Lab5: Data classification using K-Nearest Neighbor Classifier

You are given the **Pima Indians Diabetes Database** as a csv file (`pima-indians-diabetes.csv`). This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. It consists 768 tuples each having 9 attributes. The last attribute for every tuple signifies the class label (0 for non-diabetes and 1 diabetes). It is a two class problem. Other attributes are input features.

1. Write a python program to
   a. **Normalize** all the attributes, except class attribute, of `pima-indians-diabetes.csv` using **min-max** normalization to transform the data in the range [0-1]. Save the file as `pima-indians-diabetes-Normalised.csv`
   b. **Standardize**, all the attributes, except class attribute, of `pima-indians-diabetes.csv` using **z-normalization**. Save the file as `pima-indians-diabetes-Standardised.csv`
2. Split the **data of each class** from `pima-indians-diabetes.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 `diabetes-train.csv` and save the test data as `diabetes-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 `pima-indians-diabetes-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 `diabetes-train-normalise.csv` and save the test data as `diabetes-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 `pima-indians-diabetes-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 `diabetes-train-standardise.csv` and save the test data as `diabetes-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.
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.