Lab4: Data Classification using K-Nearest Neighbour Classifier and Bayes Classifier with Unimodal Gaussian Density

Deadline for submission: 02 October 2021, 10:00 PM

You are given the **Steel Plates Faults Data Set** as a csv file (**`SteelPlateFaults-2class.csv`**). This dataset comes from research by Semeion, Research Center of Sciences of Communication. The original aim of the research was to correctly classify the type of surface defects in stainless steel plates, with six types of possible defects (plus "other"). The Input vector is made up of 27 indicators that approximately describe the geometric shape of the defect and its outline. As Semeion was commissioned by the Centro Sviluppo Materiali (Italy) for this task and therefore the details of the nature of the 27 indicators used as input vectors or the types of the 6 classes of defects are confidential.

The dataset used for this assignment contains features extracted from the steel plates of types A300 and A400 to predict whether the image of the surface of steel plate contains two types of faults such as Z_Scratch and K-Scratch. It consists 1119 tuples each having 27 attributes which are indicators representing the geometric shape of the fault. The last attribute (28^{th} attribute) for every tuple signifies the class label (0 for K_Scratch fault and 1 for Z_Scratch fault). It is a two-class problem. For more information refer [1, 2].

Attribute Information:

The fault description is constituted by 27 indicators representing the geometric shape of the fault and its contour.

1. X_Minimum
2. X_Maximum
3. Y_Minimum
4. Y_Maximum
5. Pixels_Areas
6. X_Perimeter
7. Y_Perimeter
8. Sum_of_Luminosity
9. Minimum_of_Luminosity
10. Maximum_of_Luminosity
11. Length_of_Conveyer
12. TypeOfSteel_A300
13. TypeOfSteel_A400
14. Steel_Plate_Thickness
15. Edges_Index
16. Empty_Index
17. Square_Index
18. Outside_X_Index
19. Edges_X_Index
20. Edges_Y_Index
21. Outside_Global_Index
22. LogOfAreas
23. Log_X_Index
24. Log_Y_Index
25. Orientation_Index
26. Luminosity_Index
27. SigmoidOfAreas

1. Write a python program to split the data of each class from **SteelPlateFaults-2class.csv** into train data and test data. Train data contain 70% of tuples from each of the class and test data contain remaining 30% of tuples from each class. Save the train data as **SteelPlateFaults-train.csv** and save the test data as **SteelPlateFaults-test.csv**

   Note: Use the command **train_test_split** from scikit-learn given below to split the data (keep `random_state=42` to get the same random values for every students).

   ```
   [X_train, X_test, X_label_train, X_label_test] = train_test_split(X, X_label, test_size=0.3, random_state=42, shuffle=True)
   ```

   Classify every test tuple using **K-nearest neighbor (KNN)** method for the different values of *K*=1, 3, and 5. Perform the following analysis:

   a. Find **confusion matrix** (use '`confusion_matrix`' function from `scikit-learn`) for each *K*.

   b. Find the **classification accuracy** (You can use '`accuracy_score`' function) for each *K*. Note the value of *K* for which the accuracy is high.

2. Normalize all the attributes (except class attribute) of **SteelPlateFaults-train.csv** using Min-Max normalization to transform the data in the range [0-1]. Save the file as **SteelPlateFaults-train-Normalised.csv.** Normalize the test dataset using the **minimum and maximum values of train dataset** and save the test data as **SteelPlateFaults-test-normalised.csv**.

   Classify every test tuple using **K-nearest neighbor (KNN)** method for the different values of *K*=1, 3, and 5. Perform the following analysis:

   a. Find confusion matrix for each *K*.

   b. Find the classification accuracy for each *K*. Note the value of *K* for which the accuracy is high.

3. Build a **Bayes classifier** (with unimodal Gaussian density used to model the distribution of the data) on the training data **SteelPlateFaults-train.csv.** Test the performance on **SteelPlateFaults-test.csv** and give confusion matrix and accuracy.

   Note: Compute mean vector and covariance matrix from the training data of each classes separately. Use them to compute likelihood for a class. For computing likelihood use the expression of multivariate Gaussian density. (Do not use Gaussian Naïve Bayes function from sklearn).

4. Tabulate and compare the best result of KNN classifier, best result of KNN classifier on normalised data, and result of Bayes classifier using unimodal Gaussian density.

Instructions:

- **Your python program(s) should be well commented. Comment section at the beginning of the program(s) should include your name, registration number and mobile number.**

- **The python program(s) should be in the file extension .py**
- **Report should be strictly in PDF form. Write the report in word or latex form and then convert to PDF form. Template for the report (in word and latex) is uploaded.**
- **First page of your report must include your name, registration number and mobile number. Use the template of the report given in the assignment.**
- **Upload your program(s) and report in a single zip file. Give the name as <roll_*number*>_Assignment4.zip. Example: *b20001*_Assignment4.zip**
- **Upload the zip file in the link corresponding to your group only.**

In case the program found to be copied from others, both the person who copied and who help for copying will get zero as a penalty.

Reference:

[1] M Buscema, S Terzi, W Tastle, A New Meta-Classifier,in NAFIPS 2010, Toronto (CANADA),26-28 July 2010.

[2] M Buscema, MetaNet: The Theory of Independent Judges, in Substance Use & Misuse, 33(2), 439-461,1998