CSCE 478/878 Recitation 3 Handout
Binary Classification using K Nearest Neighbor
January 22, 2019

Introduction

For this recitation we will solve a binary classification problem by using the k-NN model. As with last week, we will be working in and submitting Jupyter notebooks.

A couple general instructions before we start:

- Your Jupyter notebook should be submitted via webhandin by 4:45PM, Tuesday January 22th.
    - Use the same naming convention as last week: `<lastname>_<firstname>_2.ipynb`
- Use markdown cells with **<u>bold underlined</u>** and a font size of 6 titles to denote the start of each problem, e.g.

Jupyter Notebook Pointers

Here are a couple tips to (maybe) make your life easier:

- You can run the currently selected cell with `ctrl + enter`
    - `shift + enter` will both run the currently cell and make a new one
- Tab completion is available in many circumstances
- If you want to clear memory and start over, use the 'Kernel' dropdown menu at the top and select 'Restart'
    - 'Restart and Run All' is a great way to make sure your code works when run from scratch
- You can change the cell type between code and markdown using the 'Cell' dropdown menu
- This article contains several other useful shortcuts and tips
- This article contains info on some of the most common "magic commands" In Jupyter notebooks
    - %system pwd is a nice way of reminding yourself where your notebook is running
    - %matplotlib inline is necessary for displaying your matplotlib pictures in your notebook

---

Part 1: Data Processing (40 pts)

1. Get the dataset ('winequality-white.csv') from the following URL:
   https://archive.ics.uci.edu/ml/datasets/wine+quality
   Note: For your 1st programming assignment you will use a similar dataset.

   The dataset consists of characteristics of white wine (e.g., alcohol content, density, amount of citric acid, pH, etc) with target variable "quality" representing rating of wine.

   11 Input variables (based on physicochemical tests):
   - fixed acidity
   - volatile acidity
   - citric acid
   - residual sugar
   - chlorides
   - free sulfur dioxide
   - total sulfur dioxide
   - density
   - pH
   - sulphates
   - alcohol

   Output variable (based on sensory data):
   - quality (score between 0 and 10)

2. Read in the 'winequality-white.csv' file as a pandas data frame.
3. The target will be the 'quality' column which represents rating of wine and ranges from 3 to 8. You will need to convert it into a two-category variable consisting of "good" (quality > 5) & "bad" (quality <= 5). Your target vector should have 0s (representing "bad" quality wine) and 1s (representing "good" quality wine).
4. Use the techniques from the first recitation to summarize each of the variables in the dataset in terms of mean, standard deviation, and quartiles.
5. (Optional:) You may apply the data pre-processing steps of Exploratory Data Analysis (EDA) that you learned in recitation 1.
6. Generate pair plots using the seaborn package (see first recitation notebook). You need to identify and report the redundant features.
7. Drop the redundant features and implement the steps up to 13 (you need to perform grid search in step 10 just once). See whether dropping features improves the performance measures (e.g., F1 score). If not then keep the features. You need to understand that having highly correlated features will not contribute towards meaningful distance calculation. Moreover, it will increase the time-complexity. You will have to determine this experimentally.
8. Partition the dataset into train and test set (80%-20%). It should return two feature matrices X_train, X_test; and two target vectors y_train & y_test.

9. Standardize each feature of your training & test set. You may use one of the following techniques.

   Technique 1:
   ```
   from sklearn.preprocessing import StandardScaler
   scaler = StandardScaler().fit(X)
   X = scaler.transform(X)
   ```

   Technique 2:
   ```
   from sklearn.preprocessing import scale
   X = scale(X)
   ```

Part 2: Model Evaluation (60 pts)

10. **Model Selection**: Using Scikit-Learn's GridSearchCV select the best model. For the "param_grid" use the following attributes.
    ```
       'n_neighbors': [1, 3, 5, 7, 9, 15, 21, 23, 33, 35, 37]
       'p': [1, 2, 10, 100]
       'weights': ["uniform", "distance"]}
    ```

    The "scoring" attribute should be 'f1'.
    Use 10-fold cross-validation: cv = 10

11. Train Scikit-Learn's KNeighborsClassifier model using the optimal hyperparameter values obtained from GridSearchCV.
12. Evaluate your model on the train data using the cross_val_score function and report the average accuracy.
13. Report the train data performance measures: precision, recall, F1 score and confusion matrix.
14. Generate the ROC curve for the train data.
15. Report the area under the curve (AUC) score for the ROC curve.
16. Generate the precision-recall curve for train data. Report the optimal threshold.
17. Evaluate your model on the test data and report the performance measures.
    a. Precision
    b. Recall
    c. F1 score
    d. Confusion matrix
    e. Accuracy