MACHINE LEARNING #2 (AI.A) (EXTERNAL RESOURCE)

(50.0 points possible)

Your email address will be used to identify your submission entry.

Go To Question #2

III. Classification

In this problem you will use the support vector classifiers in the **sklearn** package to learn a classification model for a chessboard-like dataset. In your starter code, you will find `input3.csv`, containing a series of data points. Open the dataset in python. Make a scatter plot of the dataset showing the two classes with two different patterns.

Use SVM with different kernels to build a classifier. Make sure you split your data into **training** (60%) and **testing** (40%). Also make sure you use **stratified sampling** (i.e. same ratio of positive to negative in both the training and testing datasets). Use **cross validation** (with the number of folds $k = 5$) instead of a validation set. Train-test splitting and cross validation are both functionalities that are readily available in sklearn.

- **SVM with Linear Kernel**. Observe the performance of the SVM with linear kernel. Search for a good setting of parameters to obtain high classification accuracy. Specifically, try values of C = [0.1, 0.5, 1, 5, 10, 50, 100]. Read about **sklearn.grid_search** and how this can help you accomplish this task. After locating the optimal parameter value by using the training data, record the corresponding **best score** (accuracy) achieved. Then apply the testing data to the model, and record the actual **test score**. Both scores will be a number between zero and one.

- **SVM with Polynomial Kernel**. (Similar to above).
  Try values of C = [0.1, 1, 3], degree = [4, 5, 6], and gamma = [0.1, 1].

- **SVM with RBF Kernel**. (Similar to above).
  Try values of C = [0.1, 0.5, 1, 5, 10, 50, 100] and gamma = [0.1, 0.5, 1, 3, 6, 10].

- **Logistic Regression**. (Similar to above).
  Try values of C = [0.1, 0.5, 1, 5, 10, 50, 100].

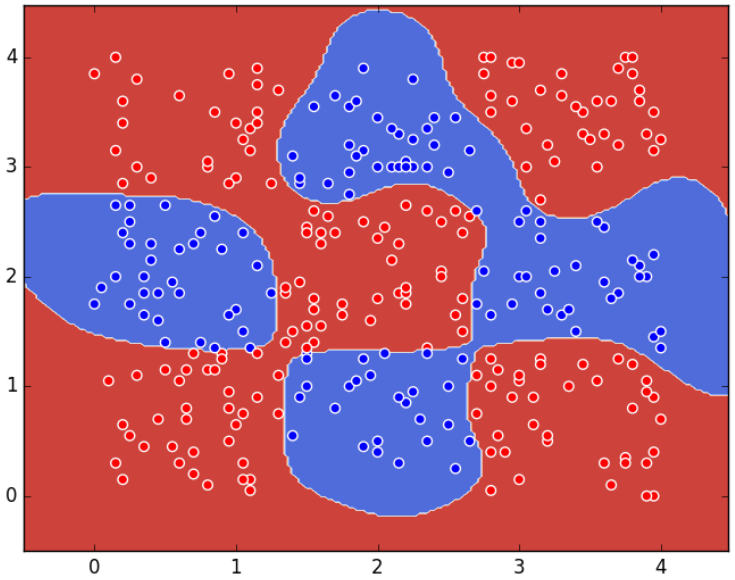
- **k-Nearest Neighbors**. (Similar to above).
  Try values of n_neighbors = [1, 2, 3, ..., 50] and leaf_size = [5, 10, 15, ..., 60].

- **Decision Trees**. (Similar to above).
  Try values of max_depth = [1, 2, 3, ..., 50] and min_samples_split = [1, 2, 3, ...,

10].

- **Random Forest**. (Similar to above).
  Try values of max_depth = [1, 2, 3, ..., 50] and min_samples_split = [1, 2, 3, ..., 10].

What To Submit. `output3.csv` (see **example**). No need to submit your actual program. The file should contain an entry for each of the seven methods used. For each method, print a comma-separated list as shown in the example, including the **method name**, **best score**, and **test score**, expressed with as many decimal places as you please. There may be more than one way to implement a certain method, and we will allow for small variations in output you may encounter depending on the specific functions you decide to use.



Optional. To visualize the result of each case of classification method, you can use **matplotlib** to output an image containing the data points and boundaries of each method. An example output showing the result of SVM with RBF kernel is shown above for reference.

MACHINE LEARNING #3 (AI.A) (EXTERNAL RESOURCE)

(100.0 points possible)

Your email address will be used to identify your submission entry.

Go To Question #3