1. %matplotlib inline

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

1. data=pd.read\_csv("emotion.csv")
2. print(data.shape)
3. data.head()
4. data
5. data.isnull().sum()
6. data.dtypes
7. data['SYSTOLIC\_PRESSURE'].mean()
8. data['DIASTOLIC\_PRESSURE'].mean()
9. data['HEART\_RATE'].mean()
10. data['BLOOD\_VISCOSITY'].mean()
11. data['SYSTOLIC\_PRESSURE']=data['SYSTOLIC\_PRESSURE'].fillna(data['SYSTOLIC\_PRESSURE'].mean())
12. data['DIASTOLIC\_PRESSURE']=data['DIASTOLIC\_PRESSURE'].fillna(data['DIASTOLIC\_PRESSURE'].mean())
13. data['BLOOD\_VISCOSITY']=data['BLOOD\_VISCOSITY'].fillna(data['BLOOD\_VISCOSITY'].mean())
14. data['HEART\_RATE']=data['HEART\_RATE'].fillna(data['HEART\_RATE'].mean())
15. data.isnull().sum()
16. data.columns
17. from sklearn.preprocessing import LabelEncoder

cat\_var =data.dtypes.loc[data.dtypes == 'object'].index

le =LabelEncoder()

for var in cat\_var:

data[var] = le.fit\_transform(data[var])

1. data
2. x=data[['SYSTOLIC\_PRESSURE', 'DIASTOLIC\_PRESSURE', 'HEART\_RATE','BLOOD\_VISCOSITY']]

y=data[['EMOTION']]

1. from sklearn.model\_selection import train\_test\_split
2. from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.05)

1. y\_train
2. y\_test
3. from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

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

1. model.predict(x\_train)
2. predicted\_train = model.predict(x\_train)

true\_value = y\_train

1. predicted\_train
2. from sklearn.metrics import accuracy\_score

print("Train Accuracy {:.2%}".format(accuracy\_score(true\_value,predicted\_train)))

1. from sklearn.metrics import confusion\_matrix,classification\_report

print("Classification Report")

print (classification\_report(true\_value,predicted\_train))

print ("Confusion matrix")

print (confusion\_matrix(true\_value,predicted\_train))

1. r2\_score=model.score(x\_train,y\_train)
2. print(r2\_score)
3. from sklearn import metrics

print(metrics.mean\_absolute\_error(true\_value,predicted\_train))

print(metrics.mean\_squared\_error(true\_value,predicted\_train))

1. print(np.sqrt(metrics.mean\_squared\_error(true\_value,predicted\_train)))
2. plt.plot(true\_value,predicted\_train)