DATA MINING AND ANALYTICS – IE-6318-001

HOMEWORK 3

Submitted by :

Deepak Ram Bala Boopathy

(1001671704)

1. For the IRIS dataset, prepare a training dataset and a testing dataset for classification model training and testing. For each class, take the first 40 samples into the training dataset, and the remaining 10 samples into the testing dataset.

**CODES:**

**import** numpy **as** np  
**import** pandas **as** pd  
iris = pd.read\_csv(**"/Users/deepakboopathy/Desktop/UTA/Data Mining/iris.csv"**)  
a = pd.DataFrame(iris)  
X1=a.iloc[0:50,0:4]  
Y1=a.iloc[0:50,4:5]**from** sklearn.model\_selection **import** train\_test\_split  
X1\_train, X1\_test , Y1\_train , Y1\_test = train\_test\_split(X1,Y1,test\_size=0.2,random\_state = 1)  
X2=a.iloc[50:100,0:4]  
Y2=a.iloc[50:100,4:5]  
X2\_train, X2\_test , Y2\_train , Y2\_test = train\_test\_split(X2,Y2,test\_size=0.2,random\_state = 1)X3=a.iloc[100:150,0:4]  
Y3=a.iloc[100:150,4:5]  
X3\_train, X3\_test , Y3\_train , Y3\_test = train\_test\_split(X3,Y3,test\_size=0.2,random\_state = 1)  
Xa\_train=X1\_train.append(X2\_train)  
X\_train=Xa\_train.append(X3\_train)  
Ya\_train=Y1\_train.append(Y2\_train)  
Y\_train=Ya\_train.append(Y3\_train)  
Xa\_test=X1\_test.append(X2\_test)  
X\_test=Xa\_test.append(X3\_test)  
Ya\_test=Y1\_test.append(Y2\_test)  
Y\_test=Ya\_test.append(Y3\_test)

2. Make a KNN classifier for the 3-class classification problem using the Minkowski distance function you made in HW2. The KNN function performs classification based on the majority voting of Knearest neighbors. Implement the KNN classifiers to the IRIS dataset using K = 3, 5, 7 for K-nearest neighbors, and r = 1, 2, 4 for the distance order of Minkowski Distance. For each parameter setting of K and r, perform classification for the testing data samples you prepared in problem 1. 1) For each KNN parameter setting, report classification accuracy and the confusion matrix. 2) Calculate and report the classification accuracy for each class at each parameter setting. 3) Assume we use the average accuracy of each class as the overall model performance measure, find the best parameter setting that generates the highest average accuracy for the 3 classes.

**CODES AND OUTPUT:**

1)Classification accuracy and confusion matrix

1. **PARAMETER SETTING: k=3 & r=1**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 3,weights=**'uniform'**,p=1,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[10 0 0]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9666666666666667

1. **PARAMETER SETTING: k=3 & r=2**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 3,weights=**'uniform'**,p=2,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

**[[10 0 0]**

**[ 0 10 0]**

**[ 0 0 10]]**

Accuracy:**0.95**

1. **PARAMETER SETTING: k=3 & r=4**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 3,weights=**'uniform'**,p=4,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[10 0 0]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9666666666666667

1. **PARAMETER SETTING: k=5 & r=1**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 5,weights=**'uniform'**,p=1,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[ 9 0 1]

[ 0 10 0]

[ 1 0 9]]

Accuracy: 0.9333333333333333

1. **PARAMETER SETTING: k=5 & r=2**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 5,weights=**'uniform'**,p=2,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[ 9 0 1]

[ 0 10 0]

[ 0 0 10]]

Classification Accuracy: 0.9666666666666667

1. **PARAMETER SETTING: k=5 & r=4**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 5,weights=**'uniform'**,p=4,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[ 9 0 1]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9666666666666667

1. **PARAMETER SETTING: k=7 & r=1**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 7,weights=**'uniform'**,p=1,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[ 9 0 1]

[ 0 10 0]

[ 2 0 8]]

Accuracy: 0.9

1. **PARAMETER SETTING: k=7 & r=2**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 7,weights=**'uniform'**,p=2,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[ 9 0 1]

[ 0 10 0]

[ 1 0 9]]

Accuracy:0.9333333333333333

1. **PARAMETER SETTING: k=7 & r=4**

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** metrics  
**from** sklearn.model\_selection **import** cross\_val\_score  
classifier = KNeighborsClassifier(n\_neighbors = 7,weights=**'uniform'**,p=4,metric=**'minkowski'**)  
classifier.fit(X\_train, Y\_train.values.ravel())  
y\_pred = classifier.predict(X\_test)  
**from** sklearn.metrics **import** confusion\_matrix  
cm = confusion\_matrix(Y\_test, y\_pred)  
print(cm)  
print(metrics.accuracy\_score(Y\_test, y\_pred))  
scores = cross\_val\_score(classifier, X\_train, Y\_train, cv=10, scoring=**'accuracy'**)  
print(scores.mean())

[[ 9 0 1]

[ 0 10 0]

[ 1 0 9]]

Accuracy:0.9333333333333333

2)Classification accuracy for each class at each parameter setting

1. **PARAMETER SETTING: k=3 & r=1**

[[10 0 0]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9416666666666668

1. **PARAMETER SETTING: k=3 & r=2**

**[[10 0 0]**

**[ 0 10 0]**

**[ 0 0 10]]**

Accuracy:**0.95**

1. **PARAMETER SETTING: k=3 & r=4**

[[10 0 0]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9583333333333333

1. **PARAMETER SETTING: k=5 & r=1**

[[ 9 0 1]

[ 0 10 0]

[ 1 0 9]]

Accuracy: 0.9416666666666667

1. **PARAMETER SETTING: k=5 & r=2**

[[ 9 0 1]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9583333333333333

1. **PARAMETER SETTING: k=5 & r=4**

[[ 9 0 1]

[ 0 10 0]

[ 0 0 10]]

Accuracy: 0.9583333333333333

1. **PARAMETER SETTING: k=7 & r=1**

[[ 9 0 1]

[ 0 10 0]

[ 2 0 8]]

Accuracy: 0.9333333333333333

1. **PARAMETER SETTING: k=7 & r=2**

[[ 9 0 1]

[ 0 10 0]

[ 1 0 9]]

Accuracy: 0.9666666666666666

1. **PARAMETER SETTING: k=7 & r=4**

[[ 9 0 1]

[ 0 10 0]

[ 1 0 9]]

Accuracy: 0.9666666666666666

3)Best Parameter Setting

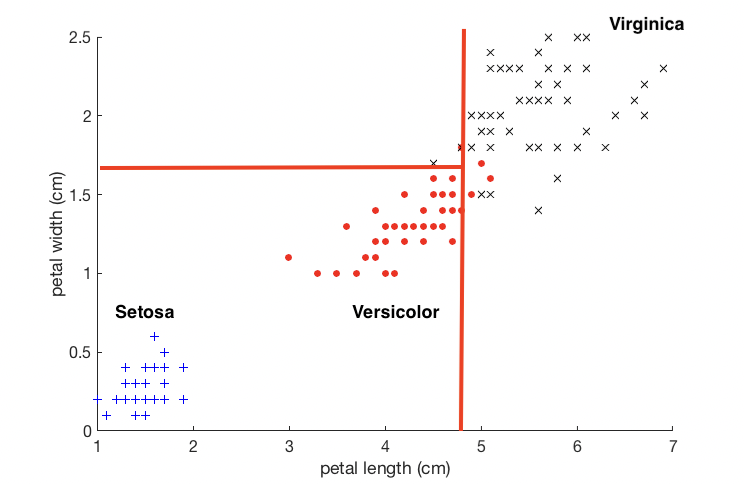
**PARAMETER SETTING: k=3 & r=4**

[[10 0 0]

[ 0 10 0]

[ 0 0 10]]

3. As shown in the plot below, a simple decision tree is constructed to classify two iris flowers: Versicolor and Virginica using two features of petal width and petal length. Assume the binary decision boundary on Petal Length is 4.8, and the decision boundary on Petal Width is 1.7. Make a function to implement this simple decision tree and use your function to classify the 100 iris samples of Versicolor and Virginica. Report the classification accuracy, sensitivity, and specificity. Here we define sensitivity = accuracy for class Versicolor, and specificity = accuracy of class Virginica.



**CODE:**

**import** pandas **as** pd  
  
data = pd.read\_csv(**"/Users/deepakboopathy/Desktop/UTA/Data Mining/irisfin.csv"**, header=**None**)  
  
data = data.iloc[50:]  
  
  
**def** decision\_tree(data):  
 pred = [**"Null" for** i **in** range(100)]  
 data[**"Prediction"**] = pred  
  
 **for** i, row **in** data.iterrows():  
 PL = row[2]  
 PW = row[3]  
 **if** (PL >= 4.8):  
 **if** (PW >= 1.7):  
 data.at[i, **"Prediction"**] = **"Iris-virginica"  
 else**:  
 **if** (PW <= 1.7):  
 data.at[i, **"Prediction"**] = **"Iris-versicolor"** Vers\_c = 0  
 Vers\_w = 0  
 Virg\_c = 0  
 Virg\_w = 0  
  
 rows\_vers = data.iloc[:50]  
 rows\_virg = data.iloc[50:]  
  
 **for** i, row **in** rows\_vers.iterrows():  
 **if** (row[4] == row[**"Prediction"**]):  
 Vers\_c += 1  
 **else**:  
 Vers\_w += 1  
  
 **for** i, row **in** rows\_virg.iterrows():  
 **if** (row[4] == row[**"Prediction"**]):  
 Virg\_c += 1  
 **else**:  
 Virg\_w += 1  
  
 accuracy = (Vers\_c + Virg\_c)  
 sensitivity = (Vers\_c / 50) \* 100  
 specificity = (Virg\_c / 50) \* 100  
  
 **return** [accuracy, sensitivity, specificity]  
  
  
[accuracy, sensitivity, specificity] = decision\_tree(data)  
print(**"Accuracy : "** + str(accuracy) + **"%"**)  
print(**"Sensitivity : "** + str(sensitivity) + **"%"**)  
print(**"Specificity : "** + str(specificity) + **"%"**)

**OUTPUT:**

Accuracy : 89%

Sensitivity : 88.0%

Specificity : 90.0%