**CODE:**

* import pandas as pd
* datafr =pd.read\_csv('hand2.csv')
* dup=datafr.duplicated().any()
* dup

False

* cate\_val=[]

con\_val=[]

for column in datafr.columns:

if datafr[column].nunique() <=10:

cate\_val.append(column)

else:

con\_val.append(column)

* cate\_val

['subject\_id',

'Test\_Name',

'Test\_Category',

'Test\_Location',

'Test\_Laterality',

'Test\_trial',

'Test\_Unit',

'Test\_Break',

'HandTest\_Name',

'HandTest\_Category',

'HandTest\_Location',

'HandTest\_Laterality',

'HandTest\_trial',

'HandTest\_Unit',

'HandTest\_Setting']

* con\_val

['MS\_Delta', 'Test\_Result', 'HandMS\_Delta', 'HandTest\_Result']

* df=pd.get\_dummies(datafr,columns=cate\_val,drop\_first=True)
* datafr.head()
* from sklearn.preprocessing import StandardScaler
* sts=StandardScaler()

datafr[con\_val]=sts.fit\_transform(datafr[con\_val])

* x=df.drop(['Test\_Break\_Yes'],axis=1)
* y = df['Test\_Break\_Yes']
* from sklearn.model\_selection import train\_test\_split
* x\_train, x\_test, y\_train, y\_test=train\_test\_split(x,y,test\_size=0.33,random\_state=42)
* from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

ss = knn.fit(x\_train, y\_train)

* from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()

ss=rf.fit(x\_train,y\_train)

ss.fit(x\_train,y\_train)

RandomForestClassifier()

* y\_p = ss.predict(x\_test)
* t\_p=ss.predict(x\_train)
* from sklearn.metrics import accuracy\_score ,confusion\_matrix,precision\_score ,recall\_score,f1\_score
* accuracy\_score(y\_test,y\_p)\*100

96.90721649484536

* accuracy\_score(y\_train,t\_p)\*100

100.0

* from sklearn.model\_selection import KFold , cross\_val\_score

kfold = KFold(n\_splits=10, random\_state=7,shuffle=True)

res = cross\_val\_score(knn, x, y, cv=kfold, scoring='accuracy')

* print("Accuracy:",res.mean(), res.std())

Accuracy: 0.917816091954023 0.06682749131103662

* print(res.mean()\*100)
* matrix = confusion\_matrix(y\_test, y\_p)

print(matrix)

* precision = precision\_score(y\_test, y\_p)

print("Precision:", precision)

* recall = recall\_score(y\_test, y\_p)

print("Recall:", recall)

* f1=f1\_score(y\_test,y\_p)

print("F1 score:", f1)

* auc = cross\_val\_score(knn, x, y, cv=kfold, scoring='roc\_auc')

print("AUC:",auc.mean(), auc.std())

* import matplotlib.pyplot as plt

import numpy as np

metrics = ['Accuracy', 'Precision', 'Recall', 'F-score', 'AUC-ROC']

scores = [res.mean(), precision, recall, f1, auc.mean()]

fig, zx = plt.subplots(figsize=(10,5))

zx.bar(np.arange(len(metrics)), scores)

zx.set\_xticks(np.arange(len(metrics)))

zx.set\_xticklabels(metrics)

zx.set\_ylabel('Score')

plt.show()

* new = pd.DataFrame({

'MS\_Delta': 0,

'Test\_Result': 33,

'HandMS\_Delta': 0,

'HandTest\_Result': 33,

'subject\_id\_1736': 1,

'subject\_id\_2598': 0,

'subject\_id\_4918': 0,

'Test\_Name\_Isometric Muscle Strength, Dorsiflexion': 0,

'Test\_Name\_Isometric Muscle Strength, Extension': 0,

'Test\_Name\_Isometric Muscle Strength, Flexion': 1,

'Test\_Location\_ELBOW JOINT': 1,

'Test\_Location\_FIRST DORSAL INTEROSSEOUS MUSCLE OF THE HAND': 0,

'Test\_Location\_HIP FLEXOR MUSCLE': 0,

'Test\_Location\_KNEE JOINT': 0,

'Test\_Location\_SHOULDER JOINT': 0,

'Test\_Location\_WRIST JOINT': 0,

'Test\_Laterality\_RIGHT': 0,

'Test\_trial\_2': 1,

'Test\_trial\_3': 0,

'HandTest\_Laterality\_RIGHT': 1,

'HandTest\_trial\_2': 1,

'HandTest\_Setting\_3': 1,

'HandTest\_Setting\_2': 0

}, index=[0])

# Ensure the columns are in the same order as they were during training

new = new[x\_train.columns]

# Now you can make predictions

p = knn.predict(new)

if p[0] == 0:

print("No ALS")

else:

print("ALS")

* import joblib

joblib.dump(knn,'model\_joblib\_als')

model = joblib.load('model\_joblib\_als')

model.predict(new)

from tkinter import \*

import joblib

def show\_entry\_fields():

p1=int(h1.get())

p2=int(h2.get())

p3=int(h3.get())

p4=int(h4.get())

p5=int(h5.get())

p6=int(h6.get())

p7=int(h7.get())

p8=int(h8.get())

p9=int(h9.get())

p10=float(h10.get())

p11=int(h11.get())

p12=int(h12.get())

p13=int(h13.get())

p14=int(h14.get())

p15=int(h15.get())

p16=int(h16.get())

p17=int(h17.get())

p18=int(h18.get())

p19=int(h19.get())

p20=int(h20.get())

p21=int(h21.get())

p22=int(h22.get())

p23=int(h23.get())

model = joblib.load('model\_joblib\_als')

result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8,p8,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,p21,p22,p23]])

if result == 0:

Label(mtr, text="No ALS").grid(row=37)

else:

Label(mtr, text="Possibility of ALS").grid(row=37)

mtr = Tk()

mtr.title("Amyotrophic Lateral Sclerosis Prediction System")

label = Label(mtr, text = "ALS Prediction System"

, bg = "blue", fg = "white"). \

grid(row=0,columnspan=2)

Label(mtr, text="Enter MS\_Delta").grid(row=1)

Label(mtr, text="Test\_Result").grid(row=2)

Label(mtr, text="Enter Value of subject\_id\_1736").grid(row=3)

Label(mtr, text="Enter Value of subject\_id\_2598").grid(row=4)

Label(mtr, text="Enter Value of subject\_id\_4918").grid(row=5)

Label(mtr, text="Enter Value of Test\_Name\_Isometric Muscle Strength, Dorsiflexion").grid(row=6)

Label(mtr, text="Enter Value of Test\_Name\_Isometric Muscle Strength, Extension").grid(row=7)

Label(mtr, text="Enter Value of Test\_Name\_Isometric Muscle Strength, Flexion").grid(row=8)

Label(mtr, text="Enter Value of Test\_Location\_ELBOW JOINT").grid(row=9)

Label(mtr, text="Enter Value of Test\_Location\_FIRST DORSAL INTEROSSEOUS MUSCLE OF THE HAND").grid(row=10)

Label(mtr, text="Enter Value of Test\_Location\_HIP FLEXOR MUSCLE").grid(row=11)

Label(mtr, text="Enter Value of Test\_Location\_KNEE JOINT").grid(row=12)

Label(mtr, text="Enter Value of Test\_Location\_SHOULDER JOINT").grid(row=13)

Label(mtr, text="Enter Value of Test\_Location\_WRIST JOINT").grid(row=14)

Label(mtr, text="Enter Value of Test\_Laterality\_RIGHT").grid(row=15)

Label(mtr, text="Enter Value of Test\_trial\_2").grid(row=16)

Label(mtr, text="Enter Value of Test\_trial\_3").grid(row=17)

Label(mtr, text="Enter Value of HandTest\_Laterality\_RIGHT").grid(row=18)

Label(mtr, text="Enter Value of HandTest\_trial\_2").grid(row=19)

Label(mtr, text="Enter Value of HandTest\_Setting\_3").grid(row=20)

Label(mtr, text="Enter Value of HandTest\_Setting\_2").grid(row=21)

Label(mtr, text="Enter Value of HandMS\_Delta").grid(row=22)

Label(mtr, text="Enter Value of HandTest\_Result").grid(row=23)

h1 = Entry(mtr)

h2 = Entry(mtr)

h3 = Entry(mtr)

h4 = Entry(mtr)

h5 = Entry(mtr)

h6 = Entry(mtr)

h7 = Entry(mtr)

h8 = Entry(mtr)

h9 = Entry(mtr)

h10 = Entry(mtr)

h11 = Entry(mtr)

h12 = Entry(mtr)

h13 = Entry(mtr)

h14 = Entry(mtr)

h15 = Entry(mtr)

h16 = Entry(mtr)

h17 = Entry(mtr)

h18 = Entry(mtr)

h19 = Entry(mtr)

h20 = Entry(mtr)

h21 = Entry(mtr)

h22 = Entry(mtr)

h23 = Entry(mtr)

h1.grid(row=1, column=1)

h2.grid(row=2, column=1)

h3.grid(row=3, column=1)

h4.grid(row=4, column=1)

h5.grid(row=5, column=1)

h6.grid(row=6, column=1)

h7.grid(row=7, column=1)

h8.grid(row=8, column=1)

h9.grid(row=9, column=1)

h10.grid(row=10, column=1)

h11.grid(row=11, column=1)

h12.grid(row=12, column=1)

h13.grid(row=13, column=1)

h14.grid(row=14, column=1)

h15.grid(row=15, column=1)

h16.grid(row=16, column=1)

h17.grid(row=17, column=1)

h18.grid(row=18, column=1)

h19.grid(row=19, column=1)

h20.grid(row=20, column=1)

h21.grid(row=21, column=1)

h22.grid(row=22, column=1)

h23.grid(row=23, column=1)

Button(mtr, text='Predict', command=show\_entry\_fields).grid()

mainloop()