LAB # 11

Statement Purpose:

This lab will introduce students to classification using K nearest neighbors. Students will also use cross validation on the classifier.

Activity Outcomes:

This lab teaches you the following topics:

- How to use k nearest neighbors for classification
- How to improve the KNN classifier using cross validation

Instructor Note:

As pre-lab activity, read "Matlab" An introduction with Applications" 4th edition by Amos Gilat, John Wiley and Sons.

1) Stage J (Journey)

Introduction:

Given a set X of n points and a distance function, k-nearest neighbor (kNN) search lets you find the k closest points in X to a query point or set of points Y. The kNN search technique and kNN-based algorithms are widely used as benchmark learning rules. The relative simplicity of the kNN search technique makes it easy to compare the results from other classification techniques to kNN results. The technique has been used in various areas such as:

- bioinformatics
- image processing and data compression
- document retrieval
- computer vision
- multimedia database
- marketing data analysis

You can use kNN search for other machine learning algorithms, such as:

- kNN classification
- local weighted regression
- missing data imputation and interpolation

- density estimation

  You can also use *k*NN search with many distance-based learning functions, such as K-means clustering.

2) Stage a1 (apply)

Lab Activities:

Activity 1:

Using k nearest neighbors for supervised learning.

Solution:

Load the iris data set and check features

```
#Import the load_iris function from datsets module
from sklearn.datasets import load_iris
```

```
#Create bunch object containing iris dataset and its attributes.
iris = load_iris()
```

```
type(iris)
```

```
sklearn.utils.Bunch
```

```
#Print the iris data
iris.data
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
```

```
#Names of 4 features (column names)
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (c
m)']
```

```
#Integers representing the species: 0 = setosa, 1=versicolor, 2=virginica
print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

```
# 3 classes of target
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

```
print(type(iris.data))
print(type(iris.target))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

```
# we have a total of 150 observations and 4 features
print(iris.data.shape)
```

```
(150, 4)
```

Create train and test splits

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(iris.data,iris.target,test_size=0.2,random_state=4)
```

```
#shape of train and test objects
print(X_train.shape)
print(X_test.shape)
```

```
(120, 4)
(30, 4)
```

```
# shape of new y objects
print(y_train.shape)
print(y_test.shape)
```

```
(120,)
(30,)
```

Train the model

```
#import the KNeighborsClassifier class from sklearn
from sklearn.neighbors import KNeighborsClassifier

#import metrics model to check the accuracy
from sklearn import metrics
#Try running from k=1 through 25 and record testing accuracy
k_range = range(1,26)
scores = {}
scores_list = []
for k in k_range:
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train,y_train)
        y_pred=knn.predict(X_test)
        scores[k] = metrics.accuracy_score(y_test,y_pred)
        scores_list.append(metrics.accuracy_score(y_test,y_pred))
```

Test the model

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X,y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
#0 = setosa, 1=versicolor, 2=virginica
classes = {0:'setosa',1:'versicolor',2:'virginica'}

#Making prediction on some unseen data
#predict for the below two random observations
x_new = [[3,4,5,2],
         [5,4,2,2]]
y_predict = knn.predict(x_new)

print(classes[y_predict[0]])
print(classes[y_predict[1]])
```

```
versicolor
setosa
```

Submission Activity:

Import wine dataset and perform knn classification