



7055CEM Lab Session 2

Basic Machine Learn with Python

In this lab you will carry out Basic Machine Learning using Python below.

If you have any additional questions on this lab email me on ab0487@coventry.ac.uk.

Add your solution on the AULA feed using the tag **#lab2**.

Simple Machine Learning – Predicting Breast Cancer

Task a)



Open Anaconda Jupyter Notebook using the instruction on AULA. [Getting Setup](#). Open python by selecting **New -> Python 3**.

Task b)



Import the libraries that are required to create a simple Machine Learning application. Such imports will allow us to import a data set and manipulate the data.

Run the cell by selecting **Cell -> Run Cells**

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
```

```
In [1]: import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
```



Explore what these imports do.

Task c)



Load the data from anaconda jupyter and get the description of the data.

This tutorial will analyze how data can be used to predict whether a women has breast cancer. By analyzing the breast cancer data, we will implement machine learning to predict breast cancer. Given a list of features (i.e., feature vectors) calculated from a digitized image of the FNA of a breast mass from a patient, the problem is how to diagnose (determine) whether or not the patient has breast cancer. This problem is treated as a 2- class (B-benign, M-malignant) classification problem in this article.

```
cancer=load_breast_cancer()  
print(cancer.DESCR)
```

```
In [3]: cancer = load_breast_cancer()  
print(cancer.DESCR)
```

What does the data look like?

Task d)



The code below creates a (569,31) shaped DataFrame with features and target of the cancer dataset as its attributes. A DataFrame is a structure the makes manipulating data easier.

```
df=pd.DataFrame(cancer.data,columns=[cancer.feature_names])  
df["target"] = cancer.target
```

```
In [9]: df=pd.DataFrame(cancer.data,columns=[cancer.feature_names])  
df["target"] = cancer.target
```

Explore what a dataframe is and what are the benefits of using them?


Task e)



Dividing the dataset into a training set and test set. In this case we put all the rows from the first 30 columns into X and put the last column in y. This splits the DataFrame into X (the data) and y (the labels).

```
X=df.iloc[:,0:30]  
y=df.iloc[:,-1]
```

```
In [14]: x=df.iloc[:,0:30]
         y=df.iloc[:, -1]
```

 We will iloc in the preprocessing section, but see if you understand what we are doing to the data.

Task f)



Using `train_test_split`, split X and y into training and test sets (`x_train`, `x_test`, `y_train`, and `y_test`). This allows us to train the machine learning model and then test the model. The first two parameters are the features and the label values, training size is 426 samples and there are 143 test samples. You will see in the future we can set the size of test set using a decimal number (0.3 being 30%). Which samples are in the test set is selected randomly based in the seed value provided.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X,y,train_size=426,test_size=143,random_state=0)
```

```
In [15]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test=train_test_split(X,y,train_size=426,test_size=143,random_state=0)
```


Task g)



Using `KNeighborsClassifier`, fit a k-nearest neighbours (knn) classifier with `X_train`, `y_train` using one nearest neighbor (`n_neighbors = 1`). Hence, we are using the k-nearest neighbours model to predict breast cancer.

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=1) #loading
model.fit(X_train,y_train) #training
```

```
In [20]: from sklearn.neighbors import KNeighborsClassifier
         model=KNeighborsClassifier(n_neighbors=1) #loading
         model.fit(X_train,y_train) #training
```

 Explore the nearest neighbour model. What does it mean if `n_neighbors=1`?

Task h)



Using your knn classifier, we predict the class labels for the test set `X_test`. We are using the model to predict whether the women in the test samples have cancer or not. We are giving the model the features for the test samples (`X_test`).

```
model.predict(X_test)
```

```
In [21]: model.predict(X_test)
```

Task i)



Create the mean Accuracy score of the knn classifier. So see how many of the predictions are correct on the test data.

```
model.score(X_test, y_test)
```

```
In [22]: model.score(X_test,y_test)
```



This will generate a float between 0 and 1. Explore what this number means.

Simple Machine Learning Digit Recognition

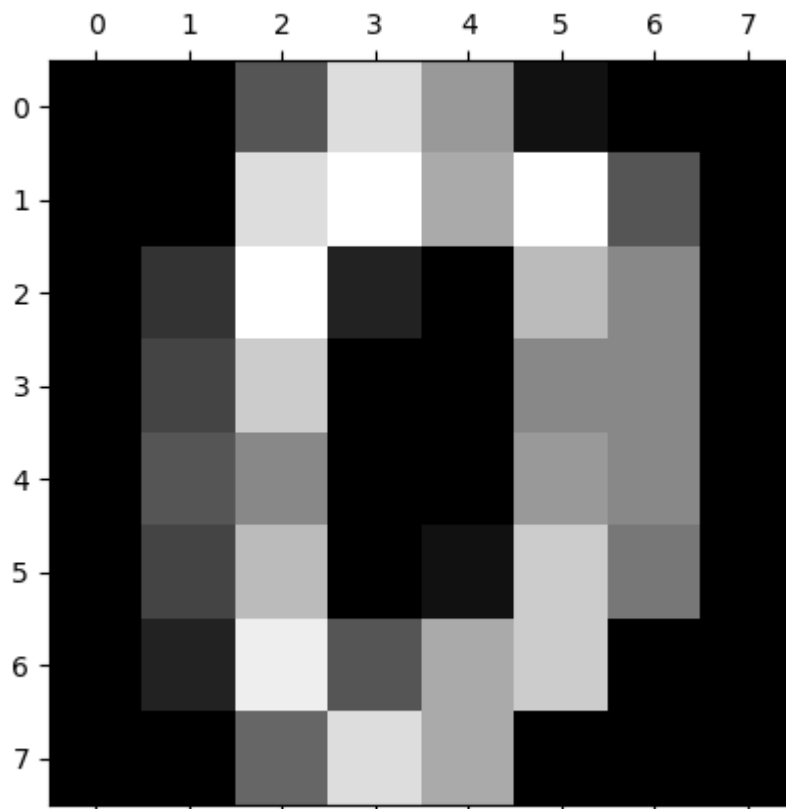
Task a)

*



Open Anaconda Jupyter Notebook using the instruction on AULA. [Getting Setup](#). Open python by selecting **New -> Python 3**.

We are going to create a basic recogniser what is able to recognise handwritten digits. The sklearn digits dataset is made up of 1797 8×8 images. Each image, like the one shown below, is of a hand-written digit. We can load the digits dataset from the **sklearn.datasets** by using the **load_digits()** method. This will save the object containing digits data and the attributes associated with it.



Task b)



Get the samples and put the data into digits. These samples include the features to recognise the digits and the labels to say what the digits are.

```
from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()
```

```
In [1]: from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()
```

Task c)



Put the features into X and put the labels into y. Print the first row of X.

```
X=digits.data
y=digits.target
print(X[0])
```

```
In [2]: X = digits.data
        y = digits.target
        print(X[0])
```

 Show both the data and the labels for the first 5-digit samples.

Task d)



Display the image of the first digit.

```
plt.gray()
plt.matshow(digits.images[0])
plt.show()
```

```
In [4]: plt.gray()
        plt.matshow(digits.images[0])
        plt.show()
```

 How would you show the fifth digit?

Task e)



Split the data for the training and the test data. 80% of the samples are used for training and 20% for testing. Train the nearest neighbour model and test. We use 7 nearest neighbours. Evaluate the model's capability to recognise the digits.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```


```
In [5]: from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42, stratify=y)

        knn = KNeighborsClassifier(n_neighbors=7)

        knn.fit(X_train, y_train)

        print(knn.score(X_test, y_test))
```

 Consider how well nearest neighbour model predicts the digits.

Task f)



Use the decision tree classifier to predict the different digits.

```
from sklearn import tree
DecTree = tree.DecisionTreeClassifier()
DecTree.fit(X_train, y_train)
print(DecTree.score(X_test, y_test))
```

```
In [34]: from sklearn import tree
DecTree = tree.DecisionTreeClassifier()
DecTree.fit(X_train, y_train)
print(DecTree.score(X_test, y_test))
```

Advanced Task - Simple Machine Learning

Task a)



Load the iris dataset from sklearn and create k-nearest neighbour and decision tree models. Evaluate how well they can classify the iris plants.

Please note that you will need to explore how to convert the target values from words to numbers. We will look at this more when looking at preprocessing Lab Session 3 pages 10-15.