 1 1 2 2 1 2 0
 2 1 2 2 2 0 2 2 1 0 1 0 2 0 1 2 2 2 0 2 1 0 0 0 0 0 1 0 2 1 2 1 2 1 2 2 1
 2 1 2 0 1 2 2 0 2]
1.0
1.0
```