Assignment: Homework 2

How to Hand It In

1. Put all your solutions in one folder. Compress this folder and name it <firstname>_<lastname>_HW2.zip. For example, "Zhiyao_Duan_HW2.zip".
2. Submit to the corresponding entry on Blackboard.

When to Hand It In

It is due at 11:59 PM on the date specified on the course calendar. **Late assignments will receive a 20% deduction of the full grade each day.**

Problems (10 points in total)

In this assignment, you will practice applying Nearest Neighbor and Decision Tree algorithms on the dataset you collected in Homework 1. We provide some skeleton code in "HW2-skeleton.ipynb" to help you smoothly get started. You may not need to restrict yourself to those skeleton codes since they are mostly just helper functions. In the notebook, we also provide an example dataset for you to do sanity check. If you need help during office hours, we suggest you use this example dataset to show TA the bugs you encountered and results. However, in your actual submission, you should use your own dataset created in Homework 1.

Note: When you created your dataset in HW1, you might not have a good idea about machine learning hence your dataset might not be appropriate for certain machine learning algorithms. Therefore, you may modify your dataset as you are more familiar with machine learning concepts. You will receive feedback about your dataset in HW1 grading comments, but you are also encouraged to discuss with TAs and the instructor if you are not sure whether your dataset is appropriate.

1. (6 points) Nearest Neighbor.
    a. (2 points) Implement a nearest neighbor (NN) function that returns the label of the nearest neighbor of the query example in the training set. If there are two nearest neighbors equally distant from the query example, randomly pick one. Your function needs to allow the following distance metrics: 1) Euclidean (L^2) distance, 2) Manhattan distance (L^1), 3) more generally L^p distance where 0<p<∞.
    b. (1 point) Apply your NN algorithm to the dataset you created in Homework 1. Apparently, you can only use the numerical features. Use those data values as is. Run your NN function in the Leave-One-Out fashion with different distance metrics, i.e., each time taking one data example as the query and the other data examples as the training set; Evaluate your NN function's classification against the ground-truth label of the query example. Report the classification accuracy. Which distance metric (try multiple different p values for L^p) do you think is better?
    c. (1 point) You may realize that different features have different scales, and those with a large scale have a huge influence on the classification result. Normalize your features so that each feature has zero mean and unit standard deviation. (Note: You may do the normalization on the entire dataset instead of only the training subset

for this question.) Run your NN function again with different distance metrics in the Leave-One-Out fashion. Report the classification accuracy and discuss your results.

d. (1 point) Modify your NN function to a k-Nearest Neighbor (kNN) function. The kNN function returns the majority label of the k nearest neighbors of the query example in the training set. If there is a tie between two labels, randomly pick one. Again, this function should support the distance metrics mentioned above.

e. (1 point) Run your kNN function to the normalized version of your dataset with different k in the Leave-One-Out fashion. Try different distance metrics and report accuracy. What value of k is the best for your dataset? Discuss your findings.

2. (4 points) Decision Tree.

   a. (0.5 point) Understand the basics. Read the official document of Decision Trees in scikit-learn. (1.10.1-1.10.6) https://scikit-learn.org/stable/modules/tree.html Answer the questions: 1) Use concise language to state the important advantages and disadvantages for decision tree algorithms in general. 2) Which tree algorithm is implemented in scikit-learn and what is its limitation compared to other tree algorithms?

   b. (1.5 point) Fit, visualize, and predict. Randomly partition the dataset you created in Homework 1 into training, validation, and test subsets with the ratio of 6:2:2. Apply decision tree in scikit-learn to the training split but only use the numerical features. We suggest that you use only a few features that you think are important. This will make your tree easier to visualize. Export the tree in Graphviz format and visualize it in Jupyter. Report the accuracy on the training split, validation split and the test split.

   c. (1 point) Tweak the hyper-parameters. Play with some of the arguments of the `DecisionTreeClassifier`, such as 'criterion', 'max_depth' and 'min_samples_leaf', and monitor the accuracy on the training and validation splits. Report the training and validation accuracies with several sets of both hyper-parameters. State what you find. Choose the hyper-parameters that resulted in the highest validation accuracy and re-learn your tree using both the training and validation data combined. Visualize the tree. Run the tree on the test data. Report the class probability of each test sample. Also report the classification accuracy of the training, validation and test splits.

   d. (1 point) Over the limit. Given the limitation of the algorithm implemented in scikit-learn, how would you apply decision trees to categorical features? Explain the preprocessing step and implement it to process your dataset. Then apply decision trees to all the features. Play with the hyper-parameters and try to get a validation accuracy that you are satisfied with. Then retrain the tree on the training and validation combined dataset using those hyper-parameters, visualize the tree, and report the classification accuracy on the training, validation and test splits.

   Note: It is noted that in c) and d), the hyper-parameters are tuned using the validation split instead of the test set. After hyper-parameter tuning, the validation

split can be regarded as part of training data, and the tree can be re-trained using the training and validation combined dataset to maximize the usage of the data. In this way, the test split is never used for training the tree or tuning its hyper-parameters, thus the accuracy computed on the test set can be regarded as a good estimate of the model's performance on other unseen data samples.