Lab session #5

Introduction to Classification

Dr Zied M'NASRI

Introduction

In this Lab session, we tackle some problems of classification, using mainly the Decision Trees and the Naïve Bayes classifier. To do the problem of this lab session, you need to use the following:

1. Python tutorial (see Lab session 1 material)
2. Colab tutorial (see Lab session 2 material)
3. Lecture 5 notes
4. Google's Colab or Jupyter notebook
5. *play_tennis.csv* file (in attachment)

I. Warm-up example: Fisher's Iris flower classifier

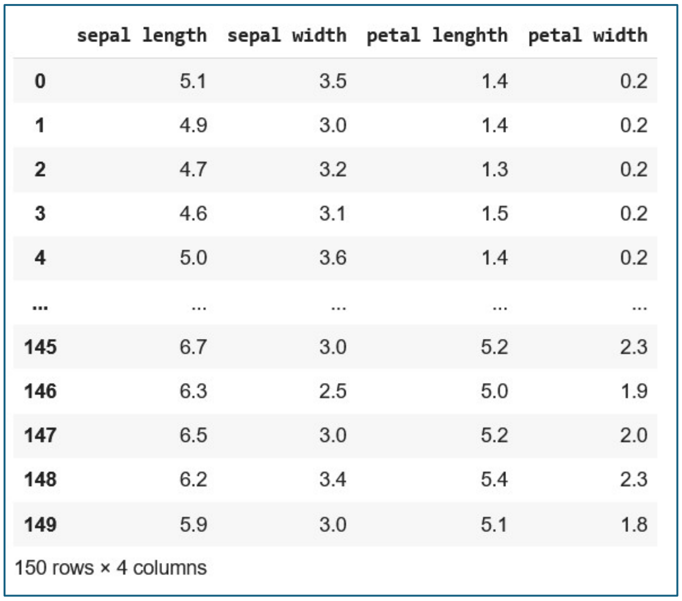
In this example, we implement and visualize a basic classifier of the Fischer's Iris flowers dataset, using Decision Trees.

I.1. Open a session in Google Colab and import the following packages:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

I.2. Import and upload the Iris dataset and visualize the created dataframe created

```
from sklearn.datasets import load_iris

df = load_iris()

pd.DataFrame(df['data'],columns=['sepal length','sepal
width','petal lenghth','petal width'])
```



	sepal length	sepal width	petal lenghth	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

Figure I.1. Samples from the Fischer's Iris dataframe

1.3.Define input features and output targets

```
X = pd.DataFrame(df['data'],columns=['sepal lemgth','sepal width','petal lenghth','petal width'])

y = pd.DataFrame(df['target'],columns=['target'])
```

1.4.Split data into training and test sets

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size = 0.3,random_state=42)
```

1.5.Apply decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier

treeClf=DecisionTreeClassifier()
```

1.6.Train the Decision Tree classifier on the extracted dataset

```
treeClf.fit(X_train,Y_train)
```

1.7. Visualise the trained Decision Tree model

```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(treeClf,filled=True)
plt.show()
```

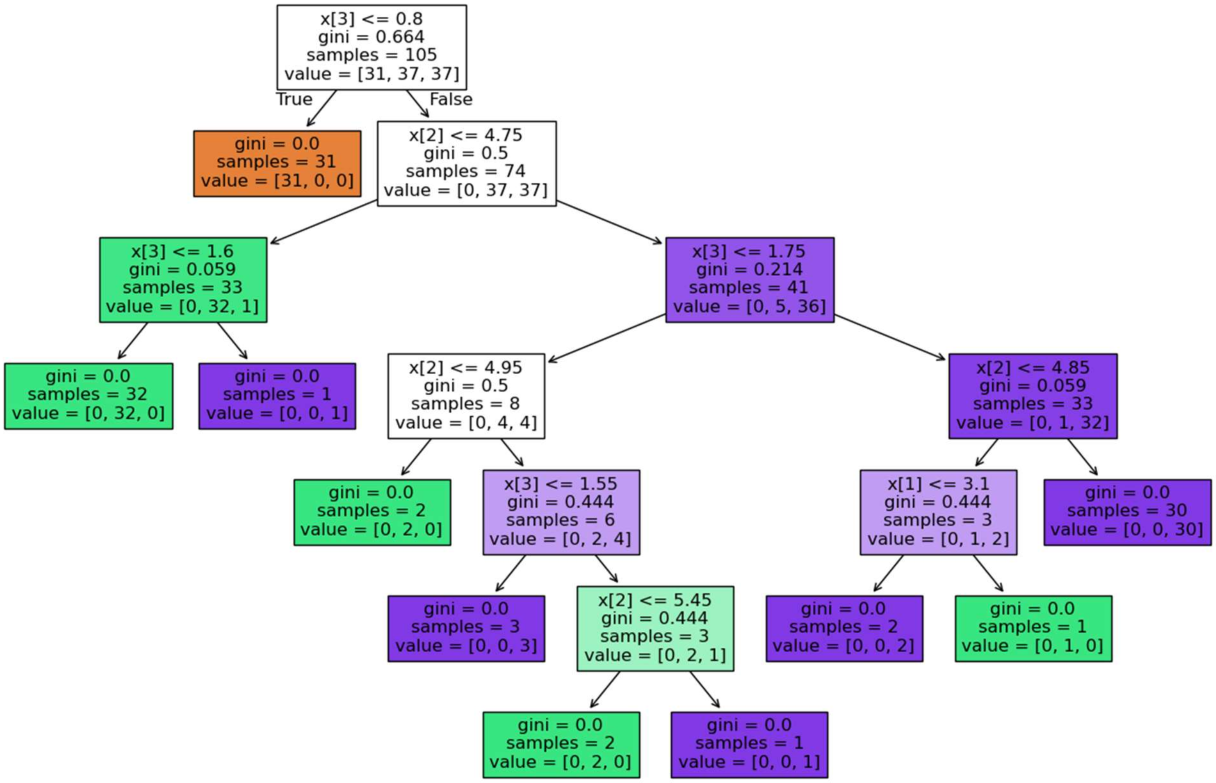


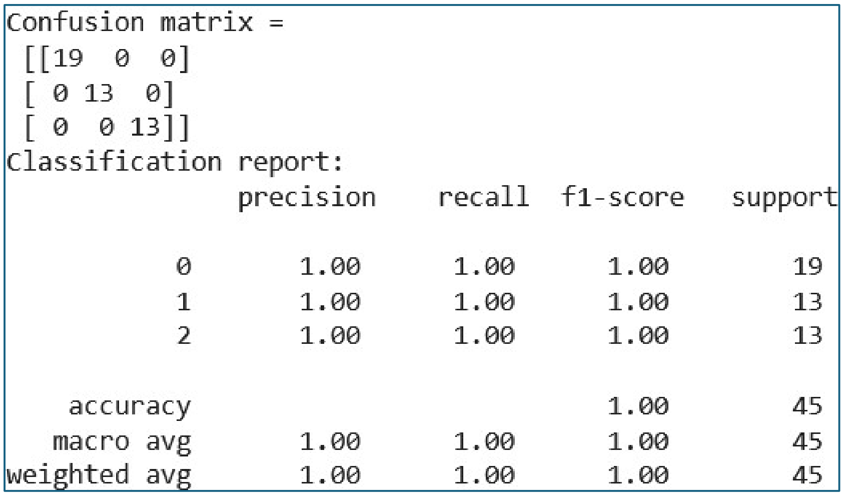
Figure 1.2. Decision tree model trained on Fisher's Iris dataset

1.8. Apply the Decision Tree model on the test set

```
Y_pred = treeClf.predict(X_test)
```

1.9. Compute and visualize the confusion matrix and the evaluation metrics (Accuracy, precision, recall and F1-score)

```
from sklearn.metrics import confusion_matrix, classification_report
print('Confusion matrix = \n', confusion_matrix(Y_test,Y_pred))
print('Classification report: \n', classification_report(Y_test,Y_pred))
```



```
Confusion matrix =
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Figure 1.3. Confusion matrix and classification report of the Decision Tree model trained on Fisher’s Iris dataset

1.10. Make new predictions by entering new input features

```
k = input('Enter the sepal length: ')
l = input('Enter the sepal width: ')
m = input('Enter the petal length: ')
n = input('Enter the petal width: ')
predicteditem = treeClf.predict([[k,l,m,n]])
print(predicteditem)
```

II. Exercises

Exercise #1 Play Tennis example using Decision Trees

In this exercise, we train a decision tree model to make predictions about the example “Play Tennis”, where input features are: {Outlook, Temperature, Humidity, Wind} and the output decision is {Yes, No} (cf. Lecture 5).

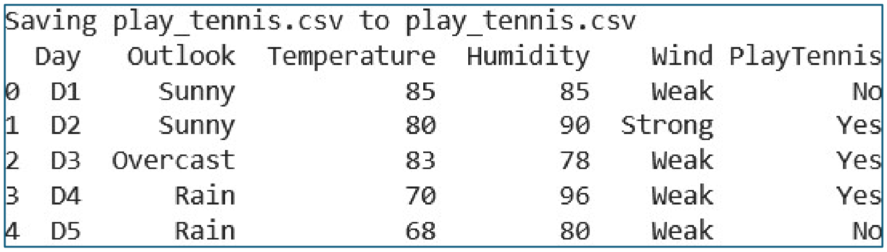
1.1.If you are using Google Colab!

Upload the file *‘Play_Tennis.csv’* to Google Colab using the following script

```
from google.colab import files
uploaded = files.upload()
```

1.2. Read the dataset and print the header using the methods

`pd.read_csv(‘filename’)` and `datasetName.head()`



```
Saving play_tennis.csv to play_tennis.csv
  Day   Outlook  Temperature  Humidity    Wind PlayTennis
0  D1     Sunny           85        85    Weak         No
1  D2     Sunny           80        90  Strong        Yes
2  D3  Overcast           83        78    Weak        Yes
3  D4      Rain           70        96    Weak        Yes
4  D5      Rain           68        80    Weak         No
```

Figure 2.1. Header of the play_tennis dataset

1.3. For the attributes *Outlook* and *Wind*, and the ouput *PlayTennis*, replace the categorical values by numerical ones as follows:

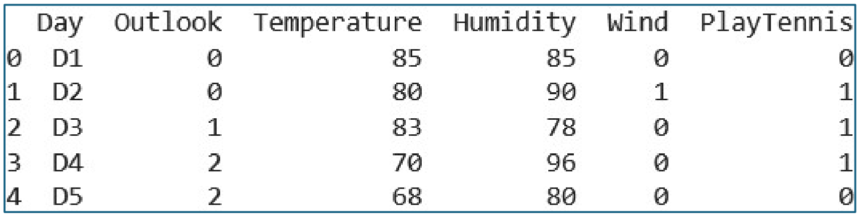
Outlook: {Sunny, Overcast, Rain} → {0,1,2}

Wind: {Weak, Strong} → {0,1}

PlayTennis: {Yes, No} → {1,0}

Hint! Use the method

```
datasetName[‘AttributeName’].replace([‘categoricalValue1’,
‘categoricalValue2’],[NumericalValue1, NumericalValue2],
inplace=True)
```



```
  Day  Outlook  Temperature  Humidity  Wind  PlayTennis
0  D1        0           85        85     0           0
1  D2        0           80        90     1           1
2  D3        1           83        78     0           1
3  D4        2           70        96     0           1
4  D5        2           68        80     0           0
```

Figure 2.2. Dataset header with numerical values

1.4. Split the input features (x) and the output (y) using the method

`datasetName.iloc[:,columnPositions].values`

Hint! In this example, x corresponds to columns positions (1:-1) and y to column (5)

1.5. Split the dataframe into training and test data `(x_train, y_train, x_test, y_test)`, as mentioned in the previous example, using a test/train split rate (0.2) and `random_state = 0`

1.6. Initialize a decision tree classifier, using the method `DecisionTreeClassifier()` with the following attributes:

`(criterion= 'entropy', random_state=0)`

Train the decision tree on `x_train` and `y_train`, using the method `classifierName.fit()`

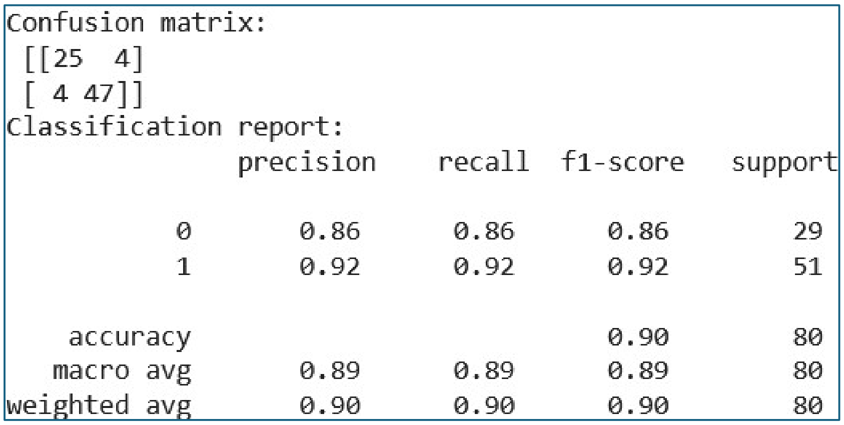
1.7. Plot the trained tree, using the method `tree.plot_tree()`

Hint! To show the split criteria on each note, activate the attribute `fillet=True`

1.8. Predict the test set results using the method `predict()`

1.9. Generate the confusion matrix and the classification report (accuracy, precision, recall and F1 score)

Hint! Import `confusion_matrix()` and `classification_report()` methods from `sklearn.metrics`



```
Confusion matrix:
 [[25  4]
 [ 4 47]]
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        29
           1       0.92      0.92      0.92        51

    accuracy                           0.90        80
   macro avg       0.89      0.89      0.89        80
weighted avg       0.90      0.90      0.90        80
```

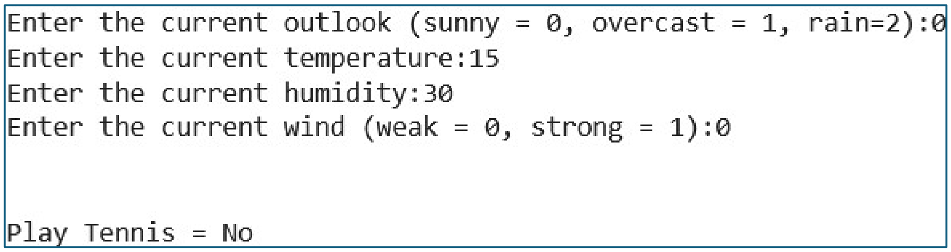
Figure 2.3. Confusion matrix and classification report for the Decision Tree classifier

1.10. Test the DT model by entering new data for each attribute

Hints!

a. You need to reshape the input data into an array of size 1xNumber of attributes

b. You need to convert the decision output into a categorical value (Yes/No)



```
Enter the current outlook (sunny = 0, overcast = 1, rain=2):0
Enter the current temperature:15
Enter the current humidity:30
Enter the current wind (weak = 0, strong = 1):0


Play Tennis = No
```

Figure 2.4. Result of the DT classifier on new user's data

1.11. Re-do the training and steps 2.6-2.11 by changing the values of the training attributes in the method `DecisionTreeClassifier()`, e.g. by setting different values to :

```
criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None,
random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0,
class_weight=None, ccp_alpha=0.0, monotonic_cst=None
```

Hint! Check the meaning an possible values of these attributes in Sickit-learn library:

DecisionTreeClassifier — scikit-learn 1.7.2 documentation

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

2.13. Compare different configurations of the DT to choose the best performing model, in terms of **Accuracy, Precision, Recall** and **F1 score**.

Exercise #2 Play Tennis example using Naïve Bayes Classifier

Re-take Exercise #1 from Step 1.6 and do the following:

2.1. Initialise a Gaussian Naïve Bayes classifier using the method `GaussianNB()`

Hint! Import `GaussianNB()` from sklearn.naive_bayes

2.2. Train the classifier on the training data using the method `classifierName.fit()`

`2.3.` Make predictions on the test set

2.4. Generate the confusion matrix and the classification report. Which classifier is performing better on the test set. Re-do the training and test using a different test/train split (<0.5).

2.4. Make predictions for new input data

Exercise #3 Play Tennis example using Random Forest Classifier

Random Forest Classifier is an extension of the Decision Tree classifier, where many DT models are combined in an ensemble model. Ensemble modelling means that several models are run simultaneously on the data, then the best performing one is picked at each try. Thus, each time a different DT model can be selected from the RF.

Re-take Exercise #1 from Step 1.6 and do the following:

3.1. Initialise a Random Forest (RF) classifier using the method

`RandomForestClassifier()` with `max_depth = 2` and `random_state=0`

Hint! Import `GaussianNB()` from `sklearn.ensemble`

3.2. Train the classifier on the training data using the method `classifierName.fit()`

3.3. Visualize the number of trees in the RF model. Plot the first 5 trees (if there are more trees).

Hint! The number of trees is the length of the attribute `estimator_` in the RF classifier name.

3.4. Make predictions on the test set

3.5. Generate the confusion matrix and the classification report. Compare the performance of RF and DT.

3.6. Make predictions for new input data

3.7. Re-do training and testing using different configurations by assigning different values of the method `RandomForestClassifier()` in the library Sickit-Learn, see [RandomForestClassifier — scikit-learn 1.7.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) or

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

3.8. Try to find the best configurations such that the (Ensemble) Random Forest classifier outperforms the (individual) DT classifier.