|  |  |
| --- | --- |
| **EX.NO:5 Bagging in Classification** | |
| IN [1] | **import** pandas **as** pd  **from** sklearn.neighbors **import** KNeighborsClassifier |
| IN [2] | df **=** pd**.**read\_csv("D://KDD\_PP.csv") |
| IN [3] | df |
| IN [4] | x **=** df**.**iloc[:,:37]  x |
| IN [5] | y **=** df**.**iloc[:,**-**1:]  y |
| IN [6] | **from** sklearn.model\_selection **import** train\_test\_split  x\_train,x\_test,y\_train,y\_test **=** train\_test\_split(x,y,test\_size**=**0.3) |
| IN [7] | **from** sklearn.naive\_bayes **import** GaussianNB  gnb **=** GaussianNB()  gnb**.**fit(x\_train, y\_train)  y\_pred **=** gnb**.**predict(x\_train) |
| IN [8] | y\_pred |
| IN [9] | **from** sklearn **import** metrics  mse\_NB **=** metrics**.**mean\_squared\_error(y\_train, y\_pred)  print("Mean squared error when only applying Naive Bayes :- " **+** str(mse\_NB)) |
| IN [10] | x\_train['NB\_Pred'] **=** y\_pred |
| IN [11] | x\_train |
| IN [12] | y\_pred\_test **=** gnb**.**predict(x\_test)  x\_test['NB\_Pred'] **=** y\_pred\_test  x\_test |
| IN [13] | **from** sklearn.linear\_model **import** LogisticRegression  lr **=** LogisticRegression(multi\_class**=**'ovr')  lr**.**fit(x\_train, y\_train)  y\_pred\_log **=** lr**.**predict\_proba(x\_train)**.**argmax(axis**=**1)  y\_pred\_log |
| IN [14] | **from** sklearn **import** metrics  mse\_NB **=** metrics**.**mean\_squared\_error(y\_train, y\_pred\_log)  print("Mean squared error when applying 2 level NB + Logistic ensemble :- " **+** str(mse\_NB)) |
| IN [15] | lr **=** LogisticRegression(multi\_class**=**'ovr')  gnb **=** GaussianNB()  knn **=** KNeighborsClassifier(n\_neighbors**=**3)  lr**.**fit(x\_train,y\_train)  gnb**.**fit(x\_train,y\_train)  knn**.**fit(x\_train,y\_train)  y\_pred\_log **=** lr**.**predict\_proba(x\_train)**.**argmax(axis**=**1)  y\_pred\_NB **=** gnb**.**predict(x\_train)  y\_pred\_knn **=** knn**.**predict(x\_train) |
| IN [16] | final\_votes **=** []  **for** i **in** range(len(y\_pred)):  sum\_class **=** y\_pred\_log[i] **+** y\_pred\_NB[i] **+** y\_pred\_knn[i]  **if** sum\_class **>=**2:  final\_votes**.**append(1)  **else**:  final\_votes**.**append(0)  mse\_vote **=** metrics**.**mean\_squared\_error(y\_train, final\_votes)  mse\_vote |
| IN [17] | **from** sklearn.ensemble **import** VotingClassifier  lr **=** LogisticRegression(multi\_class**=**'ovr')  gnb **=** GaussianNB()  knn **=** KNeighborsClassifier(n\_neighbors**=**3)  models **=** [('lr',lr),('gnb',gnb),('knn',knn)]  ensemble **=** VotingClassifier(estimators**=**models, voting**=**' |
| IN [18] | ensemble **=** ensemble**.**fit(x\_train, y\_train)  y\_pred\_ens **=** ensemble**.**predict(x\_train) |
| IN [19] | mse\_ens **=** metrics**.**mean\_squared\_error(y\_train, y\_pred\_ens)  mse\_ens |
| IN [20] | df\_NB **=** df**.**sample(frac **=** 1)  df\_log **=** df**.**sample(frac **=** 1)  df\_knn **=** df**.**sample(frac **=** 1) |
| IN [21] | x **=** df\_NB**.**iloc[:,:37]  y **=** df\_NB**.**iloc[:,**-**1:]  **from** sklearn.model\_selection **import** train\_test\_split  x\_train\_NB,x\_test\_NB,y\_train\_NB,y\_test\_NB **=** train\_test\_split(x,y,test\_size**=**0.3) |
| IN [22] | x **=** df\_log**.**iloc[:,:37]  y **=** df\_log**.**iloc[:,**-**1:]  **from** sklearn.model\_selection **import** train\_test\_split  x\_train\_log,x\_test\_log,y\_train\_log,y\_test\_log **=** train\_test\_split(x,y,test\_size**=**0.3) |
| IN [23] | x **=** df\_knn**.**iloc[:,:37]  y **=** df\_knn**.**iloc[:,**-**1:]  **from** sklearn.model\_selection **import** train\_test\_split  x\_train\_knn,x\_test\_knn,y\_train\_knn,y\_test\_knn **=** train\_test\_split(x,y,test\_size**=**0.3) |
| IN [24] | lr **=** LogisticRegression(multi\_class**=**'ovr')  gnb **=** GaussianNB()  knn **=** KNeighborsClassifier(n\_neighbors**=**3)  lr**.**fit(x\_train\_log,y\_train\_log)  gnb**.**fit(x\_train\_NB,y\_train\_NB)  knn**.**fit(x\_train\_knn,y\_train\_knn) |
| IN [25] | y\_pred\_log **=** lr**.**predict\_proba(x\_train\_log)**.**argmax(axis**=**1)  y\_pred\_NB **=** gnb**.**predict(x\_train\_NB)  y\_pred\_knn **=** knn**.**predict(x\_train\_knn) |
| IN [26] | final\_votes **=** []  **for** i **in** range(len(y\_pred)):  sum\_class **=** y\_pred\_log[i] **+** y\_pred\_NB[i] **+** y\_pred\_knn[i]  **if** sum\_class **>=**2:  final\_votes**.**append(1)  **else**:  final\_votes**.**append(0)  mse\_vote **=** metrics**.**mean\_squared\_error(y\_train, final\_votes)  mse\_vote |