dm-programming-assignment2

March 25, 2023

[1]:
```
import os
os.chdir("/Users/vitta/Downloads")
p=os.path.abspath(os.getcwd())
p
```

[1]:
```
'C:\\Users\\vitta\\Downloads'
```

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from shutil import copyfile, rmtree
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import pandas as pd
import cv2
import os
import random
from pathlib import Path
```

[3]:
```
path=r'C:\Users\vitta\Downloads\Weed-4class-45'
print(path)
```

```
C:\Users\vitta\Downloads\Weed-4class-45
```

[4]:
```
df = pd.read_csv("C:/Users/vitta/Downloads/Weed-4class-45/Weed-4class-45-labels.
  ↪csv")
df
```

[4]:
```
                    Filename  Label     Species
0      20161207-112417-0.jpg      8    Negative
1      20161207-112431-0.jpg      8    Negative
```

```
2      20161207-112802-0.jpg      8    Negative
3      20161207-112812-0.jpg      8    Negative
4      20170128-101909-0.jpg      8    Negative
…                          …    …          …
13323  20171025-172145-3.jpg      3  Parthenium
13324  20171025-172200-3.jpg      3  Parthenium
13325  20171025-172226-3.jpg      3  Parthenium
13326  20171025-172236-3.jpg      3  Parthenium
13327  20171025-172247-3.jpg      3  Parthenium

[13328 rows x 3 columns]
```

[59]:

```
d = pd.read_csv("C:/Users/vitta/Downloads/Weed-4class-45/Weed-4class-45-labels.
 ↪csv")
f="Filename"
speciess="Species"
labl="Label"
cl=list(d[labl])
fl=list(d[f])
ln=len(cl)
print(len(cl))
Parthenium,Prickly,Siam,Lantana,Negative=[],[],[],[],[]
for i in range(0,ln):
    if cl[i] == 1:
        Lantana.append(fl[i])
    elif cl[i] == 3:
        Parthenium.append(fl[i])
    elif cl[i] == 4:
        Prickly.append(fl[i])
    elif cl[i] == 6:
        Siam.append(fl[i])
    elif cl[i] == 8:
        Negative.append(fl[i])
    else:
        continue
Total=[Parthenium,Prickly,Siam,Lantana,Negative]
blist=[[],[],[],[],[]]
for x in range(0,5):
    blist[x]=[]
    for each in Total[x]:
        s=os.path.join(path,each)
        image=plt.imread(s)
        gr_image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        (n,bins)=np.histogram(gr_image,256,[0,256])
        blist[x].append(n)
```

```
13328
```

[47]:

```
combine=[blist[0],blist[1],blist[2],blist[3],blist[4]]
hist_list=[]
for x in combine:
    for y in x:
        hist_list.append(y)
print(len(hist_list))
print(len(combine))
```

```
13328
5
```

[48]:

```
from sklearn.model_selection import train_test_split

train,test=train_test_split(hist_list,test_size=0.2,random_state=1)
Parthenium_train, Parthenium_test= train_test_split(blist[0], test_size=0.2,␣
 ↪random_state=5)

print("Parthenium split:")
print(len(Parthenium_train))
print(len(Parthenium_test))

Prickly_train, Prickly_test= train_test_split(blist[1], test_size=0.2,␣
 ↪random_state=12)

print("Prickly split:")
print(len(Prickly_train))
print(len(Prickly_test))

Siam_train, Siam_test= train_test_split(blist[2], test_size=0.2,␣
 ↪random_state=14)

print("Siamsplit:")
print(len(Siam_train))
print(len(Siam_test))

Lantana_train, Lantana_test= train_test_split(blist[3], test_size=0.2,␣
 ↪random_state=8)

print("Lantanasplit:")
print(len(Lantana_train))
print(len(Lantana_test))

Negative_train,Negative_test= train_test_split(blist[4], test_size=0.2,␣
 ↪random_state=8)
print("Negative:")
print(len(Negative_train))
print(len(Negative_test))
```

```
Parthenium split:
817
205
Prickly split:
849
213
Siamsplit:
859
215
Lantanasplit:
851
213
Negative:
7284
1822
```

[11]:

```
combined_train=[Parthenium_train,Prickly_train,Siam_train,Lantana_train,Negative_train]
combine_trn=[]
for x in combined_train:
    for y in x:
        combine_trn.append(y)
```

[49]:

```
from sklearn.neighbors import KNeighborsClassifier
X_train, X_test, y_train, y_test = train_test_split(hist_list, cl,␣
 ↪random_state=4)
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print('accuracy: ', knn.score(X_test, y_test))
```

```
accuracy:  0.7418967587034814
```

[50]:

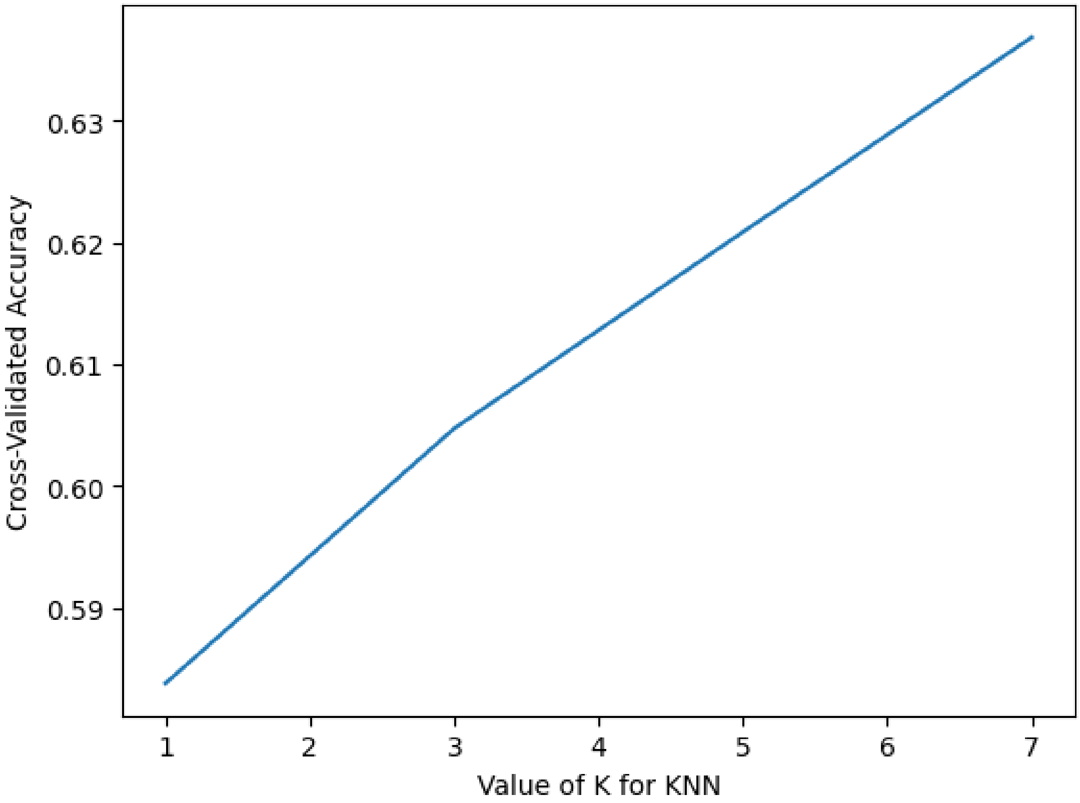
```
from sklearn.model_selection import cross_val_score
knn = KNeighborsClassifier(n_neighbors = 5)
scores = cross_val_score(knn, hist_list, cl, cv=5, scoring='accuracy')
print(scores)
print(scores.mean())
```

```
[0.56939235 0.68192048 0.66804201 0.684803   0.50018762]
0.6208690915693277
```

[51]:

```
import matplotlib.pyplot as plt
%matplotlib inline
k_range =[1,3,5,7]
k_scores = []
for k in k_range:
 knn = KNeighborsClassifier(n_neighbors=k)
 scores = cross_val_score(knn, hist_list, cl, cv=5, scoring='accuracy')
 print(scores)
 k_scores.append(scores.mean())
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```

```
[0.55701425 0.63578395 0.60277569 0.64390244 0.47954972]
[0.56789197 0.66541635 0.64253563 0.66078799 0.48705441]
[0.56939235 0.68192048 0.66804201 0.684803   0.50018762]
[0.5783946  0.69692423 0.69054764 0.70318949 0.515197  ]
```



[53]:
```
from sklearn.neighbors import KNeighborsClassifier
X_train, X_test, y_train, y_test = train_test_split(hist_list, cl,␣
 ↪random_state=4)
knn = KNeighborsClassifier(n_neighbors = 7)
scores = cross_val_score(knn, hist_list, cl, cv=5, scoring='accuracy')
print(scores.mean())
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print('accuracy: ', knn.score(X_test,y_test))
```

```python
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
0.6368505916347754
accuracy:  0.7545018007202882
0.7545018007202882
```

[55]:

```python
#decision tree
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.tree import DecisionTreeClassifier
clfr = DecisionTreeClassifier(criterion='entropy', random_state=0)
clfr.fit(X_train,y_train)
pr = clfr.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, pr))
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm2 = confusion_matrix(y_test, pr)
print(cm2)
print(classification_report(y_test, pr))
```

```
0.6848739495798319
[[ 130   20    3   21  104]
 [  21  100   11   22  103]
 [   6   15  125   34   91]
 [  24   16   14   76  130]
 [ 103   88   96  128 1851]]
              precision    recall  f1-score   support

           1       0.46      0.47      0.46       278
           3       0.42      0.39      0.40       257
           4       0.50      0.46      0.48       271
           6       0.27      0.29      0.28       260
           8       0.81      0.82      0.81      2266

    accuracy                           0.68      3332
   macro avg       0.49      0.49      0.49      3332
weighted avg       0.68      0.68      0.68      3332
```

[56]:

```python
#SVM
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score
```

```

svm_model = SVC(kernel = 'rbf', C = 10)
svm_scores = cross_val_score(svm_model,X_train, y_train, cv=5)
svm_model.fit(X_train, y_train)
acc = svm_model.score(X_test, y_test)

print(f"SVM Classifier scores: {svm_scores}")
print(f"Average score = {svm_scores.mean()*100:0.2f}")

y_pred = svm_model.predict(X_test)
acc = svm_model.score(X_test, y_test)
print("Accuracy: {:.2f}%".format(acc * 100))
```

```
SVM Classifier scores: [0.785      0.78389195 0.78789395 0.7913957  0.7923962 ]
Average score = 78.81
Accuracy: 79.65%
```

[58]:

```
#Naïve Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


X_train, X_test, y_train, y_test = train_test_split(hist_list,cl, test_size=0.
 ↪2,random_state=42)


nb_model = GaussianNB()


nb_model.fit(X_train, y_train)


y_pred = nb_model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)

print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 22.84%
```