**LOGISTIC REGRESSION**

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn **import** preprocessing

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.model\_selection **import** KFold

col\_names**=**['x1','x2','result']

ds**=**pd**.**read\_csv(r'/home/inspiron/Downloads/stduni.csv',header**=None**,names**=**col\_names)

print(ds)

*# input*

x **=** ds**.**iloc[:, [0, 1]]**.**values

*# output*

y **=** ds**.**iloc[:, 2]**.**values

xp**=**preprocessing**.**scale(x)

kf**=**KFold(n\_splits**=**5)

**for** train\_index, test\_index **in** kf**.**split(xp):

xtrain, xtest, ytrain, ytest **=** train\_test\_split(xp, y, test\_size **=** 0.20, random\_state **=** 0)

x1**=**xtrain[:,0]

x2**=**xtrain[:,1]

b0**=**0.0

b1**=**0.0

b2**=**0.0

epoch**=**10000

alpha**=**0.001

**while**(epoch**>**0):

**for** i **in** range(len(xtrain)):

prediction**=**1**/**(1**+**np**.**exp(**-**(b0**+**b1**\***x1[i]**+**b2**\***x2[i])))

b0**=**b0**+**alpha**\***(ytrain[i]**-**prediction)**\***prediction**\***(1**-**prediction)**\***1.0

b1**=**b1**+**alpha**\***(ytrain[i]**-**prediction)**\***prediction**\***(1**-**prediction)**\***x1[i]

b2**=**b2**+**alpha**\***(ytrain[i]**-**prediction)**\***prediction**\***(1**-**prediction)**\***x2[i]

epoch**=**epoch**-**1

print(b0)

print(b1)

print(b2)

final\_prediction**=**[]

x3**=**xtest[:,0]

x4**=**xtest[:,1]

print(ytest)

y\_pred**=**[0]**\***len(xtest)

**for** i **in** range(len(xtest)):

y\_pred[i]**=**np**.**round(1**/**(1**+**np**.**exp(**-**(b0**+**b1**\***x3[i]**+**b2**\***x4[i]))))

final\_prediction**.**append(np**.**ceil(y\_pred[i]))

print(final\_prediction)

**from** sklearn.metrics **import** accuracy\_score

print("Accuracy",accuracy\_score(ytest,y\_pred))

**PCA**

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.preprocessing **import** StandardScaler

df**=**pd**.**read\_csv('iris.csv')

df**.**head()

X **=** df**.**drop(['species'],axis**=**1)

X\_scaled **=** StandardScaler()**.**fit\_transform(X)

X\_scaled[:5]

y**=**df['species']

features **=** X\_scaled**.**T

cov\_matrix **=** np**.**cov(features)

cov\_matrix[:5]

values, vectors **=** np**.**linalg**.**eig(cov\_matrix)

values[:5]

vectors[:5]

explained\_variances **=** []

**for** i **in** range(len(values)):

explained\_variances**.**append((values[i] **/** np**.**sum(values))**\***100)

print("variances of each feature",explained\_variances)

plt**.**figure(figsize**=**(8,4))

plt**.**bar(range(4),explained\_variances, alpha**=**0.6)

plt**.**ylabel('Percentage of explained variance')

plt**.**xlabel('Dimensions')

projected\_1 **=** X\_scaled**.**dot(vectors**.**T[0])

projected\_2 **=** X\_scaled**.**dot(vectors**.**T[1])

res **=** pd**.**DataFrame(projected\_1, columns**=**['PC1'])

res['PC2'] **=** projected\_2

res['Y'] **=** y

res**.**head()

sns**.**FacetGrid(res, hue**=**"Y", height**=**6)**.**map(plt**.**scatter, 'PC1', 'PC2')**.**add\_legend()

plt**.**show()

**NAÏVE BAYES**

**import** pandas **as** pd

**import** numpy **as** np

data **=** pd**.**read\_csv('covid.csv')

data

**from** sklearn **import** preprocessing

le **=** preprocessing**.**LabelEncoder()

pc\_encoded**=**le**.**fit\_transform(data['pc']**.**values)

wbc\_encoded**=**le**.**fit\_transform(data['wbc']**.**values)

mc\_encoded**=**le**.**fit\_transform(data['mc']**.**values)

ast\_encoded**=**le**.**fit\_transform(data['ast']**.**values)

bc\_encoded**=**le**.**fit\_transform(data['bc']**.**values)

ldh\_encoded**=**le**.**fit\_transform(data['ldh']**.**values)

Y**=**le**.**fit\_transform(data['diagnosis']**.**values)

X**=**np**.**array(list(zip(pc\_encoded,wbc\_encoded,mc\_encoded,ast\_encoded,bc\_encoded,ldh\_encoded)))

X

Y

**from** sklearn.naive\_bayes **import** MultinomialNB

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.metrics **import** classification\_report

model **=** MultinomialNB()

**from** sklearn.model\_selection **import** train\_test\_split

X\_train,X\_test,Y\_train,Y\_test**=**train\_test\_split(X,Y)

model**.**fit(X\_train, Y\_train)

y\_pred **=** model**.**predict(X\_test)

print("Accuracy:",accuracy\_score(Y\_test, y\_pred))

print("\nReport")

print(classification\_report(Y\_test,y\_pred))

**Random Forest**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

data **=** pd**.**read\_csv('pima.csv')

data**.**head()

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.datasets **import** make\_classification

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.preprocessing **import** StandardScaler, MinMaxScaler

**import** pandas\_profiling

**from** matplotlib **import** rcParams

**import** warnings

data**.**columns

X**=**data**.**drop("Outcome",axis**=**1)

y**=**data["Outcome"]

scaler**=**StandardScaler()

X\_scaled**=**scaler**.**fit\_transform(X)

X\_train,X\_test,Y\_train,Y\_test**=**train\_test\_split(X\_scaled,y,stratify**=**y,test\_size**=**0.10,random\_state**=**34)

classifier **=** RandomForestClassifier(n\_estimators**=**100)

classifier**.**fit(X\_train,Y\_train)

y\_pred **=** classifier**.**predict(X\_test)

print("Accuracy:",accuracy\_score(Y\_test,y\_pred))

feature\_importances\_df **=** pd**.**DataFrame(

{"feature":list(X**.**columns),"importance":classifier**.**feature\_importances\_}

)**.**sort\_values("importance",ascending**=False**)

feature\_importances\_df

**from** sklearn.tree **import** DecisionTreeClassifier

clf**=**DecisionTreeClassifier()

clf**.**fit(X\_train,Y\_train)

Y\_pred **=** clf**.**predict(X\_test)

**from** sklearn.metrics **import** accuracy\_score

print("Accuracy-DecisionTree :",accuracy\_score(Y\_test,Y\_pred))

**SVM**

**from** sklearn.svm **import** SVC

**from** sklearn **import** svm

**import** numpy **as** np

X**=**np**.**array([[3,4],[1,4],[2,3],[6,**-**1],[7,**-**1],[5,**-**3]])

y**=**np**.**array([**-**1,**-**1,**-**1,1,1,1])

l**=**SVC(C**=**1e5,kernel**=**'linear')

l**.**fit(X,y)

print('w= ',l**.**coef\_)

print('b= ',l**.**intercept\_)

print('Indices of support vectors= ',l**.**support\_)

print('Support vectors= ',l**.**support\_vectors\_)

print('No. of support vectors from each class= ',l**.**n\_support\_)

print('coefficient of support vectors in decision function= ',np**.**abs(l**.**dual\_coef\_))

**import** pandas **as** pd

data**=**pd**.**read\_csv('glass.csv')

data**.**head()

x**=**data**.**drop('Type',axis**=**1)

y**=**data**.**Type

**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)

linear**=**svm**.**SVC(kernel**=**'linear')

linear**.**fit(x\_train,y\_train)

print(linear**.**support\_vectors\_)

print(linear**.**n\_support\_)

y\_pred**=**linear**.**predict(x\_test)

**from** sklearn.metrics **import** accuracy\_score

print(accuracy\_score(y\_test,y\_pred))

**from** sklearn.metrics **import** confusion\_matrix

print(confusion\_matrix(y\_test,y\_pred))

**from** sklearn.metrics **import** classification\_report

print(classification\_report(y\_test,y\_pred))

model1**=**SVC(kernel**=**'sigmoid')

model2**=**SVC(kernel**=**'poly')

model3**=**SVC(kernel**=**'rbf')

model1**.**fit(x\_train,y\_train)

model2**.**fit(x\_train,y\_train)

model3**.**fit(x\_train,y\_train)

y\_pred1**=**model1**.**predict(x\_test)

y\_pred2**=**model2**.**predict(x\_test)

y\_pred3**=**model3**.**predict(x\_test)

print("prediction by model1 ",accuracy\_score(y\_test,y\_pred1))

print("prediction by model2",accuracy\_score(y\_test,y\_pred2))

print("prediction by model3",accuracy\_score(y\_test,y\_pred1))