

A) Read Data from your disk and make predictions using the KNN Classifier.

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('/Users/test/Downloads/breast-cancer-wisconsin.data.txt')

df.replace('?', -99999, inplace=True)
df.drop(['id'], 1, inplace=True)

X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

clf = KNeighborsClassifier(n_neighbors=1)
clf.fit(X_train, y_train)

print (clf.score(X_train, y_train))
print (clf.score(X_test, y_test))

data = np.array([4,3,3,2,1,2,1,1,2])
prediction= clf.predict(data.reshape(1,-1))

print(prediction)
```

Tasks


1) Change number of neighbors (ie, K) and test the accuracy of both training and testing - print results.

2) Try NOT to drop the 'id' column then evaluate both training and testing outcomes (scores).

You may notice that both scores are less accurate, explain why?

3) Explore other parameters of the **KNeighborsClassifier** function, for instance, can we apply Manhattan distance instead of the Euclidean. If so, test its use and compare with the results you had earlier.

B) Read Data from your disk and make predictions using the KNN Classifier.



```
19 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
20
21
22 training_accuracy = []
23 test_accuracy = []
24 # try n_neighbors from 1 to 10
25 neighbors_settings = range(1, 11)
26
27 for n_neighbors in neighbors_settings:
28     # build the model
29     clf = KNeighborsClassifier(n_neighbors=n_neighbors)
30     clf.fit(X_train, y_train)
31     # record training set accuracy
32     training_accuracy.append(clf.score(X_train, y_train))
33     # record generalization accuracy
34     test_accuracy.append(clf.score(X_test, y_test))
35
36 plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
37 plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
38 plt.ylabel("Accuracy")
39 plt.xlabel("n_neighbors")
40 plt.legend()
```

Tasks

Note: the code until the red arrow is copied from the previous example (you still should use it).

1) If you run the code above you will produce a figure - print it or save it and discuss/explore what it means. What is your understanding from the produced figure?

C) Do it again, now with breast cancer data loaded directly from sklearn

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=66)
```

Tasks

The above lines of code will load same/similar dataset from **sklearn**.

- 1) Explore the data, the attributes, is there any missing data?
- 2) Fit a classifier (Knn) and evaluate accuracy corresponding to different “model” settings (K, distance, etc)
- 3) Do you remember Confusion Matrix (check material on Week 3) — execute the following snippet and discuss results.

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```

- 4) Can you produce F1 score on data test split (refer to sklearn documentation, or would you implement it yourself?).
- 5) Check feature scales? —investigate sample central tendency and dispersion and explore sklearn methods to normalise dataset, if needed.

Visualise classifier's boundary — toy example on slide number 23 in Lecture Notes can be produced by:

```
import numpy as np
import pandas as pd

import mglearn

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

%matplotlib inline

X, y = mglearn.datasets.make_forge()
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
clf = KNeighborsClassifier(n_neighbors=3)

fig, axes = plt.subplots(1, 3, figsize=(10, 3))

for n_neighbors, ax in zip([1, 3, 9], axes):
    clf = KNeighborsClassifier(n_neighbors=n_neighbors).fit(X, y)
    mglearn.plots.plot_2d_separator(clf, X, fill=True, eps=0.5, ax=ax, alpha=.4)
    mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
    ax.set_title("{} neighbor(s)".format(n_neighbors))
    ax.set_xlabel("feature 0")
    ax.set_ylabel("feature 1")
axes[0].legend(loc=3)
```

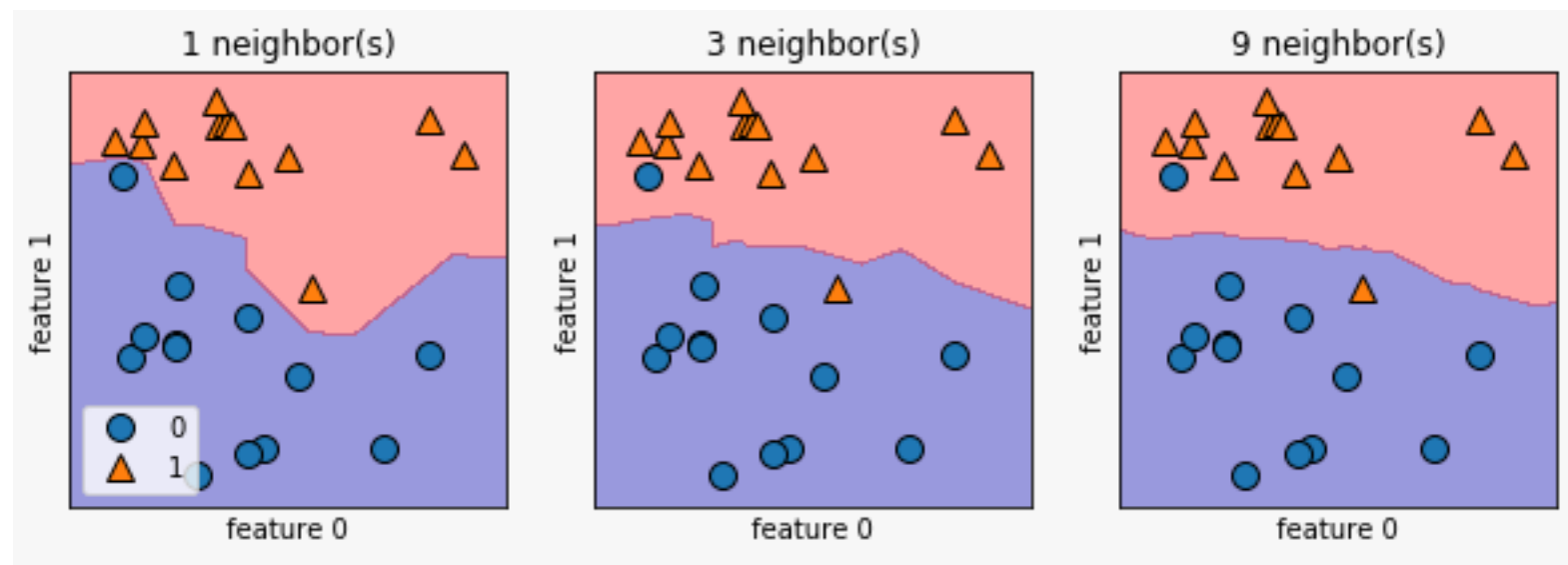
Note: you will need to install mglearn library (a supporting library made developed by A Muller, useful mainly for plotting). Installing it could simply be by:

```
pip install mglearn
```

if in Jupyter notebook you may need to write it as follows:

```
!pip install mglearn
```

Outcome:



- Can you draw a classifier boundary for your breast cancer algorithm in relation to different K?

Finally (extra): Would you be able to code a Knn from scratch (ie, no sklearn help) — please don't copy code from online sources, try it yourself and compare with Jason Brownlee attempt.