Lab5: Data classification using K-Nearest Neighbor Classifier

You are given the **Steel Plates Faults Data Set** as a csv file (`SteelPlateFaults-2class.csv`). This dataset contains features extracted from the steel plates of types A300 and A400 to predict whether an image contains two types of faults such as Z_Scratch and K-Scratch. It consists 581 tuples each having 28 attributes. The last 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. Other attributes are input features. For more information refer [1, 2].

1. Write a python program to
    a. **Normalize** all the attributes, except class attribute, of `SteelPlateFaults-2class.csv` using **min-max** normalization to transform the data in the range [0-1]. Save the file as `SteelPlateFaults-2class-Normalised.csv`
    b. **Standardize**, all the attributes, except class attribute, of `SteelPlateFaults-2class.csv` using **z-normalization**. Save the file as `SteelPlateFaults-2class-Standardised.csv`
2. 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-2class-train.csv` and save the test data as `SteelPlateFaults-2class-test.csv`
    a. Classify every test tuple using **K-nearest neighbor (KNN)** method for the different values of K **(1, 3, 5, 7, 9, 11, 13, 15, 17, 21)**. Perform the following analysis :
        i. Find **confusion matrix (**use '*confusion_matrix'***)** for each K.
        ii. Find the **classification accuracy** (You can use '*accuracy_score'*) for each K. Note the value of K for which the accuracy is high.
3. Split the **data of each class** from `SteelPlateFaults-2class-Normalised.csv` into **train data** and **test data**. Train data should contain same **70%** of tuples in Question 2 from each of the class and test data contain remaining same **30%** of tuples from each class. Save the train data as `SteelPlateFaults-2class-train-normalise.csv` and save the test data as `SteelPlateFaults-2class-test-normalise.csv`
    a. Classify every test tuple using **K-nearest neighbor (KNN)** method for the different values of K **(1, 3, 5, 7, 9, 11, 13, 15, 17, 21)**. Perform the following analysis :
        i. Find **confusion matrix (**use '*confusion_matrix'***)** for each K.
        ii. Find the **classification accuracy** (You can use '*accuracy_score'*) for each K. Note the value of K for which the accuracy is high.
4. Split the **data of each class** from `SteelPlateFaults-2class-Standardised.csv` into **train data** and **test data**. Train data should contain same **70%** of tuples in Question 2 from each of the class and test data contain remaining same **30%** of tuples from each class. Save the train data as `SteelPlateFaults-2class-train-standardise.csv` and save the test data as `SteelPlateFaults-2class-test-standardise.csv`
    a. Classify every test tuple using **K-nearest neighbor (KNN)** method for the different values of K **(1, 3, 5, 7, 9, 11, 13, 15, 17, 21)**. Perform the following analysis :
        i. Find **confusion matrix (**use '*confusion_matrix'***)** for each K.

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

5. Plot and the **classification accuracy vs K**. for each cases (original, normalized and standardized) in a same graph and compare & observe how it is behaving.

6. Why the value of K is considered as **odd** integer?

Note:

1. Note that while splitting the data (original, normalized and standardized) into train and test set, use the same seed value for random split of all 3 cases. This is to ensure that same training and test samples will be there in all 3 cases.

   Sample code:

   ```
   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)
   ```

   Keep the value for `random_state` same for all the cases. This will ensure that for all three cases, same samples will be in train and test set.

2. You can import the `KNeighborsClassifier` class from the `sklearn.neighbors` library
3. You can use the functions `StandardScaler` for standardization and `MinMaxScaler` for min-max normalization in scikit-learn.
4. Refer the slide uploaded in moodle about the performance evaluation to know more about confusion matrix and Accuracy.

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