

**MACHINE LEARNING
(UCS611)**

IONOSPHERE CLASSIFICATION

ASSIGNMENT - 3

Submitted by: -

Leeza 101715078

Naman Jain 101715093

Submitted to: -

Dr. Surbhi Sharma



**THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)**

Department of Computer Science and Engineering
Thapar Institute of Engineering and Technology

Patiala (Punjab) -147004
January – May 2020

TABLE OF CONTENTS

1. Objective.....	1
2. Software specifications.....	1
3. Theory.....	1
4. Model snapshots and Visualization Graphs.....	2
a. KNN Classifier.....	2
(i) Test-size: 30%	3
(ii) Test-size: 40%.....	7
(iii) Test-size: 50%.....	10
b. Decision Tree Classifier.....	13
(i) Test-size: 30%.....	15
(ii) Test-size: 40%.....	18
(iii) Test-size: 50%.....	22
c. Random Forest Classifier.....	25
(i) Test-size: 30%.....	27
(ii) Test-size: 40%.....	30
(iii) Test-size: 50%.....	34
5. Observations.....	37
6. Important points.....	37
7. References.....	38

OBJECTIVE: - Compare the training and testing phase performances of three classifiers on Ionosphere dataset.

Classifiers to be used: 1. KNN Classifier. 2. Decision Tree Classifier. 3. Random Forest Classifier

1. Performance with any five values of K and test ratio 30%, 40% and 50% for KNN.
2. Performance with any five values of max_depth and test ratio 30%, 40% and 50% for Decision Trees.
3. Performance with any five values of n_estimators and test ratio 30%, 40% and 50% for Random Forest.

SOFTWARE SPECIFICATIONS: -

1. Language: - Python
2. Coding Environment: -Anaconda 3(Spyder 4.0)
3. Packages: - Numpy, Matplotlib, Pandas.
4. ML model used: - K-Nearest Neighbours Classifier, Decision Tree Classifier, Random Forest classifier.

THEORY: -

1. **K-Nearest Neighbours Classifier:** - In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbour.
2. **Decision Tree Classifier:** - Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
3. **Random Forest classifier:** - Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.^{[1][2]} Random decision forests correct for decision trees' habit of overfitting to their training set.

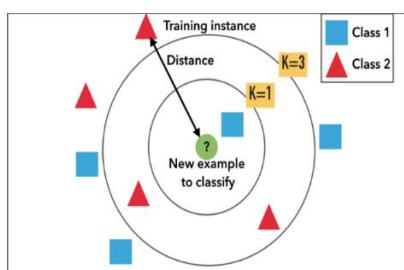


Fig. 1.a KNN Classifier

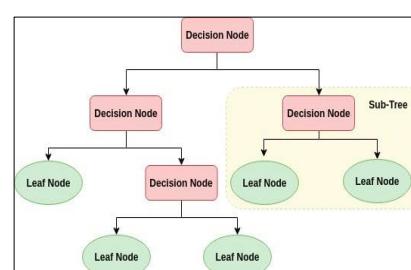


Fig. 1.b Decision Tree Classifier

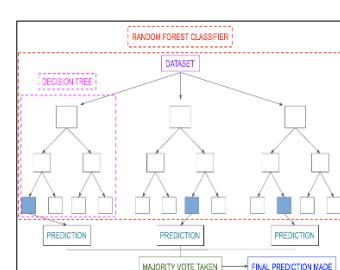


Fig. 1.c Random Forest Classifier

Models Snapshots and Graphs: -

1. K-Nearest Neighbours Classifier: -

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('ionosphere.csv', header = None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,
random_state = 0)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 1, metric = 'minkowski', p
= 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```

Using the label encoder:

- **Good 'g' output is encoded as 1**
- **Bad 'b' output is encoded as 0**

```

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('KNN (Training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('KNN (Test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

```

a. Test-size: - 30%

➤ K-value: - 3

Accuracy: - 77.35%

The screenshot shows a Jupyter Notebook interface with three main panes:

- Code Editor:** The file `temp.py` contains the Python code provided above.
- Variable Explorer:** A table showing the variables used in the analysis. Key entries include:

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[26 18] [6 56]]
X	Array of float64	(351, 34)	[[1 ... 0. 0.99...]
X1	Array of float64	(776, 821)	[[-3.14221767 -3.13221767 ...]
X2	Array of float64	(776, 821)	[[-3.08534372 ...]
X_set	Array of float64	(106, 2)	[[-3.43546095 -1.64864671]
X_test	Array of float64	(106, 2)	[[-0.3003702 -2.08534372]
X_train	Array of float64	(245, 2)	[[-3.10637101 -2.08534372]
accuracy	float64	1	0.7735849056683774
classifier	neighbors._classification.KNeighborsClassifier	1	KNeighborsClassifier object
dataset	DataFrame	(351, 35)	Column names: 0, 1, 2, 3, ...
explained_variance	Array of float64	(2,)	[0.31906397 0.13082327]
- Console:** The output of the code execution, showing the confusion matrix and accuracy score.

```

In [4]:
.... # Fitting K-NN to the Training set
.... from sklearn.neighbors import KNeighborsClassifier
.... classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
.... classifier.fit(X_train, y_train)

.... # Predicting the Test set results
.... y_pred = classifier.predict(X_test)

.... # Making the Confusion Matrix
.... from sklearn.metrics import confusion_matrix, accuracy_score
.... Confusion_matrix = confusion_matrix(y_test, y_pred)
.... accuracy = accuracy_score(y_test, y_pred)
.... print(accuracy)
0.7735849056683774

```

Fig. 1.1.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

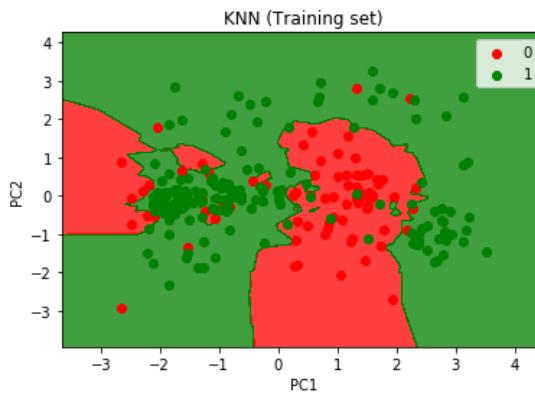


Fig. 1.1.1.b Training Set Graph

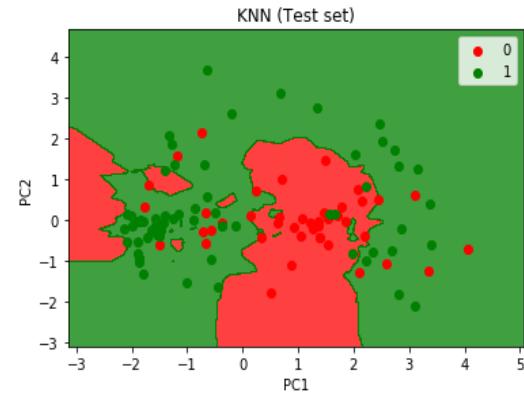


Fig. 1.1.1.c Test Set Graph

➤ K-value: - 4

Accuracy: - 78.30%

```

C:\Users\leezo\Documents\ML Projects\ionosphere\knn.py
temp.py x knnPy ...
9 # K-Nearest Neighbors (K-NN)
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-NN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 4, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49

```

Confusion_matrix - NumPy ...

0	1
0	29
1	8
	37

Name Type Size Value

Confusion_matrix Array of int64 (2, 2) [[29 15] [8 34]]

X Array of float64 (351, 34) [[1. ... [-3.14221767 -3.13221767 ...]]]

X1 Array of float64 (776, 821) [[1. ... [-3.14221767 -3.13221767 ...]]]

X2 Array of float64 (776, 821) [[1. ... [-3.14221767 -3.13221767 ...]]]

X_set Array of float64 (106, 2) [[-0.43544696 -1.64886471] [3.10037101 -2.08543727] ...]

X_test Array of float64 (106, 2) [[-0.43544696 -1.64886471] [3.10037101 -2.08543727] ...]

X_train Array of float64 (245, 2) [[1.357563 -0.00007543] [2.78877995 -1.54053569] ...]

accuracy float64 1 0.7830188679245284

classifier neighbors.classification.KNeighborsClassifier 1 KNeighborsClassifier object

dataset DataFrame (351, 35) Column names: 0, 1, 2, 3, ...

explained_variance Array of float64 (2,) [0.31906397 0.13042327]

In [6]:

Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 4, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

Predicting the Test set results
y_pred = classifier.predict(X_test)

Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
0.7830188679245284

Fig. 1.1.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

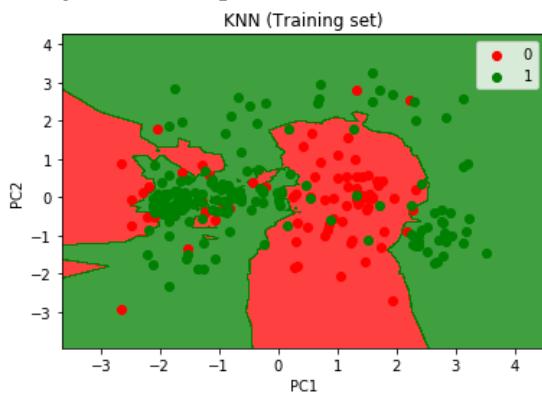


Fig. 1.1.2.b Training Set Graph

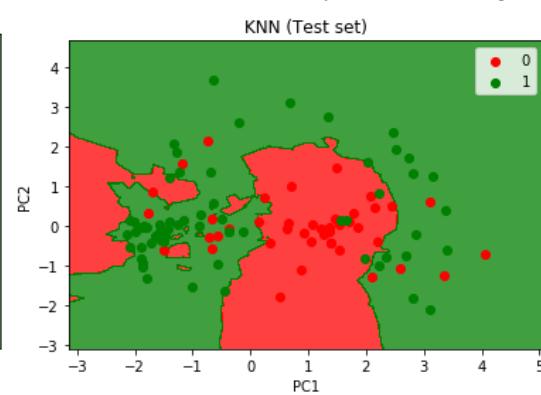
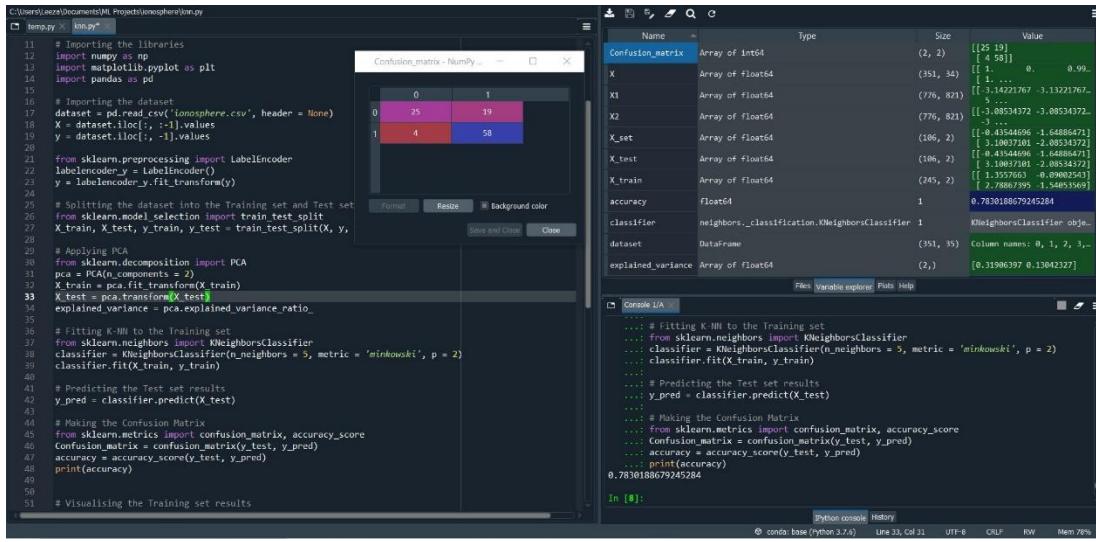


Fig. 1.1.2.c Test Set Graph

➤ **K-value: - 5**
Accuracy: - 78.30%



```

11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-MN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49
50 # Visualising the Training set results

```

Confusion matrix - NumPy -

	0	1
0	25	19
1	4	58

Console 1/A:

```

... # Fitting K-MN to the Training set
... from sklearn.neighbors import KNeighborsClassifier
... classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
... classifier.fit(X_train, y_train)
...
... # Predicting the Test set results
... y_pred = classifier.predict(X_test)
...
... # Making the Confusion Matrix
... from sklearn.metrics import confusion_matrix, accuracy_score
... confusion_matrix = confusion_matrix(y_test, y_pred)
... accuracy = accuracy_score(y_test, y_pred)
... print(accuracy)
0.7830186792452784

```

Fig.

1.1.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

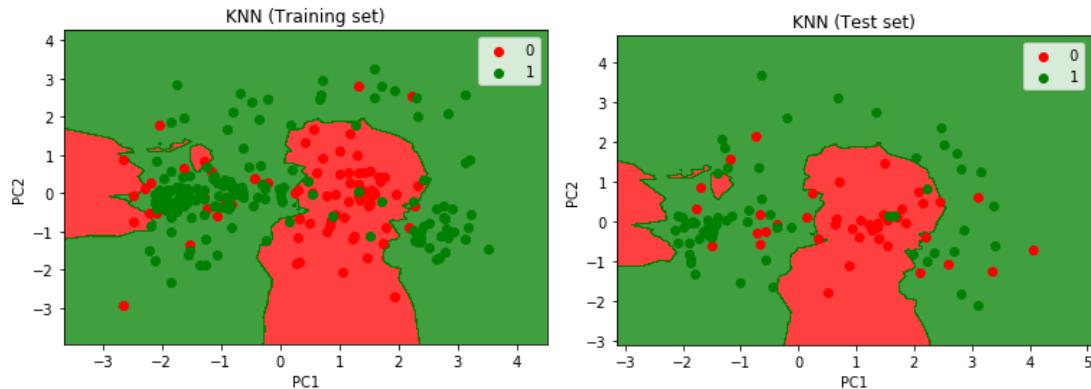
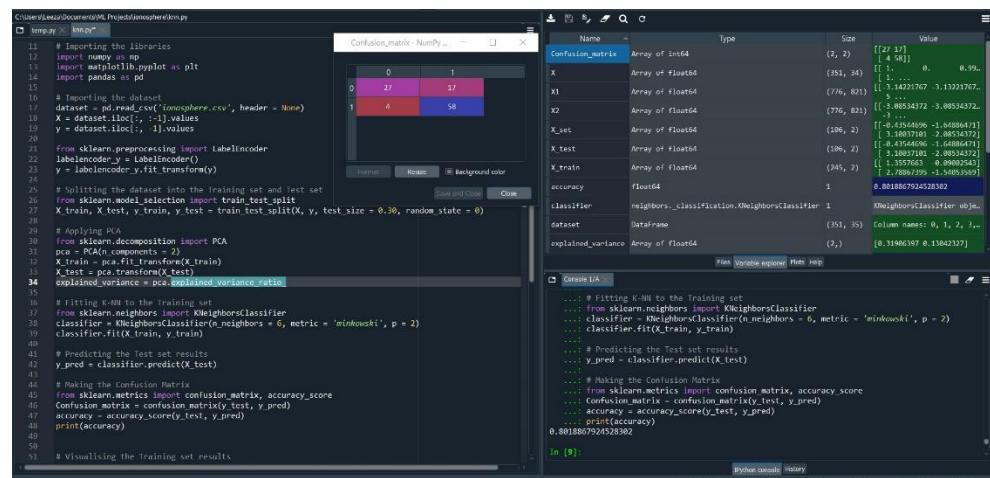


Fig. 1.1.3.b Training Set Graph

Fig. 1.1.3.c Test Set Graph

➤ **K-value: - 6**
Accuracy: - 80.18%



```

11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-MN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 6, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49
50 # Visualising the Training set results

```

Confusion matrix - NumPy -

	0	1
0	27	17
1	6	58

Console 1/A:

```

... # Fitting K-MN to the Training set
... from sklearn.neighbors import KNeighborsClassifier
... classifier = KNeighborsClassifier(n_neighbors = 6, metric = 'minkowski', p = 2)
... classifier.fit(X_train, y_train)
...
... # Predicting the Test set results
... y_pred = classifier.predict(X_test)
...
... # Making the Confusion Matrix
... from sklearn.metrics import confusion_matrix, accuracy_score
... confusion_matrix = confusion_matrix(y_test, y_pred)
... accuracy = accuracy_score(y_test, y_pred)
... print(accuracy)
0.801867924528892

```

Fig. 1.1.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

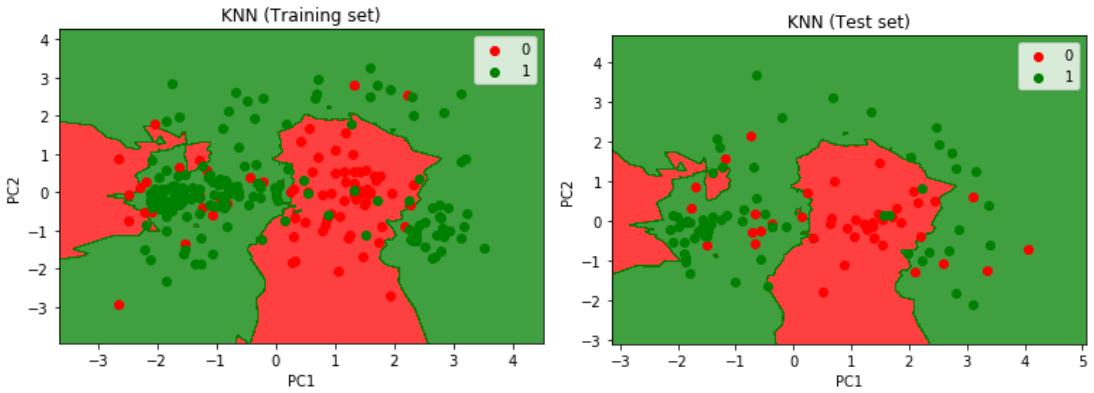


Fig. 1.1.4.b Training Set Graph

Fig. 1.1.4.c Test Set Graph

➤ K-value: -7

Accuracy: - 80.18%

```

9 # K-Nearest Neighbors (K-NN)
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-NN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 7, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49

```

Fig. 1.1.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

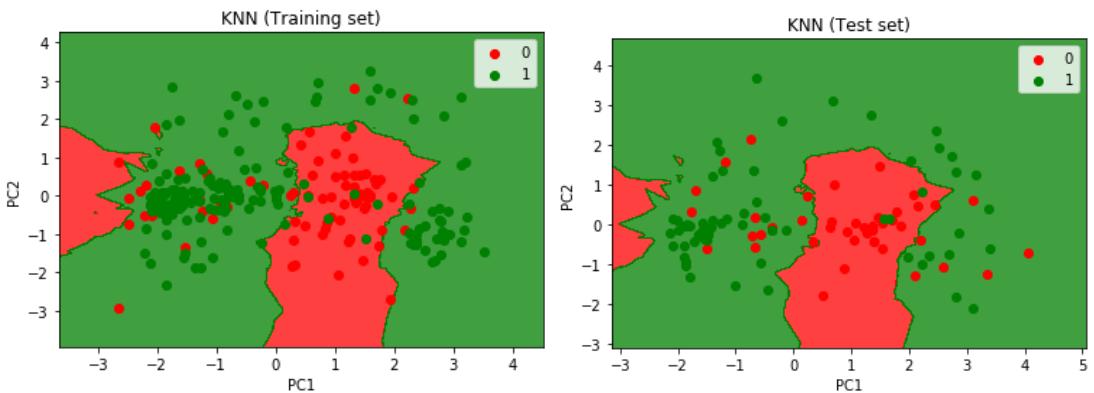


Fig. 1.1.5.b Training Set Graph

Fig. 1.1.5.c Test Set Graph

Conclusion: - The classification with test size = 30% gives best accuracy score = 80.18% with K-value = 6 and 7.

b. Test-size: - 40%

➤ K-value: - 3

Accuracy: - 80.85%

```

temp.py | [temp.py]
Created on Sat May 22 23:07:23 2020
Author: Leesa
...
# K-Nearest Neighbors (K-NN)
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split(X, y, test_size = 0.40, random_state = 0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
0.80851061897723

```

Confusion_matrix - NumPy		
0	1	
0	17	26
1	2	77

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[17 26] [2 77]]
X	Array of float64	(351, 34)	[[1. ... 0. ... 0. ...]]
X1	Array of float64	(761, 819)	[[-1.1668775 3.9848875 ...]]
X2	Array of float64	(761, 819)	[[-2.9393396 -2.9393396 ...]]
X_set	Array of float64	(141, 2)	[[4.4591288 1.7563692 ...]]
X_test	Array of float64	(141, 2)	[[-0.4591298 -1.7563682 ...]]
X_train	Array of float64	(218, 2)	[[2.0693764 0.8136463 ...]]
accuracy	float64	1	0.80851061897723
classifier	KNeighborsClassifier	1	KNeighborsClassifier object
dataset	Dataframe	(351, 35)	Column names: 0, 1, 2, 3, ...
explained_variance	Array of float64	(2,)	[0.31151855 0.12930431]

Fig. 1.2.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

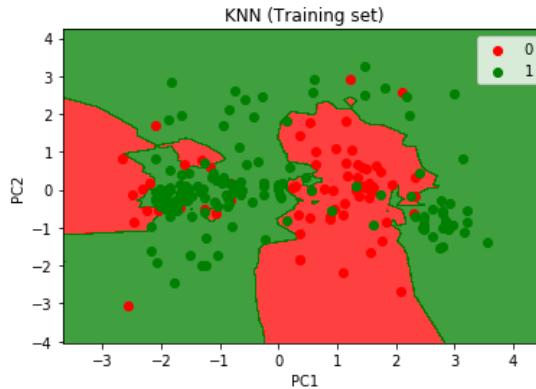


Fig. 1.2.1.b Training Set Graph

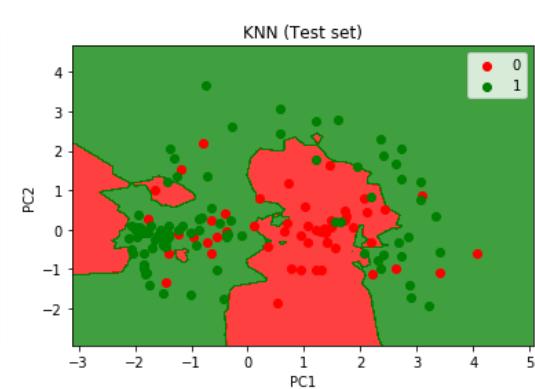


Fig. 1.2.1.c Test Set Graph

➤ K-value: - 4

Accuracy: - 80.14%

```

temp.py | [temp.py]
Created on Sat May 22 23:07:23 2020
Author: Leesa
...
# K-Nearest Neighbors (K-NN)
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split(X, y, test_size = 0.40, random_state = 0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 4, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
0.8014848439716312

```

Confusion_matrix - NumPy		
0	1	
0	38	19
1	9	75

```

temp.py | [temp.py]
Created on Sat May 22 23:07:23 2020
Author: Leesa
...
# K-Nearest Neighbors (K-NN)
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split(X, y, test_size = 0.40, random_state = 0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 4, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
0.8014848439716312

```

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[38 19] [9 75]]
X	Array of float64	(351, 34)	[[1. ... 0. ... 0. ...]]
X1	Array of float64	(761, 819)	[[-1.1668775 3.9848875 ...]]
X2	Array of float64	(761, 819)	[[-2.9393396 -2.9393396 ...]]
X_set	Array of float64	(141, 2)	[[4.4591288 1.7563692 ...]]
X_test	Array of float64	(141, 2)	[[-0.4591298 -1.7563682 ...]]
X_train	Array of float64	(218, 2)	[[2.0693764 0.8136463 ...]]
accuracy	float64	1	0.8014848439716312
classifier	KNeighborsClassifier	1	KNeighborsClassifier object
dataset	Dataframe	(351, 35)	Column names: 0, 1, 2, 3, ...
explained_variance	Array of float64	(2,)	[0.31151855 0.12930431]

Fig. 1.2.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

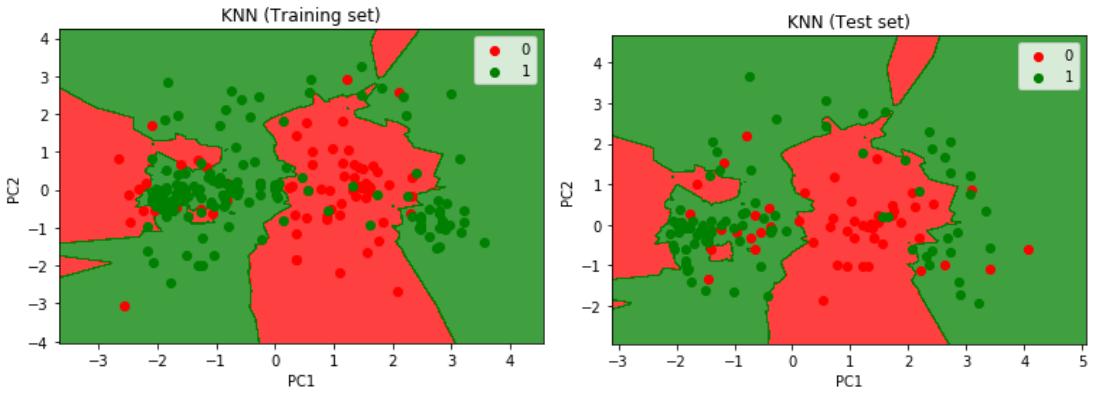


Fig. 1.2.2.b Training Set Graph

Fig. 1.2.2.c Test Set Graph

➤ K-value: - 5

Accuracy: - 79.43%

```

4
5 #author: Leesa
6
7
8
9 # K-Nearest Neighbors (K-NN)
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-NN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49
50

```

Fig. 1.2.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

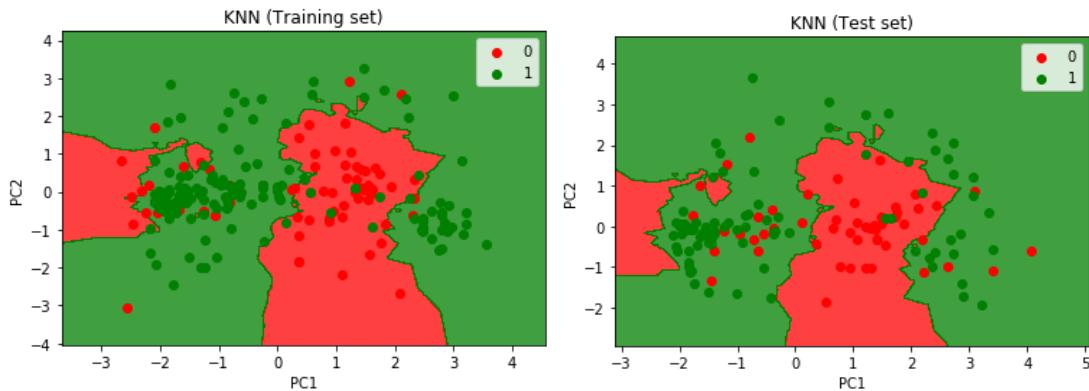


Fig. 1.2.3.b Training Set Graph

Fig. 1.2.3.c Test Set Graph

➤ K-value: - 6

Accuracy: - 79.43%

The screenshot shows a Jupyter Notebook interface with the following components:

- Code Cell:** The code for a K-Nearest Neighbors classifier using PCA and K=6.
- Output Cell:** A confusion matrix table showing 35 True Positives (TP), 22 False Positives (FP), 7 False Negatives (FN), and 72 True Negatives (TN).
- Variable Explorer:** Shows variables like X, X1, X2, X_set, X_test, X_train, accuracy, classifier, dataset, and explained_variance.
- Console Log:** Displays the accuracy value: 0.7943262411347518.

Fig. 1.2.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

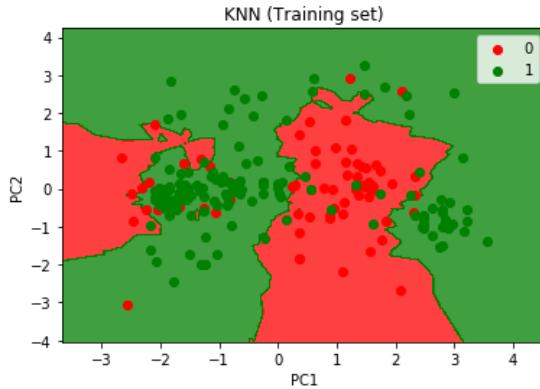


Fig. 1.2.4.b Training Set Graph

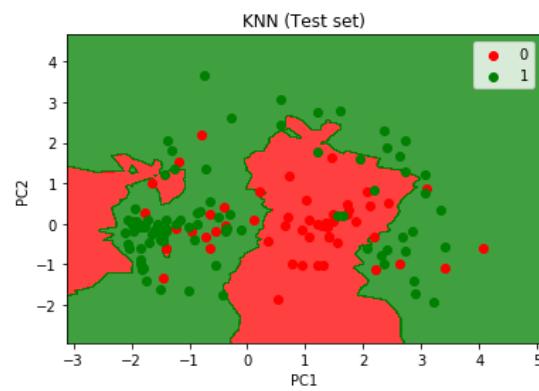


Fig. 1.2.4.c Test Set Graph

➤ K-value: - 7

Accuracy: - 79.43%

The screenshot shows a Jupyter Notebook interface with the following components:

- Code Cell:** The same K-Nearest Neighbors classifier code as before, but with K=7.
- Output Cell:** A confusion matrix table showing 33 True Positives (TP), 26 False Positives (FP), 5 False Negatives (FN), and 72 True Negatives (TN).
- Variable Explorer:** Shows variables like X, X1, X2, X_set, X_test, X_train, accuracy, classifier, dataset, and explained_variance.
- Console Log:** Displays the accuracy value: 0.7943262411347518.

Fig. 1.2.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

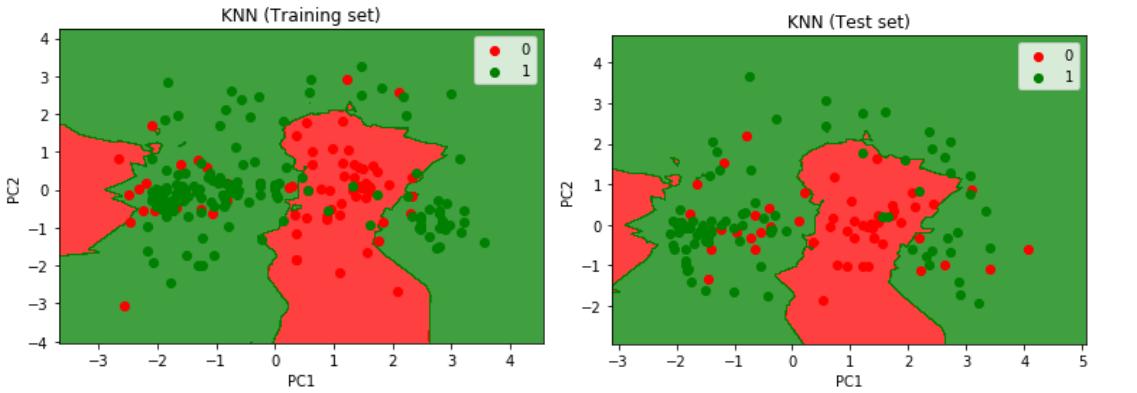


Fig. 1.2.5.b Training Set Graph

Fig. 1.2.5.c Test Set Graph

Conclusion: - The classification with test size = 40% gives best accuracy score = 80.85% with K-value = 3.

c. Test-size: - 50%

➤ K-value: - 3

Accuracy: - 78.97%

```

C:\Users\Leeza\Documents\ML Projects\ionosphere\knn.py
temp.ipynb > kNN.py
3   Created on Sat May 23 23:07:23 2020
4   @Author: Leeza
5
6
7
8
9   # K-Nearest Neighbors (K-NN)
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
15
16  # Importing the dataset
17  dataset = pd.read_csv('ionosphere.csv', header = None)
18  X = dataset.iloc[:, :-1].values
19  y = dataset.iloc[:, -1].values
20
21  from sklearn.preprocessing import LabelEncoder
22  labelencoder_y = LabelEncoder()
23  y = labelencoder_y.transform(y)
24
25  # Splitting the dataset into the Training set and Test set
26  from sklearn.model_selection import train_test_split
27  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29  # Applying PCA
30  from sklearn.decomposition import PCA
31  pca = PCA(n_components = 2)
32  X_train = pca.fit_transform(X_train)
33  X_test = pca.transform(X_test)
34  explained_variance_ratio_ = pca.explained_variance_ratio_
35
36  # Fitting K-NN to the Training set
37  from sklearn.neighbors import KNeighborsClassifier
38  classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
39  classifier.fit(X_train, y_train)
40
41  # Predicting the Test set results
42  y_pred = classifier.predict(X_test)
43
44  # Making the Confusion Matrix
45  from sklearn.metrics import confusion_matrix, accuracy_score
46  confusion_matrix = confusion_matrix(y_test, y_pred)
47  accuracy = accuracy_score(y_test, y_pred)
48  print(accuracy)
49

```

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[44 31] [6 95]]
X	Array of float64	(351, 34)	[1. ...]
X1	Array of float64	(780, 857)	[1.47780644 -3.46780644 ...]
X2	Array of float64	(780, 857)	[-3.11979976 -3.11979976 ...]
X_set	Array of float64	(176, 2)	[1.37446914 -1.53215052]
X_test	Array of float64	(176, 2)	[-0.37646914 -1.63415052]
X_train	Array of float64	(175, 2)	[2.87199818 -1.67881872]
accuracy	float64	1	0.7897727272727273
classifier	neighbors._classification.KNeighborsClassifier	1	KNeighborsClassifier object
dataset	DataFrame	(351, 35)	Column names: 0, 1, 2, 3, ...
explained_variance	Array of float64	(2,)	[0.32942732 0.14880766]

Fig. 1.3.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

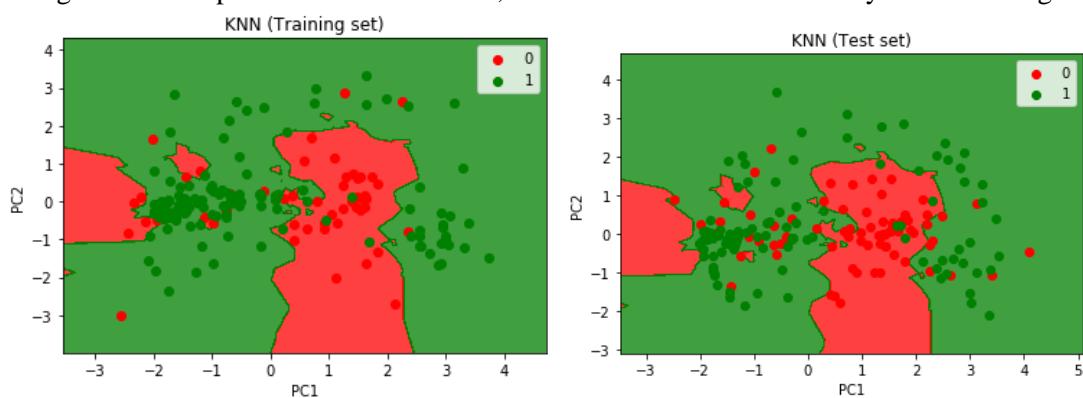


Fig. 1.3.1.b Training Set Graph

Fig. 1.3.1.c Test Set Graph

➤ **K-value: - 4**

Accuracy: - 82.38%

The screenshot shows a Jupyter Notebook interface with the following components:

- Code Cell:** The code for a K-Nearest Neighbors classifier using scikit-learn.
- Output Cell:** A confusion matrix table and a console log showing the accuracy.
- Variable Explorer:** Shows variables like Confusion_matrix, X, X1, X2, X_set, X_test, X_train, accuracy, classifier, dataset, and explained_variance.
- Console Log:** Displays the command to fit the classifier, predict the test set, and calculate accuracy.

```

C:\Users\Leeza\Documents\ML Projects\ionosphere\knn.py
temp.py | Runpy |
4
5 #author: Leeza
6
7
8
9 # K-Nearest Neighbors (K-NN)
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-NN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 4, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49
50

```

Confusion_matrix	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[52 23] [8 93]]
X	Array of float64	(351, 34)	[[1. ... 0. 0.99...]
X1	Array of float64	(788, 857)	[[3.11079976 -3.11079976...]
X2	Array of float64	(788, 857)	[[-0.37646934 -1.63415952...]
X_set	Array of float64	(176, 2)	[[0.35959077 -2.11079976...]
X_test	Array of float64	(176, 2)	[[3.15595077 -2.11079976...]
X_train	Array of float64	(175, 2)	[[2.87109818 -1.6781872...]
accuracy	float64	1	0.8238636363636364
classifier	neighbors._classification.KNeighborsClassifier	1	KNeighborsClassifier object
dataset	DataFrame	(351, 35)	Column names: 0, 1, 2, 3, ...
explained_variance	Array of float64	(2,)	[0.32942732 0.14880766]

Fig. 1.3.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

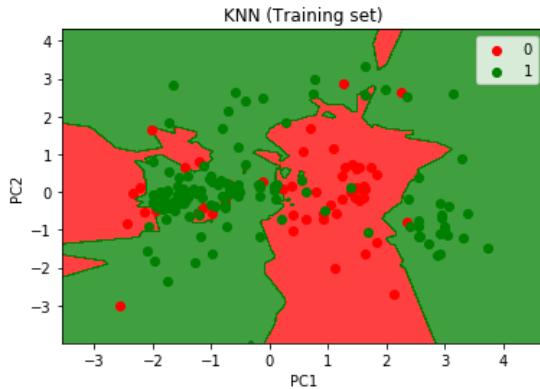


Fig 1.3.2.b Training Set Graph

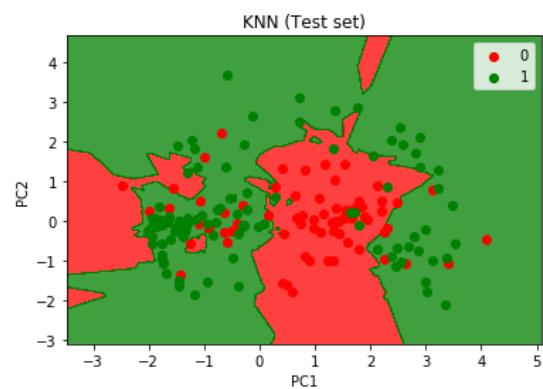


Fig. 1.3.2.c Test Set Graph

➤ **K-value: - 5**

Accuracy: - 81.81%

The screenshot shows a Jupyter Notebook interface with the following components:

- Code Cell:** The code for a K-Nearest Neighbors classifier using scikit-learn.
- Output Cell:** A confusion matrix table and a console log showing the accuracy.
- Variable Explorer:** Shows variables like Confusion_matrix, X, X1, X2, X_set, X_test, X_train, accuracy, classifier, dataset, and explained_variance.
- Console Log:** Displays the command to fit the classifier, predict the test set, and calculate accuracy.

```

C:\Users\Leeza\Documents\ML Projects\ionosphere\knn.py
temp.py | Runpy |
4
5 #author: Leeza
6
7
8
9 # K-Nearest Neighbors (K-NN)
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting K-NN to the Training set
37 from sklearn.neighbors import KNeighborsClassifier
38 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49
50

```

Confusion_matrix	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[48 27] [5 96]]
X	Array of float64	(351, 34)	[[1. ... 0. 0.99...]
X1	Array of float64	(788, 857)	[[-3.4788644 5.46788644...]
X2	Array of float64	(788, 857)	[[3.11079976 -3.11079976...]
X_set	Array of float64	(176, 2)	[[-0.37646934 -1.63415952...]
X_test	Array of float64	(176, 2)	[[-3.55959077 -2.11079976...]
X_train	Array of float64	(175, 2)	[[2.87109818 -1.6781872...]
accuracy	float64	1	0.8181818181818182
classifier	neighbors._classification.KNeighborsClassifier	1	KNeighborsClassifier object
dataset	DataFrame	(351, 35)	Column names: 0, 1, 2, 3, ...
explained_variance	Array of float64	(2,)	[0.3942732 0.14880766]

Fig. 1.3.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

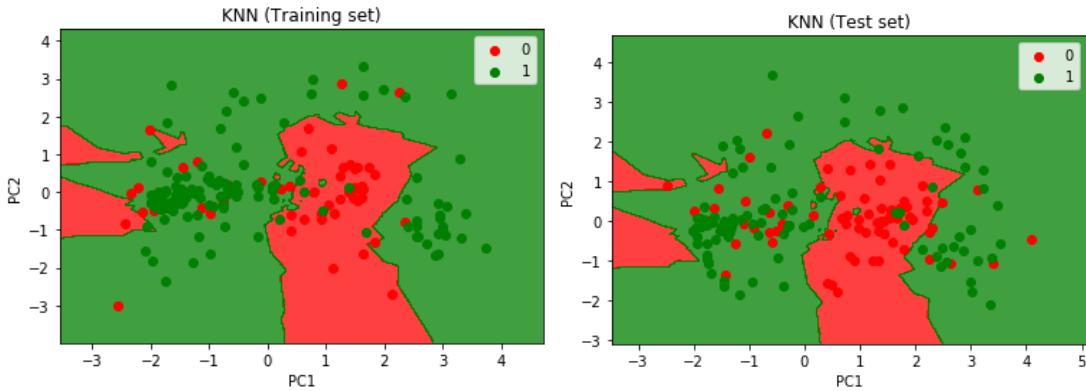


Fig. 1.3.3.b Training Set Graph

Fig. 1.3.3.c Test Set Graph

➤ K-value: - 6

Accuracy: - 82.18%

```

5 #author: Leesa
6
7
8 # K-Nearest Neighbors (K-NN)
9
10 # Importing the libraries
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pandas as pd
14
15 # Importing the dataset
16 dataset = pd.read_csv('ionosphere.csv', header = None)
17 X = dataset.loc[:, :-1].values
18 y = dataset.loc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24 # Splitting the dataset into the Training set and Test set
25 from sklearn.model_selection import train_test_split
26 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
27
28 # Applying PCA
29 from sklearn.decomposition import PCA
30 pca = PCA(n_components = 2)
31 X_train = pca.fit_transform(X_train)
32 X_test = pca.transform(X_test)
33 explained_variance = pca.explained_variance_ratio_
34
35 # Fitting K-NN to the Training set
36 from sklearn.neighbors import KNeighborsClassifier
37 classifier = KNeighborsClassifier(n_neighbors = 6, metric = 'minkowski', p = 2)
38 classifier.fit(X_train, y_train)
39
40 # Predicting the Test set results
41 y_pred = classifier.predict(X_test)
42
43 # Making the Confusion Matrix
44 from sklearn.metrics import confusion_matrix, accuracy_score
45 Confusion_matrix = confusion_matrix(y_test, y_pred)
46 accuracy = accuracy_score(y_test, y_pred)
47 print(accuracy)
48
49
50 # Visualizing the Training set results

```

Fig. 1.3.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

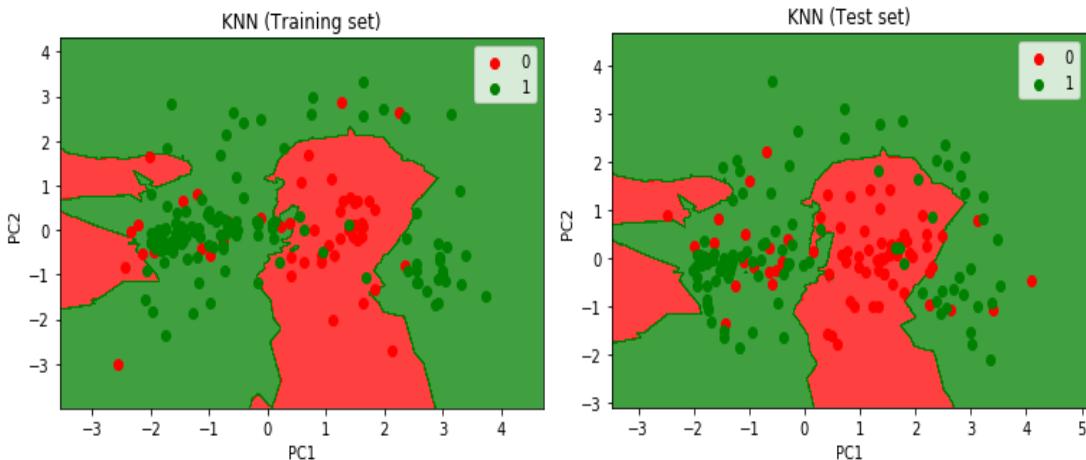


Fig. 1.3.4.b Training Set Graph

Fig. 1.3.4.c Test Set Graph

➤ K-value: -7

Accuracy: - 80.68%

The screenshot shows a Jupyter Notebook interface with several windows:

- Code Cell:** Contains Python code for KNN classification, including importing libraries, reading the ionosphere dataset, splitting it into training and test sets, applying PCA, fitting a KNN classifier with K=7, predicting the test set, and calculating accuracy.
- Output Cell:** Displays the confusion matrix as a 2x2 table:

0	48
1	27
0	7
1	94
- Variable Explorer:** Shows variables and their values, including the confusion matrix, X_set, X_train, X_test, accuracy, classifier, dataset, and explained_variance.
- Console:** Shows the command history and the output of the accuracy calculation: `accuracy = accuracy_score(y_test, y_pred)` followed by the value `0.8068181818181818`.

Fig. 1.3.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

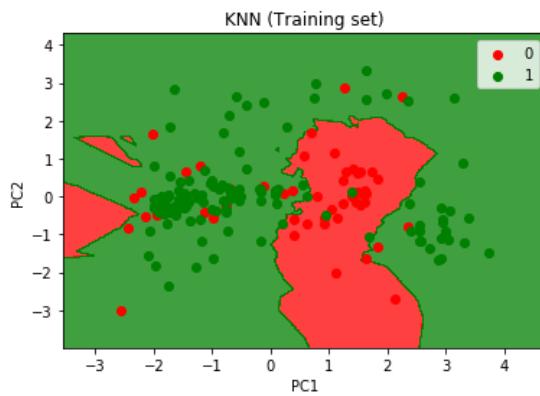


Fig. 1.3.5.b Training Set Graph

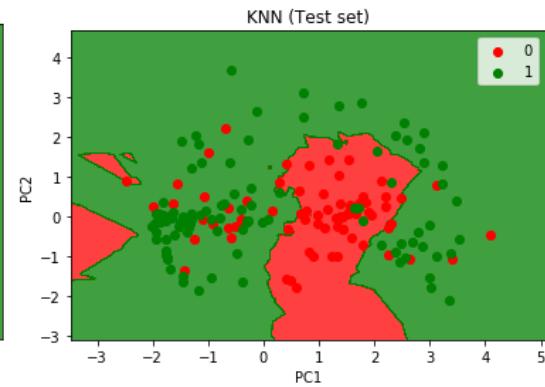


Fig. 1.3.5.c Test Set Graph

Conclusion: - The classification with test size = 50% gives best accuracy score = 82.38% with K-value = 4.

2. Decision Tree Classifier: -

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('ionosphere.csv', header = None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

```

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
//values of test size, random_state = 0)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth =
//value of max_depth of the tree, random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree (Training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),

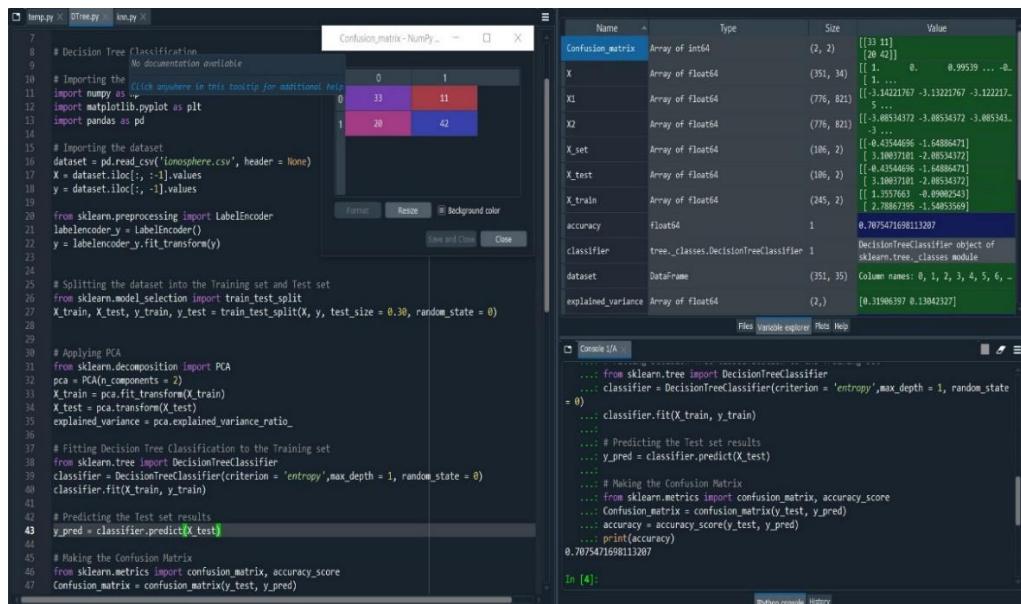
```

```

        alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree (Test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

```

- a. Test-size: - 30%**
 ➤ **max_depth = 1**
 Accuracy: - 70.75%



The screenshot shows a Jupyter Notebook interface with three main sections:

- Code Cell:** Contains Python code for loading the dataset, splitting it into training and test sets, applying PCA, fitting a Decision Tree classifier with max_depth=1, and predicting the test set.
- Confusion Matrix:** A 2x2 matrix showing the count of correctly and incorrectly classified samples. The matrix is:

0	33
1	11
20	42
- Console Log:** Displays the accuracy of the classifier, which is 0.707541698113207.

Fig. 2.1.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

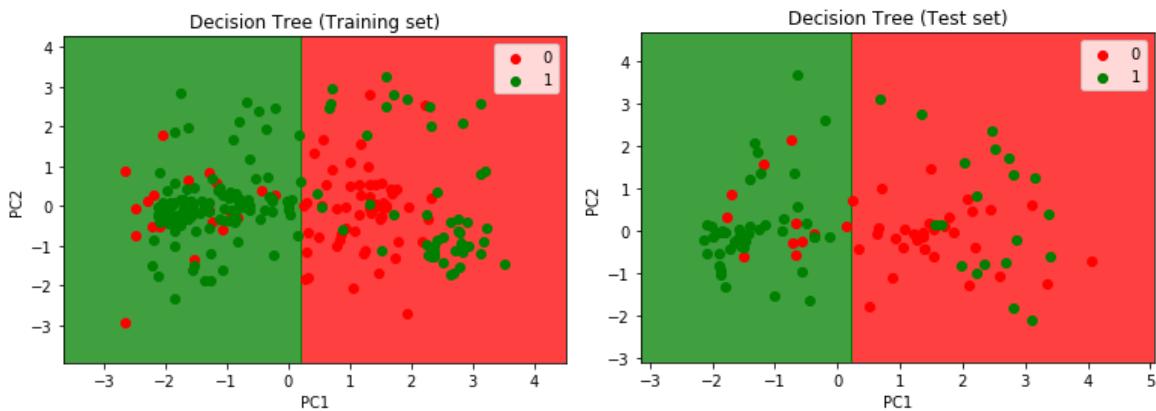


Fig. 2.1.1.b Training Set Graph

Fig. 2.1.1.c Test Set Graph

➤ **max_depth = 2**

Accuracy: - 77.35%

The screenshot shows a Jupyter Notebook interface with several panes:

- Code pane:** Contains Python code for fitting a Decision Tree classifier with max_depth=2 to a dataset and predicting on a test set.
- Output pane:** Displays a confusion matrix table and a console log showing the accuracy score.
- Variable explorer:** Shows variables like X, X_train, X_test, y_train, y_test, classifier, accuracy, etc.
- Console:** Shows the command to import DecisionTreeClassifier and the resulting classifier object.
- Plots:** A scatter plot titled "Decision tree (Training set)" showing data points colored by class (red and green) and decision regions.

Fig. 2.1.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

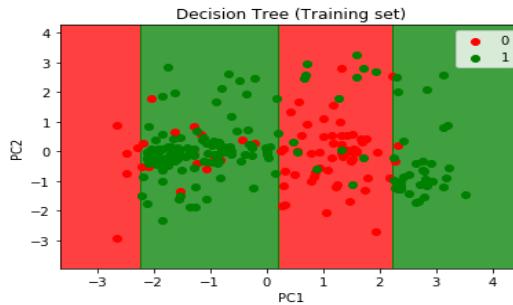


Fig. 2.1.2.b Training Set Graph

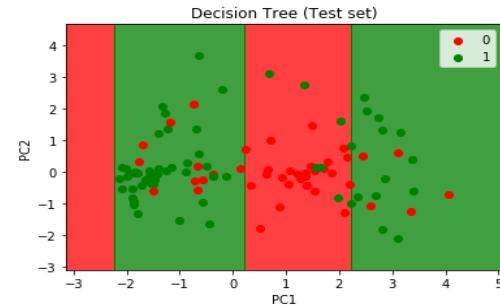


Fig. 2.1.2.c Test Set Graph

➤ **max_depth = 3**

Accuracy: - 79.24%

The screenshot shows a Jupyter Notebook interface with several panes:

- Code pane:** Contains Python code for fitting a Decision Tree classifier with max_depth=3 to a dataset and predicting on a test set.
- Output pane:** Displays a confusion matrix table and a console log showing the accuracy score.
- Variable explorer:** Shows variables like X, X_train, X_test, y_train, y_test, classifier, accuracy, etc.
- Console:** Shows the command to import DecisionTreeClassifier and the resulting classifier object.
- Plots:** Two scatter plots titled "Decision tree (Training set)" and "Decision tree (Test set)" showing data points colored by class (red and green) and decision regions. The axes are the same as in Fig. 2.1.2.b and 2.1.2.c.

Fig. 2.1.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

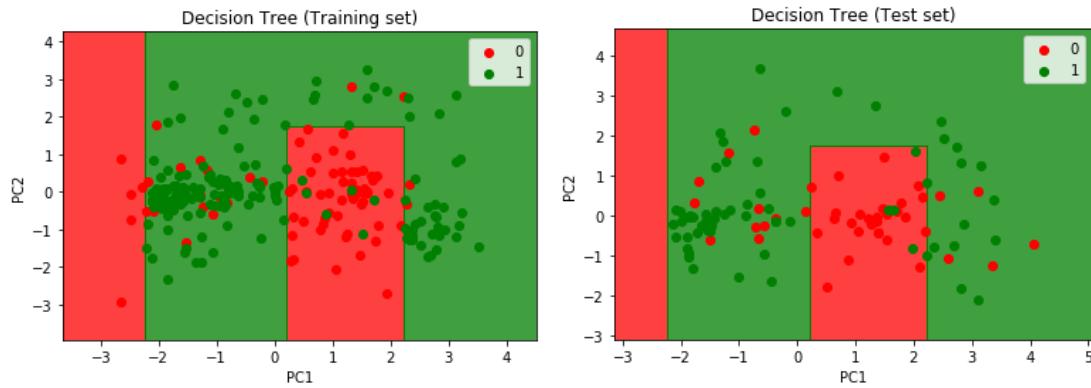


Fig. 2.1.3.b Training Set Graph

Fig. 2.1.3.c Test Set Graph

➤ **max_depth = 6**
Accuracy: - 78.30%

```

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min(), stop = X_set[:, 0].max(), step = 0.05),
                     np.arange(start = X_set[:, 1].min(), stop = X_set[:, 1].max(), step = 0.05))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(['red', 'green']))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(['red', 'green'])(j), label = j)
plt.title('Decision Tree (Training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

# Visualising the Test set results
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min(), stop = X_set[:, 0].max(), step = 0.05),
                     np.arange(start = X_set[:, 1].min(), stop = X_set[:, 1].max(), step = 0.05))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(['red', 'green']))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(['red', 'green'])(j), label = j)
plt.title('Decision Tree (Test set)')


```

Fig. 2.1.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

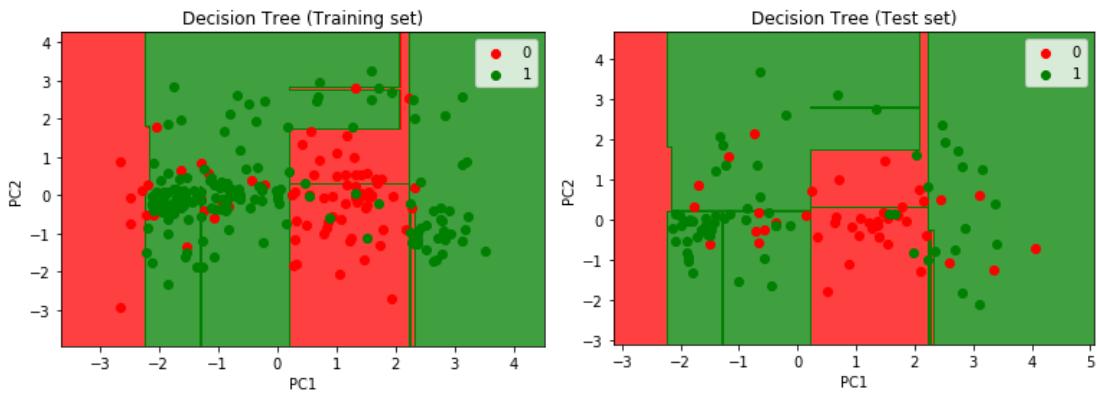


Fig. 2.1.4.b Training Set Graph

Fig. 2.1.4.c Test Set Graph

➤ **max_depth = 7**
Accuracy: - 78.30%

The screenshot shows a Jupyter Notebook interface. On the left, the code for a Decision Tree classifier is displayed. In the center, a confusion matrix is shown with values: 0 28 16, 1 7 55. On the right, the accuracy is printed as 0.7830188679245284.

```

temp.py < JTree.ipynb > In[1]
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('lenses.csv', header = None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 7, random_state = 0)
classifier.fit(X_train, y_train)
# Predicting the Test set
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

```

Fig. 2.1.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

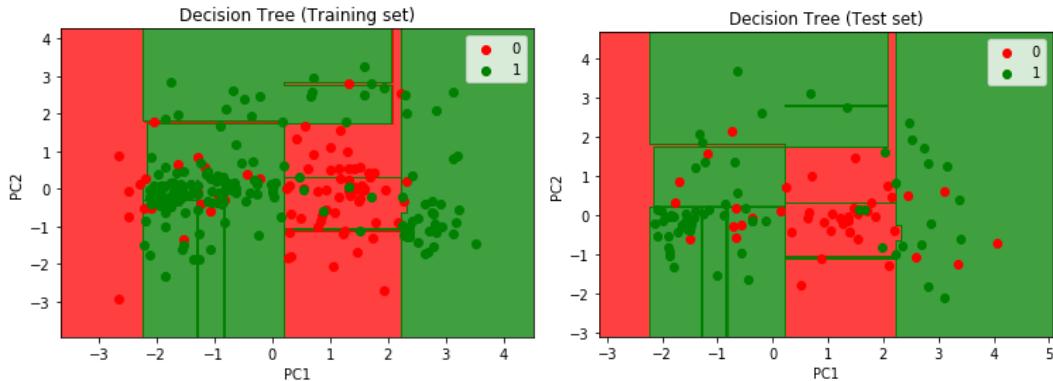


Fig. 2.1.5.b Training Set Graph

Fig. 2.1.5.c Test Set Graph

Conclusion: - The classification model with max_depth = 30% yields best accuracy of 79.24% among all the values of max_depth.

b. **Test-size: - 40%**
➤ **max_depth = 1**
Accuracy: - 68.79%

The screenshot shows a Jupyter Notebook interface. On the left, the code for a Decision Tree classifier is displayed. In the center, a confusion matrix is shown with values: 0 41 16, 1 28 56. On the right, the accuracy is printed as 0.6879432624113475.

```

temp.py < JTree.ipynb > In[1]
# *-* coding: utf-8 *-
# Created on Sat May 23 23:43:59 2020
# @author: Leesa
# Decision Tree Classification
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('lenses.csv', header = None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 1, random_state = 0)
classifier.fit(X_train, y_train)
# Predicting the Test set
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

```

Fig. 2.2.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

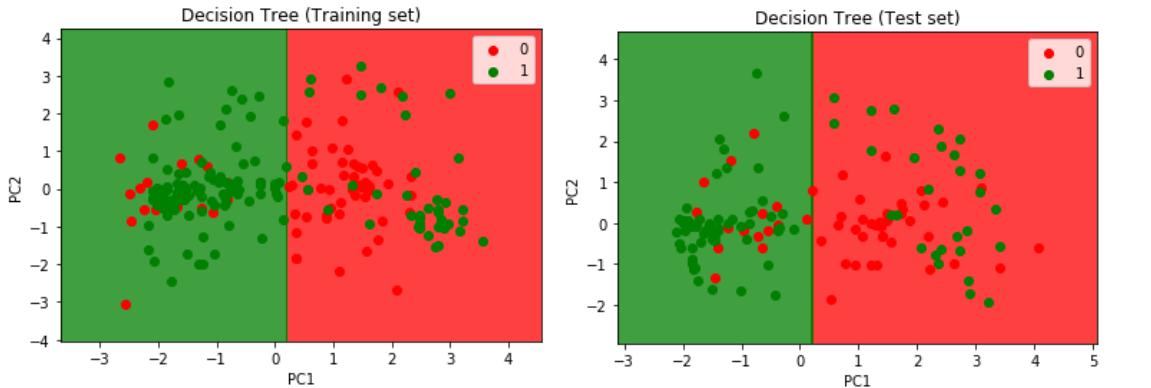


Fig. 2.2.1.b Training Set Graph

Fig. 2.2.1.c Test Set Graph

➤ **max_depth = 2**
Accuracy: - 75.88%

```

C:\Users\ewa\Documents\ML\Projects\iris\iris.py
In [1]: # Importing the required libraries
17 X = dataset.iloc[:, :-1].values
18 y = dataset.iloc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24 # Splitting the dataset into the Training set and Test set
25 from sklearn.model_selection import train_test_split
26 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
27
28 # Applying PCA
29 from sklearn.decomposition import PCA
30 pca = PCA(n_components=2)
31 X_train = pca.fit_transform(X_train)
32 X_test = pca.transform(X_test)
33 explained_variance = pca.explained_variance_ratio_
34
35 # Fitting Decision tree classification to the training set
36 from sklearn.tree import DecisionTreeClassifier
37 classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 2, random_state = 0)
38 classifier.fit(X_train, y_train)
39
40 # Predicting the Test set results
41 y_pred = classifier.predict(X_test)
42
43 # Making the Confusion Matrix
44 from sklearn.metrics import confusion_matrix, accuracy_score
45 confusion_matrix = confusion_matrix(y_test, y_pred)
46 accuracy = accuracy_score(y_test, y_pred)
47 print(accuracy)
48
49 # Visualising the Training set results
50 from matplotlib.colors import ListedColormap
51 X_set, y_set = X_train, y_train
52 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.5),
53                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.5))
54
55 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
56               alpha = 0.75)
57
58 # Visualising the Test set results
59 from matplotlib.colors import ListedColormap
60 X_set, y_set = X_test, y_test
61 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.5),
62                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.5))
63
64 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
65               alpha = 0.75)
66
67 # Making the Confusion Matrix
68 from sklearn.metrics import confusion_matrix, accuracy_score
69 confusion_matrix = confusion_matrix(y_test, y_pred)
70 accuracy = accuracy_score(y_test, y_pred)
71 print(accuracy)
72
73 # Applying PCA
74 from sklearn.decomposition import PCA
75 pca = PCA(n_components=2)
76 X_train = pca.fit_transform(X_train)
77 X_test = pca.transform(X_test)
78 explained_variance = pca.explained_variance_ratio_
79
80 # Fitting Decision tree classification to the training set
81 from sklearn.tree import DecisionTreeClassifier
82 classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 2, random_state = 0)
83 classifier.fit(X_train, y_train)
84
85 # Predicting the Test set results
86 y_pred = classifier.predict(X_test)
87
88 # Making the Confusion Matrix
89 from sklearn.metrics import confusion_matrix, accuracy_score
90 confusion_matrix = confusion_matrix(y_test, y_pred)
91 accuracy = accuracy_score(y_test, y_pred)
92 print(accuracy)
93
94 # Visualising the Training set results
95 from matplotlib.colors import ListedColormap
96 X_set, y_set = X_train, y_train
97 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.5),
98                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.5))
99
100 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
101               alpha = 0.75)
102
103 # Visualising the Test set results
104 from matplotlib.colors import ListedColormap
105 X_set, y_set = X_test, y_test
106 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.5),
107                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.5))
108
109 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
110               alpha = 0.75)
111
112 # Making the Confusion Matrix
113 from sklearn.metrics import confusion_matrix, accuracy_score
114 confusion_matrix = confusion_matrix(y_test, y_pred)
115 accuracy = accuracy_score(y_test, y_pred)
116 print(accuracy)

```

Fig. 2.2.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

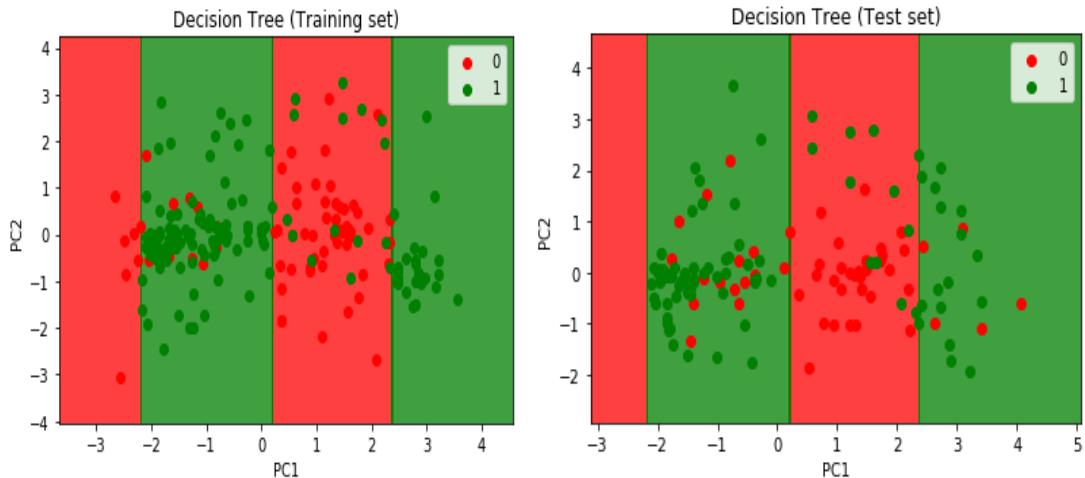


Fig. 2.2.2.b Training Set Graph

Fig. 2.2.2.c Test Set Graph

➤ **max_depth = 3**

Accuracy: -79.43%

The screenshot shows a Jupyter Notebook interface with several panes. The code cell contains Python code for splitting the dataset, applying PCA, fitting a DecisionTreeClassifier with max_depth=3, and calculating accuracy. The 'Confusion matrix' cell displays a 2x2 matrix:

0	36
1	8

The 'In [26]' cell shows the classifier's fit and predict steps, resulting in an accuracy of 0.7943262411347518.

Fig. 2.2.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

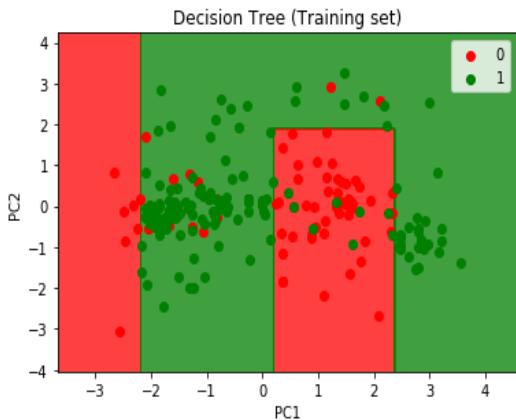


Fig. 2.2.3.b Training Set Graph

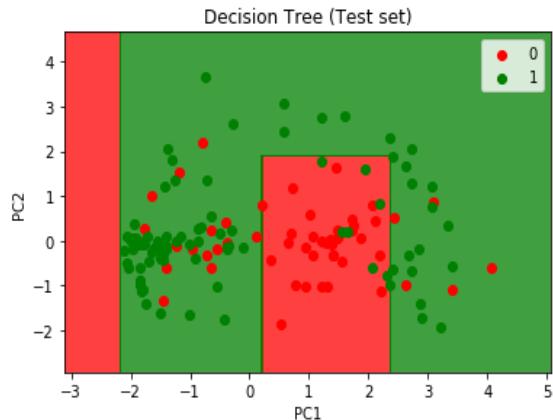


Fig. 2.2.3.c Test Set Graph

➤ **max_depth = 6**

Accuracy: - 79.43%

The screenshot shows a Jupyter Notebook interface with several panes. The code cell contains Python code for splitting the dataset, applying PCA, fitting a DecisionTreeClassifier with max_depth=6, and calculating accuracy. The 'Confusion matrix' cell displays a 2x2 matrix:

0	36
1	8

The 'In [23]' cell shows the classifier's fit and predict steps, resulting in an accuracy of 0.7943262411347518.

Fig. 2.2.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

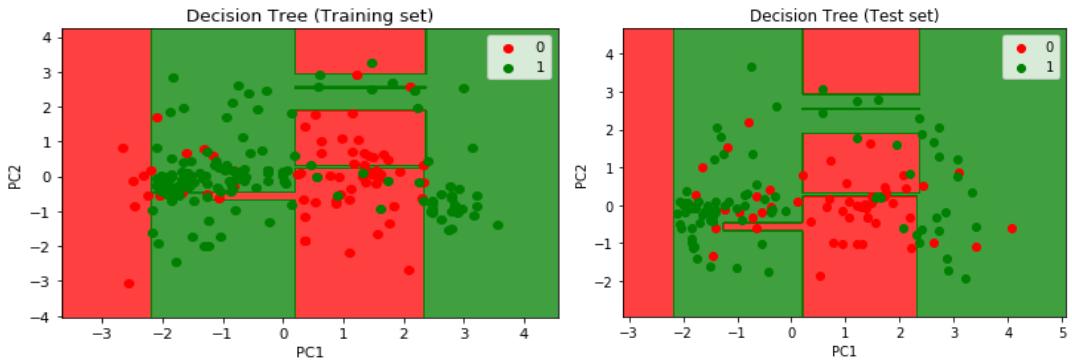


Fig. 2.2.4.b Training Set Graph

Fig. 2.2.4.c Test Set Graph

➤ **max_depth = 7**
Accuracy: =78.72%

```

C:\Users\esra\Documents\N_Project\jupyter\011.ipynb
[1]: # Importing the dataset
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('lenses.csv', header = None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 7, random_state = 0)
classifier.fit(X_train, y_train)
# Predicting the Test set
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

```

Confusion matrix:

	0	1
0	37	20
1	10	74

Console log output:

```

In [20]: from sklearn.tree import DecisionTreeClassifier
... classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 7,
random_state = 0)
... classifier.fit(X_train, y_train)
...
... # Predicting the Test set results
... y_pred = classifier.predict(X_test)
...
... # Making the Confusion Matrix
... from sklearn.metrics import confusion_matrix, accuracy_score
... Confusion_matrix = confusion_matrix(y_test, y_pred)
... accuracy = accuracy_score(y_test, y_pred)
... print(accuracy)
0.7872340642531915

```

Fig. 2.2.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

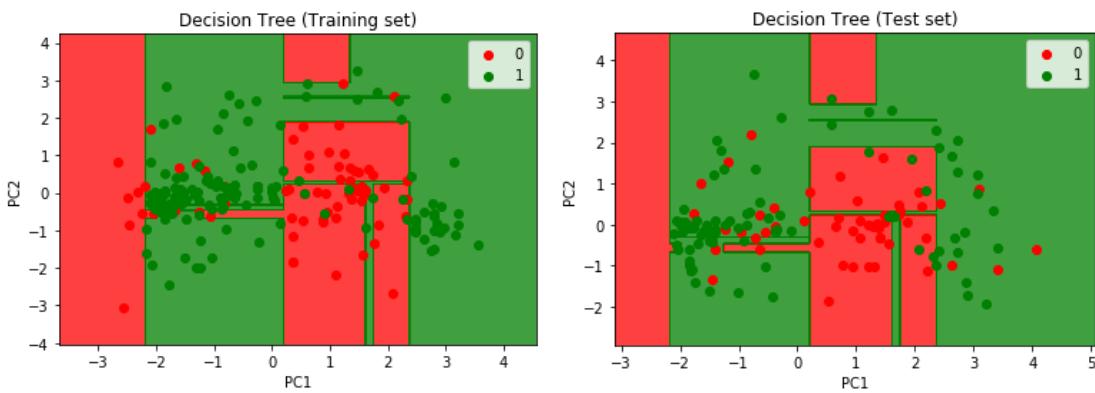


Fig. 2.2.5.b Training Set Graph

Fig. 2.2.5.c Test Set Graph

Conclusion: - The classification model with $\text{max_depth} = 40\%$ yields best accuracy of 79.43% among all the values of max_depth .

- c. Test-size: - 50%
- max_depth = 1
- Accuracy: =78.72%

The screenshot shows a Jupyter Notebook interface. On the left, the code for a Decision Tree classifier is displayed. In the center, a confusion matrix is shown as a 2x2 grid:

0	57
1	18

On the right, the console log displays the following information:

```

Name      Type     Size   Value
Confusion_matrix Array of int64 (2, 2) [[37 18]
X          Array of float64 (351, 34) [[1. ...
X1         Array of float64 (76, 82) [[-3.14221767 ...
X2         Array of float64 (76, 82) [[-3.08534372 ...
X_set        Array of float64 (106, 2) [[-0.43544696 ...
X_test       Array of float64 (141, 2) [[-0.15522938 ...
X_train      Array of float64 (210, 2) [[-2.01633764 ...
accuracy     float64    1      0.7872340425531915
classifier   tree._classes.DecisionTreeClassifier object of ...
dataset      DataFrame (351, 35) Column names: 0, 1, 2, 3, 4, 5, 6, ...
explained_variance Array of float64 (2,) [0.31151855 0.12938431]

```

Fig. 2.3.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

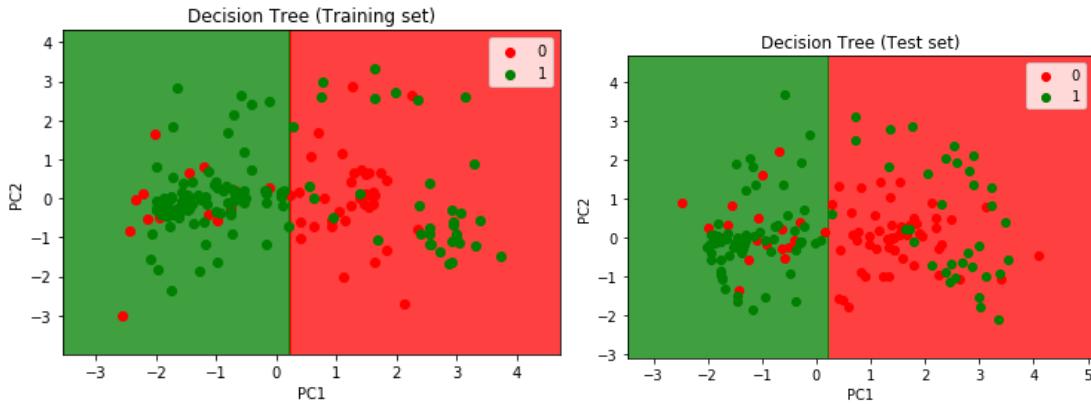


Fig. 2.3.1.b Training Set Graph

Fig. 2.3.1.c Test Set Graph

- max_depth = 2
- Accuracy: =69.31%

The screenshot shows a Jupyter Notebook interface. On the left, the code for a Decision Tree classifier with max_depth=2 is displayed. In the center, a confusion matrix is shown as a 2x2 grid:

0	19
1	35

On the right, the console log displays the following information:

```

Name      Type     Size   Value
Confusion_matrix Array of int64 (2, 2) [[16 19]
X          Array of float64 (351, 34) [[1. ...
X1         Array of float64 (76, 819) [[1.1608775 ...
X2         Array of float64 (76, 819) [[2.9359396 ...
X_set        Array of float64 (141, 2) [[-0.784694 ...
X_test       Array of float64 (176, 2) [[-0.784694 ...
X_train      Array of float64 (175, 2) [[0.54946805 ...
accuracy     float64    1      0.6931818181818182
classifier   tree._classes.DecisionTreeClassifier object of ...
dataset      DataFrame (351, 25) Column names: 0, 1, 2, 3, 4, 5, 6, ...
explained variance Array of float64 (2,) [0.52942732 0.1408866]

```

Fig. 2.3.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

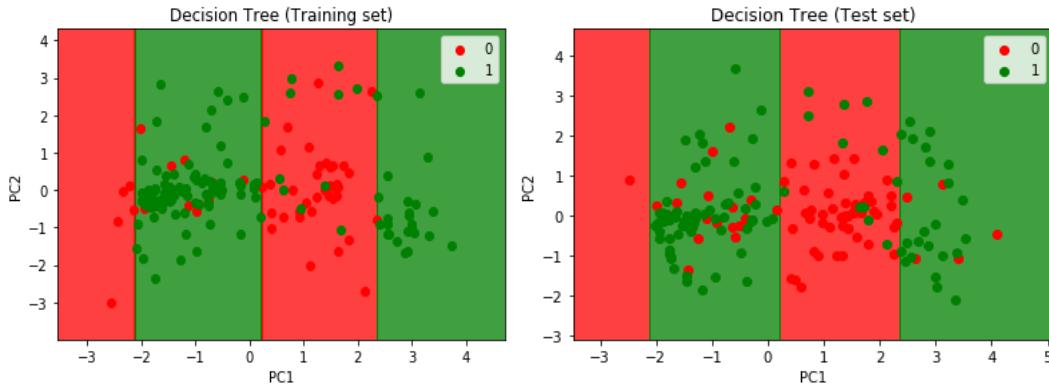


Fig. 2.3.2.b Training Set Graph

Fig. 2.3.2.c Test Set Graph

➤ **max_depth = 3**

Accuracy: =82.29%

```

10 # Importing the libraries
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pandas as pd
14
15 # Importing the dataset
16 dataset = pd.read_csv('iris.csv', header = None)
17 X = dataset.iloc[:, :-1].values
18 y = dataset.iloc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29
30 # Applying PCA
31 from sklearn.decomposition import PCA
32 pca = PCA(n_components = 2)
33 X_train = pca.fit_transform(X_train)
34 X_test = pca.transform(X_test)
35 explained_variance = pca.explained_variance_ratio_
36
37 # Fitting Decision Tree Classification to the Training set
38 from sklearn.tree import DecisionTreeClassifier
39 classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3, random_state = 0)
40 classifier.fit(X_train, y_train)
41
42 # Predicting the Test set results
43 y_pred = classifier.predict(X_test)
44
45 # Making the Confusion Matrix
46 from sklearn.metrics import confusion_matrix, accuracy_score
47 cm = confusion_matrix(y_test, y_pred)
48 accuracy = accuracy_score(y_test, y_pred)
49 print(accuracy)

```

Fig. 2.3.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

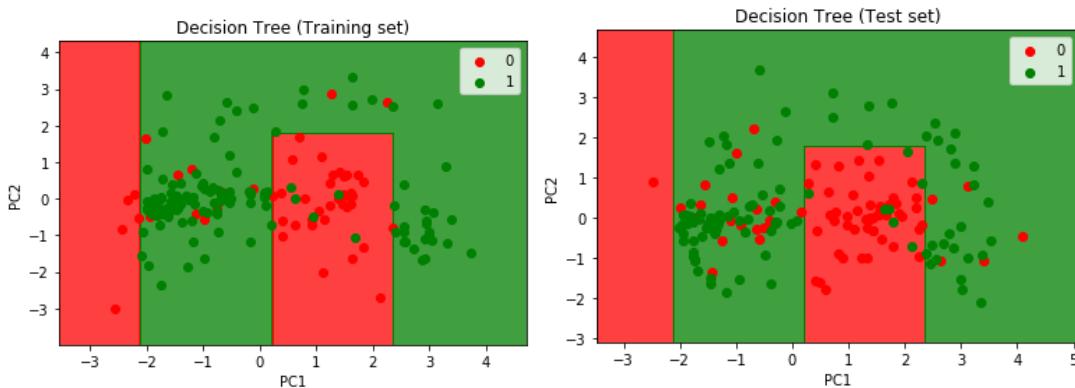


Fig. 2.3.3.b Training Set Graph

Fig. 2.3.3.c Test Set Graph

➤ **max_depth = 6**

Accuracy: =78.97%

The screenshot shows a Jupyter Notebook interface with several cells. The code cell contains Python code for loading the 'ionosphere' dataset, applying PCA, fitting a DecisionTreeClassifier with max_depth=6, and calculating accuracy. The 'Confusion matrix - NumPy' cell displays a 2x2 matrix with values: 58 (top-left), 25 (top-right), 12 (bottom-left), and 89 (bottom-right). The 'Variable explorer' cell shows variables like 'Confusion_matrix', 'X', 'X1', 'X2', 'X_set', 'X_test', 'X_train', 'accuracy', 'classifier', 'dataset', and 'explained_variance'. The 'Console' cell shows the command to import DecisionTreeClassifier and fit it to the training data, followed by the prediction of the test set and calculation of accuracy (0.7897722727272723).

```

9
10 # Importing the libraries
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pandas as pd
14
15 # Importing the dataset
16 dataset = pd.read_csv('ionosphere.csv', header = None)
17 X = dataset.iloc[:, :-1].values
18 y = dataset.iloc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29
30 # Applying PCA
31 from sklearn.decomposition import PCA
32 pca = PCA(n_components = 2)
33 X_train = pca.fit_transform(X_train)
34 X_test = pca.transform(X_test)
35 explained_variance = pca.explained_variance_ratio_
36
37 # Fitting Decision Tree Classification to the Training set
38 from sklearn.tree import DecisionTreeClassifier
39 classifier = DecisionTreeClassifier(criterion = 'entropy',max_depth = 6, random_state = 0)
40 classifier.fit(X_train, y_train)
41
42 # Predicting the Test set results
43 y_pred = classifier.predict(X_test)
44
45 # Making the Confusion Matrix
46 from sklearn.metrics import confusion_matrix, accuracy_score
47 Confusion_matrix = confusion_matrix(y_test, y_pred)
48 accuracy = accuracy_score(y_test, y_pred)
49 print(accuracy)

```

Fig. 2.3.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

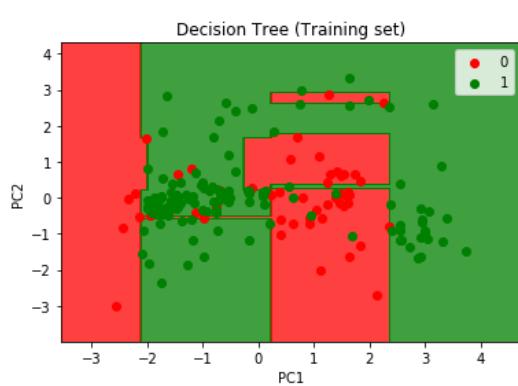


Fig. 2.3.4.b Training Set Graph

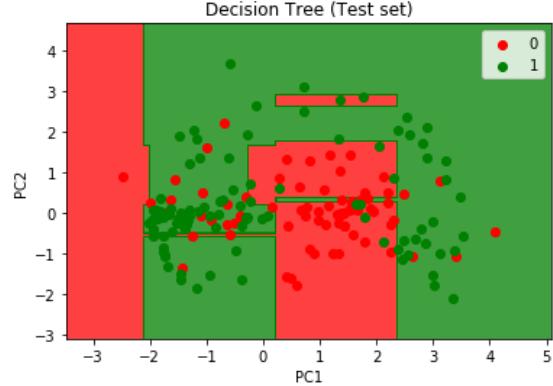


Fig. 2.3.4.c Test Set Graph

➤ **max_depth = 7**

Accuracy: =78.40%

The screenshot shows a Jupyter Notebook interface with several cells. The code cell contains Python code for loading the 'ionosphere' dataset, applying PCA, fitting a DecisionTreeClassifier with max_depth=7, and calculating accuracy. The 'Confusion matrix - NumPy' cell displays a 2x2 matrix with values: 69 (top-left), 26 (top-right), 12 (bottom-left), and 89 (bottom-right). The 'Variable explorer' cell shows variables like 'Confusion_matrix', 'X', 'X1', 'X2', 'X_set', 'X_test', 'X_train', 'accuracy', 'classifier', 'dataset', and 'explained_variance'. The 'Console' cell shows the command to import DecisionTreeClassifier and fit it to the training data, followed by the prediction of the test set and calculation of accuracy (0.7840000000000001).

```

10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25
26 # Splitting the dataset into the Training set and Test set
27 from sklearn.model_selection import train_test_split
28 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
29
30
31 # Applying PCA
32 from sklearn.decomposition import PCA
33 pca = PCA(n_components = 2)
34 X_train = pca.fit_transform(X_train)
35 X_test = pca.transform(X_test)
36 explained_variance = pca.explained_variance_ratio_
37
38 # Fitting Decision Tree Classification to the Training set
39 from sklearn.tree import DecisionTreeClassifier
40 classifier = DecisionTreeClassifier(criterion = 'entropy',max_depth = 7, random_state = 0)
41 classifier.fit(X_train, y_train)
42
43 # Predicting the Test set results
44 y_pred = classifier.predict(X_test)
45
46 # Making the Confusion Matrix
47 from sklearn.metrics import confusion_matrix, accuracy_score
48 Confusion_matrix = confusion_matrix(y_test, y_pred)
49 accuracy = accuracy_score(y_test, y_pred)
50 print(accuracy)

```

Fig. 2.3.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

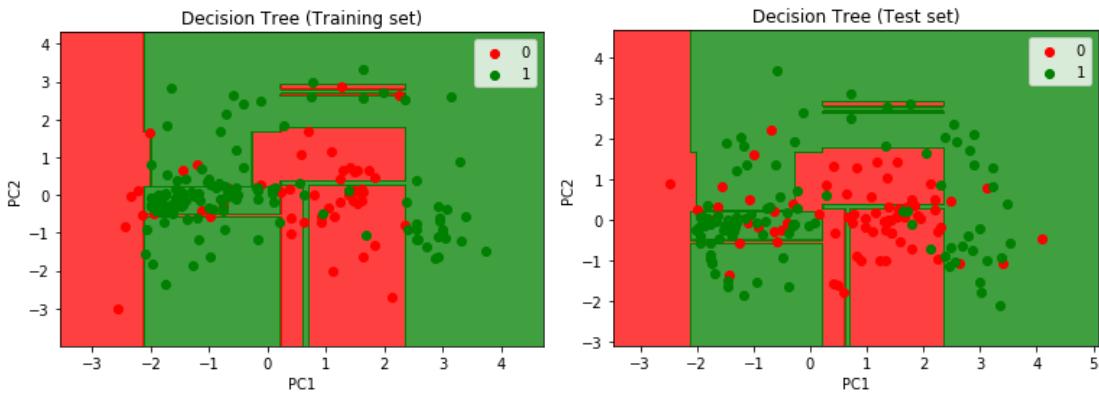


Fig. 2.3.5.b Training Set Graph

Fig. 2.3.5.c Test Set Graph

Conclusion: - The classification model with `max_depth = 50%` yields best accuracy of 82.29% among all the values of `max_depth`.

3. Random Forest classifier: -

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('ionosphere.csv', header = None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
//test size value, random_state = 0)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators //value of
n_estimators, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest (Training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest (Test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()

```

a. Test-size: - 30%

➤ N_estimators: - 3

Accuracy: - 73.58%

```

C:\Users\Lezel\Documents\ML\Projects\osphere.ipynb
In [4]: # Random Forest Classification
1 # Importing the libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5
6 # Importing the dataset
7 dataset = pd.read_csv('ionosphere.csv', header = None)
8 X = dataset.iloc[:, :-1].values
9 y = dataset.iloc[:, -1].values
10
11 from sklearn.preprocessing import LabelEncoder
12 labelencoder_y = LabelEncoder()
13 y = labelencoder_y.fit_transform(y)
14
15 # Applying PCA
16 from sklearn.decomposition import PCA
17 pca = PCA(n_components = 2)
18 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
19
20 # Applying PCA
21 X_train_pca = pca.fit_transform(X_train)
22 X_test_pca = pca.transform(X_test)
23 explained_variance = pca.explained_variance_ratio_
24
25 # Fitting Random Forest Classification to the Training set
26 from sklearn.ensemble import RandomForestClassifier
27 classifier = RandomForestClassifier(n_estimators = 3, criterion = 'entropy', random_state = 0)
28 classifier.fit(X_train, y_train)
29
30 # Predicting the Test set results
31 y_pred = classifier.predict(X_test)
32
33 # Making the Confusion Matrix
34 from sklearn.metrics import confusion_matrix, accuracy_score
35 confusion_matrix = confusion_matrix(y_test, y_pred)
36 accuracy = accuracy_score(y_test, y_pred)
37 print(accuracy)
38
39
40
41
42
43
44
45
46
47
48
49

```

Confusion matrix: NumPy array

	0	1
0	29	15
1	13	49

accuracy

RandomForestClassifier object of 'sklearn.ensemble._forest module'

dataset

explained_variance

Fig. 3.1.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

Random Forest (Training set)

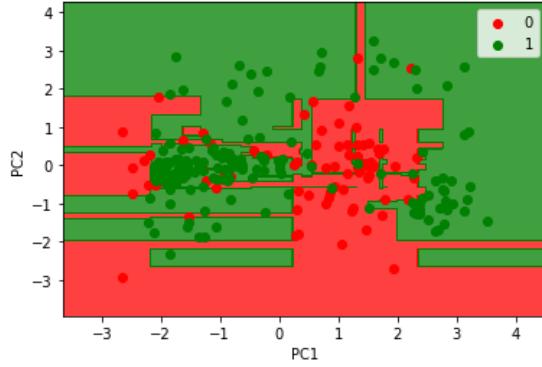


Fig. 3.1.1.b Training Set Graph

Random Forest (Test set)

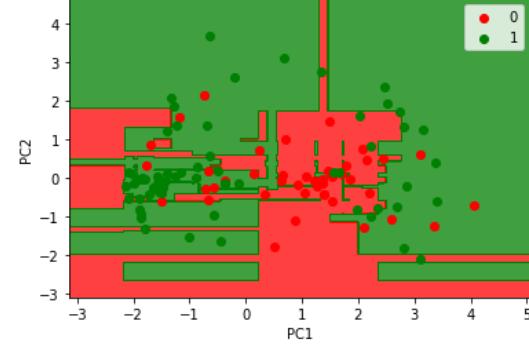


Fig. 3.1.1.c Test Set Graph

➤ N_estimators: - 4

Accuracy: - 73.58%

```

C:\Users\Lezel\Documents\ML\Projects\osphere.ipynb
In [4]: # Random Forest Classification
1 # Importing the libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5
6 # Importing the dataset
7 dataset = pd.read_csv('ionosphere.csv', header = None)
8 X = dataset.iloc[:, :-1].values
9 y = dataset.iloc[:, -1].values
10
11 from sklearn.preprocessing import LabelEncoder
12 labelencoder_y = LabelEncoder()
13 y = labelencoder_y.fit_transform(y)
14
15 # Applying PCA
16 from sklearn.decomposition import PCA
17 pca = PCA(n_components = 2)
18 X_train_pca = pca.fit_transform(X_train)
19 X_test_pca = pca.transform(X_test)
20 explained_variance = pca.explained_variance_ratio_
21
22 # Fitting Random Forest classification to the Training set
23 from sklearn.ensemble import RandomForestClassifier
24 classifier = RandomForestClassifier(n_estimators = 4, criterion = 'entropy', random_state = 0)
25 classifier.fit(X_train, y_train)
26
27 # Predicting the Test set results
28 y_pred = classifier.predict(X_test)
29
30 # Making the Confusion Matrix
31 from sklearn.metrics import confusion_matrix, accuracy_score
32 confusion_matrix = confusion_matrix(y_test, y_pred)
33 accuracy = accuracy_score(y_test, y_pred)
34 print(accuracy)
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

Confusion matrix: NumPy array

	0	1
0	38	14
1	14	65

accuracy

RandomForestClassifier object of 'sklearn.ensemble._forest module'

dataset

explained_variance

Fig. 3.1.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

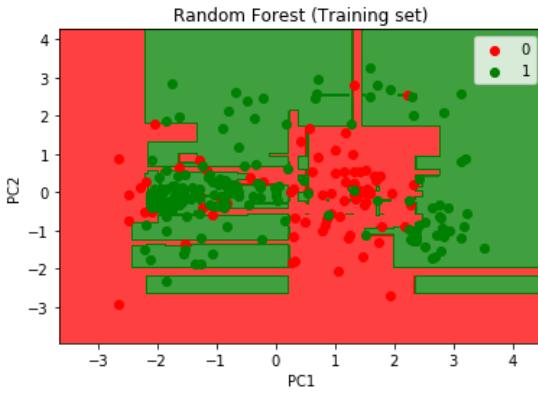


Fig. 3.1.2.b Training Set Graph

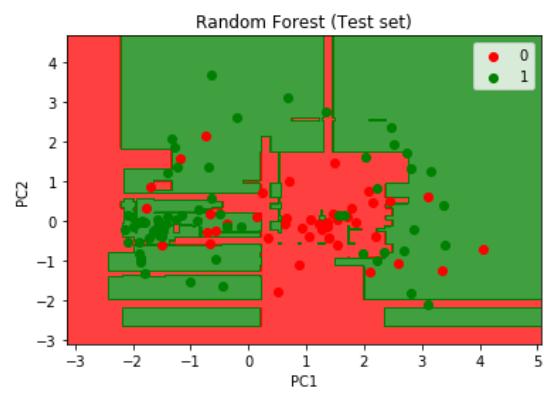


Fig. 3.1.2.c Test Set Graph

➤ N_estimators: - 5
Accuracy: - 74.52%

```

9 # Random Forest Classification
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting Random Forest Classification to the Training set
37 from sklearn.ensemble import RandomForestClassifier
38 classifier = RandomForestClassifier(n_estimators = 5, criterion = 'entropy', random_state = 0)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49

```

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[29 15] [12 58]]
X	Array of float64	(351, 34)	[[], 1, ..., 0, ..., 0.99539 ...]
X1	Array of float64	(776, 821)	[[-3.14223767, -3.13223767, -3.122 ...]
X2	Array of float64	(776, 821)	[[-3.08534372, -3.08534372, -3.085 ...]
X_set	Array of float64	(106, 2)	[[-0.43544696, -1.64886471], [3.10837101, -2.08834372]]
X_test	Array of float64	(106, 2)	[[-0.43544696, -1.64886471], [3.10837101, -2.08834372]]
X_train	Array of float64	(245, 2)	[[], 1, 3557643, -0.09002543], [2.78867395, -1.56053569]
accuracy	float64	1	0.7452810188679245
classifier	ensemble._forest.RandomForestClassifier	1	RandomForestClassifier object of 'sklearn.ensemble._forest module'
dataset	Dataframe	(351, 35)	Column names: 0, 1, 2, 3, 4, 5, ...
explained_variance	Array of float64	(, 2)	[0.3190639, 0.13642327]

```

Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 5, criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
Confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
0.7452810188679245

```

Fig. 3.1.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

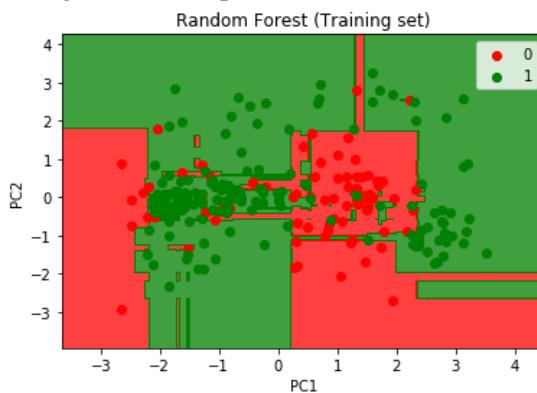


Fig. 3.1.3.b Training Set Graph

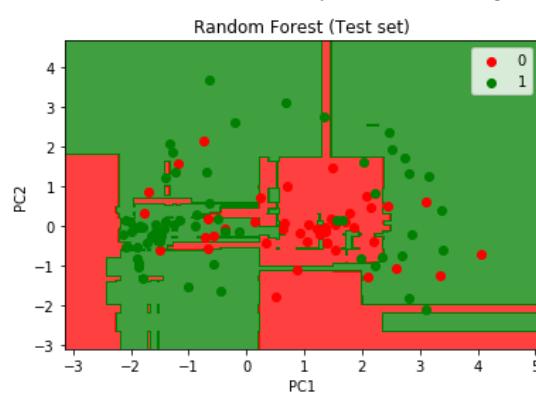


Fig. 3.1.3.c Test Set Graph

➤ N_estimators: - 6
Accuracy: - 75.47%

```

8 # Random Forest Classification
9
10 # Importing the libraries
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pandas as pd
14
15 # Importing the dataset
16 dataset = pd.read_csv('ionosphere.csv', header = None)
17 X = dataset.iloc[:, :-1].values
18 y = dataset.iloc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24 # Splitting the dataset into the Training set and Test set
25 from sklearn.model_selection import train_test_split
26 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
27
28 # Applying PCA
29 from sklearn.decomposition import PCA
30 pca = PCA(n_components = 2)
31 X_train = pca.fit_transform(X_train)
32 X_test = pca.transform(X_test)
33 explained_variance = pca.explained_variance_ratio_
34
35 # Fitting Random Forest Classification to the Training set
36 from sklearn.ensemble import RandomForestClassifier
37 classifier = RandomForestClassifier(n_estimators = 6, criterion = 'entropy', random_state = 0)
38 classifier.fit(X_train, y_train)
39
40 # Predicting the Test set results
41 y_pred = classifier.predict(X_test)
42
43 # Making the Confusion Matrix
44 from sklearn.metrics import confusion_matrix, accuracy_score
45 confusion_matrix = confusion_matrix(y_test, y_pred)
46 accuracy = accuracy_score(y_test, y_pred)
47 print(accuracy)
48

```

	0	1
0	28	16
1	10	52

Confusion matrix - NumPy -

RandomForestClassifier object of sklearn.ensemble._forest module

accuracy: 0.7547169811120755

Fig. 3.1.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

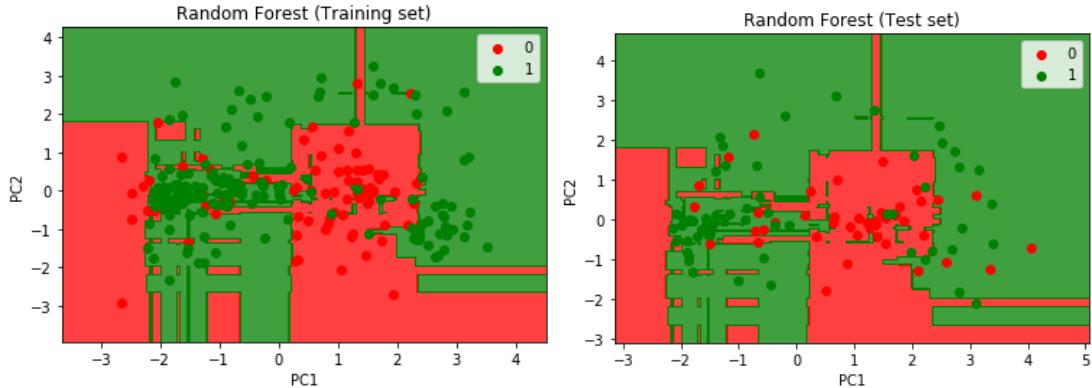


Fig. 3.1.4.b Training Set Graph

Fig. 3.1.4.c Test Set Graph

➤ N_estimators: - 7
Accuracy: - 75.47%

```

8 # Random Forest Classification
9
10 # Importing the libraries
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pandas as pd
14
15 # Importing the dataset
16 dataset = pd.read_csv('ionosphere.csv', header = None)
17 X = dataset.iloc[:, :-1].values
18 y = dataset.iloc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24 # Splitting the dataset into the Training set and Test set
25 from sklearn.model_selection import train_test_split
26 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
27
28 # Applying PCA
29 from sklearn.decomposition import PCA
30 pca = PCA(n_components = 2)
31 X_train = pca.fit_transform(X_train)
32 X_test = pca.transform(X_test)
33 explained_variance = pca.explained_variance_ratio_
34
35 # Fitting Random Forest Classification to the Training set
36 from sklearn.ensemble import RandomForestClassifier
37 classifier = RandomForestClassifier(n_estimators = 7, criterion = 'entropy', random_state = 0)
38 classifier.fit(X_train, y_train)
39
40 # Predicting the Test set results
41 y_pred = classifier.predict(X_test)
42
43 # Making the Confusion Matrix
44 from sklearn.metrics import confusion_matrix, accuracy_score
45 confusion_matrix = confusion_matrix(y_test, y_pred)
46 accuracy = accuracy_score(y_test, y_pred)
47 print(accuracy)
48

```

	0	1
0	28	16
1	10	52

Confusion_matrix - NumPy -

RandomForestClassifier object of sklearn.ensemble._forest module

accuracy: 0.7547169811120755

In [13]:

Fig. 3.1.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

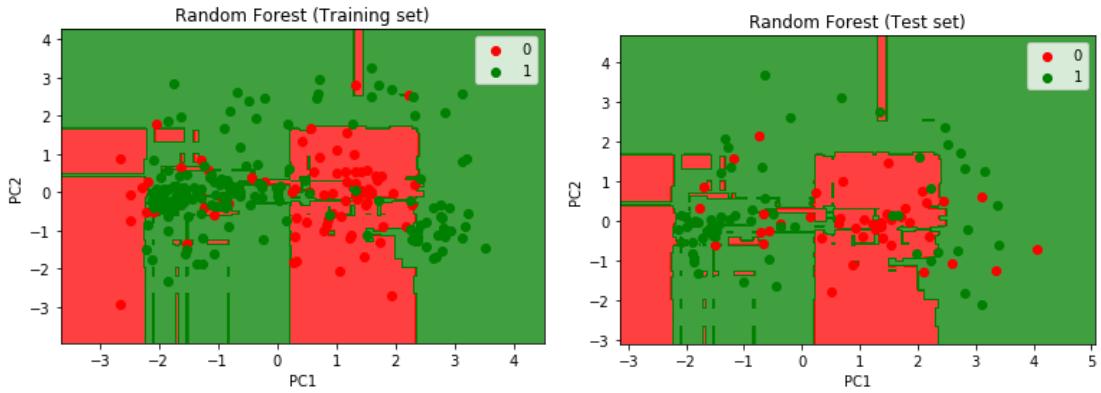


Fig. 3.1.5.b Training Set Graph

Fig. 3.1.5.c Test Set Graph

Conclusion: - The classification with test size = 30% gives best accuracy score = 75.47% with N_estimators = 6 and 7.

b. Test-size: - 40%

➤ N_estimators: - 3

Accuracy: - 74.46%

```

8 # Random Forest Classification
9
10 # Importing the libraries
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import pandas as pd
14
15 # Importing the dataset
16 dataset = pd.read_csv('ionosphere.csv', header = None)
17 X = dataset.iloc[:, :-1].values
18 y = dataset.iloc[:, -1].values
19
20 from sklearn.preprocessing import LabelEncoder
21 labelencoder_y = LabelEncoder()
22 y = labelencoder_y.fit_transform(y)
23
24 # Splitting the dataset into the Training set and Test set
25 from sklearn.model_selection import train_test_split
26 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 0)
27
28 # Applying PCA
29 from sklearn.decomposition import PCA
30 pca = PCA(n_components = 2)
31 X_train = pca.fit_transform(X_train)
32 X_test = pca.transform(X_test)
33 explained_variance = pca.explained_variance_ratio_
34
35 # Fitting Random Forest Classification to the Training set
36 from sklearn.ensemble import RandomForestClassifier
37 classifier = RandomForestClassifier(n_estimators = 3, criterion = 'entropy', random_state = 0)
38 classifier.fit(X_train, y_train)
39
40 # Predicting the Test set results
41 y_pred = classifier.predict(X_test)
42
43 # Making the Confusion Matrix
44 from sklearn.metrics import confusion_matrix, accuracy_score
45 Confusion_matrix = confusion_matrix(y_test, y_pred)
46 accuracy = accuracy_score(y_test, y_pred)
47 print(accuracy)

```

Fig. 3.2.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

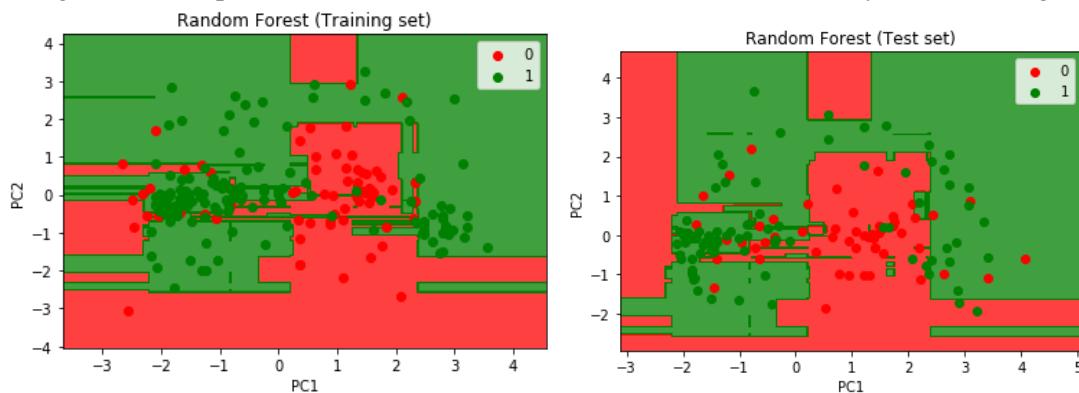


Fig. 3.2.1.b Training Set Graph

Fig. 3.2.1.c Test Set Graph

➤ N_estimators: - 4
Accuracy: - 74.46%

The screenshot shows a Jupyter Notebook environment. On the left, the code for a Random Forest classifier is displayed. In the center, a 'Confusion matrix - NumPy' dialog box shows a 2x2 matrix: [[36, 21], [15, 69]]. On the right, the Python console displays the following output:

```

Name      Type     Size   Value
Confusion_matrix  Array of int64 (2, 2) [[ 36, 21], [15, 69]]
X         Array of float64 (351, 34) [[...]
X1        Array of float64 (761, 819) [[...]
X2        Array of float64 (761, 819) [[...]
X_set     Array of float64 (141, 2) [[ 2.41197298, 1.75639396]
X_test    Array of float64 (141, 2) [[ 0.41192972, 1.75639396]
X_train   Array of float64 (210, 2) [[ 2.6163784, 0.8778421]
accuracy  Float64    1       0.7446008510638298
classifier ensemble_Forest.RandomForestClassifier 1
dataset   Dataframe  (351, 35) Column names: 0, 1, 2, 3, 4, 5, ...
explained_variance  Array of float64 (2,) [0.31151855 0.12930431]

```

Fig. 3.2.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

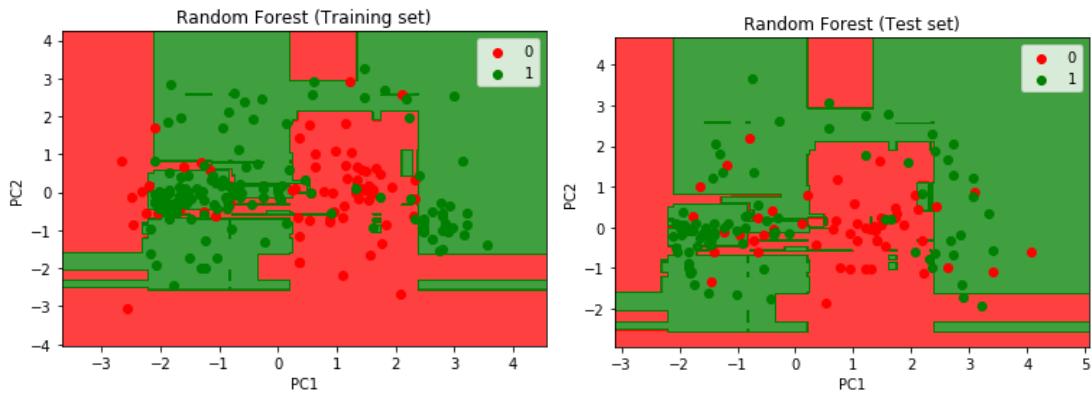


Fig. 3.2.2.b Training Set Graph

➤ N_estimators: - 5
Accuracy: - 78.01%

Fig. 3.2.2.c Test Set Graph

The screenshot shows a Jupyter Notebook environment. On the left, the code for a Random Forest classifier is displayed. In the center, a 'Confusion matrix - NumPy' dialog box shows a 2x2 matrix: [[34, 23], [8, 76]]. On the right, the Python console displays the following output:

```

Name      Type     Size   Value
Confusion_matrix  Array of int64 (2, 2) [[ 34, 23], [ 8, 76]]
X         Array of float64 (351, 34) [[...]
X1        Array of float64 (761, 819) [[...]
X2        Array of float64 (761, 819) [[...]
X_set     Array of float64 (141, 2) [[ 2.41188775, -3.10688775, -3.496...
X_test    Array of float64 (141, 2) [[ 0.41189272, 1.75639396]
X_train   Array of float64 (210, 2) [[ 2.6163784, 0.8778421]
accuracy  Float64    1       0.780118439716312
classifier ensemble_Forest.RandomForestClassifier 1
dataset   Dataframe  (351, 35) Column names: 0, 1, 2, 3, 4, 5, ...
explained_variance  Array of float64 (2,) [0.31151855 0.12930431]

```

Fig. 3.2.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

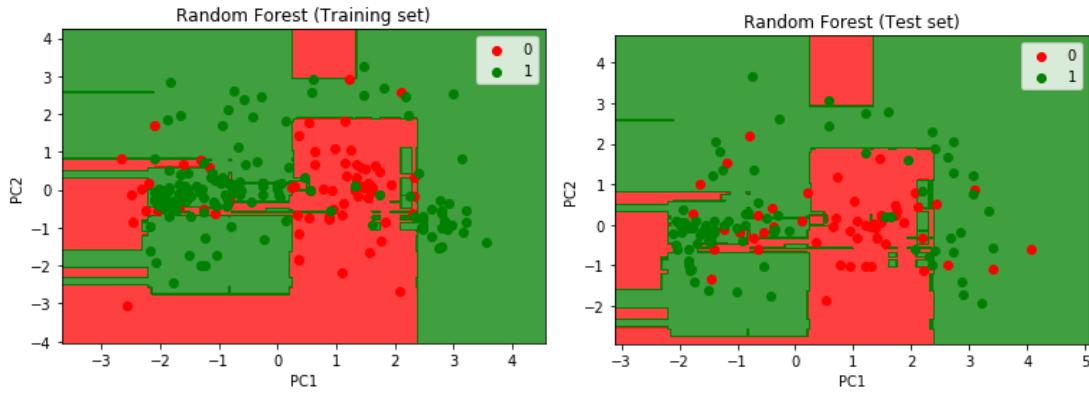


Fig. 3.2.3.b Training Set Graph

Fig. 3.2.3.c Test Set Graph

➤ N_estimators: - 6
Accuracy: - 78.72%

```

8
9 # Random Forest Classification
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('lensesph.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting Random Forest Classification to the Training set
37 from sklearn.ensemble import RandomForestClassifier
38 classifier = RandomForestClassifier(n_estimators = 6, criterion = 'entropy', random_state = 0)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)

```

Name	Type	Size	Value						
Confusion_matrix	Array of int64	(2, 2)	<table border="1"> <tr><td>78</td><td>19</td></tr> <tr><td>11</td><td>73</td></tr> </table>	78	19	11	73		
78	19								
11	73								
X	Array of float64	(351, 36)	<table border="1"> <tr><td>1</td><td>...</td></tr> <tr><td>3</td><td>...</td></tr> <tr><td>5</td><td>...</td></tr> </table>	1	...	3	...	5	...
1	...								
3	...								
5	...								
X2	Array of float64	(351, 89)	<table border="1"> <tr><td>1</td><td>...</td></tr> <tr><td>3</td><td>...</td></tr> <tr><td>5</td><td>...</td></tr> </table>	1	...	3	...	5	...
1	...								
3	...								
5	...								
X_set	Array of float64	(341, 2)	<table border="1"> <tr><td>-0.0591298</td><td>-1.764082</td></tr> <tr><td>1.2016872</td><td>1.9559336</td></tr> <tr><td>1.4016872</td><td>1.9559336</td></tr> </table>	-0.0591298	-1.764082	1.2016872	1.9559336	1.4016872	1.9559336
-0.0591298	-1.764082								
1.2016872	1.9559336								
1.4016872	1.9559336								
X_test	Array of float64	(141, 2)	<table border="1"> <tr><td>2.4016872</td><td>-1.9559336</td></tr> <tr><td>2.4016974</td><td>0.03740011</td></tr> <tr><td>1.4016974</td><td>0.0548579</td></tr> </table>	2.4016872	-1.9559336	2.4016974	0.03740011	1.4016974	0.0548579
2.4016872	-1.9559336								
2.4016974	0.03740011								
1.4016974	0.0548579								
X_train	Array of float64	(210, 2)	<table border="1"> <tr><td>-2.9593936</td><td>-2.9559336</td></tr> <tr><td>-2.9593936</td><td>-2.9559336</td></tr> <tr><td>-2.9593936</td><td>-2.9559336</td></tr> </table>	-2.9593936	-2.9559336	-2.9593936	-2.9559336	-2.9593936	-2.9559336
-2.9593936	-2.9559336								
-2.9593936	-2.9559336								
-2.9593936	-2.9559336								
accuracy	float64	1	0.787248402551015						
classifier	ensemble._forest.RandomForestClassifier	1	ensemble._forest.RandomForestClassifier object of sklearn.ensemble._forest module						
dataset	Dataframe	(351, 36)	column names: 0, 1, 2, 3, 4, 5, ...						
explained_variance	Array of float64	(2,)	[0.3125185, 0.1285841]						

Fig. 3.2.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

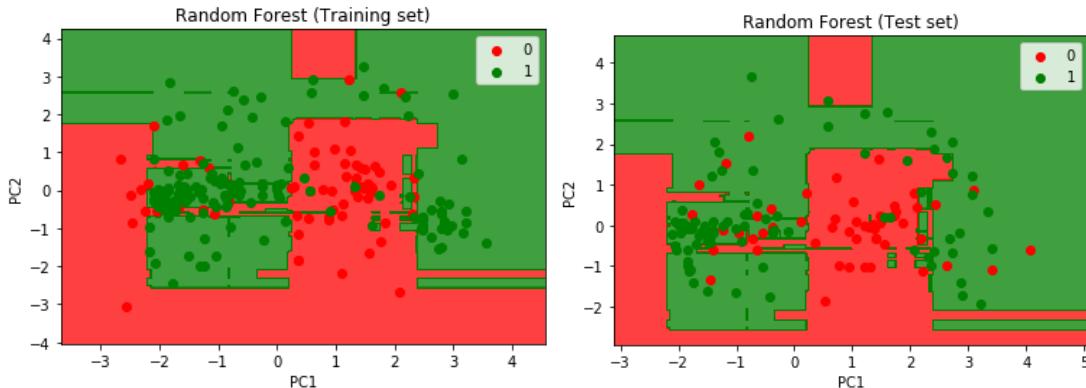


Fig. 3.2.4.b Training Set Graph

Fig. 3.2.4.c Test Set Graph

➤ N_estimators: -7
Accuracy: - 76.59%

```

9 # Random Forest Classification
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 labelencoder_y = LabelEncoder()
23 y = labelencoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting Random Forest Classification to the Training set
37 from sklearn.ensemble import RandomForestClassifier
38 classifier = RandomForestClassifier(n_estimators = 7, criterion = 'entropy', random_state = 0)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)
49

```

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[35 22] [11 73]]
X	Array of float64	(351, 34)	[[1. ... 0.99539 ...
X1	Array of float64	(761, 819)	[[1. ... 0.99539 ...
X2	Array of float64	(761, 819)	[[-3.11608775 -3.10608775 -3.096... 5 ...
X_set	Array of float64	(141, 2)	[[-3.93593396 -2.93593396 -2.935... -2 ...
X_test	Array of float64	(141, 2)	[[-3.93593396 -2.93593396 -2.935... 3.20148877 -1.93593396
X_train	Array of float64	(210, 2)	[[-2.01693764 0.81978402 1.464941 0.85140519]
accuracy	float64	1	0.7659574468085106
classifier	RandomForestClassifier object of sklearn.ensemble module	1	
dataset	DataFrame	(351, 35)	Column names: 0, 1, 2, 3, 4, 5, ...
explained_variance	Array of float64	(2,)	[0.31151855 0.12930431]

Fig. 3.2.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

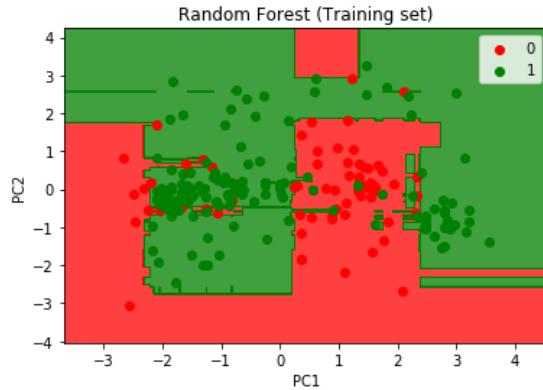


Fig. 3.2.5.b Training Set Graph

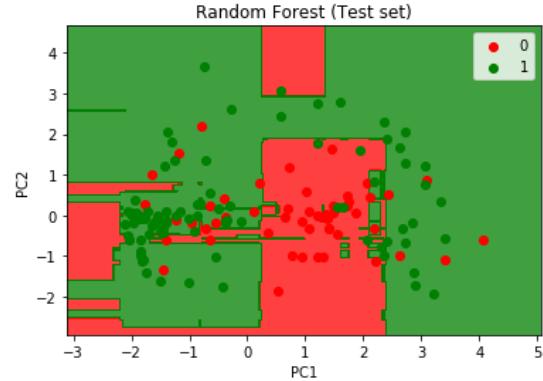


Fig. 3.2.5.c Test Set Graph

Conclusion: - The classification with test size = 30% gives best accuracy score = 78.72% with N_estimators = 6.

c. Test-size: - 50%

➤ N_estimators: - 3

Accuracy: - 75%

The screenshot shows a Jupyter Notebook cell with Python code for Random Forest Classification. The code imports libraries, reads the 'ionosphere' dataset, splits it into training and test sets, applies PCA, fits a Random Forest classifier with 3 estimators, and prints the accuracy. A confusion matrix is displayed in a separate window.

```

8
9 # Random Forest Classification
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 LabelEncoder_y = LabelEncoder()
23 y = LabelEncoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting Random Forest Classification to the Training set
37 from sklearn.ensemble import RandomForestClassifier
38 classifier = RandomForestClassifier(n_estimators = 3, criterion = 'entropy', random_state = 0)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)

```

Fig. 3.3.1.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

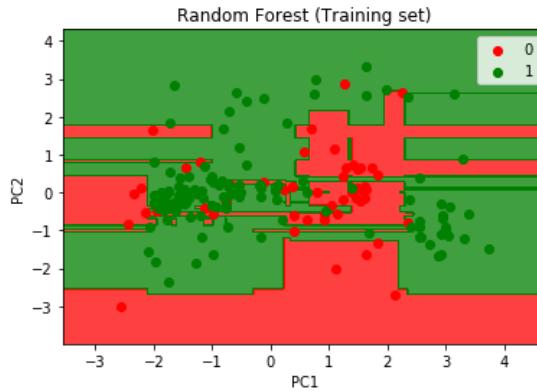


Fig. 3.3.1.b Training Set Graph

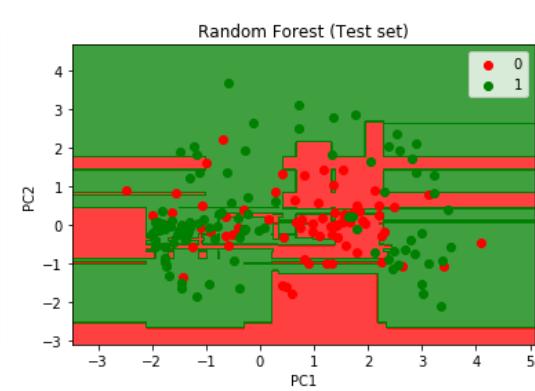


Fig. 3.3.1.c Test Set Graph

➤ N_estimators: - 4

Accuracy: - 76.70%

The screenshot shows a Jupyter Notebook cell with Python code for Random Forest Classification. The code imports libraries, reads the 'ionosphere' dataset, splits it into training and test sets, applies PCA, fits a Random Forest classifier with 4 estimators, and prints the accuracy. A confusion matrix is displayed in a separate window.

```

8
9 # Random Forest Classification
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import pandas as pd
15
16 # Importing the dataset
17 dataset = pd.read_csv('ionosphere.csv', header = None)
18 X = dataset.iloc[:, :-1].values
19 y = dataset.iloc[:, -1].values
20
21 from sklearn.preprocessing import LabelEncoder
22 LabelEncoder_y = LabelEncoder()
23 y = LabelEncoder_y.fit_transform(y)
24
25 # Splitting the dataset into the Training set and Test set
26 from sklearn.model_selection import train_test_split
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
28
29 # Applying PCA
30 from sklearn.decomposition import PCA
31 pca = PCA(n_components = 2)
32 X_train = pca.fit_transform(X_train)
33 X_test = pca.transform(X_test)
34 explained_variance = pca.explained_variance_ratio_
35
36 # Fitting Random Forest Classification to the Training set
37 from sklearn.ensemble import RandomForestClassifier
38 classifier = RandomForestClassifier(n_estimators = 4, criterion = 'entropy', random_state = 0)
39 classifier.fit(X_train, y_train)
40
41 # Predicting the Test set results
42 y_pred = classifier.predict(X_test)
43
44 # Making the Confusion Matrix
45 from sklearn.metrics import confusion_matrix, accuracy_score
46 Confusion_matrix = confusion_matrix(y_test, y_pred)
47 accuracy = accuracy_score(y_test, y_pred)
48 print(accuracy)

```

Fig. 3.3.2.a Snapshot of Error free code, confusion matrix and Accuracy in console log.



Fig. 3.3.2.b Training Set Graph

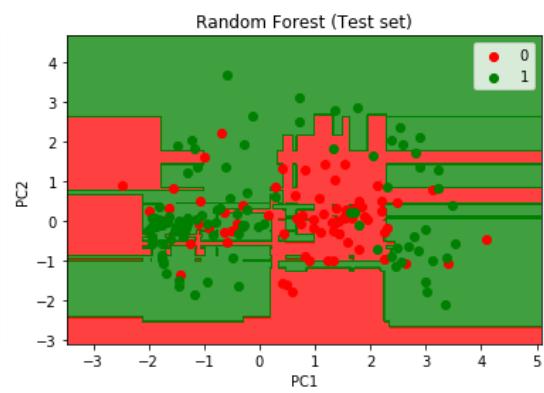


Fig. 3.3.2.c Test Set Graph

➤ N_estimators: - 5
Accuracy: - 76.13%

```

C:\Users\steve\Documents\ML\Projects\ionosphere.ipynb
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.decomposition import PCA
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import confusion_matrix, accuracy_score
        dataset = pd.read_csv('ionosphere.csv', header = None)
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, -1].values
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50, random_state = 0)
        pca = PCA(n_components = 2)
        X_train = pca.fit_transform(X_train)
        X_test = pca.transform(X_test)
        explained_variance = pca.explained_variance_ratio_
        classifier = RandomForestClassifier(n_estimators = 5, criterion = 'entropy', random_state = 0)
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        Confusion_matrix = confusion_matrix(y_test, y_pred)
        accuracy = accuracy_score(y_test, y_pred)
        print(accuracy)
    
```

		0	1
0	43	32	
1	10	91	

Name	Type	Size	Value
Confusion_matrix	Array of int64	(2, 2)	[[43 32] [10 91]]
X	Array of float64	(351, 34)	([...], 0, 0.99539 ..., 1, ...)
X1	Array of float64	(788, 857)	([...], 3.47788644, -3.46788644, -4.457..., 5, ...)
X2	Array of float64	(788, 857)	([...], 3.13079976, -3.11079976, -3.110..., 1, ...)
X_set	Array of float64	(176, 2)	([...], 0.3764691, -1.63415052)
X_test	Array of float64	(176, 2)	([...], 3.3959077, -2.11079976)
X_train	Array of float64	(175, 2)	([...], 3.3959077, -2.11079976)
accuracy	float64	1	0.7613636363636364
classifier	ensemble.Forest.RandomForestClassifier	1	RandomForestClassifier object of 'sklearn.ensemble._Forest' module
dataset	DataFrame	(351, 35)	Column names: 0, 1, 2, 3, 4, 5, ...
explained_variance	Array of float64	(2,)	[0.32942732 0.14088766]

Fig. 3.3.3.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

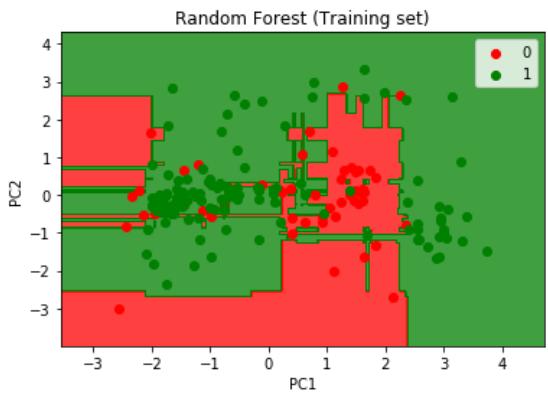


Fig. 3.3.3.b Training Set Graph

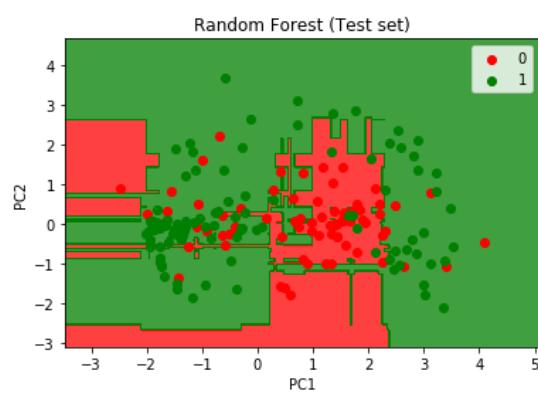


Fig. 3.3.3.c Test Set Graph

➤ N_estimators: - 6

Accuracy: - 78.97%

The screenshot shows a Jupyter Notebook interface. On the left is the code for a Random Forest classifier. In the center is a 'Confusion_matrix - NumPy' window displaying a 2x2 matrix:

0	1	
0	49	26
1	11	90

To the right is a variable explorer showing various data structures like X, X1, X2, X_set, X_test, X_train, accuracy, classifier, dataset, and explained_variance.

Fig. 3.3.4.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

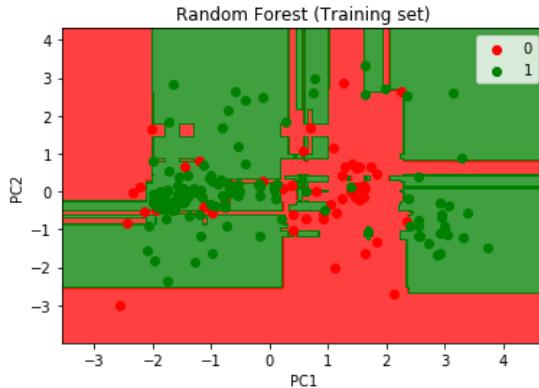


Fig. 3.3.4.b Training Set Graph

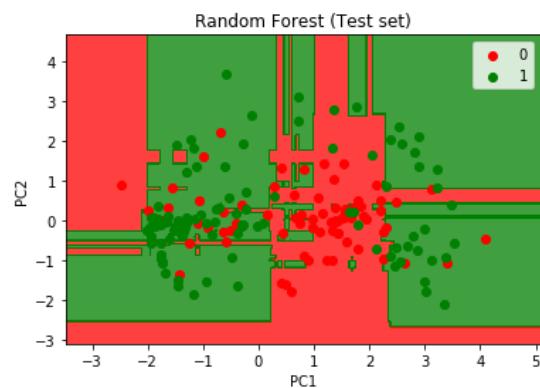


Fig. 3.3.4.c Test Set Graph

➤ N_estimators: - 7

Accuracy: - 79.54%

The screenshot shows a Jupyter Notebook interface. On the left is the same code as in Fig. 3.3.4.a. In the center is a 'Confusion_matrix - NumPy' window displaying a 2x2 matrix:

0	1	
0	47	28
1	8	93

To the right is a variable explorer showing the same data structures as before.

Fig. 3.3.5.a Snapshot of Error free code, confusion matrix and Accuracy in console log.

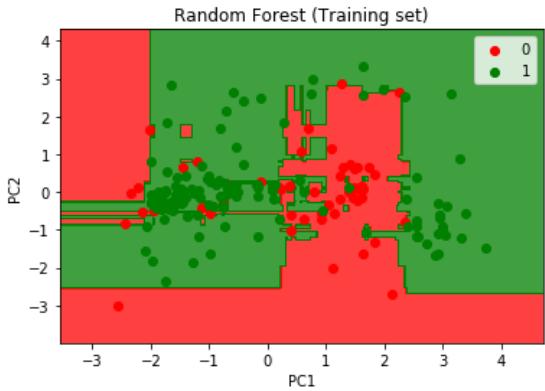


Fig. 3.3.5.b Training Set Graph



Fig. 3.3.5.c Test Set Graph

Conclusion: - The classification with test size = 30% gives best accuracy score = 79.54% with N_estimators = 7.

OBSERVATIONS: - The following points have been concluded from the above analysis of data using various classifier models: -

- With K nearest neighbour classifier model the best accuracy score is yielded with test size = 50% and k-value = 4. **ACCURACY_SCORE – 82.38%**
- With decision tree classifier model the best accuracy score is yielded with test size = 50% and max_depth = 3. **ACCURACY_SCORE – 82.29%**
- With random forest classifier model the best accuracy score is yielded with test size = 50% and n_estimator = 7. **ACCURACY_SCORE – 79.54%**

Hence, the best classification model considering the above constraint values is K-Nearest Neighbours with accuracy of 82.38 %, test size = 50 % for k-value = 4.

NEED FOR APPLYING PCA: - The above models have been implemented with PCA (Principle Component Analysis) in order to visualize the models graphically. Since the original dataset had 34 features so it's not possible to plot them so we used PCA with n_components = 2 to have a plot for the dataset as it's easy to conclude from the visualization of dataset.

CONFUSION MATRIX: - A confusion matrix is a matrix (table) that can be used to measure the performance of a machine learning algorithm, usually a supervised learning one. Each row of the confusion matrix represents the instances of an actual class and each column represents the instances of a predicted class.

	<i>Class 1 Predicted</i>	<i>Class 2 Predicted</i>
<i>Class 1 Actual</i>	TP	FN
<i>Class 2 Actual</i>	FP	TN

TP: True Positive
FP: False Positive
FN: - False Negative
TN: - True Negative

Fig. 1.d Confusion Matrix

ACCURACY SCORE :-

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Fraction predicted
correctly

REFERENCES :-

- [https://en.wikipedia.org/wiki/K-nearest neighbors algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)
- <https://www.geeksforgeeks.org/decision-tree/>
- [https://en.wikipedia.org/wiki/Random forest](https://en.wikipedia.org/wiki/Random_forest)
- <https://medium.com/@sonish.sivarajkumar/k-nearest-neighbours-knn-algorithm-9900c1427726>
- <https://www.datacamp.com/community/tutorials/decision-tree-classification-python>
- <https://medium.com/analytics-vidhya/machine-learning-decision-trees-and-random-forest-classifiers-81422887a544>
- https://www.python-course.eu/confusion_matrix.php#:~:text=A%20confusion%20matrix%20is%20a,instances%20of%20a%20predicted%20class.
- <https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>
- <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>