Project 3: Breast Cancer

CS 5473: Data Mining Fall 2022

Instructor: Dr. Mohammad Imran Chowdhury

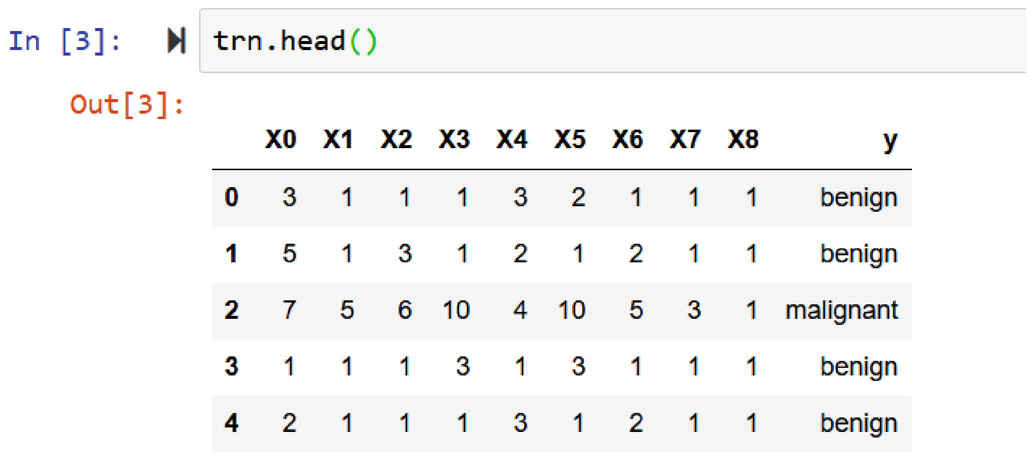
Total Points: 55

Due: 10/20/2022 11:59 PM

In this project, I invite you to do the following:

1. Import the Breast Cancer training and testing datasets.
2. Apply the kNN model to the training data.
3. Train and optimize the kNN model.
4. Plot the accuracy of the parameters.
5. Graph the confusion matrix.
6. Calculate the overall accuracy of the model on the testing data.

Task 1: Load the Breast Cancer training and testing datasets provided to you as **'data/BreastCancer_trn.csv'** file and **'data/BreastCancer_tst.csv'** into the Jupyter Notebook. After loading the dataset, if you look at the first few rows of the training dataset, then the output should be as follows: **(5 points)**



```
In [3]:  trn.head()
Out[3]:
```

	X0	X1	X2	X3	X4	X5	X6	X7	X8	y
0	3	1	1	1	3	2	1	1	1	benign
1	5	1	3	1	2	1	2	1	1	benign
2	7	5	6	10	4	10	5	3	1	malignant
3	1	1	1	3	1	3	1	1	1	benign
4	2	1	1	1	3	1	2	1	1	benign

Task 2: Apply the kNN model to the training data. **(5 points)**

To train a kNN model, set up a **KNeighborsClassifier** object and **fit it to training data**. You can assume **n_neighbors=5**. Note that for the train data set, you need to separate the attributes X0-X8 into X_trn and the class variable into y_trn. Same goes for the test data set. That is for the test dataset, you need to separate the attributes X0-X8 into X_tst and the class variable into y_tst.

Task 3: Train and optimize the kNN model. **(15 points)**

The challenge in training a kNN model is to determine the optimal number of neighbors. To find the optimal parameters, **GridSearchCV** object can be used. The output should be as follows:

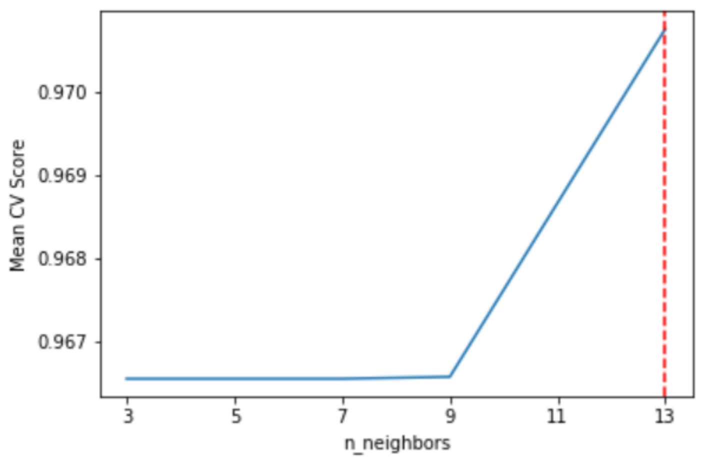
```
Out[6]: {'algorithm': 'auto',
          'leaf_size': 30,
          'metric': 'minkowski',
          'metric_params': None,
          'n_jobs': None,
          'n_neighbors': 13,
          'p': 2,
          'weights': 'uniform'}
```

Task 4: Plot the accuracy of the parameters. **(15 points)**

Once the optimal parameters are found, the accuracy for different parameters can be compared by plotting. The grid variable has an attribute **cv_results_**, which is a dictionary of key value pairs and stores the cross validation accuracy for each parameter.

The output should be close to as follows:

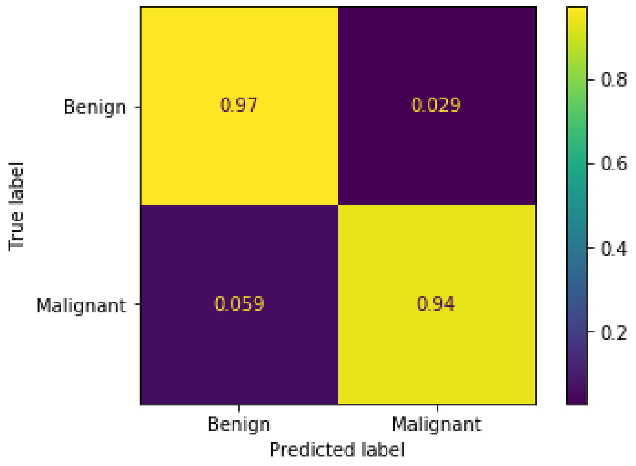
```
Out[7]: <matplotlib.lines.Line2D at 0x29a41635390>
```



Task 5: Graph the confusion matrix. **(10 points)**

In this task, you'll evaluate the accuracy of the trained kNN model on the test set. A good evaluation measure is the **confusion matrix** that gives the fraction of true positives, true negatives, false positives, and false negatives.

After plotting the data, the output should be as follows: **(10 points)**



Tasks 6: Calculate the overall accuracy of the model on both the training and testing datasets. **(5 points)**

The output should be close to as follows for the training dataset:

```
Accuracy on training data: 97.91%
```

And, the output should be close to as follows for the test dataset:

```
Accuracy on testing data: 96.10%
```

The submission grading rubric is as follows (points out of 55 total):

Project element	**Points**
Task 1	5
Task 2	5
Task 3	15
Task 4	15
Task 5	10
Task 6	5

Submission Instructions: Create a compressed file (.zip or .tar.gz files are accepted) with your all source files such as .ipynb files and data files. Generally speaking to complete Task1 through Task6, you just need one .ipynb file. But it's better to submit everything as a compressed file. Submit the compressed file to Blackboard.

Late submission policy: As described in the syllabus, any late submission will the penalized with 10% off after each 24 hours late. For example, an assignment worth 100 points turned in 2 days late will receive a 20 point penalty. Assignments turned in 5 or more days after the due date will receive a grade of 0.