Homework#2
CS576 Machine Learning, Fall 2023

Due: October 12

Instruction
- Compile your work into a single file, naming it "*YourLastName_FirstName*_CS576_HW2.zip"
- Structure your submission with individual directories for each part, namely Part I, Part II, Part III, etc.

- For each part, you are required to submit:
  (1) **Program Code**: Annotate the relevant lines in your code with the corresponding task numbers for clarity.
  (2) **Execution Results**: Present results for specified tasks, labeling each with its corresponding task number for easy identification.

Part I. Model Evaluation with Nearest Neighbor Classifier

In Part I, you will conduct a machine learning experiment with **k-Nearest Neighbor** classifier and assess various model performance metrics.

0. Initially, download the provided ***toydata.csv*** provided. This toy dataset incorporates two numeric features ('Feature1', 'Feature2') and one binary class feature ('Label'), represented by 0 and 1.

Implement a program which performs the following tasks (a) – (m).

1. [Data Exploration]
- (a) Load the given dataset into your program.
- (b) Visualize the dataset using a **scatter plot**, distinguishing data points by color (e.g., 'blue', 'green') according to their class label.
- (c) Exhibit the first few rows of the dataset
- (d) Present basic statistics of the descriptive features within the dataset.

2. [Data Preparation]
- (e) Apply the **holdout** method to split the data, allocating 2**0%** for testing and the remaining **80%** for training. Ensure consistent splitting every time the code executes, by using, for example, "42" as the random seed value in Python.
- (f) **Standardize** the descriptive features in both the training and test sets, excluding the label feature.

3. [Classification]
- (g) Employ the **k-Nearest Neighbor** (**k=1**) classifier to predict the class label of test instances. Notice that the k-NN classifier does not have a training stage.
- (h) Show all test instances alongside their predicted classes, e.g.,

```
Test Instances and Predicted Classes:
Instance: [-1.53760567 -2.1124287], True Label: 0, Predicted Label: 0
Instance: [0.93499408 1.15684075], True Label: 1, Predicted Label: 1
```

4. [Model Evaluation]

(i) Create a **confusion matrix** for the test data predictions.

(j) Display **TP, TN, FP and FN** values like the following:

```
The True Positive (TP) value is: xx
The True Negative (TN) value is: xx
   …
```

(k) Calculate various model performance metrics such as **Accuracy**, **Misclassification Rate**, **Precision**, **Recall** and **F1 score** utilizing the TP, TN, FP and FN values. Display the performance measure values like the following:

```
Accuracy = (TP + TN) / (TP + TN + FP + FN) = xx.xx%
Misclassification Rate = (FP + FN) / (TP + TN + FP + FN) = xx.xx%
…
```

For performance measure calculations, **do NOT use library functions** such as `precision_recall_fscore_support` and `accuracy_scor` in Python. Instead, compute with confusion matrix values.

5. [Visualization]

(l) Generate a **Voronoi** diagram for the training data

(m) Place the test data points, annotated with their class label, on the Voronoi diagram.

Submit the followings:

(1) A program code that addresses tasks (a) through (m). Please annotate the relevant code lines with their corresponding task numbers for clarity.

(2) The execution results for tasks (b), (c), (d), (h), (i), (j), (k), (l), and (m). Ensure each result is labeled with its respective task number.

Part II. Model Comparison with Random Forests

In Part II, you will conduct a machine learning experiment to compare the performance of **Random Forests** under varying hyper-parameter values.

0. To start, download the provide **adult.data**. This census dataset originates from a Machine Learning Repository. For a detailed description of the data, refer to https://archive.ics.uci.edu/dataset/2/adult. The dataset has a class feature, INCOME, with two categorical values: '<= 50K' and '> 50K'. The task involves classifying whether income exceeds $50K/yr.

Implement a program to accomplish the following tasks (a) – (l).

1. [Data Exploration]

(a) Load the given dataset.

(b) Exhibit the initial few rows of the dataset. Show the count of instances and descriptive features in the original data.

(c) The Adult dataset represents missing values with '?'. Show the count of missing values per each feature.

(d) Eliminate instances containing missing values. Subsequently, display the updated instance count.

(e) Illustrate a histogram representing instance counts per INCOME class.

2. [Data Preparation]

(f) The class feature, INCOME, comprises two categorical values: '<=50K' and '>50K'. Transform this feature into binary 0/1

(g) Implement **One-hot Encoding** for the categorical variables.

(h) Present the first few rows of the processed data. How many descriptive features does the data now include?

(i) Allocate **70%** of the data for training, and the remaining **30%** for testing. While splitting the data, ensure that the distribution of classes in the target feature is consistent in both the training and test sets using **stratified sampling**.

3. [Random Forests and Performance Evaluation]

(j) Construct four **Random Forests** models, each varying by the number of trees in the forest (the hyper-parameter m), each **5, 10, 50 and 500**.

(k) Generate and plot the **ROC curves** of each model collectively.

(l) Explain which random forests classifier(s) exhibit(s) superior performance.

Submit the following:

(1) A program code that addresses tasks (a) through (l).

(2) The execution results for tasks (b), (c), (d), (e), (h), and (k), and answer for (l).

Part III. Hyper- Parameter Tuning with Gradient Boosting

In Part III, you will conduct a hyper-parameter tuning experiment with **Gradient Boosting**.

0. For this experiment, utilize the same adult dataset (**adult.data**) as in Part II.

1. [Data Preparation]

(a) Load the designated dataset.
(b) Exhibit the first few rows of the dataset and show the count of instances and descriptive features in the original data.
(c) Eliminate instances containing missing values.
(d) The class feature, INCOME, has two categorical values: ‘<=50K’ and ‘>50K’. Alter the target feature to binary 0/1, although it’s generally not a requisite for the Gradient Boosting algorithm.
(e) Execute **Label Encoding** for categorical variables.
(f) Illustrate the first few rows of the modified data. How many descriptive features does the data contain? Explain the difference from the prior **one-hot encoding**
(g) Split the data for model training and testing, allocating **30%** for testing and the remaining **70%** for training.

2. [Hyper-parameter Tuning]

In this experiment, we are primarily altering two hyper-parameters: the number of base learners and the learning rate, for the Gradient Boosting classifier. For **the number of individual decision trees** for the base learners, employ **5, 10, and 50**, and for the **learning rate**, select **0.01, 0.05, and 0.1**. Therefore, a total of 9 combinations will be considered to identify the optimum hyper-parameters.

(h) Execute a **grid search** to find the most considerable hyper-parameter values among the provided combinations of values. **During the search**, utilize the prepared training data and a **3-fold cross-validation** schema for training and validation**.** **For testing**, employee the prepared test data, and use **accuracy** as the scoring metric.
(i) For every combination of the stated parameter values, present the **average test score**, **standard deviation of test scores**, and **rank test score** (1, 2, 3..)
(j) Present the performance report of the model with the superior parameter setting, incorporating metrics such as **accuracy**, **precision**, **recall**, **F1-score**, etc.

Submit the following:

(1) A program code which performs for all the tasks (a) through (j).

(2) The execution results for tasks (b), (f), (i) and (j), and answer for (f).

Part IV. Prediction with Linear Regression

In Part IV, you will an experiment with linear regression.

0. Download the provided **auto.csv** file. This dataset will be used to predict the target feature, **mpg**.

1. [Data Exploration]

(a) Load the given dataset.
(b) Display the initial rows of the dataset and the basic information of features.
(c) Convert the **horsepower** feature to numeric, if it is not in numeric type within your program.
(d) <u>Eliminate any features that are non-numeric.</u>
(e) Utilize **plots** to visualize the pairwise relationships of each descriptive feature with the target feature, **mpg**.

2. [Data Preparation]

For this experiment, you will conduct **a simple linear regression** focusing on the **horsepower** feature and the target, **mpg**.

(f) If any missing values are detected in the 'horsepower', impute them with the feature's **mean** value. For validation, display both original missing values alongside their imputed replacements.
(g) Divide the data into training and test subsets using the **holdout** method; allocate **30%** of testing, and the remaining **70%** for training.

3. [Linear Regression Model]

(h) Develop a **regression model** utilizing the training data and assess it using the test data.
(i) Show the **linear regression equation** in the form Y=α+βX.
(j) Illustrate **the regression line** and data points via a scatter plot. <u>If the intercept is not vividly represented, annotate the intercept value on the plot.</u>

3. [Model Evaluation]

(k) Present a **prediction-true plot** (y-y plot) where the x-axis represent the true target value and the y-axis denoted the predicted target value.
(l) Evaluate the model using performance metrics: Mean Absolute Error (**MAE**), Mean Squared Error (**MSE**), Root Mean Squared Error (**RMSE**), and Coefficient of Determination ($\boldsymbol{R}^2$).
(m) Lastly, exhibit a **3D surface plot depicting the error space**, with the x-axis representing the coefficient of 'Horsepower', the y-axis representing the intercept, and the z-axis indicating the mean square error.

Submit the following:

(1) A program code that addresses tasks (a) through (m).

(2) The execution results for tasks (b), (e), (f), (i), (j), (k), (l), and (m).

Part V. Support Vector Machine Learning

In Part V, you will experiment with **Support Vector Machine (SVM)** learning.

0. For this experiment, you will be utilizing the **toydata.csv** used in Part I. This toy dataset contains two numeric features ('Feature1', 'Feature2') and one binary class feature ('Label'), denoted by 0 and 1.

Implement a program to execute the tasks (a) – (f) as mentioned below.

1. [Data Preparation]

(a) Load the dataset

(b) Partition the data using the **holdout** method, allocating 2**0%** of testing, and the remaining **80%** for training.

2. [SVM Models]

(c) Construct four **Support Vector Machine (SVM) models**, each employing different kernel types: **linear, poly, rbf,** and **sigmoid.**

(d) Assess the models using the test data.

3. [Model Evaluation and Visualization]

(e) Calculate the **accuracy** of each SVM model

(f) Per each SVM model, generate a **scatter plot** to visualize the data points in the feature space, coloring them by their true labels. Integrate a contour plot to visualize the **decision boundary** and **margins** of each model.

Submit the following:

(1) A program code that addresses tasks (a) through (f).

(2) The execution results for tasks (e) and (f).