CS550_W3_HWQ30_19556_SAHITI_EMANI

GITHUB LINK:

https://github.com/sahitiemani96/CS550_MACHINE_LEARNING_19556_SAHITI_EMANI/blob/main/IRIS_KNN_19556.ipynb

```
import pandas as pdimport numpy as npimport sklearnimport seaborn as snsimport matplotlib.pyplot as pltfrom sklearn.preprocessing import StandardScalerfrom sklearn import metricsfrom sklearn.metrics import *from sklearn.model_selection import *from sklearn.model_selection import train_test_splitfrom sklearn.neighbors import KNeighborsClassifier
```

In [37]:

```
from google.colab import files

uploaded=files.upload()

import io

iris=pd.read_csv(io.BytesIO(uploaded['iris.data']),names=['sepal_length','sepal_width','petal_length','petal_width','species'])

iris.shape

col_list = iris.columns

print(type(col_list))

print(col_list[:])

iris['species'].value_counts()

iris_data = iris.iloc[:,0:4] # select all the rows and col indices 0 to 3

iris_lables = iris.iloc[:,4:] # select all trhe rows and 4th cloumn

iris_data.shape

iris_data.head(2)

)

iris_lables.shape

iris_lables.head(2)

#standardizing using sklearn pre-processing

iris_standard = StandardScaler().fit_transform(iris_data)  # this has transformed dataframe to numpy N-dimensional array,
```

```
#each row in df is a list we will have n inner lists in a outer list,thats why length of iris_standard is 150 and
#length of each inner list is 4.
print('length of iris_standard is ',len(iris_standard))
print('length of inner list is',len(iris_standard[0]))
print('sample elements are')
print((iris_standard[0:3]))
#splitting dataset into train and test


iris_lables_np = iris_lables.values.reshape(1,150)
x_train, x_test, y_train, y_test = train_test_split(iris_standard, iris_lables_np[0], test_size=0.33,
random_state=42)
print(x_test[0:2],y_test[0:2])
print(len(x_test),len(y_test))
print(len(x_train),len(y_train))
#Training using K_NN


neigh = KNeighborsClassifier(n_neighbors=5)
neigh.fit(x_train, y_train)
#predicting


predict_array = neigh.predict(x_test)
print(metrics.accuracy_score(y_test, predict_array))
#print(predict_array[0])
#print(y_test[0])
```

```
for i in range(len(predict_array)):
  if (predict_array[i] != y_test[i]):
     print('actual is {} but predicted is {}'.format(y_test[i],predict_array[i]))
     print('Wrong')
#prediction on non standardized data
x_train, x_test, y_train, y_test = train_test_split(iris_data, iris_lables_np[0], test_size=0.33,
random_state=42)
neigh2 = KNeighborsClassifier(n_neighbors=5)
neigh2.fit(x_train, y_train)
predict_array = neigh2.predict(x_test)
print(metrics.accuracy_score(y_test, predict_array))
#cross validation using 10 folds,cv=10


k_list= [1,3,5,7,9]
cv_scores=[]


for i in k_list:
   cross_neigh = KNeighborsClassifier(n_neighbors=i)
   scores = cross_val_score(cross_neigh,x_train, y_train,cv=10)
   cv_scores.append(np.mean(scores))


print(len(cv_scores))
print(cv_scores)
```

```
cv_score_zip=zip(k_list,cv_scores)


for i in cv_score_zip:
    print(i)


#plot for K-value and accuracy using 10 fold cv.


plt.figure('Iris_KNN')
plt.xlabel('k-value')
plt.ylabel('cv_score')
plt.grid()
plt.plot(k_list,cv_scores)
plt.show()


# based on above observations we are getting maximum accuracy when k=7,
#So we will use K-value 7 and predict on test datsset and see accuracy.


neigh_K7 = KNeighborsClassifier(n_neighbors=7)
neigh_K7.fit(x_train, y_train)
predict_array_k7 = neigh_K7.predict(x_test)
print(metrics.accuracy_score(y_test, predict_array_k7))
predict_probability = neigh_K7.predict_proba(x_test)
```

```
#zipped_pobability = zip(predict_array_k7,predict_probability)
#for i in zipped_pobability:
#    print(i)


cross_predict = cross_val_predict(cross_neigh,x_test,y_test,cv=10)
print(metrics.accuracy_score(y_test, cross_predict))
#confusion matrix and classification_report
#precision = TP/TP+FP
#Recall = TP/TP+FN


print(metrics.confusion_matrix(y_test, cross_predict))
print(metrics.classification_report(y_test, cross_predict))
```