**PRINCIPLE COMPONENT ANALYSIS**

In [ ]:

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

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

data**.**head(5)

Out[ ]:

|  | **sepal length** | **sepal width** | **petal length** | **petal width** | **species** |
| --- | --- | --- | --- | --- | --- |
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 1 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 1 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 1 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | 1 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | 1 |

In [ ]:

data**.**describe()

Out[ ]:

|  | **sepal length** | **sepal width** | **petal length** | **petal width** | **species** |
| --- | --- | --- | --- | --- | --- |
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.057333 | 3.758000 | 1.199333 | 2.000000 |
| **std** | 0.828066 | 0.435866 | 1.765298 | 0.762238 | 0.819232 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 | 1.000000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 | 1.000000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 | 2.000000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 | 3.000000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 | 3.000000 |

**Splitting data**

In [ ]:

*# Output data*

species **=** data["species"]**.**tolist()

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

*# Input data*

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

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

print(y[:5])

sepal length sepal width petal length petal width

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

0 1

1 1

2 1

3 1

4 1

Name: species, dtype: int64

**Normalising or Standardising the data**

In [ ]:

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

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

x\_scaled[:4]

Out[ ]:

array([[-0.90068117, 1.01900435, -1.34022653, -1.3154443 ],

[-1.14301691, -0.13197948, -1.34022653, -1.3154443 ],

[-1.38535265, 0.32841405, -1.39706395, -1.3154443 ],

[-1.50652052, 0.09821729, -1.2833891 , -1.3154443 ]])

**Calculations**

In [ ]:

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

*# Covariance Matrix*

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

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

covMatrix

Out[ ]:

array([[ 1.00671141, -0.11835884, 0.87760447, 0.82343066],

[-0.11835884, 1.00671141, -0.43131554, -0.36858315],

[ 0.87760447, -0.43131554, 1.00671141, 0.96932762],

[ 0.82343066, -0.36858315, 0.96932762, 1.00671141]])

In [ ]:

*# Eigen values and Eigen vector*

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

print(values, "\n")

print(vectors)

[2.93808505 0.9201649 0.14774182 0.02085386]

[[ 0.52106591 -0.37741762 -0.71956635 0.26128628]

[-0.26934744 -0.92329566 0.24438178 -0.12350962]

[ 0.5804131 -0.02449161 0.14212637 -0.80144925]

[ 0.56485654 -0.06694199 0.63427274 0.52359713]]

In [ ]:

*# 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)

Variance of each feature [72.9624454132999, 22.850761786701725, 3.6689218892828612, 0.5178709107154993]

**Plotting graphs**

In [ ]:

**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()

In [20]:

*# Based on the explained variances and the bar graph*

*# So we select first 2 features to be Principle Components*

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)

Out[20]:

|  | **PC1** | **PC2** | **Y** |
| --- | --- | --- | --- |
| **0** | -2.264703 | -0.480027 | 1 |
| **1** | -2.080961 | 0.674134 | 1 |
| **2** | -2.364229 | 0.341908 | 1 |
| **3** | -2.299384 | 0.597395 | 1 |
| **4** | -2.389842 | -0.646835 | 1 |
| **5** | -2.075631 | -1.489178 | 1 |
| **6** | -2.444029 | -0.047644 | 1 |
| **7** | -2.232847 | -0.223148 | 1 |
| **8** | -2.334640 | 1.115328 | 1 |
| **9** | -2.184328 | 0.469014 | 1 |

In [22]:

*# Scatter Plot*

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

plt**.**show()

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')

In [ ]:

**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

In [ ]:

**import** warnings

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

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

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

In [ ]:

data**.**sample(5)

Out[ ]:

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **668** | 6 | 98 | 58 | 33 | 190 | 34.0 | 0.430 | 43 | 0 |
| **324** | 2 | 112 | 75 | 32 | 0 | 35.7 | 0.148 | 21 | 0 |
| **624** | 2 | 108 | 64 | 0 | 0 | 30.8 | 0.158 | 21 | 0 |
| **690** | 8 | 107 | 80 | 0 | 0 | 24.6 | 0.856 | 34 | 0 |
| **473** | 7 | 136 | 90 | 0 | 0 | 29.9 | 0.210 | 50 | 0 |

In [ ]:

data**.**columns

Out[ ]:

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],

dtype='object')

In [ ]:

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

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

In [ ]:

scaler**=**StandardScaler()

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

In [ ]:

X\_train,X\_test,Y\_train,Y\_test**=**train\_test\_split(X\_scaled,y,stratify**=**y,test\_size**=**0.10,random\_state**=**42)

In [ ]:

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

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

Out[ ]:

RandomForestClassifier()

In [ ]:

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

In [ ]:

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

Accuracy: 0.8051948051948052

In [ ]:

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

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

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

feature\_importances\_df

Out[ ]:

|  | **feature** | **importance** |
| --- | --- | --- |
| **1** | Glucose | 0.265153 |
| **5** | BMI | 0.152950 |
| **7** | Age | 0.142551 |
| **6** | DiabetesPedigreeFunction | 0.120932 |
| **2** | BloodPressure | 0.083460 |
| **0** | Pregnancies | 0.082878 |
| **4** | Insulin | 0.078441 |
| **3** | SkinThickness | 0.073634 |

In [ ]:

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

clf**=**DecisionTreeClassifier()

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

Out[ ]:

DecisionTreeClassifier()

In [ ]:

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

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

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

Accuracy-DecisionTree : 0.7532467532467533

**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\_))

w = [[ 0.25 -0.25]]

b = [-0.75]

Indices of support vectors= [2 3]

Support vectors=

[[ 2. 3.]

[ 6. -1.]]

No. of support vectors fro each class= [1 1]

coefficient of support vectors in decision function= [[0.0625 0.0625]]

In [80]:

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

data**=**pd**.**read\_csv('Lab 08 glass.csv')

data**.**head()

Out[80]:

|  | **Id** | **RI** | **Na** | **Mg** | **Al** | **Si** | **K** | **Ca** | **Ba** | **Fe** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 1.52101 | 13.64 | 4.49 | 1.10 | 71.78 | 0.06 | 8.75 | 0.0 | 0.0 | 1 |
| **1** | 2 | 1.51761 | 13.89 | 3.60 | 1.36 | 72.73 | 0.48 | 7.83 | 0.0 | 0.0 | 1 |
| **2** | 3 | 1.51618 | 13.53 | 3.55 | 1.54 | 72.99 | 0.39 | 7.78 | 0.0 | 0.0 | 1 |
| **3** | 4 | 1.51766 | 13.21 | 3.69 | 1.29 | 72.61 | 0.57 | 8.22 | 0.0 | 0.0 | 1 |
| **4** | 5 | 1.51742 | 13.27 | 3.62 | 1.24 | 73.08 | 0.55 | 8.07 | 0.0 | 0.0 | 1 |

In [81]:

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

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

In [82]:

**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 [83]:

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

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

Out[83]:

SVC(kernel='linear')

In [84]:

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

[[7.00000e+01 1.52300e+00 1.33100e+01 3.58000e+00 8.20000e-01 7.19900e+01

1.20000e-01 1.01700e+01 0.00000e+00 3.00000e-02]

[1.46000e+02 1.51839e+00 1.28500e+01 3.67000e+00 1.24000e+00 7.25700e+01

6.20000e-01 8.68000e+00 0.00000e+00 3.50000e-01]

[7.20000e+01 1.51848e+00 1.36400e+01 3.87000e+00 1.27000e+00 7.19600e+01

5.40000e-01 8.32000e+00 0.00000e+00 3.20000e-01]

[1.47000e+02 1.51769e+00 1.36500e+01 3.66000e+00 1.11000e+00 7.27700e+01

1.10000e-01 8.60000e+00 0.00000e+00 0.00000e+00]

[1.62000e+02 1.51934e+00 1.36400e+01 3.54000e+00 7.50000e-01 7.26500e+01

1.60000e-01 8.89000e+00 1.50000e-01 2.40000e-01]

[1.63000e+02 1.52211e+00 1.41900e+01 3.78000e+00 9.10000e-01 7.13600e+01

2.30000e-01 9.14000e+00 0.00000e+00 3.70000e-01]

[1.76000e+02 1.52119e+00 1.29700e+01 3.30000e-01 1.51000e+00 7.33900e+01

1.30000e-01 1.12700e+01 0.00000e+00 2.80000e-01]

[1.65000e+02 1.51915e+00 1.27300e+01 1.85000e+00 1.86000e+00 7.26900e+01

6.00000e-01 1.00900e+01 0.00000e+00 0.00000e+00]

[1.75000e+02 1.52058e+00 1.28500e+01 1.61000e+00 2.17000e+00 7.21800e+01

7.60000e-01 9.70000e+00 2.40000e-01 5.10000e-01]

[1.84000e+02 1.51969e+00 1.45600e+01 0.00000e+00 5.60000e-01 7.34800e+01

0.00000e+00 1.12200e+01 0.00000e+00 0.00000e+00]

[1.85000e+02 1.51115e+00 1.73800e+01 0.00000e+00 3.40000e-01 7.54100e+01

0.00000e+00 6.65000e+00 0.00000e+00 0.00000e+00]

[1.81000e+02 1.51299e+00 1.44000e+01 1.74000e+00 1.54000e+00 7.45500e+01

0.00000e+00 7.59000e+00 0.00000e+00 0.00000e+00]

[1.77000e+02 1.51905e+00 1.40000e+01 2.39000e+00 1.56000e+00 7.23700e+01

0.00000e+00 9.57000e+00 0.00000e+00 0.00000e+00]

[1.88000e+02 1.52315e+00 1.34400e+01 3.34000e+00 1.23000e+00 7.23800e+01

6.00000e-01 8.83000e+00 0.00000e+00 0.00000e+00]

[1.86000e+02 1.51131e+00 1.36900e+01 3.20000e+00 1.81000e+00 7.28100e+01

1.76000e+00 5.43000e+00 1.19000e+00 0.00000e+00]]

In [85]:

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

[1 2 3 3 4 2]

In [86]:

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

In [87]:

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

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

0.9846153846153847

In [88]:

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

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

[[25 0 0 0 0 0]

[ 0 26 0 0 0 0]

[ 0 0 3 0 0 0]

[ 0 0 1 0 0 0]

[ 0 0 0 0 2 0]

[ 0 0 0 0 0 8]]

In [89]:

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

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

precision recall f1-score support

1 1.00 1.00 1.00 25

2 1.00 1.00 1.00 26

3 0.75 1.00 0.86 3

5 0.00 0.00 0.00 1

6 1.00 1.00 1.00 2

7 1.00 1.00 1.00 8

accuracy 0.98 65

macro avg 0.79 0.83 0.81 65

weighted avg 0.97 0.98 0.98 65

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

In [90]:

*#with different kernels*

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

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

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

In [91]:

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

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

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

Out[91]:

SVC()

In [92]:

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

In [93]:

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

In [94]:

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

In [95]:

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

0.7384615384615385

In [96]:

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

0.9538461538461539

In [97]:

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

0.9230769230769231