Project: A Gaussian Classifier for Bank Marketing

Data Science and Machine Learning

INSTRUCTOR: *William Klement*

CLASS: *AML 1113*

- *Individual task*
- *Due date: Sunday December 15, 2024 23:59*
- *Total marks 100. Please submit:*
  - *Python code of your project (instructions are on the last page)*
  - *A word document including your report*

Part 0 [0 marks]:

Download the **Bank Marketing** (https://archive.ics.uci.edu/dataset/222/bank+marketing) dataset from the UC Irvine Machine Learning repository (https://archive.ics.uci.edu/) to use in the questions of this assignment.

Download Instructions

- After extracting the zipped data file, use only bank-full.csv
- In bank-full.csv, replace all semi-colon (;) with a comma (,) so you can load it as a comma-separated file in excel. Basically, change the field separator to comma
- Alternatively, you can read it into Python as a text file by setting field separator to semi-colon (sep = ';')
- The file bank-names.txt contains complete data description for convenience
- Please discard the additional data found in bank-additional.zip and bank.csv

Part I [40 marks in total]:

The task in this project is to construct a Naïve Bayes classifier from the Bank Marketing dataset you downloaded. Step by step, you will train the model, test it and experiment with different proportions of splitting the data to show best performance using accuracy and F1-score metrics.

Write a Python program to do the following:

1. Load the above dataset file into a data frame called df
2. Map each categorical variables into an integer number starting at 0. This applies to the columns: 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'poutcome', and the target column 'y' (see the mapping example below)
3. Remove any rows that have missing (NA) values from df
4. Set X = all feature columns in the data frame excluding the target column y
5. Set y = the target column 'y'
6. In a loop of 10 iterations:
    a. Split the dataset in X and y randomly into 70% for training and 30% for testing using stratified random sampling on the target variable (so that both training and testing data have the same proportions of the target class as the original dataset)
    b. Build a Gaussian classifier by calling: **GaussianNB()**
    c. Fit the model to the training data
    d. Obtain model predictions on test data

To evaluate the classification performance of your model, use the following Python code:

```
# import performance metric libraries
from sklearn.metrics import (
   accuracy_score,
   confusion_matrix,
   ConfusionMatrixDisplay,
   f1_score,
   classification_report,
)
# measure the performance
accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")
print("The accuracy of my Naive Bayes Model is:", accuray)
print("The F1 Score of my Naive Bayes Model is:", f1)
```

The code above assumes that y_pred and y_test contain predicted and original classes respectively.

Useful Hints:

- To map the categorical column 'loan' to integer values, you may use the following Python code fragment:

```
d = {"yes":1,"no":0}              # list integer values for each label in d
df['loan'] = df['loan'].map(d)    # map d on all rows of df['loan']
```

- For illustration, a good example on how to construct a Naïve Bayes classifier, read the following link: https://www.geeksforgeeks.org/naive-bayes-classifiers/
- Do not forget to import the necessary libraries! You will need: **sklearn.naive_bayes**

Part II [20 marks]:

Run your program from Part I repeatedly and report the average and standard deviation of accuracy and F1-scores over 10 trials. In your report, present a table of Accuracy and F1-score values for each trial and summarize with mean and standard deviation at the end.

Hint: make sure you use a different seed value in each trial so that you obtain variable random splits for training and testing.

Part III [20 marks]:

Repeat the experiment of part II but split the data for training and testing using 30% for training and 70% for testing. Again, present your results in a table in your report as described in part II and compare the results of parts II and III in your discussion.

Part IV [20 marks]:

Run your program 8 times, and for each iteration, use an increasing percentage for training data by an additional 10% in each iteration starting at 10% as follows:

iteration 1: use 10% for training and 90% for testing

iteration 2: use 20% for training and 80% for testing

iteration 3: use 30% for training and 70% for testing

……..

iteration 8: use 80% for training and 20% for testing

Plot your results for an increasing size of the training data and discuss any observations you can make based on your findings. Please try to explain any unusual behavior of your model.

Instructions:

- At most 10 marks will be subtracted for incorrect submissions!
- Write your python code all into a single, error free text file with the extension "py" with the file name as described below.
- Submit a word document (with the same naming scheme as below but with the extension .docx) that contains your report, tables, and discussion of your findings.
- Submit the two files named as:
  - StudentID_AML1113-2024F_Project.py
  - StudentID_AML1113-2024F_Project.docx
  - Where StudentID is your student id of c0XXXXXX
- Add the following header as comments at the top of your file containing your python:

```
######################################################
#  Student Name: <enter your name here>
# Student ID: <enter your student ID here>
# Course Code: AML-1213
# Project Due Date: <enter due date>
######################################################
```