Lab Session 3

«Naive Bayes, Linear discriminant analysis (LDA) and Quadratic discriminant analysis (QDA)»

1. Introduction

In this lab session, you will train and test your dataset using methodes namely : Naive Bayes, LDA, QDA. A set of exercises is proposed to test and compare the accuray of different methods.

2. Learning outcome

On successful completion of this lab session you will be able to :

- Handle mglearn datasets.
- Understand and implement Naive bayes
- Build models using LDA
- Build models using QDA

3. Ressources

Libraries Documentation

- Python : https://docs.python.org/3/
- NumPy : https://numpy.org/doc/
- SciPy : https://docs.scipy.org/doc/scipy/
- Matplotlib : https://matplotlib.org/3.5.1/
- Panda : https://pandas.pydata.org/docs/
- mglearn : https://libraries.io/pypi/mglearn
- sklearn : https://scikit-learn.org/stable/

**to proceed with the lab session, refer to this section*

1. Naive Bayes

In this example we will use the following dataset to explain the working principle of Naive Bayes

```
import pandas as pd
# Create dataset
Weather_dataset = {
'weather'  : ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy'],
'temperature'  : ['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild'],
'play'  : ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
}
Weather_dataset = pd.DataFrame(Weather_dataset)

display(Weather_dataset)
```

We need to encode the features [Weather, Temperature], and target by converting string labels into numbers

```
# Import LabelEncoder
from sklearn import preprocessing

#creating labelEncoder
labelEncoder = preprocessing.LabelEncoder()

# Converting string labels into numbers.
encoded_weather = labelEncoder.fit_transform(Weather_dataset.weather)
encoded_temperature = labelEncoder.fit_transform(Weather_dataset.temperature)
target = labelEncoder.fit_transform(Weather_dataset.play)
#print data
encoded_weather, encoded_temperature, target
```

encoded features are then combined

```
features = [list(i) for i in zip(encoded_weather,encoded_temperature)]
```

Next we generate a model using naive bayes classifier

```
#Import Categorical Naive Bayes model
from sklearn.naive_bayes import CategoricalNB

# Create the model
model = CategoricalNB()

# Train the model
model.fit(features, target)
```

Our model is ready to be tested

```
#Predict Ouput (sunny and hot)
predicted= model.predict([[0,1]])
print("Can i play ? ",labelEncoder.inverse_transform(predicted))
```

Exercice 1

In this exercise you will train a Naive Bayes model using the blood_transfursion dataset https://github.com/INRIA/scikit-learn-mooc/blob/main/datasets/blood_transfusion.csv

1. Import the blood_transfusion dataset, and display the donated and not donated samples.
2. train your Naive Bayes model
3. Predict the class for this following vectors (0, 50, 1000, 28) and (4, 10, 4000, 64)

NB : vectors should be encoded according to the dataset encoding

Exercise 2

Marks secured by students in highschool are classified according to different caracteristics (gender, race, etc), take a look at the dataset : https://raw.githubusercontent.com/rashida048/Datasets/master/StudentsPerformance.csv

1. Does parental level of eduction affect the reading score ?
2. Does test preparation course help to score better ?
3. what is the ethnicity of the students ?
4. Using the Naive Bayes method, train your model to predict the different scores (math, reading, writing) of the following students caracteristics :
   - [female, group A, some college, free/reduced, completed]
   - [male, group D, high school, standard, none]
   - [female, group B, associate's degree, standard, none]
   - [male, group C, master's degree, free/reduced, none]

2. LDA

in this example we use the iris dataset to train an LDA model

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

#load iris dataset
iris = load_iris()

#convert dataset to pandas DataFrame
df = pd.DataFrame(data = np.c_[iris['data'], iris['target']], columns = iris['feature_names'] + ['target'])
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
df.columns = ['s_length', 's_width', 'p_length', 'p_width', 'target', 'species']

X = df[['s_length', 's_width', 'p_length', 'p_width']]
y = df['species']
```

```
#define data to plot
X = iris.data
y = iris.target

model = LinearDiscriminantAnalysis()
data_plot = model.fit(X, y).transform(X)
target_names = iris.target_names
```

After LDA tranformation we can plot our new dataset

```
#create LDA plot
plt.figure()
colors = ['red', 'green', 'blue']
lw = 2
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(data_plot[y == i, 0], data_plot[y == i, 1], alpha=.8, color=color, label=target_name)

#add legend to plot
plt.legend(loc='best', shadow=False, scatterpoints=1)

print('*************** LDA Summary ***************')
print('Classes: ', lda.classes_)
print('Priors: ', lda.priors_)
print('Explained variance ratio: ', lda.explained_variance_ratio_)

#display LDA plot
plt.show()
```

Exercise

In this exercise, we will simplify and display a multicritera classification dataset. You will use a record of league of legends ranked games 🕹️, you can explore and visualize the dataset from the following link : https://www.kaggle.com/datasets/gyejr95/league-of-legends-challenger-ranked-games2020

first import challenger dataset from the data explorer

1. Does having first blood affect on winning the game ?
2. what is the average game duration of challenger ranked game ?
3. is your dataset separable based on winning and losing classification ?

Optional

4. Merge the 3 datasets and add column relative to rank
5. is your dataset now seperable ?

3. QDA

the olivetti faces

We need to import the Olivetti faces data-set (https://scikit-learn.org/stable/datasets/real_world.html?highlight=olivetti).

```
from sklearn.datasets import fetch_olivetti_faces

faces, labels = fetch_olivetti_faces(return_X_y=True, shuffle=True, random_state=42)
n_samples, n_features = faces.shape

print("Dataset consists of %d faces" % n_samples)
print("Each face has %d features" % n_features)
```

We can see all labels that exist in the dataset by runing this code :

```
import numpy as np
print(np.unique(labels))
```

Exercise

Now, let's split the data into a training and a testing set, with the test set being made out of 20% of the data of the original dataset.

Now let's implement a LDA on this dataset

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA()
# Fit transform the data
lda.fit(X_train,y_train)
# Print the results
print('*************** LDA Summary ***************')
print('Classes: ', lda.classes_)
print('Priors: ', lda.priors_)
print('Explained variance ratio: ', lda.explained_variance_ratio_)
```

Exercise

1. Print the accuracy of the LDA, the number of faces in total (in the test dataset) and the number of faces with an incorrect prediction.
2. Is the LDA accurate in this situation ?

Now, let's make a QDA !

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

qda = QDA()
# Fit transform the data
qda.fit(X_train,y_train)

# Print the results
print('*************** LDA Summary ***************')
print('Classes: ', lda.classes_)
print('Priors: ', lda.priors_)
print('Explained variance ratio: ', lda.explained_variance_ratio_)
```

Exercise

1. Print the accuracy of the QDA, the number of faces in total (in the test dataset) and the number of faces with an incorrect prediction.
2. Is the QDA accurate in this situation ?

Exercise

1. Using the credit card dataset (shown below), use a QDA to predict fraud (the Class column has a value of 1 for a fraud).
2. Print the accuracy of the QDA, the number of cases in total (in the test dataset) and the number of cases with an incorrect prediction.
3. Is the QDA accurate in this situation ?

Optional

4. Display the results in an interesting way

df = pd.read_csv('https://raw.githubusercontent.com/nsethi31/Kaggle-Data-Credit-Card-Fraud-Detection/master/creditcard.csv')