UNIVERSITY OF IOANNINA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Assignment 1 - Classification

Terizi Chrysoula, Student ID: 430

May 1, 2020

Dataset

The given dataset consists of 1000 records in total. For each entry, 17 features and its label are recorded. There are two categories of data, the first one is called "1" and the second one "2". All the features take real values, positive and negative. The first two attributes range from -2.5 to +2.5, and the rest of the attributes range from -5 to +5.

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges. In our case, the features have different ranges, and therefore, the new scale ranges from 0 to +1 for all the features (insights into Appendix B).

Classifiers

Four classifiers are considered in this assignment, and these are the Nearest Neighbor classifier (KNN), Naïve Bayes, Support Vector Machines (SVM) using linear and RBF kernel functions and the Decision Trees. A brief description of the construction of the classifiers follows below. The initial step of all methods is to load the data.

- **K-Nearest Neighbor**: The algorithm assumes that similar things exist in close proximity, that is, the similar things are near to each other. The user should initialize the K number of neighbors. For each example in the data, the distance between the test example and the current example from the data is calculated. After that, the distances are sorted from smallest to largest and the K entries and their labesl are picked simultaneously from this collection. The label of the test example is the mode of the K labels.
- **Naïve Bayes**: This classifier is based on the Bayes theorem, $P(A,B) = \frac{P(B|A)P(A)}{P(B)}$, where B is the evidence and A is the hypothesis. The assumption of this method is that the features are independent, this means that the presence of one particular feature does not affect the other. Firstly, the data are converting into a frequency table. After that, a likelihood table is created by finding the probabilities for each distinct element. Finally, we calculate the posterior probability for each class using Naive Bayesian equation. The class with the highest posterior probability is the outcome of prediction. The type of the Naive Bayes classifier that we use is the **Gaussian**, which assumes that the features follow a normal distribution.
- **Support Vector Machines (SVM)**: There are many possible hyperplanes (decision boundaries) that could be chosen to separate the two classes of data points. This method try to find a plane that has the maximum margin. The dimension of the hyperplane depends upon the number of features. SVM algorithms use a set of mathematical functions that

are defined as the kernel function. The kernel function take as input the data points and transform them into the required form. **Linear kernel SVM** is used if the data is linearly separable. That is that a single line can seperate the data points. **RBF kernel SVM** has two parameters, C and $Gamma$. C is the penalty for misclassifying a data point. If C is large, the classifier is heavily penalized for misclassified data and therefore bends over backwards avoid any misclassified data points. $Gamma$ can be thought of as the spread of the kernel and therefore the decision region. If $Gamma$ is low, the curve of the decision boundary is very low and thus the decision region is very broad. If $Gamma$ is high, the curve of the decision boundary is high, which creates islands of decision boundaries around data points.

- **Decision Trees**: This classifier performs better due to their high adaptability and computationally effective features. It consists of nodes (test for the value of a certain attribute), edges (correspond to the outcome of a test and connect to the next node or leaf) and leaf nodes (terminal nodes that predict the class labels). The main idea of this method is that the training set is broken down into smaller and smaller subsets while an associated decision tree get incrementally developed. It starts at the tree root and split the data on the feature that results in the largest information gain.

Results

The verification of the classifiers' quality is one of the most important stages of the classification problem. There are several metrics that measure the successful prediction of a classifier. In this project, we use the accuracy metric. Accuracy calculates how closely the measured value of a quantity corresponds to its true value. Furthermore, we use 10-fold cross validation method (insights into Appendix A). The results for each one of the four selected classifiers are presented below.

K-Nearest Neighbor

The Table 0.1 shows the accuracy score for each one of the tested values of K parameter. A general observation of these simulations is that the accuracy is low. The values range between $[0.546, 0.602]$. Nevertheless, the case $k = 9$ succeeds the best rate equals to 0.602.

K neighbors	**Accuracy**
1	0.576
2	0.546
3	0.558
4	0.550
5	0.559
6	0.563
7	0.575
8	0.564
9	**0.602**

Table 0.1: Shows the accuracy score of the K-Nearest Neighbor classifier while the number of neighbors k ranges between $k \in [1, 9]$.

Naïve Bayes

The next tested classifier is the Naïve Bayes classifier. This classifier does not require any initialization of hyperparameters. The experiments of the classifier show that the success rate is not high. The rate is close to 0.50.

Support Vector Machines - Linear kernel

The result of the SVM classifier using linear kernel function follows. The accuracy of this SVM linear classifier is equal to 0.512 and tends to be equal to the classification accuracy of the random classifier which is 0.50. This performance is not satisfactory, however, this performance was expected because of the number of features which is not two. Therefore, a simple line would not be able to predict the category of our data which have been trained using 17 features.

Support Vector Machines - RBF kernel

The results of the SVM classifier using radial kernel function listed below. This method was tested with various values of C and $Gamma$ parameters. C parameter ranges between $[10^3, 10^6]$ by step $10k$ and $Gamma$ parameter ranges between $[10^{-5}, 10^2]$ by step 0.001. The Table 0.2 shows the best C and $Gamma$ combinations which achieve high accuracy which exceed 0.80.

Gamma	*C*	**Accuracy**
0.03	10000	0.803
0.01	110000	0.801
0.01	210000	0.805
0.01	**510000**	**0.815**
0.01	**910000**	**0.813**

Table 0.2: Shows the accuracy of the Support Vector Machine classifier using RBF kernel function for various values of $C \in [10^3, 10^6]$ and $Gamma \in [10^{-5}, 10^2]$ hyperparameters.

A remark of these results is that the best accuracy score takes place while C is a few hundred thousand and $Gamma$ a few millimeters. More specifically, if C gets 510k and $Gamma$ gets 0.01, then the classification accuracy is 0.815.

Decision Trees

The last classifier that was tested is the Decision Tree classifier. About the results of this classifier, many experiments implemented testing various parameters, such as the maximum depth [1] of the tree and the size of the leaf [2]. The maximum depth tested interval is $[1, 10^3]$ and the size of the leaf tested interval is $[1, 500]$. The Table 0.3 represents the best combinations of maximum depth and size of the leaf achieving a high success rate.

Max depth	*Leaf size*	**Accuracy**
517	**4**	**0.708**
132	5	0.704
296	9	0.704
888	9	0.701
145	**10**	**0.707**
598	10	0.703
800	12	0.702
715	13	0.702

Table 0.3: Shows the accuracy of the Decision Tree classifier for various values of $max\ depth \in [1, 10^3]$ and $leaf\ size \in [1, 500]$ hyperparameters.

Looking at the Table 0.3, we notice that the best combination of these two hyperparameters are $max\ depth = 517$ and $leaf\ size = 4$ achieving accuracy 0.708.

[1]The number of nodes along the longest path from the root node down to the farthest leaf node.

[2]The minimum number of samples required to be at a leaf node.

Comparison of methods

In this section, we are going to compare the accuracy score of the four tested classifiers, these are the K-Nearest Neighbor(KNN), Naïve Bayes, Support Vector Machines (SVM) using linear and RBF kernel functions and the Decision Trees. The best results for each one of these classifiers are presented in Table 0.4.

Order	Method	*Hyperparameters*	**Accuracy**
1	SVM RBF	Gamma = 0.01, C = 510k	**0.815**
2	Decision Tree	Max depth = 517, Leaf size = 4	0.708
3	KNN	K = 9	0.602
4	SVM Linear	-	0.512
5	Naïve Bayes	-	0.50

Table 0.4: Table presents the accuracy score for each one of the classifiers, KNN, Naïve Bayes, SVM using linear and radial kernel functions and Decision Trees. The methods are sorted in descending order based on the accuracy score.

Observing the Table 0.4, the classifier with the highest accuracy score, 0.815, is the SVM using radial basis function kernel. The most appropriate values for the C and $Gamma$ parameters are 510k and 0.01 respectively. Follows, the Decision Tree classifier with a difference of 0.107 points. Follows, the KNN classifier using 9 number of neighbors, SVM linear kernel classifier and in the last position is the Naïve Bayes classifier.

Appendix A

This Appendix mentions the libraries which were used for the execution of the selected classifiers, these are KNN, Naïve Bayes, SVM and Decision Trees. Furthermore, includes information about the implementation of the k-fold cross validation and the calculation of the accuracy metric.

The simulations were developed in Python. The initial step is to split the input data into train and test sets using 10-fold cross validation procedure. The code presented below,

```python
from sklearn.model_selection import KFold

k = 10
kf = KFold(n_splits = k, shuffle = True)
for train_index, test_index in kf.split(data):
        ''' Set train set into X_train variable,
        Set the labels of the train set into y_train variable '''
        X_train = data[train_index]
        y_train = labels[train_index]

        ''' Set test set into X_test variable,
        Set the labels of the test set into y_test variable '''
        X_test = data[test_index]
        y_test = labels[test_index]
```

Follows the implementation for each one of the four selected classifiers. There are three basic and similar steps for every method. These are presented below,

1. Build the classifier
2. Train the classifier using the training sets
3. Predict the label for the test dataset

Python provides the following procedure for the K-Nearest Neighbor classifier,

```python
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = k) # k = [1, 2, ... , 9]
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

Follows the code of the Naïve Bayes classifier,

```python
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb = gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
```

The next classifier is the Support Vector Machine classifier. The code of the SVM Linear kernel and the SVM RBF kernel is presented,

```python
from sklearn.svm import SVC

if(kernel == 'linear'):
        svm_linear = SVC(kernel='linear')
elif(kernel == 'rbf'):
        # gamma = [1000, 1000000] by step 10000
        # C = [0.00001, 100] by step 0.001
        svm = SVC(kernel='rbf', random_state = 0, gamma = gamma, C = c)
svm_linear = svm_linear.fit(X_train, y_train)
y_pred = svm_linear.predict(X_test)
```

The last used classifier is the Decision Trees.

```python
from sklearn.tree import DecisionTreeClassifier

# depth = [1, 1000] by step 1
# sizeLeaf = [1, 500] by step 1
clf = DecisionTreeClassifier(max_depth = depth, min_samples_leaf = sizeLeaf)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

The next step is to measure the successful prediction rate. The metric that has been used for this current assignment is the accuracy score. Python provides the next procedure,

```python
from sklearn import metrics

metrics.accuracy_score(y_test, y_pred)
```

Due to the k-fold cross validation, the final accuracy score for every classifier is the average of all accuracy scores.

Appendix B

If x ranges between $[a, b]$, then $(x - a)$ ranges between $[0, b - a]$, so $\frac{(x-a)}{(b-a)}$ ranges between $[0, 1]$, so $\frac{(d-c)(x-a)}{b-a}$ ranges between $[0, d - c]$, so

$$\frac{(d-c)(x-a)}{(b-a)} + c$$

ranges between $[c, d]$.