Indian Institute of Technology Dharwad

CS214: Artificial Intelligence Laboratory

Lab 4: Dimensionality Reduction and K-Nearest Neighbor Classifier

You are given the Wisconsin Diagnostic Breast Cancer (WDBC) dataset as a csv file (***"WisconsinDiagnosticBreastCancer.csv"***). The dataset features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/ street/images/. The separating plane (separating plane in the 3-D space of Worst Area, Worst Smoothness and Mean Texture) was obtained using Multisurface Method-Tree (MSM-T) [1], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes. The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in [2].

The dataset details

- Number of instances/records: 569
- Number of attributes: 32 (ID, diagnosis, 30 real-valued input features)
- Attribute information
    1. ID number (Attribute 1)
    2. Diagnosis (M = malignant, B = benign) (Attribute 2)
    3. Ten real-valued features are computed for each cell nucleus: (Attribute 3-32)
        (a) radius (mean of distances from center to points on the perimeter)
        (b) texture (standard deviation of gray-scale values)
        (c) perimeter
        (d) area
        (e) smoothness (local variation in radius lengths)
        (f) compactness ($perimeter^2/area - 1.0$)
        (g) concavity (severity of concave portions of the contour)
        (h) concave points (number of concave portions of the contour)
        (i) symmetry
        (j) fractal dimension ("coastline approximation" - 1)

  The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, Attribute 3 is Mean Radius, Attribute 13 is Radius SE, Attribute 23 is Worst Radius.
- Class distribution: 357 benign, 212 malignant.

Write a python program to complete the following tasks.

Part A: Data Preparation

1. Load the WDBC dataset and display the first five rows.
2. Drop the *'id'* column, replace the *'diagnosis'* column values with binary (M=1, B=0), and check for missing values.
3. Compute and visualize the correlation matrix using a heatmap.
4. Split the dataset into training (60%), validation (20%), and testing (20%) sets for each class label (diagnosis). Save them as ***WDBC_Train.csv***, ***WDBC_Validation.csv***, and ***WDBC_Test.csv***, respectively. You need to load these CSV files later when needed.

Part B: Standardization

1. Load the training data from csv file (***WDBC_Train.csv***).
2. Calculate the mean and standard deviation of each training data attribute except the class attribute (diagnosis).
3. Standardize all the train data attributes except the class attribute. Save the standardized training data as ***WDBC_Scaled_Train.csv***.
4. Use the mean and standard deviation of training data to perform the standardization of validation and test data. Save the standardized validation and test data as ***WDBC_Scale -d_Validation.csv*** and ***WDBC_Scaled_Test.csv***, respectively.

Part C: Principal Component Analysis (PCA)

1. Load the training data from csv file (***WDBC_Train.csv***).
2. Perform the PCA on all attributes except the class attribute. You need to fit and transform the PCA on the original training data. Then, observe the eigenvalues and corresponding eigenvectors.
   (a) Plot the eigenvalues in the descending order of their values.
   (b) Compute the covariance matrix of transformed data and observe its nature and variances.
3. Perform the PCA on all attributes except the class attribute (i.e., "diagnosis"). You need to fit and transform the PCA for two components ($l = 2$) on the original training data. Using the same fit function, transform the original validation and test data for two components ($l = 2$). Finally, save the transformed training, validation, and test data as ***WDBC_PCA2_Train.csv***, ***WDBC_PCA2_Validation.csv***, and ***WDBC_PCA2_ -Test.csv***, respectively.
4. Perform the PCA on all attributes except the class attribute (i.e., "diagnosis"). You need to fit and transform the PCA for two components ($l = 10$) on the original training data. Using the same fit function, transform the original validation and test data for two components ($l = 10$). Finally, save the transformed training, validation, and test data as

WDBC_PCA10_Train.csv, ***WDBC_PCA10_Validation.csv***, and ***WDBC_PC-A10_Test.csv***, respectively.

Note: Do not forget to include the class attribute in the transformed data files.

Part D: K-Nearest Neighbors (KNN) classification on original data

Perform the KNN classification and performance metric evaluation as listed below.

1. Load train, validation, and test data from ***WDBC_Train.csv***, ***WDBC_Validation.csv***, and ***WDBC_Test.csv***, respectively.
2. Implement KNN classification with $K = 1, 7, 11$.
3. Evaluate and compare the performance metrics (listed below) on validation data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score
4. Plot the classification accuracy vs. K for validation data and choose the best K for the KNN classifier.
5. Now, use the KNN classifier of best K to evaluate and compare the performance metrics (listed below) on the test data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score

Part E: KNN classification on standardized data

Perform the KNN classification and performance metric evaluation as listed below.

1. Load train, validation, and test data from ***WDBC_Scaled_Train.csv***, ***WDBC_Scale-d_Validation.csv***, and ***WDBC_Scaled_Test.csv***, respectively.
2. Implement KNN classification with $K = 1, 7, 11$.
3. Evaluate and compare the performance metrics (listed below) on validation data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score
4. Plot the classification accuracy vs. K for validation data and choose the best K for the KNN classifier.
5. Now, use the KNN classifier of best K to evaluate and compare the performance metrics (listed below) on the test data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score

Part F: KNN classification on PCA ($l = 2$) transformed data

Perform the KNN classification and performance metric evaluation as listed below.

1. Load train, validation, and test data from ***WDBC_PCA2_Train.csv***, ***WDBC_PC--A2_Validation.csv***, and ***WDBC_PCA2_Test.csv***, respectively.
2. Implement KNN classification with $K = 1, 7, 11$.
3. Evaluate and compare the performance metrics (listed below) on validation data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score
4. Plot the classification accuracy vs. K for validation data and choose the best K for the KNN classifier.
5. Now, use the KNN classifier of best K to evaluate and compare the performance metrics (listed below) on the test data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score

Part G: KNN classification on PCA ($l = 10$) transformed data

Perform the KNN classification and performance metric evaluation as listed below.

1. Load train, validation, and test data from ***WDBC_PCA10_Train.csv***, ***WDBC_P--CA10_Validation.csv***, and ***WDBC_PCA10_Test.csv***, respectively.
2. Implement KNN classification with $K = 1, 7, 11$.
3. Evaluate and compare the performance metrics (listed below) on validation data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score
4. Plot the classification accuracy vs. K for validation data and choose the best K for the KNN classifier.
5. Now, use the KNN classifier of best K to evaluate and compare the performance metrics (listed below) on the test data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score

Note

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

   1st line separates the train (60%) and 2nd line separate the validation and test data equally from rest 40% data.

   1. *X_train, X_temp, X_label_train, X_label_temp = train_test_split(X, X_label, test_size=0.4, stratify=X_label, random_state=42)*
   2. *X_valid, X_test, X_label_valid, X_label_test = train_test_split(X_temp, X_label_temp, test_size=0.5, stratify=X_label_temp, random_state=42)*

   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 *PCA* from the *sklearn.decomposition* library. Use *explained_variance_ratio_* function to get the variance ratio of each component.

3. You can import the *KNeighborsClassifier* class from the *sklearn.neighbors* library

4. You can use the function *StandardScaler* for standardization in scikit-learn.

5. Refer to the slide uploaded on Moodle about the performance evaluation (the confusion matrix, accuracy, etc.). You can import *confusion_matrix*, *accuracy_score*, *precision_score*, *recall_score*, *f1_score*, and *classification_report* from *sklearn.metrics* library.

Please upload the completed Jupyter Notebook for each problem statement separately to Moodle for evaluation. Save the files using the format:

your_rollnumber_problem_statement1.ipynb.

References

[1] K. P. Bennett, ”Decision Tree Construction Via Linear Programming.” Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992

[2] K. P. Bennett and O. L. Mangasarian: ”Robust Linear Programming Discrimination of Two Linearly Inseparable Sets”, Optimization Methods and Software 1, 1992, 23-34