PCA

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

data **=** pd**.**read\_csv("iris.csv")

data**.**head(5)

y **=** data["species"]

*# Input data*

X **=** data**.**drop("species", 1)

print(X[:5], "\n")

print(y[:5])

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

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

x\_scaled[:4]

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

*# Covariance Matrix*

features **=** x\_scaled**.**T

covMatrix **=** np**.**cov(features)

covMatrix

*# Eigen values and Eigen vector*

values, vectors **=** np**.**linalg**.**eig(covMatrix)

print(values, "\n")

print(vectors)

*# Variance of each feature w.r.t eigen vlaues*

explained\_variance **=** []

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

res **=** values[i]**/**np**.**sum(values)**\***100

explained\_variance**.**append(res)

print("Variance of each feature", explained\_variance)

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

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

*# Bar graph*

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

plt**.**bar(range(4), explained\_variance, alpha**=**0.8)

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

plt**.**xlabel("Dimensions")

plt**.**show()

pro\_1 **=** x\_scaled**.**dot(vectors**.**T[0])

pro\_2 **=** x\_scaled**.**dot(vectors**.**T[1])

result **=** pd**.**DataFrame(pro\_1, columns**=**["PC1"])

result["PC2"] **=** pro\_2

result["Y"] **=** y

result**.**head(10)

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

plt**.**show()

NAIVE 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))

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= ')

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

print('No. of support vectors fro 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)

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

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(accuracy\_score(y\_test,y\_pred1))

print(accuracy\_score(y\_test,y\_pred2))

print(accuracy\_score(y\_test,y\_pred3))

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')

**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

warnings**.**filterwarnings("ignore")

rcParams["figure.figsize"]**=**10,6

np**.**random**.**seed(42)

data**.**sample(5)

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**=**42)

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))

Accuracy: 0.8051948051948052

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))