In [1]: import numpy as np import pandas as pd In [2]: | dataset=pd.read csv('DataB.csv') X=dataset.iloc[:,:-1].values y=dataset.iloc[:,-1].values In [3]: X=pd.DataFrame(X) In [4]: X Out[4]: 1 2 3 0 **0** 5.1 3.5 1.4 0.2 **1** 4.9 3.0 1.4 0.2 **2** 4.7 3.2 1.3 0.2 **3** 4.6 3.1 1.5 0.2 **4** 5.0 3.6 1.4 0.2 **145** 6.7 3.0 5.2 2.3 **146** 6.3 2.5 5.0 1.9 **147** 6.5 3.0 5.2 2.0 **148** 6.2 3.4 5.4 2.3 **149** 5.9 3.0 5.1 1.8 150 rows × 4 columns from sklearn.preprocessing import MinMaxScaler In [5]: MMC=MinMaxScaler() X=MMC.fit transform(X) In [6]: X [0.16666667, 0.41666667, 0.06779661, 0.04166667],[0.11111111, 0.5 , 0.05084746, 0.04166667], [0.08333333, 0.45833333, 0.08474576, 0.04166667],[0.19444444, 0.66666667, 0.06779661, 0.04166667],[0.30555556, 0.79166667, 0.11864407, 0.125 [0.08333333, 0.58333333, 0.06779661, 0.08333333],[0.19444444, 0.58333333, 0.08474576, 0.04166667],[0.02777778, 0.375 , 0.06779661, 0.04166667], [0.16666667, 0.45833333, 0.08474576, 0. [0.30555556, 0.70833333, 0.08474576, 0.04166667],[0.13888889, 0.58333333, 0.10169492, 0.04166667],[0.13888889, 0.41666667, 0.06779661, 0.], , 0.41666667, 0.01694915, 0. [0.41666667, 0.83333333, 0.03389831, 0.04166667],[0.38888889, 1. , 0.08474576, 0.125 [0.30555556, 0.79166667, 0.05084746, 0.125 [0.22222222, 0.625 , 0.06779661, 0.08333333], , 0.11864407, 0.08333333], [0.38888889, 0.75 [0.2222222, 0.75 , 0.08474576, 0.08333333], [0.30555556, 0.58333333, 0.11864407, 0.04166667],[0.22222222, 0.70833333, 0.08474576, 0.125 [0.08333333, 0.66666667, 0. , 0.04166667], [0.22222222, 0.54166667, 0.11864407, 0.16666667],[0.13888889, 0.58333333, 0.15254237, 0.04166667],[0.19444444, 0.41666667, 0.10169492, 0.04166667],[0.19444444, 0.58333333, 0.10169492, 0.125 [0.25 , 0.625 , 0.08474576, 0.04166667], [0.25 , 0.58333333, 0.06779661, 0.04166667], [0.11111111, 0.5 , 0.10169492, 0.04166667], [0.13888889, 0.45833333, 0.10169492, 0.04166667],[0.30555556, 0.58333333, 0.08474576, 0.125 , 0.875 , 0.08474576, 0.], [0.25 [0.33333333, 0.91666667, 0.06779661, 0.04166667], [0.16666667, 0.45833333, 0.08474576, 0.], [0.19444444, 0.5 , 0.03389831, 0.04166667], [0.33333333, 0.625 , 0.05084746, 0.04166667], [0.16666667, 0.45833333, 0.08474576, 0.], [0.02777778, 0.41666667, 0.05084746, 0.04166667],[0.22222222, 0.58333333, 0.08474576, 0.04166667],[0.19444444, 0.625 , 0.05084746, 0.08333333], [0.05555556, 0.125 , 0.05084746, 0.08333333], [0.02777778, 0.5 , 0.05084746, 0.04166667], [0.19444444, 0.625 , 0.10169492, 0.20833333], [0.22222222, 0.75 , 0.15254237, 0.125], [0.13888889, 0.41666667, 0.06779661, 0.08333333],[0.22222222, 0.75 , 0.10169492, 0.04166667], [0.08333333, 0.5 , 0.06779661, 0.04166667], [0.27777778, 0.70833333, 0.08474576, 0.04166667],[0.19444444, 0.54166667, 0.06779661, 0.04166667],[0.75 , 0.5 , 0.62711864, 0.54166667], [0.58333333, 0.5 , 0.59322034, 0.58333333], [0.72222222, 0.45833333, 0.66101695, 0.58333333],[0.33333333, 0.125 , 0.50847458, 0.5 [0.61111111, 0.33333333, 0.61016949, 0.58333333],[0.38888889, 0.33333333, 0.59322034, 0.5], [0.5555556, 0.54166667, 0.62711864, 0.625 [0.16666667, 0.16666667, 0.38983051, 0.375 [0.63888889, 0.375 , 0.61016949, 0.5 [0.25 , 0.29166667, 0.49152542, 0.54166667], [0.19444444, 0. , 0.42372881, 0.375], [0.44444444, 0.41666667, 0.54237288, 0.58333333],[0.47222222, 0.08333333, 0.50847458, 0.375], [0.5 , 0.375 , 0.62711864, 0.54166667], [0.36111111, 0.375 , 0.44067797, 0.5], [0.66666667, 0.45833333, 0.57627119, 0.54166667],[0.36111111, 0.41666667, 0.59322034, 0.58333333],[0.41666667, 0.29166667, 0.52542373, 0.375 [0.52777778, 0.08333333, 0.59322034, 0.58333333],[0.36111111, 0.20833333, 0.49152542, 0.41666667],[0.44444444, 0.5 , 0.6440678 , 0.70833333], [0.5 , 0.33333333, 0.50847458, 0.5 [0.55555556, 0.20833333, 0.66101695, 0.58333333],, 0.33333333, 0.62711864, 0.45833333], [0.58333333, 0.375 , 0.55932203, 0.5], [0.63888889, 0.41666667, 0.57627119, 0.54166667],[0.69444444, 0.33333333, 0.6440678, 0.54166667],[0.66666667, 0.41666667, 0.6779661, 0.66666667],[0.47222222, 0.375 , 0.59322034, 0.58333333], [0.38888889, 0.25 , 0.42372881, 0.375], [0.33333333, 0.16666667, 0.47457627, 0.41666667],[0.33333333, 0.16666667, 0.45762712, 0.375], [0.41666667, 0.29166667, 0.49152542, 0.45833333],[0.47222222, 0.29166667, 0.69491525, 0.625], [0.30555556, 0.41666667, 0.59322034, 0.58333333],[0.47222222, 0.58333333, 0.59322034, 0.625], [0.66666667, 0.45833333, 0.62711864, 0.58333333],[0.55555556, 0.125 , 0.57627119, 0.5 [0.36111111, 0.41666667, 0.52542373, 0.5 [0.33333333, 0.20833333, 0.50847458, 0.5 [0.33333333, 0.25, 0.57627119, 0.45833333],, 0.41666667, 0.61016949, 0.54166667], [0.5 [0.41666667, 0.25 , 0.50847458, 0.45833333], [0.19444444, 0.125 , 0.38983051, 0.375], [0.36111111, 0.29166667, 0.54237288, 0.5], [0.38888889, 0.41666667, 0.54237288, 0.45833333],[0.38888889, 0.375 , 0.54237288, 0.5], [0.52777778, 0.375 , 0.55932203, 0.5 [0.22222222, 0.20833333, 0.33898305, 0.41666667],[0.38888889, 0.33333333, 0.52542373, 0.5 [0.55555556, 0.54166667, 0.84745763, 1. [0.41666667, 0.29166667, 0.69491525, 0.75 [0.77777778, 0.41666667, 0.83050847, 0.83333333], [0.55555556, 0.375, 0.77966102, 0.70833333],[0.61111111, 0.41666667, 0.81355932, 0.875 [0.91666667, 0.41666667, 0.94915254, 0.83333333],[0.16666667, 0.20833333, 0.59322034, 0.66666667], [0.83333333, 0.375 , 0.89830508, 0.70833333], [0.66666667, 0.20833333, 0.81355932, 0.70833333],[0.80555556, 0.66666667, 0.86440678, 1. [0.61111111, 0.5 , 0.69491525, 0.79166667], [0.58333333, 0.29166667, 0.72881356, 0.75 [0.69444444, 0.41666667, 0.76271186, 0.83333333], [0.38888889, 0.20833333, 0.6779661, 0.79166667],[0.41666667, 0.33333333, 0.69491525, 0.95833333],[0.58333333, 0.5 , 0.72881356, 0.91666667], [0.61111111, 0.41666667, 0.76271186, 0.70833333],[0.94444444, 0.75 , 0.96610169, 0.875 [0.94444444, 0.25 , 1. , 0.91666667], [0.47222222, 0.08333333, 0.6779661, 0.58333333],[0.72222222, 0.5 , 0.79661017, 0.91666667], [0.36111111, 0.33333333, 0.66101695, 0.79166667],[0.94444444, 0.33333333, 0.96610169, 0.79166667],[0.55555556, 0.29166667, 0.66101695, 0.70833333],[0.66666667, 0.54166667, 0.79661017, 0.83333333],[0.80555556, 0.5 , 0.84745763, 0.70833333], [0.52777778, 0.33333333, 0.6440678, 0.70833333],, 0.41666667, 0.66101695, 0.70833333], [0.58333333, 0.33333333, 0.77966102, 0.83333333],[0.80555556, 0.41666667, 0.81355932, 0.625 [0.86111111, 0.33333333, 0.86440678, 0.75 , 0.75 [1. , 0.91525424, 0.79166667], [0.58333333, 0.33333333, 0.77966102, 0.875 [0.55555556, 0.33333333, 0.69491525, 0.58333333],, 0.25 , 0.77966102, 0.54166667], [0.94444444, 0.41666667, 0.86440678, 0.91666667],[0.55555556, 0.58333333, 0.77966102, 0.95833333],[0.58333333, 0.45833333, 0.76271186, 0.70833333],[0.47222222, 0.41666667, 0.6440678, 0.70833333],[0.72222222, 0.45833333, 0.74576271, 0.83333333],[0.66666667, 0.45833333, 0.77966102, 0.95833333],[0.72222222, 0.45833333, 0.69491525, 0.91666667],[0.41666667, 0.29166667, 0.69491525, 0.75 [0.69444444, 0.5 , 0.83050847, 0.91666667], [0.66666667, 0.54166667, 0.79661017, 1.], [0.66666667, 0.41666667, 0.71186441, 0.91666667],[0.5555556, 0.20833333, 0.6779661 , 0.75], [0.611111111, 0.41666667, 0.71186441, 0.79166667],[0.52777778, 0.58333333, 0.74576271, 0.91666667],[0.44444444, 0.41666667, 0.69491525, 0.70833333]]) In [7]: y Out[7]: array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica'], dtype=object) In [8]: from sklearn.preprocessing import LabelEncoder le=LabelEncoder() y=le.fit transform(y) In [9]: # y=pd.DataFrame(y) In [10]: y 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, In [11]: y=y.ravel() In [12]: y 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, In [13]: 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, shuffle=True, random state=42) from sklearn.svm import LinearSVC from sklearn.multiclass import OneVsRestClassifier Lin=LinearSVC(multi class='ovr', random state=42) # Lin=Lin.fit(X_train,y_train) In [15]: from sklearn.multiclass import OneVsOneClassifier LIN=LinearSVC(random_state=0) In [16]: # Lin.predict(X_test) In [17]: | ovr=OneVsRestClassifier(Lin) In [18]: Ono=OneVsOneClassifier(LIN) In [19]: ovr.fit(X train, y train) Out[19]: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit intercept=True, intercept scaling=1, loss='squared_hinge', max_iter=1000, multi class='ovr', penalty='12', random state=42, tol=0.0001, verbose=0), n_jobs=None) In [20]: Ono.fit(X_train, y_train) Out[20]: OneVsOneClassifier(estimator=LinearSVC(C=1.0, class weight=None, dual=True, fit intercept=True, intercept scaling=1, loss='squared hinge', max iter=1000, multi class='ovr', penalty='12', random state=0, tol=0.0001, verbose=0), n jobs=None) In [21]: y pred=ovr.predict(X test) In [22]: ypred=Ono.predict(X_test) In [23]: y_pred Out[23]: array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 2, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 2, 0, 0, 0, 2, 2, 2, 0, 0]) In [24]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score ACC_Ovr=accuracy_score(y_test, y_pred) Precis=precision_score(y_pred, y_test, average='micro') Recall=recall_score(y_pred, y_test, average='micro') F score=f1 score(y pred, y test, average='micro') In [25]: ACC Ovr Out[25]: 0.9111111111111111 In [26]: Precis Out[26]: 0.9111111111111111 In [27]: Recall Out[27]: 0.911111111111111 In [28]: F score Out[28]: 0.9111111111111111 In [29]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score ACC_Ono=accuracy_score(ypred, y_test) Pren=precision_score(ypred, y_test,average='micro') Rec=recall_score(ypred, y_test,average='micro') Fscore=f1_score(ypred, y_test,average='micro') In [30]: ACC Ono Out[30]: 1.0 In [31]: Pren Out[31]: 1.0 In [32]: Rec Out[32]: 1.0 In [33]: Fscore Out[33]: 1.0 In [34]: from sklearn.tree import DecisionTreeClassifier DEC=DecisionTreeClassifier(random state=0) DEC.fit(X_train, y_train) Out[34]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min impurity decrease=0.0, min impurity split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random state=0, splitter='best') In [35]: y pred=DEC.predict(X test) In [36]: from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score ACC_DEC=accuracy_score(ypred, y_test) Precis_DEC=precision_score(y_pred, y_test,average='micro') Recall_DEC=recall_score(y_pred, y_test,average='micro') F_score_DEC=f1_score(y_pred, y_test,average='micro') In [37]: ACC_DEC Out[37]: 1.0 In [38]: Precis_DEC Out[38]: 1.0 In [39]: Recall_DEC Out[39]: 1.0 F_score_DEC In [40]: Out[40]: 1.0 In []: