INTELLIGENT SYSTEMS LAB-8 (18/10/2021)

__NAME__: Harshvardhan Agarwal | __SEC__: A | __REG NO__.: 201800524

PROBLEM STATEMENT

Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set.

Assignment should contain

1. Submit pdf with code and graph.
2. Print both correct and wrong predictions.
3. Analyze with different k-Nearest numbers.
4. Perform experiments on at least 2 datasets.

PROBLEM SOLUTION

Dataset 1: Iris

SOURCE CODE

```
#importing iris dataset
from sklearn.datasets import load_iris

#load dataset
data_iris = load_iris()

#Assign features and target labels to respective variables
X, y = data_iris['data'], data_iris['target']
```

```
#train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#training model with k=4
k = 4
knn_iris = KNeighborsClassifier(n_neighbors=k)
knn_iris.fit(X_train, y_train)
y_hat_iris = knn_iris.predict(X_test)
```

```
#Checking accuracy
print("Train set Accuracy: ", metrics.accuracy_score(y_train, knn_iris.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, y_hat_iris))
```

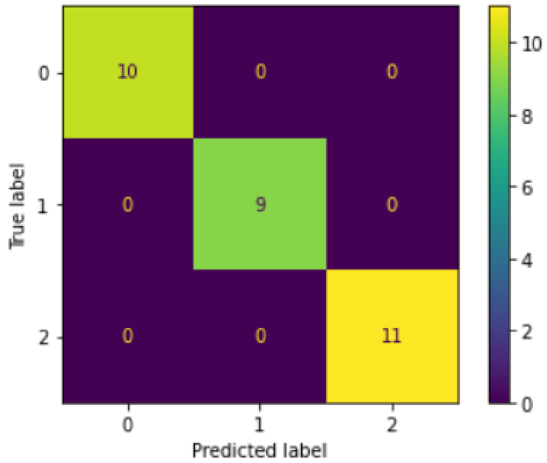
```
Train set Accuracy:  0.9583333333333334
Test set Accuracy:  1.0
```

OUTPUT:

Plotting the confusion matrix:

```
from sklearn.metrics import ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_hat_iris, labels=knn_iris.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=knn_iris.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1999b7623d0>
```



Dataset 2: teleCust1000t.csv

<u>About dataset</u>: Imagine a telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups. If demographic data can be used to predict group membership, the company can customize offers for individual prospective customers. It is a classification problem. That is, given the dataset, with predefined labels, we need to build a model to be used to predict the class of a new or unknown case. The example focuses on using demographic data, such as region, age, and marital, to predict usage patterns. The target field, called custcat, has four possible values that correspond to the four customer groups, as follows: **1- Basic Service 2- E-Service 3- Plus Service 4- Total Service**

SOURCE CODE

Load Data From CSV File

In [2]:
```
df = pd.read_csv('teleCust1000t.csv')
df.head()
```

Out[2]:

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64	4	5	0	0	2	1
1	3	11	33	1	7	136	5	5	0	0	6	4
2	3	68	52	1	24	116	1	29	0	1	2	3
3	2	33	33	0	12	33	2	0	0	1	1	1
4	2	23	30	1	9	30	1	2	0	0	4	3

Data Visualization and Anylisis

Let's see how many of each class is in our data set

In [3]:
```
df['custcat'].value_counts()
```

Out[3]:
```
3    281
1    266
4    236
2    217
Name: custcat, dtype: int64
```

281 Plus Service, 266 Basic-service, 236 Total Service, and 217 E-Service customers

Feature set

Lets defind feature sets, X:

```
df.columns
```

```
Index(['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside', 'custcat'],
      dtype='object')
```

To use scikit-learn library, we have to convert the Pandas data frame to a Numpy array:

```
X = df[['region', 'tenure','age', 'marital', 'address', 'income', 'ed', 'employ','retire', 'gender', 'reside']].values
X[0:5]
```

```
array([[  2,  13,  44,   1,   9,  64,   4,   5,   0,   0,   2],
       [  3,  11,  33,   1,   7, 136,   5,   5,   0,   0,   6],
       [  3,  68,  52,   1,  24, 116,   1,  29,   0,   1,   2],
       [  2,  33,  33,   0,  12,  33,   2,   0,   0,   1,   1],
       [  2,  23,  30,   1,   9,  30,   1,   2,   0,   0,   4]],
      dtype=int64)
```

What are our lables?

```
y = df['custcat'].values
y[0:5]
```

```
array([1, 4, 3, 1, 3], dtype=int64)
```

Normalize Data

Data Standardization give data zero mean and unit variance, it is good practice, especially for algorithms such as KNN which is based on distance of cases:

```
X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]
```

```
array([[-0.02696767, -1.055125  ,  0.18450456,  1.0100505 , -0.25303431,
        -0.12650641,  1.0877526 , -0.5941226 , -0.22207644, -1.03459817,
        -0.23065004],
       [ 1.19883553, -1.14880563, -0.69181243,  1.0100505 , -0.4514148 ,
         0.54644972,  1.9062271 , -0.5941226 , -0.22207644, -1.03459817,
         2.55666158],
       [ 1.19883553,  1.52109247,  0.82182601,  1.0100505 ,  1.23481934,
         0.35951747, -1.36767088,  1.78752803, -0.22207644,  0.96655883,
        -0.23065004],
       [-0.02696767, -0.11831864, -0.69181243, -0.9900495 ,  0.04453642,
        -0.41625141, -0.54919639, -1.09029981, -0.22207644,  0.96655883,
        -0.92747794],
       [-0.02696767, -0.58672182, -0.93080797,  1.0100505 , -0.25303431,
        -0.44429125, -1.36767088, -0.89182893, -0.22207644, -1.03459817,
         1.16300577]])
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (800, 11) (800,)
Test set: (200, 11) (200,)
```

Classification

K nearest neighbor (K-NN)

Import library

Classifier implementing the k-nearest neighbors vote.

```
from sklearn.neighbors import KNeighborsClassifier
```

Training

Lets start the algorithm with k=4 for now:

```
k = 4
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

```
KNeighborsClassifier(n_neighbors=4)
```

Predicting

we can use the model to predict the test set:

```
yhat = neigh.predict(X_test)
yhat[0:5]
```

```
array([1, 1, 3, 2, 4], dtype=int64)
```

Accuracy evaluation

In multilabel classification, **accuracy classification score** function computes subset accuracy. This function is equal to the jaccard_similarity_score function. Essentially, it calculates how match the actual labels and predicted labels are in the test set.

```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
Train set Accuracy:  0.5475
Test set Accuracy:  0.32
```

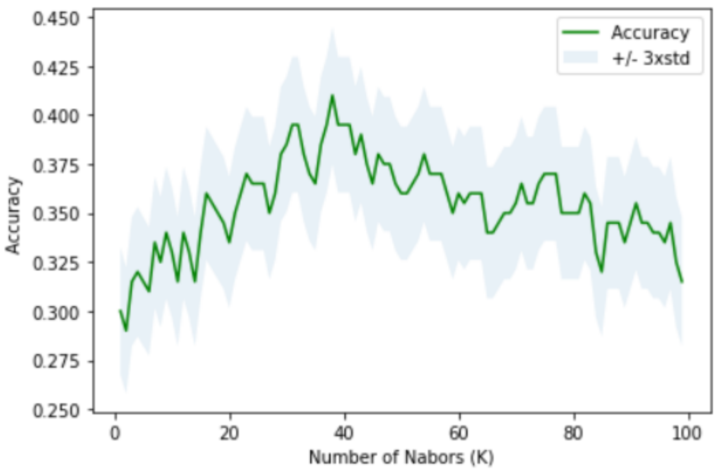
```
#For different values of K
Ks = 100
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)
    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])
mean_acc
```

```
array([0.3  , 0.29 , 0.315, 0.32 , 0.315, 0.31 , 0.335, 0.325, 0.34 ,
       0.33 , 0.315, 0.34 , 0.33 , 0.315, 0.34 , 0.36 , 0.355, 0.35 ,
       0.345, 0.335, 0.35 , 0.36 , 0.37 , 0.365, 0.365, 0.365, 0.35 ,
       0.36 , 0.38 , 0.385, 0.395, 0.395, 0.38 , 0.37 , 0.365, 0.385,
       0.395, 0.41 , 0.395, 0.395, 0.395, 0.38 , 0.39 , 0.375, 0.365,
       0.38 , 0.375, 0.375, 0.365, 0.36 , 0.36 , 0.365, 0.37 , 0.38 ,
       0.37 , 0.37 , 0.37 , 0.36 , 0.35 , 0.36 , 0.355, 0.36 , 0.36 ,
       0.36 , 0.34 , 0.34 , 0.345, 0.35 , 0.35 , 0.355, 0.365, 0.355,
       0.355, 0.365, 0.37 , 0.37 , 0.37 , 0.35 , 0.35 , 0.35 , 0.35 ,
       0.36 , 0.355, 0.33 , 0.32 , 0.345, 0.345, 0.345, 0.335, 0.345,
       0.355, 0.345, 0.345, 0.34 , 0.34 , 0.335, 0.345, 0.325, 0.315])
```

<u>OUTPUT:</u>

Plot model accuracy for Different number of Neighbors

```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()
```




```
print( "The best accuracy was", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was 0.41 with k= 38
```