CPS 844 Lab 5

03/09/2022
Section 1
Ryan Soliven
500840952

#2

Read the dataset located here 'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data'

Code

```
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)
```

#3

(10 points) ‘Recycle’ the code from lab2 to pre-process the data: assign new headers to the DataFrame, drop the 'Sample code number' attribute, convert the '?' to NaN, discard the data points that contain missing values, convert the values for the 'Bare Nuclei' attribute to numerical values, drop row duplicates.

Code

```
data.columns = ['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',
                'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
                'Normal Nucleoli', 'Mitoses','Class']
data = data.drop(['Sample code number'],axis=1)
data = data.replace('?',np.NaN)
data = data.dropna()
data = data.drop_duplicates()
```

#4

(15 points) Continue with the preprocessing: separate the features from the target class, standardize the features. Modify the target values: from the description, the malignant class labels are indicated with the value 4. The ‘benign’ labels are indicated with the value ‘2’. Replace (or ‘map’) the values such as the label ‘4’ becomes the integer ‘1’, and the label ‘2’ becomes the integer ‘0’.

Code

```
classData = data['Class']
attributeData = data.drop(['Class'], axis=1)
attributeData = preprocessing.scale(attributeData)
classData = classData.replace(4, 1)
classData = classData.replace(2, 0)
```

#5

(5 points) Use the sklearn library to construct a NearestNeighbors classifier. Keep the default value for the number of neighbors (which is 5).

Code

```
dataTrain, dataTest, classTrain, classTest = train_test_split(attributeData, classData,
test_size = 0.4, random_state=1)
clf = KNeighborsClassifier(n_neighbors = 5)
clf.fit(dataTrain, classTrain)
predC = clf.predict(dataTest)
```

#6

(40 points) Compute and print out the averages of the accuracies, f1-scores, precisions and recall measurements of the nearest neighbor classifier, using 10-fold cross validation.

Code

```
scores = cross_val_score(clf, attributeData, classData, cv=10)
print("Average of accuracy: ", scores.mean())
scores = cross_val_score(clf, attributeData, classData, cv=10, scoring='f1')
print("Average of fl-scores: ", scores.mean())
scores = cross_val_score(clf, attributeData, classData, cv=10, scoring='precision')
print("Average of precisions: ", scores.mean())
scores = cross_val_score(clf, attributeData, classData, cv=10, scoring='recall')
print("Average of recall measurements: ", scores.mean())
```

Results

```
Average of accuracy:  0.9488383838383838
Average of fl-scores:  0.9512263469391129
Average of precisions:  0.9561375218983915
Average of recall measurements:  0.948731884057971
```

#7

(30 points) The goal here is to create and display one confusion matrix. For that, you can create a training and test set from your pre-processed data, train the nearest neighbor classifier on the training set, and predict the labels of the test set using the trained classifier. Then summarize the prediction results using a confusion matrix. As a reminder, the confusion matrix will summarize the number of correct and incorrect predictions, broken down by each class.

Code

```
fig=plot_confusion_matrix(clf, dataTest, classTest,display_labels=["Benign","Malignant"])
fig.figure_.suptitle("Confusion Matrix for Dataset")
plt.show()
```

Results