University of Pittsburgh
CS2750 Machine Learning
Professor Milos Hauskrecht

Handout 6
February 24, 2022

Problem assignment 6

Due: March 4, 2022

In this assignment we continue our investigation of the "Pima" classification dataset by developing and testing two new classification models covered in the class: Decision Trees (SVMs) and a non-parametric classification model based on kNN (MLP). As in the previous assignment, you are given the complete Pima dataset (*pima.csv*), as well as, *pima_train.csv* and *pima_test.csv* to be used for model training and testing purposes. You have also received the code learning and testing the vanilla logistic regression model in file *hw6_LogReg.py*. We will use this model as a baseline model. The code also illustrates the calculation of various evaluation statistics you will be asked to replicate with the new models.

Problem 1. Decision Tree Classifier

Part a. Write and submit a program *hw6_DT.py* that will train and evaluate the basic decision tree algorithm. You can build your code using the *hw6_LogReg.py* as a starting point. To implement decision tree please use its scikit-learn implementation based on DecisionTreeClassifier class and its default parameters. Run your decision tree code and report its training and testing stats. Analyze the statistics. Do you think the model is overfitting? Compare its performance to the baseline Logistic regression model. Report any differences you notice in the performance between the two models.

Part b. The decision tree model consists of a sequence of conditions on the input attributes organized in the tree structures. The complexity of the tree is defined by the number of nodes in the tree. You can extract the number of nodes in the tree after it is trained using the *tree_.node_count* attribute. What is the number of nodes in the tree from Part a? Please note that the new version of the scikit learn library allows you to visualize the tree structure, the conditions used in different nodes, impurity of the models, etc using *tree.plot_tree* function.

Part c. The decision tree algorithm implemented in the scikit learn library comes with multiple parameters. These include the choice of the impurity measure, the limit on the depth of the tree, the minimum number of training instances covered by the leaf nodes, etc. Please experiment with the different parameter values for (a) the limit on the depth of the tree, (b) the minimum number of training instances covered by leaf nodes, and see if you can improve the performance of the default model (Part a.) on the testing data. Please

report the results for at least two different parameter settings that are different from the defaults in Part a. In addition to training and testing stats please include the number of nodes for the respective trees.

Problem 2. kNN classifier

Part a. Write and submit a program *hw6_kNN.py* that will train and evaluate the kNN classifier algorithm. Once again, please start building your code using the *hw6_LogReg.py*. To implement the kNN classifier please use its scikit-learn implementation based on KNeighborsClassifier class and its default parameters (number of neighbors=5, Euclidean distance). Run your code and report its training and testing stats. Analyze the statistics and possible overfitting? Compare its performance to the baseline Logistic regression model and the Decision tree model from Problem 1. Report any differences you notice in the performance with respect to these two models.

Part b. The kNN algorithm implemented in the scikit learn library lets one adjust multiple parameters. These include the choice of the distance metric, the number of neighbors, etc. Please experiment with the different number of neighbors to see if you can improve the performance of the default model (Part a.) on the testing data. Please report the results for at least two other different parameter settings that are different from the defaults in Part a. Report and analyze your results and any conclusions you can make about the improved performance of the model.