Linear Regression:

```
import numpy as np

class LinearRegression:
    def __init__(self, lr=0.001, n_iters=1000):
        self.lr = lr
        self.n_iters = n_iters
        self.w = None
        self.b = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.w = np.zeros(n_features)
        self.b = 0

        for _ in range(self.n_iters):
            y_pred = np.dot(X, self.w) + self.b

            dw = (1/n_samples) * np.dot(X.T, (y_pred - y))
            db = (1/n_samples) * np.sum(y_pred - y)

            self.w = self.w - self.lr * dw
            self.b = self.b - self.lr * db

    def predict(self, X):
        y_pred = np.dot(X, self.w) + self.b
        return y_pred
```

Logistic Regression:

```
import numpy as np

def sigmoid(x):
    return 1/(1+np.exp(-x))

class LogisticRegression():

    def __init__(self, lr=0.001, n_iters=1000):
        self.lr = lr
        self.n_iters = n_iters
        self.w = None
        self.b = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.w = np.zeros(n_features)
        self.b = 0.0

        for i in range(self.n_iters):
            linear_pred = np.dot(X, self.w) + self.b
            predictions = sigmoid(linear_pred)

            dw = (1/n_samples) * np.dot(X.T, (predictions - y))
            db = (1/n_samples) * np.sum(predictions-y)

            self.w = self.w - self.lr*dw
            self.b = self.b - self.lr*db


    def predict(self, X):
        linear_pred = np.dot(X, self.w) + self.b
        y_pred = sigmoid(linear_pred)
        class_pred = [0 if y<=0.5 else 1 for y in y_pred]
        return class_pred
```

1-6

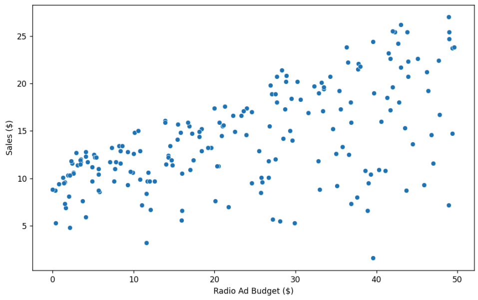
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from LinearRegression import LinearRegression
from sklearn.metrics import mean_squared_error

# load dataset
df = pd.read_csv("dataset/advertising/advertising.csv")

# i. Understand the Dataset & cleanup (if required).
print("\nSample Data: \n", df.head())

# data info
print("Data Info: \n")
df.info()

# describe data
print("\nData Description: \n", df.describe())

# check for missing data
print("\nMissing Values: \n", df.isnull().sum())

# if data consists missing data
df = df.dropna(subset=["TV Ad Budget ($)",  "Radio Ad Budget ($)",
"Newspaper Ad Budget ($)",  "Sales ($)"])

# Visualize data
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x="Radio Ad Budget ($)", y="Sales ($)")
plt.show()

# sales w.r.t Radio features.
X1 = df[['Radio Ad Budget ($)']]
y1 = df['Sales ($)']

# sales w.r.t attribute tv.
X2 = df[['TV Ad Budget ($)']]
y2 = df['Sales ($)']
```

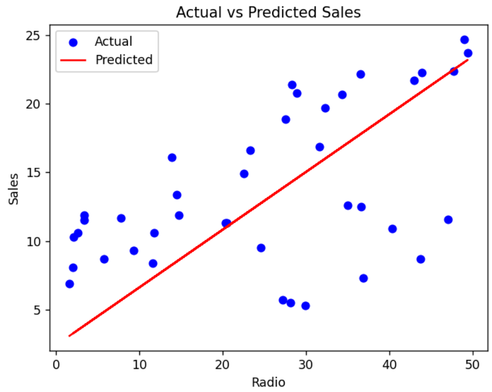
```python
# sales w.r.t attribute newspaper.
X3 = df[['Newspaper Ad Budget ($)']]
y3 = df['Sales ($)']

# sales w.r.t Radio and TV
X4 = df[['Radio Ad Budget ($)', 'TV Ad Budget ($)']]
y4 = df['Sales ($)']

# sales w.r.t Newspaper and TV
X5 = df[['Newspaper Ad Budget ($)', 'TV Ad Budget ($)']]
y5 = df['Sales ($)']

# sales w.r.t Newspaper and Radio
X6 = df[['Newspaper Ad Budget ($)', 'Radio Ad Budget ($)']]
y6 = df['Sales ($)']

X_train, X_test, y_train, y_test = train_test_split(X1, y1,
test_size=0.2, random_state=42)

# use only for tv column
# X_train = X_train/10
# X_test = X_test/10

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# iii. Also evaluate the model using scores RMSE

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")


# 1-3
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Predicted')
plt.title('Actual vs Predicted Sales')
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

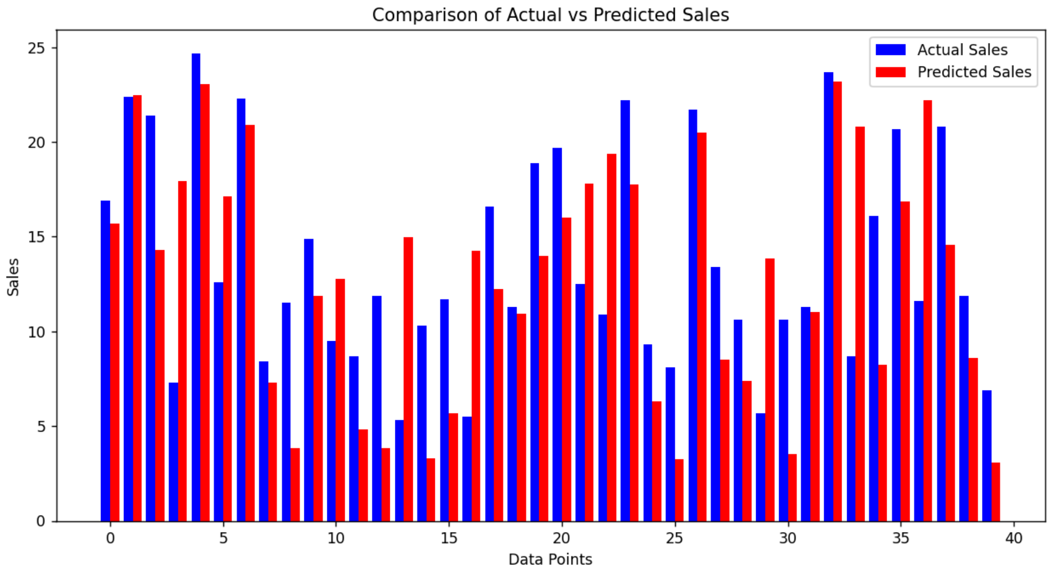
```python
# 4-6
plt.figure(figsize=(12,6))

indices = np.arange(len(y_test))
plt.bar(indices - 0.2, y_test, width=0.4, label='Actual Sales',
color='blue')
plt.bar(indices + 0.2, y_pred, width=0.4, label='Predicted Sales',
color='red')

plt.title('Comparison of Actual vs Predicted Sales')
plt.xlabel('Data Points')
plt.ylabel('Sales')
plt.legend()

plt.show()
```

```
Sample Data:
   Unnamed: 0  TV Ad Budget ($)  Radio Ad Budget ($)  Newspaper Ad Budget ($)  Sales ($)
0           1             230.1                 37.8                     69.2       22.1
1           2              44.5                 39.3                     45.1       10.4
2           3              17.2                 45.9                     69.3        9.3
3           4             151.5                 41.3                     58.5       18.5
4           5             180.8                 10.8                     58.4       12.9
Data Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0               200 non-null    int64
 1   TV Ad Budget ($)         200 non-null    float64
 2   Radio Ad Budget ($)      200 non-null    float64
 3   Newspaper Ad Budget ($)  200 non-null    float64
 4   Sales ($)                200 non-null    float64
dtypes: float64(4), int64(1)
```

```
Data Description:
       Unnamed: 0  TV Ad Budget ($)  Radio Ad Budget ($)  Newspaper Ad Budget ($)   Sales ($)
count  200.000000        200.000000           200.000000               200.000000  200.000000
mean   100.500000        147.042500            23.264000                30.554000   14.022500
std     57.879185         85.854236            14.846809                21.778621    5.217457
min      1.000000          0.700000             0.000000                 0.300000    1.600000
25%     50.750000         74.375000             9.975000                12.750000   10.375000
50%    100.500000        149.750000            22.900000                25.750000   12.900000
75%    150.250000        218.825000            36.525000                45.100000   17.400000
max    200.000000        296.400000            49.600000               114.000000   27.000000

Missing Values:
 Unnamed: 0                 0
TV Ad Budget ($)           0
Radio Ad Budget ($)        0
Newspaper Ad Budget ($)    0
Sales ($)                  0
dtype: int64
Root Mean Squared Error (RMSE): 5.845791284816158
```


Sales ($)
Radio Ad Budget ($)

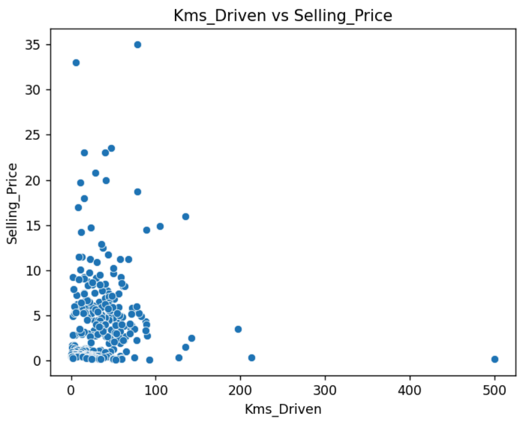

Actual vs Predicted Sales
Actual
Predicted
Sales
Radio

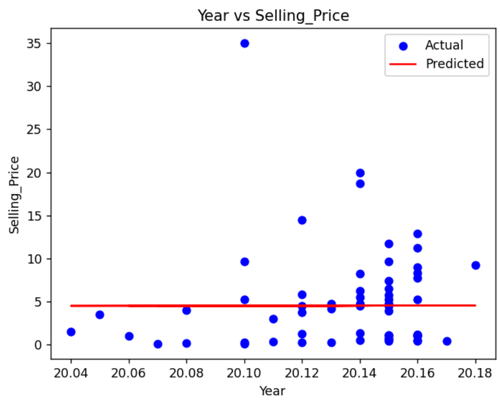

Comparison of Actual vs Predicted Sales
Actual Sales
Predicted Sales
Sales
Data Points

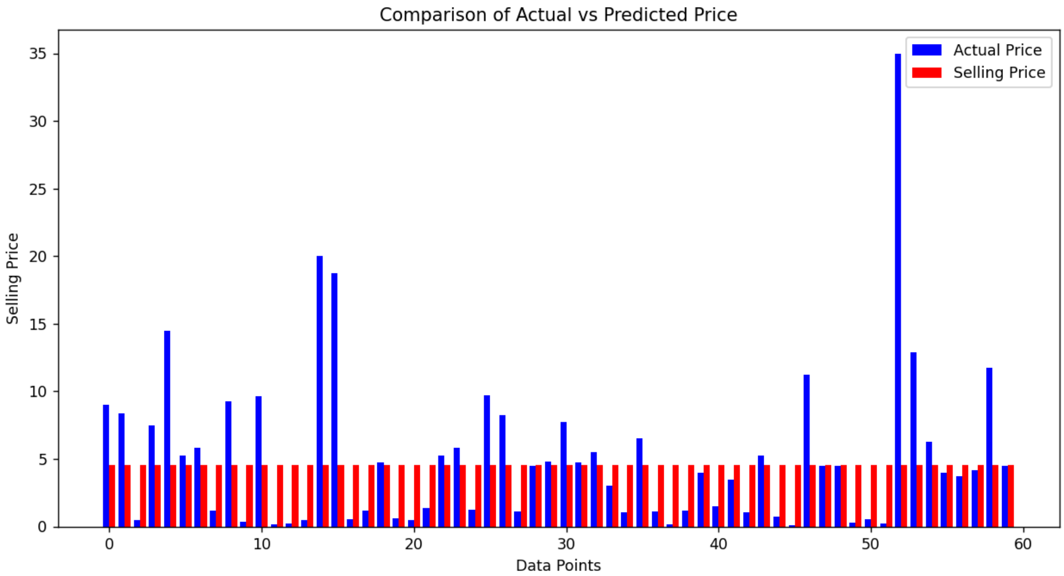
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from LinearRegression import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("dataset/car/car data.csv")

print(df.head())
print(df.info())

# Convert 'transmission' column to binary values: 0 for 'Manual', 1 for
'Automatic'
df["Transmission"] = df["Transmission"].map({"Manual": 0, "Automatic":
1})
df["Fuel_Type"] = df["Fuel_Type"].map({"Petrol": 0, "Diesel": 1})
df["Seller_Type"] = df["Seller_Type"].map({"Dealer": 0, "Individual":
1})
df["Year"] = df["Year"] / 100
df["Kms_Driven"] = df["Kms_Driven"] / 1000
df = df.dropna()

sns.scatterplot(x="Kms_Driven", y="Selling_Price", data=df)
plt.title("Kms_Driven vs Selling_Price")
plt.show()

# 7. Selling prices w.r.t year bought
X1 = df[["Year"]]  # Input
y1 = df["Selling_Price"]  # Output

# 8. Selling prices w.r.t km driven
X2 = df[["Kms_Driven"]]  # Input
y2 = df["Selling_Price"]  # Output

# 9. Selling prices w.r.t transmission
X3 = df[["Transmission"]]  # Input
y3 = df["Selling_Price"]  # Output

# 10. Selling prices w.r.t owner
```

```
X4 = df[["Owner"]]  # Input
y4 = df["Selling_Price"]  # Output

# 11. Selling prices w.r.t year bought and km driven
X5 = df[["Year", "Kms_Driven"]]  # Input
y5 = df["Selling_Price"]  # Output

# 12. Selling prices w.r.t year bought and transmission
X6 = df[["Year", "Transmission"]]  # Input
y6 = df["Selling_Price"]  # Output

# 13-14. Selling prices w.r.t year bought and owner
X7 = df[["Year", "Owner"]]  # Input
y7 = df["Selling_Price"]  # Output

# 15. Selling prices w.r.t km driven and transmission
X8 = df[["Kms_Driven", "Transmission"]]  # Input
y8 = df["Selling_Price"]  # Output

# 16. Selling prices w.r.t km driven and owner
X9 = df[["Kms_Driven", "Owner"]]  # Input
y9 = df["Selling_Price"]  # Output

# 17-18. Selling prices w.r.t transmission and owner
X10 = df[["Transmission", "Owner"]]  # Input
y10 = df["Selling_Price"]  # Output

X11 = df.drop(["Selling_Price", "Car_Name"], axis=1)
y11 = df["Selling_Price"]

X_train, X_test, y_train, y_test = train_test_split(
    X8, y8, test_size=0.2, random_state=42
)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_pred = np.clip(y_pred, a_min=0, a_max=None)

rmse = np.sqrt(np.mean((y_test - y_pred) ** 2))
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
# 7-10
# plt.scatter(X_test, y_test, color='blue', label='Actual')
# plt.plot(X_test, y_pred, color='red', label='Predicted')
# plt.title('Year vs Selling_Price')
# plt.xlabel('Year')
# plt.ylabel('Selling_Price')
# plt.legend()
# plt.show()

 # 11-18
plt.figure(figsize=(12,6))
indices = np.arange(len(y_test))
plt.bar(indices - 0.2, y_test, width=0.4, label='Actual Price',
color='blue')
plt.bar(indices + 0.2, y_pred, width=0.4, label='Selling Price',
color='red')
plt.title('Comparison of Actual vs Predicted Price')
plt.xlabel('Data Points')
plt.ylabel('Selling Price')
plt.legend()
plt.show()
```






Comparison of Actual vs Predicted Price
Actual Price
Selling Price
Selling Price
Data Points
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60

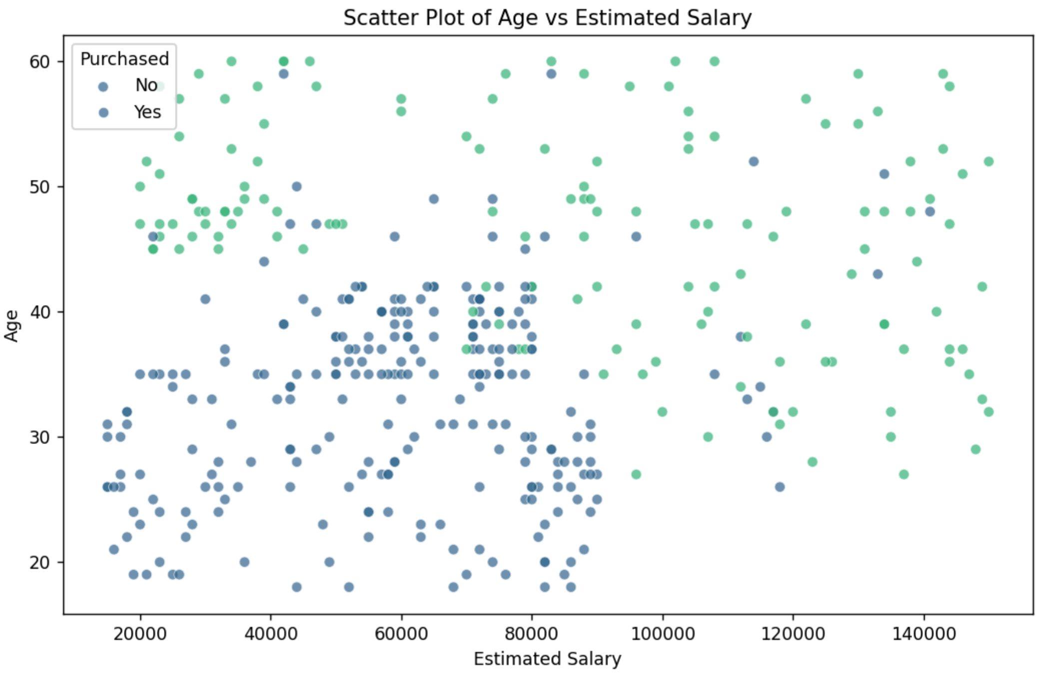
19-22

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as snb
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score

df = pd.read_csv("dataset/Social_Network_Ads/Social_Network_Ads.csv")

print(df.head())
print(df.info())
print(df.describe())

print(df.isnull().sum())

df.dropna(inplace=True)

plt.figure(figsize=(10, 6))
snb.scatterplot(data=df, x='EstimatedSalary', y='Age', hue='Purchased',
palette='viridis', alpha=0.7)
plt.title('Scatter Plot of Age vs Estimated Salary')
plt.xlabel('Estimated Salary')
plt.ylabel('Age')
plt.legend(title='Purchased', loc='upper left', labels=['No', 'Yes'])
plt.show()

# Encoding categorical data
df['Gender'] = pd.factorize(df['Gender'])[0] # Male=0, Female=1
# Features and target variable
X = df[['Gender', 'Age', 'EstimatedSalary']]
y = df['Purchased']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train the SVM model
svm_classifier_linear = SVC(kernel='linear')
svm_classifier_linear.fit(X_train, y_train)

y_pred_linear = svm_classifier_linear.predict(X_test)
```

```python
# confusion matrix
cm = confusion_matrix(y_test, y_pred_linear)
print("Confusion Matrix for linear kernel:\n", cm)

# metrics
accuracy_linear = accuracy_score(y_test, y_pred_linear)
recall_linear = recall_score(y_test, y_pred_linear)
precision_linear = precision_score(y_test, y_pred_linear)
f1_linear = f1_score(y_test, y_pred_linear)

print("For linear kernel")
print(f'Accuracy: {accuracy_linear:.2f}')
print(f'Recall: {recall_linear:.2f}')
print(f'Precision: {precision_linear:.2f}')
print(f'F1 Score: {f1_linear:.2f}')
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/
py
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
               User ID         Age  EstimatedSalary   Purchased
count  4.000000e+02  400.000000       400.000000  400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500
std    7.165832e+04   10.482877     34096.960282    0.479864
min    1.556669e+07   18.000000     15000.000000    0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
Confusion Matrix for linear kernel:
 [[48  4]
 [ 9 19]]
For linear kernel
Accuracy: 0.84
Recall: 0.68
Precision: 0.83
F1 Score: 0.75
```



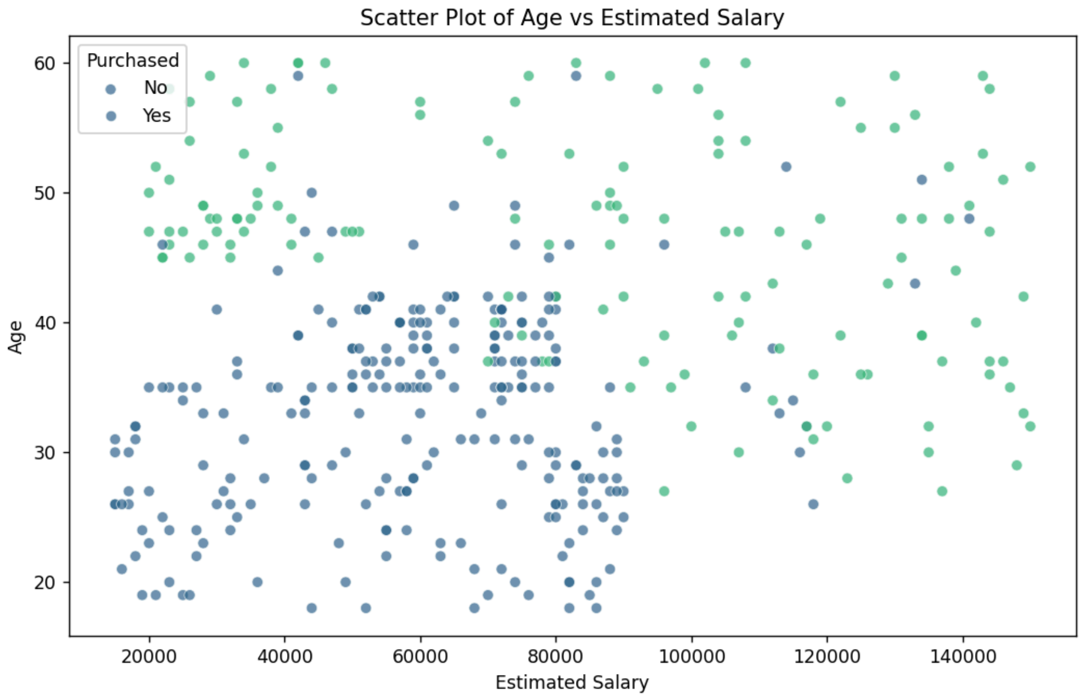
23-26

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as snb
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score

df = pd.read_csv("dataset/Social_Network_Ads/Social_Network_Ads.csv")

print(df.head())
print(df.info())
print(df.describe())

print(df.isnull().sum())
df.dropna(inplace=True)

plt.figure(figsize=(10, 6))
snb.scatterplot(data=df, x='EstimatedSalary', y='Age',
hue='Purchased',palette='viridis', alpha=0.7)
plt.title('Scatter Plot of Age vs Estimated Salary')
plt.xlabel('Estimated Salary')
plt.ylabel('Age')
plt.legend(title='Purchased', loc='upper left', labels=['No', 'Yes'])
plt.show()

# Encoding categorical data
df['Gender'] = pd.factorize(df['Gender'])[0] # Male=0, Female=1
# Features and target variable
X = df[['Gender', 'Age', 'EstimatedSalary']]
y = df['Purchased']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)

y_pred = svm_model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
```

```python
print("Confusion matrix: \n", cm)

accuracy= accuracy_score(y_test, y_pred)
recall= recall_score(y_test, y_pred)
precision= precision_score(y_test, y_pred)
f1= f1_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
print(f'Recall: {recall:.2f}')
print(f'Precision: {precision:.2f}')
print(f'F1 Score: {f1:.2f}')
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/pyt
py
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
None
```

```
            User ID         Age  EstimatedSalary   Purchased
count  4.000000e+02  400.000000       400.000000  400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500
std    7.165832e+04   10.482877     34096.960282    0.479864
min    1.556669e+07   18.000000     15000.000000    0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
Confusion matrix:
 [[49  3]
 [18 10]]
Accuracy: 0.74
Recall: 0.36
Precision: 0.77
F1 Score: 0.49
```



Scatter Plot of Age vs Estimated Salary
Purchased
No
Yes
Age
Estimated Salary
20000
40000
60000
80000
100000
120000
140000
20
30
40
50
60

27-30

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import seaborn as snb

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.head())
print(df.info())
print(df.describe())

df['Species'] = pd.factorize(df['Species'])[0]

print(df.isnull().sum())
df.dropna(inplace=True)

X = df.drop(['Id','Species'],axis=1)
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

svm_classifier_linear = SVC(kernel='linear')
svm_classifier_linear.fit(X_train, y_train)

y_pred_linear = svm_classifier_linear.predict(X_test)

cm = confusion_matrix(y_test, y_pred_linear)
print("Confusion Matrix for linear kernel:\n", cm)

accuracy_linear = accuracy_score(y_test, y_pred_linear)
recall_linear = recall_score(y_test, y_pred_linear, average='weighted')
precision_linear = precision_score(y_test, y_pred_linear,
average='weighted')
f1_linear = f1_score(y_test, y_pred_linear, average='weighted')

print(f'Accuracy: {accuracy_linear:.2f}')
print(f'Recall: {recall_linear:.2f}')
```

```
print(f'Precision: {precision_linear:.2f}')
print(f'F1 Score: {f1_linear:.2f}')
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe
py
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

```
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Confusion Matrix for linear kernel:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Accuracy: 1.00
Recall: 1.00
Precision: 1.00
F1 Score: 1.00
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import seaborn as snb

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.head())
print(df.info())
print(df.describe())

df['Species'] = pd.factorize(df['Species'])[0]

print(df.isnull().sum())
df.dropna(inplace=True)

X = df.drop(['Id','Species'],axis=1)
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

svm_classifier_rbf = SVC(kernel='rbf')
svm_classifier_rbf.fit(X_train, y_train)

y_pred_rbf = svm_classifier_rbf.predict(X_test)

cm = confusion_matrix(y_test, y_pred_rbf)
print("Confusion Matrix for rbf kernel:\n", cm)

accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
recall_rbf = recall_score(y_test, y_pred_rbf, average='weighted')
precision_rbf = precision_score(y_test, y_pred_rbf, average='weighted')
f1_rbf = f1_score(y_test, y_pred_rbf, average='weighted')


print(f'Accuracy: {accuracy_rbf:.2f}')
print(f'Recall: {recall_rbf:.2f}')
```

```python
print(f'Precision: {precision_rbf:.2f}')
print(f'F1 Score: {f1_rbf:.2f}')
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:/College/SEM-
py
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              150 non-null    int64
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

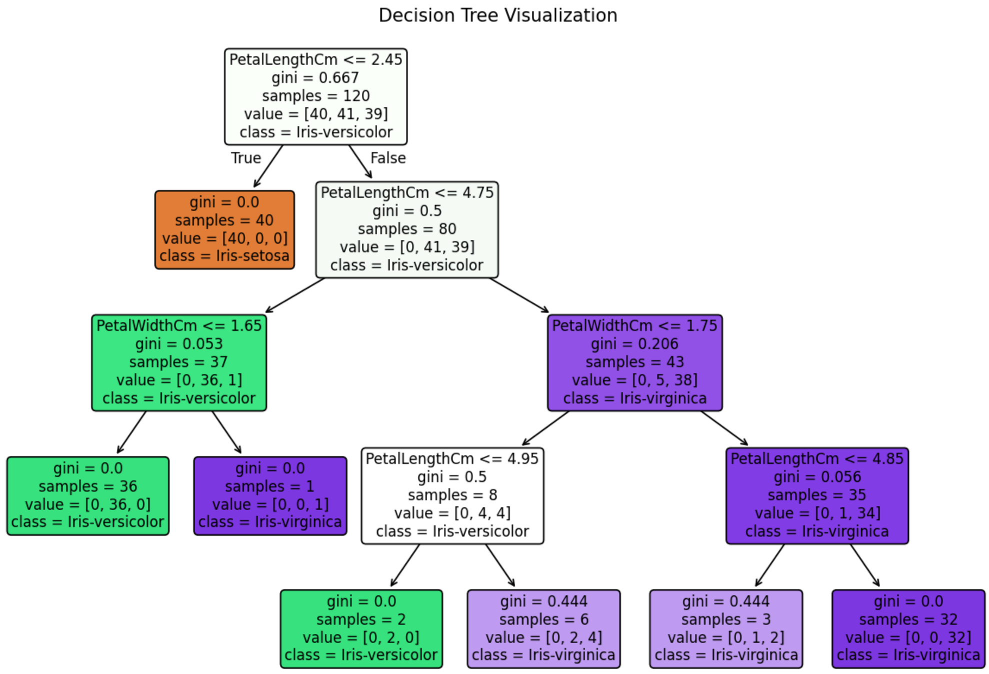
```
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Confusion Matrix for rbf kernel:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Accuracy: 1.00
Recall: 1.00
Precision: 1.00
F1 Score: 1.00
```

```python
#35. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a decision tree classifier to predict the iris plant
category? (Use GINI INDEX criteria, use
# max_depth=4, min_samples_split=2)
# iii. Evaluate the model using Accuracy.

import pandas as pd
import seaborn as snb
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, log_loss

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]

print(df.isnull().sum())

df.dropna(inplace=True)

X = df.drop(['Id', 'Species'], axis=1)
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

dt_classifier = DecisionTreeClassifier(criterion="gini", max_depth=4,
min_samples_split=2, random_state=42)
dt_classifier.fit(X_train, y_train)

y_pred = dt_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy Score: {accuracy}")
```

```
plt.figure(figsize=(12,8))
plot_tree(dt_classifier, filled=True, feature_names=X.columns,
class_names=["Iris-setosa","Iris-versicolor","Iris-virginica"],
rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/pyth
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
116  117            6.5           3.0            5.5           1.8   Iris-virginica
66    67            5.6           3.0            4.5           1.5  Iris-versicolor
15    16            5.7           4.4            1.5           0.4      Iris-setosa
130  131            7.4           2.8            6.1           1.9   Iris-virginica
61    62            5.9           3.0            4.2           1.5  Iris-versicolor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
Id               0
SepalLengthCm    0
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Accuracy Score: 1.0
```


Decision Tree Visualization
PetalLengthCm <= 2.45
gini = 0.667
samples = 120
value = [40, 41, 39]
class = Iris-versicolor
True
False
gini = 0.0
samples = 40
value = [40, 0, 0]
class = Iris-setosa
PetalLengthCm <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = Iris-versicolor
PetalWidthCm <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
class = Iris-versicolor
PetalWidthCm <= 1.75
gini = 0.206
samples = 43
value = [0, 5, 38]
class = Iris-virginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = Iris-versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
PetalLengthCm <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = Iris-versicolor
PetalLengthCm <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
class = Iris-virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
gini = 0.444
samples = 6
value = [0, 2, 4]
class = Iris-virginica
gini = 0.444
samples = 3
value = [0, 1, 2]
class = Iris-virginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = Iris-virginica

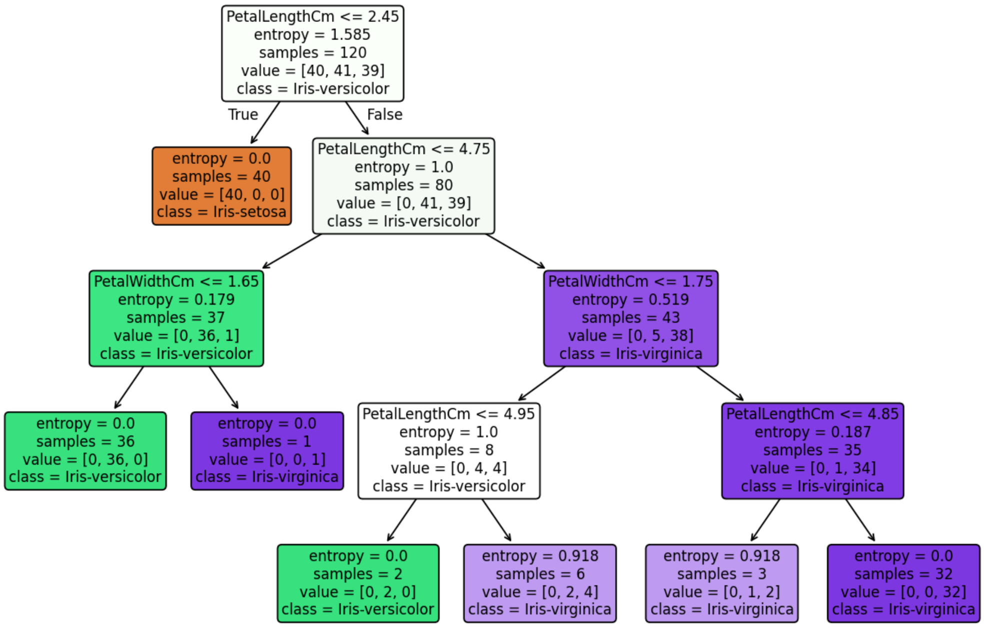
```
# 36. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a decision tree classifier to predict the iris plant
category. (Use Entropy criteria, use
# max_depth=4, min_samples_split=2)
# iii. Evaluate the model using Accuracy.

import pandas as pd
import seaborn as snb
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, log_loss

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]

print(df.isnull().sum())

df.dropna(inplace=True)

X = df.drop(['Id', 'Species'], axis=1)
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

dt_classifier = DecisionTreeClassifier(criterion="entropy",
max_depth=4, min_samples_split=2, random_state=42)
dt_classifier.fit(X_train, y_train)

y_pred = dt_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy Score: {accuracy}")
```

```python
plt.figure(figsize=(12,8))
plot_tree(dt_classifier, filled=True, feature_names=X.columns,
class_names=["Iris-setosa","Iris-versicolor","Iris-virginica"],
rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
115  116            6.4           3.2            5.3           2.3   Iris-virginica
132  133            6.4           2.8            5.6           2.2   Iris-virginica
94    95            5.6           2.7            4.2           1.3  Iris-versicolor
28    29            5.2           3.4            1.4           0.2      Iris-setosa
57    58            4.9           2.4            3.3           1.0  Iris-versicolor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Accuracy Score: 1.0
```

Decision Tree Visualization

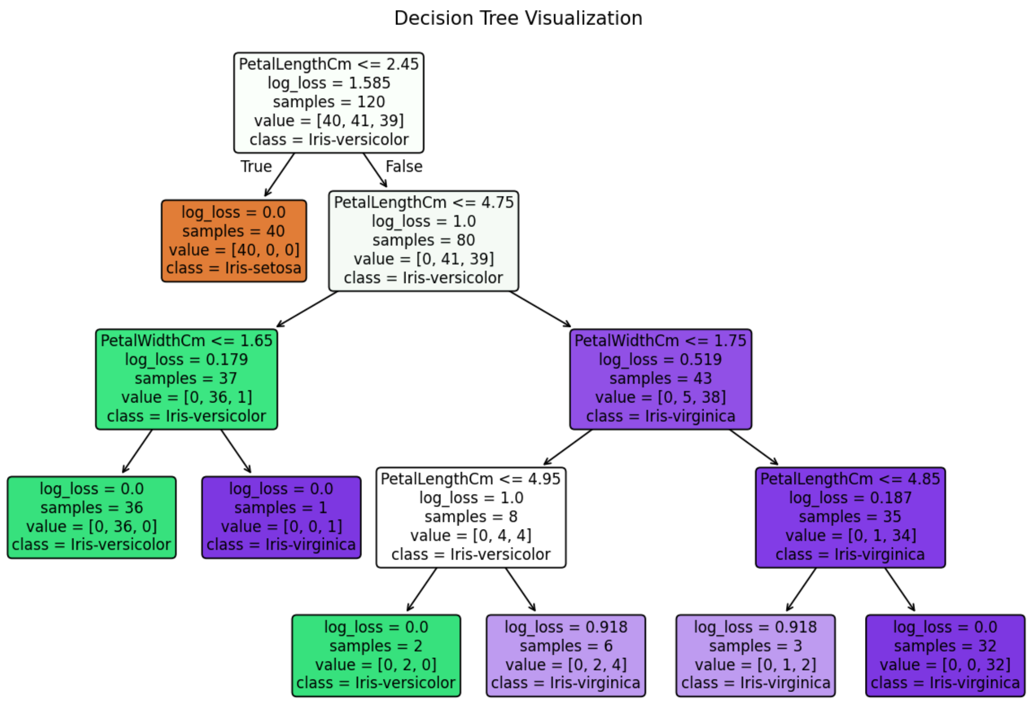


```
#37. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a decision tree classifier to predict the iris plant
category? (Use log loss criteria, use
# max_depth=4, min_samples_split=2)
# iii. Evaluate the model using Accuracy.

import pandas as pd
import seaborn as snb
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, log_loss

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]

print(df.isnull().sum())

df.dropna(inplace=True)

X = df.drop(['Id', 'Species'], axis=1)
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

dt_classifier = DecisionTreeClassifier(criterion="log_loss",
max_depth=4, min_samples_split=2, random_state=42)
dt_classifier.fit(X_train, y_train)

y_pred = dt_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy Score: {accuracy}")
```

```
plt.figure(figsize=(12,8))
plot_tree(dt_classifier, filled=True, feature_names=X.columns,
class_names=["Iris-setosa","Iris-versicolor","Iris-virginica"],
rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:/C
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
64    65            5.6           2.9            3.6           1.3  Iris-versicolor
52    53            6.9           3.1            4.9           1.5  Iris-versicolor
125  126            7.2           3.2            6.0           1.8   Iris-virginica
98    99            5.1           2.5            3.0           1.1  Iris-versicolor
127  128            6.1           3.0            4.9           1.8   Iris-virginica
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
None
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Accuracy Score: 1.0
```


Decision Tree Visualization
PetalLengthCm <= 2.45
log_loss = 1.585
samples = 120
value = [40, 41, 39]
class = Iris-versicolor
True
False
log_loss = 0.0
samples = 40
value = [40, 0, 0]
class = Iris-setosa
PetalLengthCm <= 4.75
log_loss = 1.0
samples = 80
value = [0, 41, 39]
class = Iris-versicolor
PetalWidthCm <= 1.65
log_loss = 0.179
samples = 37
value = [0, 36, 1]
class = Iris-versicolor
PetalWidthCm <= 1.75
log_loss = 0.519
samples = 43
value = [0, 5, 38]
class = Iris-virginica
log_loss = 0.0
samples = 36
value = [0, 36, 0]
class = Iris-versicolor
log_loss = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
PetalLengthCm <= 4.95
log_loss = 1.0
samples = 8
value = [0, 4, 4]
class = Iris-versicolor
PetalLengthCm <= 4.85
log_loss = 0.187
samples = 35
value = [0, 1, 34]
class = Iris-virginica
log_loss = 0.0
samples = 2
value = [0, 2, 0]
class = Iris-versicolor
log_loss = 0.918
samples = 6
value = [0, 2, 4]
class = Iris-virginica
log_loss = 0.918
samples = 3
value = [0, 1, 2]
class = Iris-virginica
log_loss = 0.0
samples = 32
value = [0, 0, 32]
class = Iris-virginica

```
# The Iris data set contains 3 classes of 50 instances each, where each
class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a logistic Regression classifier to predict the iris plant
category?
# iii. Evaluate the model using Accuracy

import pandas as pd
from sklearn.model_selection import train_test_split
from LogisticRegression import LogisticRegression
from sklearn.metrics import accuracy_score

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]
print(df.head())

print(df.isnull().sum())
df.dropna(inplace=True)

print(df['Species'].value_counts())

X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)

y_pred = logistic_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Logistic Regression Accuracy: {accuracy:.2f}")
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:/College/S
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
143  144            6.8           3.2            5.9           2.3   Iris-virginica
65    66            6.7           3.1            4.4           1.4  Iris-versicolor
11    12            4.8           3.4            1.6           0.2      Iris-setosa
120  121            6.9           3.2            5.7           2.3   Iris-virginica
74    75            6.4           2.9            4.3           1.3  Iris-versicolor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1            5.1           3.5            1.4           0.2        0
1   2            4.9           3.0            1.4           0.2        0
2   3            4.7           3.2            1.3           0.2        0
3   4            4.6           3.1            1.5           0.2        0
4   5            5.0           3.6            1.4           0.2        0
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Species
0    50
1    50
2    50
Name: count, dtype: int64
Logistic Regression Accuracy: 0.30
```

```
# 39. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a Bagging Classifier model to predict the iris plant
category?
# iii. Evaluate the model using Accuracy.

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]
print(df.head())

print(df.isnull().sum())
df.dropna(inplace=True)

print(df['Species'].value_counts())

X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

bagging_model = BaggingClassifier()
bagging_model.fit(X_train, y_train)

y_pred = bagging_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Bagging Classifier Accuracy: {accuracy:.2f}")
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:/Col
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
75    76            6.6           3.0            4.4           1.4  Iris-versicolor
8      9            4.4           2.9            1.4           0.2      Iris-setosa
135  136            7.7           3.0            6.1           2.3   Iris-virginica
43    44            5.0           3.5            1.6           0.6      Iris-setosa
66    67            5.6           3.0            4.5           1.5  Iris-versicolor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1            5.1           3.5            1.4           0.2        0
1   2            4.9           3.0            1.4           0.2        0
2   3            4.7           3.2            1.3           0.2        0
3   4            4.6           3.1            1.5           0.2        0
4   5            5.0           3.6            1.4           0.2        0
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Species
0    50
1    50
2    50
Name: count, dtype: int64
Bagging Classifier Accuracy: 1.00
```

```python
# 40. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a Random Forest Classifier model to predict the iris plant
category?
# iii. Evaluate the model using Accuracy

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]
print(df.head())

print(df.isnull().sum())
df.dropna(inplace=True)

print(df['Species'].value_counts())

X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

random_forest_model = RandomForestClassifier()
random_forest_model.fit(X_train, y_train)

y_pred = random_forest_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forest Classifier Accuracy: {accuracy:.2f}")
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:/College/SEM-VII/ML/PRACS/q40.py
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
10    11            5.4           3.7            1.5           0.2      Iris-setosa
52    53            6.9           3.1            4.9           1.5  Iris-versicolor
108  109            6.7           2.5            5.8           1.8   Iris-virginica
107  108            7.3           2.9            6.3           1.8   Iris-virginica
116  117            6.5           3.0            5.5           1.8   Iris-virginica
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1            5.1           3.5            1.4           0.2        0
1   2            4.9           3.0            1.4           0.2        0
2   3            4.7           3.2            1.3           0.2        0
3   4            4.6           3.1            1.5           0.2        0
4   5            5.0           3.6            1.4           0.2        0
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Species
0    50
1    50
2    50
Name: count, dtype: int64
Random Forest Classifier Accuracy: 1.00
```

```
# 41. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a Gradient Boost Classifier model to predict the iris plant
category?
# iii. Evaluate the model using Accuracy

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]
print(df.head())

print(df.isnull().sum())
df.dropna(inplace=True)

print(df['Species'].value_counts())

X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

gradient_boosting_model = GradientBoostingClassifier()
gradient_boosting_model.fit(X_train, y_train)

y_pred = gradient_boosting_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Gradient Boosting Classifier Accuracy: {accuracy:.2f}")
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python311/python.exe" e:/College/SEM-V
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
130  131            7.4           2.8            6.1           1.9   Iris-virginica
106  107            4.9           2.5            4.5           1.7   Iris-virginica
71    72            6.1           2.8            4.0           1.3  Iris-versicolor
63    64            6.1           2.9            4.7           1.4  Iris-versicolor
90    91            5.5           2.6            4.4           1.2  Iris-versicolor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1            5.1           3.5            1.4           0.2        0
1   2            4.9           3.0            1.4           0.2        0
2   3            4.7           3.2            1.3           0.2        0
3   4            4.6           3.1            1.5           0.2        0
4   5            5.0           3.6            1.4           0.2        0
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Species
0    50
1    50
2    50
Name: count, dtype: int64
Gradient Boosting Classifier Accuracy: 1.00
```

```python
# 42. The Iris data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.
# i. Understand the Dataset & cleanup (if required).
# ii. Build a AdaBoost Classifier model to predict the iris plant
category?
# iii. Evaluate the model using Accuracy

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]
print(df.head())

print(df.isnull().sum())
df.dropna(inplace=True)

print(df['Species'].value_counts())

X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

ada_boost_model = AdaBoostClassifier()
ada_boost_model.fit(X_train, y_train)

y_pred = ada_boost_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Ada Boost Classifier Accuracy: {accuracy:.2f}")
```

```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
108  109            6.7           2.5            5.8           1.8  Iris-virginica
137  138            6.4           3.1            5.5           1.8  Iris-virginica
145  146            6.7           3.0            5.2           2.3  Iris-virginica
115  116            6.4           3.2            5.3           2.3  Iris-virginica
4      5            5.0           3.6            1.4           0.2     Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1            5.1           3.5            1.4           0.2        0
1   2            4.9           3.0            1.4           0.2        0
2   3            4.7           3.2            1.3           0.2        0
3   4            4.6           3.1            1.5           0.2        0
4   5            5.0           3.6            1.4           0.2        0
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Species
0    50
1    50
2    50
Name: count, dtype: int64
C:\Users\Om Shete\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ensemble\_weight_bo
ME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circu
  warnings.warn(
Ada Boost Classifier Accuracy: 1.00
```

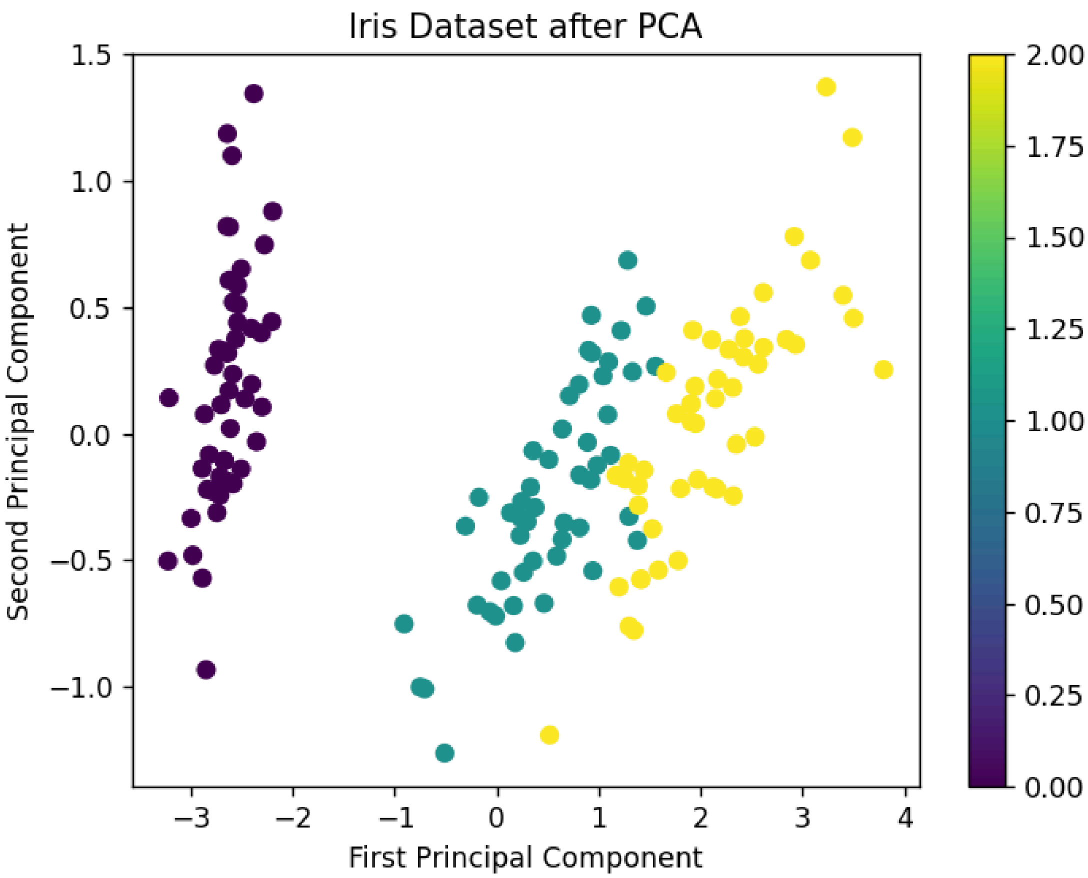
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA

df = pd.read_csv("dataset/Iris/Iris.csv")

print(df.sample(5))
print(df.info())

df['Species'] = pd.factorize(df['Species'])[0]

print(df.sample(5))

print(df.isnull().sum())
df.dropna(inplace=True)

X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Without PCA

random_forest_model = RandomForestClassifier(random_state=42)
random_forest_model.fit(X_train, y_train)

y_pred = random_forest_model.predict(X_test)

accuracy_without_pca = accuracy_score(y_test, y_pred)

print("Confusion Matrix without PCA: \n", confusion_matrix(y_test,
y_pred))
print("Accuracy without PCA: \n", accuracy_without_pca)

# With PCA

pca = PCA(n_components=2)
```

```python
X_pca = pca.fit_transform(X)

X_train_pca, X_test_pca, y_train_pca, y_test_pca =
train_test_split(X_pca, y,test_size=0.2, random_state=42)

rf_classifier_pca = RandomForestClassifier(random_state=42)
rf_classifier_pca.fit(X_train_pca, y_train_pca)

y_pred_pca = rf_classifier_pca.predict(X_test_pca)

accuracy_with_pca = accuracy_score(y_test_pca, y_pred_pca)
print("Confusion Matrix:\n", confusion_matrix(y_test_pca, y_pred_pca))
print(f"Accuracy with PCA: {accuracy_with_pca:.2f}")

#

print(f"Accuracy before PCA: {accuracy_without_pca:.2f}")
print(f"Accuracy after PCA: {accuracy_with_pca:.2f}")



#

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.title('Iris Dataset after PCA')
plt.colorbar()
plt.show()
```

```
PS E:\College\SEM-VII\ML\PRACS> & "C:/Users/Om Shete/AppData/Local/Programs/Python/Python3

      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Species
77    78            6.7           3.0            5.0           1.7  Iris-versicolor
121  122            5.6           2.8            4.9           2.0   Iris-virginica
74    75            6.4           2.9            4.3           1.3  Iris-versicolor
134  135            6.1           2.6            5.6           1.4   Iris-virginica
8      9            4.4           2.9            1.4           0.2      Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
```

```
123  124            6.3           2.7            4.9           1.8        2
126  127            6.2           2.8            4.8           1.8        2
4      5            5.0           3.6            1.4           0.2        0
110  111            6.5           3.2            5.1           2.0        2
142  143            5.8           2.7            5.1           1.9        2
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
Confusion Matrix without PCA:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Accuracy without PCA:
 1.0
```

```
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Accuracy with PCA: 1.00
Accuracy before PCA: 1.00
Accuracy after PCA: 1.00
```


Iris Dataset after PCA
Second Principal Component
First Principal Component
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00