2024 AIML231 Techniques in ML: Assignment 1

This assignment has **100 marks** in total and is due on ***23:59pm, 25th March 2024***. Please submit **your report as a single *.pdf* file** including figures and tables as required, and your **source code as separate files** (a Jupyter notebook as a *.ipynb* file and/or *.py* files). Make sure you read the *Assessment* and *Submission* sections at the end of this assignment description. This assignment contributes **20%** to your overall course grade.

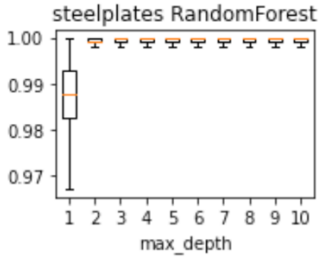
Objectives

This assignment involves trying out a variety of classification algorithms. It requires use of *python*, *numpy*, *matplotlib*, and *scikit-learn*, and serves as an introduction to all those tools. You can run Python based on the **template Jupyter notebook and Python code templates** provided.

1 Classification using the SKLearn library [55 marks]

This part of the assignment is to explore six classifiers supported by scikit-learn and investigate the performance impact of the control hyperparameter of each of these classifiers on three datasets by setting the hyperparameters to a range of plausible values and examining how well the classifiers performs on "held out" (i.e., test) data.

To do this you can use the train-test-split() function from **scikit-learn**. To get better estimates, simply **repeat 50 times with different random splits** (set the seed to get reproducible results). For simplicity, use a **50:50 train:test split** in all cases. For each setting of the hyperparameter, you then have a distribution over 50 different classification accuracies on the test set. A nice way to visualize these scores is to produce a box plot where the x-axis gives options for the hyperparameter of the classifier, while the y-axis indicates the spread for classification accuracies of the classifier. **Titles and Axis Labels** are needed for clarity.



1.1 Machine Learning Models:

You will experimentally study the following classifiers in scikit-learn:

(a) KNeighborsClassifier (K nearest neighbors)
(b) DecisionTreeClassifier (A decision tree (DT))
(c) LogisticRegression (essentially, a perceptron)
(d) RandomForestClassifier (Random Forest)
(e) MLPClassifier (Neural Network)
(f) SVC (SVM classifier)

1.2 Datasets:

You will test the above models on the following three classification datasets (you can easily download the datasets from the .csv files provided with this assignment).

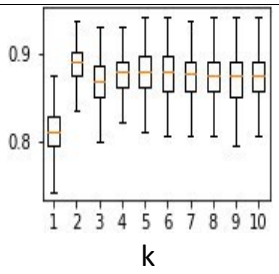
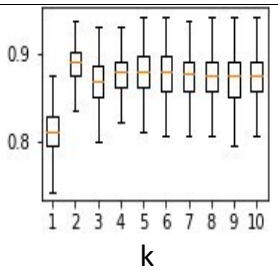
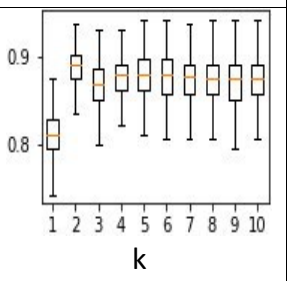
1. steel-plates-fault (csv file provided as part of this assignment)
2. ionosphere (csv file provided as part of this assignment)
3. banknotes (csv file provided as part of this assignment)

Please note that you need to use function **StandardScaler().fit_transform()** provided by the Python module **sklearn.preprocessing** to standardize the value range of each feature in the dataset by removing the mean and scaling to unit variance.

1.3 Tasks:

You should:

(i) Build a 6-by-3 (6 classifiers and 3 datasets) table to present the boxplots on the classifier accuracy versus parameter values (as the example table below). It probably won't fit into one summary figure. If so, you can structure it as separate main plots, one per classifier, each consisting of several subplots for different datasets.

	steel-plates-fault	ionosphere	banknotes
KNN			
DT	...	...	...
...	...	...	...

You need to consider the hyperparameter and the corresponding values for each classifier as summarized in Table 1 below. Other hyperparameters should be left with their default values.

Table 1: Parameters for 6 classifiers

ID	Classifier	Hyper Parameter	Values
1	KNeighborsClassifier	"n_neighbors"	[1,2,3,4,5]
2	DecisionTreeClassifier	"max_depth"	[1,2,3,4,5,6,7,8,9,10]
3	LogisticRegression	"C"	[0.1,0.5,1.0,2.0,5.0]
4	RandomForestClassifier	"max_depth"	[1,2,3,4,5,6,7,8,9,10]
5	MLPClassifier	"alpha"	[1e-5,1e-3,0.1,10.0]
6	SVC	"kernel"	['linear','poly','rbf', 'sigmoid']

(ii) Present *two* summary tables, with rows being classifiers, and columns being datasets.
Table (1) is to contain the lowest mean test errors (not the classification accuracies on the test set; not the classification accuracies on the training set; not the test errors on the training set);
Table (2) is to contain the corresponding hyperparameter values for obtaining the lowest mean test errors obtained in Table (1).

(iii) Complete the template code to generate the ROC curves with respect to the SVC and the RandomForestClassifier on the three classification datasets. For SVC, use the **RBF kernel function**. For RandomForestClassifier, use the **max depth setting of 5**.

(iv) Write a short report to compare and analyze the overall results as captured in the two tables in (ii), including discussions regarding which model has the best performance and why. Discuss how sensitive these classifiers are to the control hyperparameters. In addition, compare the ROC curves with respect to the SVC and the RandomForestClassifier. Discuss which of the two classifiers appear to perform better with respect to each classification dataset, based on the corresponding ROC curves.

2 Implement a simple K-Nearest Neighbor classifier [15 marks]

In this part of the assignment, you will gain hands-on experience with one of the most straightforward and intuitive machine learning algorithms: the k-Nearest Neighbors (k-NN) classifier. You will **implement the k-NN algorithm in Python based on the**

code template (i.e., **knn_template.py**) provided and then test your implementation on a small dataset.

You need to conduct the following tasks for this part of the assignment.

(1) Implement a simple k-NN Classifier:

- Write a function **knn_classifier** that takes in three parameters: training data (**train_data**), training labels (**train_labels**), and test data (**test_data**), and an optional **k** parameter with a **default value of 3**.
- Your function should calculate the Euclidean distance between every point in **test_data** and each point in **train_data**.
- Based on these distances, identify the k nearest neighbors for each test data point and determine the most common class label among them.
- The function should return the predicted class labels for the **test_data**.

(2) Load the classification dataset:

- Write Python code to load the **banknotes** dataset (csv file provided as part of this assignment).
- Use function **StandardScaler().fit_transform()** provided by the Python module **sklearn. preprocessing** to standardize the value range of each feature in the dataset by removing the mean and scaling to unit variance.
- Split the data into a training set (50%) and a testing set (50%) with a fixed random seed, such as 0.

(3) Testing your classifier:

- After implementing the k-NN algorithm, apply it to the **banknotes** dataset.
- Report the accuracy[1] of your classifier on the test set.

(4) Experimentation:

- Experiment with different values of k (e.g., 1, 3, 5, 7) and observe how the test accuracy changes with different k values. Provide a short paragraph to interpret your observations.

3 Implement a basic Decision Tree classifier [30 marks]

In this part of the assignment, you will gain hands-on experience of building a decision tree learning algorithm. You will **implement the decision tree learning**

[1] The percentage of test data instances that have been classified correctly.

algorithm in Python based on the code template (i.e., **decision_tree_template.py**) provided and then test your implementation (test code is provided in the code template).

You need to conduct the following tasks for this part of the assignment:

(1) Complete the implementation of the decision tree learning algorithm:

- Based on the code template provided, complete the implementation of the decision tree learning algorithm.
- The Python code template for the decision tree classifier follows a **binary tree structure**. This means that each internal node has two separate branches, i.e., the left child node and the right child node. At each internal node, a condition is checked to determine whether a categorial feature has a specific value (we assume that all features of the classification problem are categorical features). If the condition is met (i.e., the feature has the specific categorial value), we move to the left child node to continue the classification process. Conversely, if the condition is not met (i.e., the feature has a different value), the classification process proceeds down the right child node.
- At each internal node of the decision tree classifier, the splitting method used is the **information gain metric**. In other words, we choose a condition "feature==value" that maximizes the information gain after splitting an internal node.
- To control the complexity of the decision tree classifier, the code template currently uses the maximum tree depth as a control parameter (the depth of the root node is 0). Whenever a node of the decision tree reaches the maximum tree depth provided by the user, no further splitting will be performed. The corresponding node will be treated as a leaf node. As an enhancement, **add an additional complexity control mechanism based on the minimum partition size**. In other words, whenever the number of training instances associated with a node is less than a threshold provided by the user, the node will be treated as a leaf node.

(2) Test the implemented algorithm:

- Run the test code provided in **decision_tree_template.py** to test the correct functioning of the decision tree learning algorithm implemented.
- You should record the predicted class labels for two test data instances included in **decision_tree_template.py**. You should also present the detailed

structure of the decision tree constructed by your decision tree learning algorithm.

(3) Report:

- Write a short report that includes:
  - A brief explanation of how the decision tree learning algorithm works. Your explanation should be supported by either a flowchart diagram or an algorithm pseudo-code.
  - Present the predicted class labels for the two test data instances included in **decision_tree_template.py**. Explain how your learned decision tree makes its classification decisions with respect to each test data instance.
  - Present a diagram that shows the detailed structure of the decision tree constructed by your decision tree learning algorithm.
  - Identify the feature and its value selected by the root node of the constructed decision tree. Calculate the entropy of the dataset associated with the root node and the information gain achieved by the corresponding split at the root node. Show the details of the calculation, including the mathematical formulas used.

Assessment

Format: You can use any font to write the report, with a minimum of single spacing and 11-point size (handwriting is not permitted unless with approval from the lecturers). Reports are expected to be 2-8 pages that cover all the three parts described above. Reports exceeding this maximum page limit will be penalized.

Communication: A key skill required of a scientist is the ability to communicate effectively. No matter the scientific merit of a report, if it is illegible, grammatically incorrect, mispunctuated, ambiguous, or contains misspellings, it is less effective and marks will be deducted.

Late Penalties: Late submissions for assignments will be managed under the "**Three Late Day Policy**". You will have three automatic extension days, which can be applied to any assignments throughout the course. No formal application is required; instead, any remaining late hours will be automatically deducted when submitting assignments after the due date. You have the flexibility to use only a portion of your late day and retain the remainder for future use. Please note that **these three days are for the whole course, not for each assignment**. The penalty for assignments that are handed in late without prior arrangement (or use of "late days") is one grade reduction per day. Assignments that are more than one week late will not be marked.

Plagiarism: Plagiarism in programming (copying someone else's code) is just as serious as written plagiarism and is treated in the same manner. Make sure you explicitly write down where you got code from (and how much of it) if you use any other resources asides from the course material. Using excessive amounts of others' code (including code generated by AI tools) may result in the loss of marks, but plagiarism could result in zero marks!

Submission

You are required to submit a single *.pdf* report PLUS the python code files (.ipynb and/or .py) through the web submission system from the AIML231 course website *by the due time*. Provide a *README.txt* file if you use any non-standard python libraries.