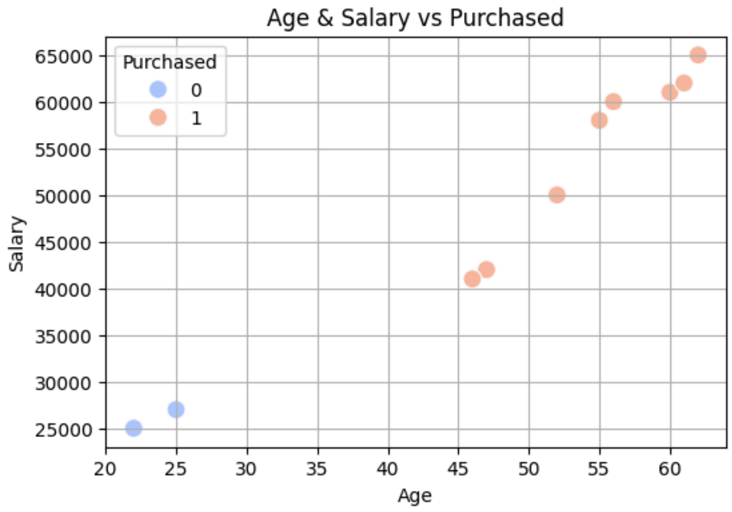
```python
# logistic regression

# step 1   # import all the libraries

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

# step 2   # creating data set

data = {
    'Age': [22, 25, 47, 52, 46, 56, 55, 60, 62, 61],
    'Salary': [25000, 27000, 42000, 50000, 41000, 60000, 58000, 61000, 65000, 62000],
    'Purchased': [0, 0, 1, 1, 1, 1, 1, 1, 1, 1]  # 0 = No, 1 = Yes
}

df = pd.DataFrame(data)


# step 3    # x,y split (x values,and y values)

x = df[['Age', 'Salary']]
y = df['Purchased'] # Changed to a Series

# step 4   # train-test split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 42)

# step 5    # model train

model = LogisticRegression()
model.fit(x_train, y_train)
print(f"Training samples: {x_train.shape[0]}, Test samples: {x_test.shape[0]}")
print('Loan Default model is trained successfully')

# step 6   # prediction

y_pred = model.predict(x_test)

# step 7   # accuracy

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


# step 8   # scottor plot visualize

plt.figure(figsize=(6,4))
sns.scatterplot(x = 'Age', y = 'Salary', hue = 'Purchased',
               data = df, s = 100, palette = 'coolwarm')

plt.title('Age & Salary vs Purchased')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.grid(True)
plt.show()
```

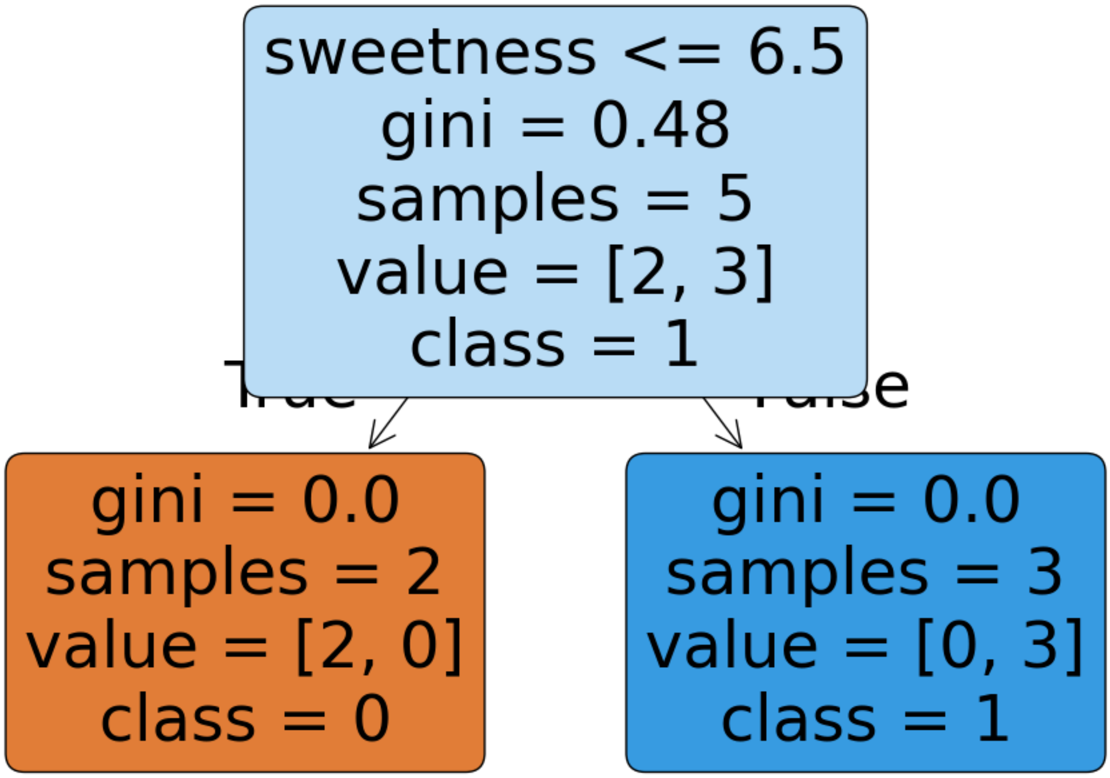
```
Training samples: 7, Test samples: 3
Loan Default model is trained successfully
accuracy: 1.00
```

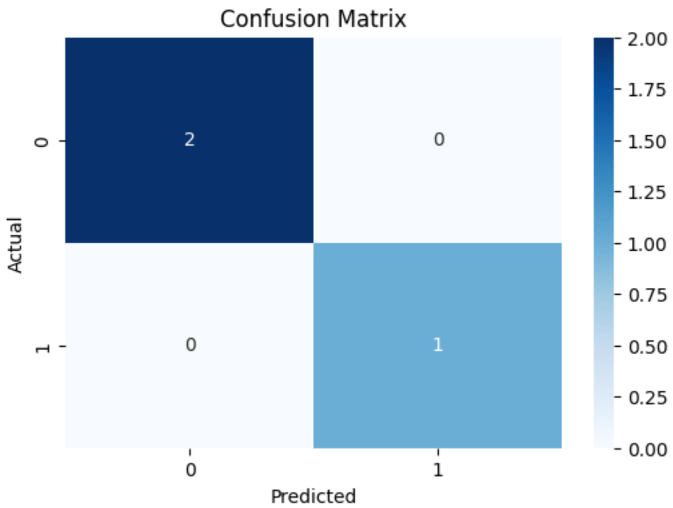



```
# decision tree

# step 1

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

# fruite size and sweetness

# step 2    # create dataset

date = {
    'size': [3,4,5,7,9,6,2, 8],
    'sweetness':[4,5,6,8,9,9,7,3],
    'fruitetype':[0,0,0,1,1,1,1,0]
}

df = pd.DataFrame(date)

# step 3   # splitting x,y

x = df[['size', 'sweetness']]
y = df['fruitetype']

# step 4   # train-test split

x_train, x_test, y_train, y_test = train_test_split (x,y, test_size = 0.3, random_state = 42)

# step 5    # model train

model = DecisionTreeClassifier(max_depth = 3, random_state = 42)
model.fit(x_train, y_train)

# step 6   # prediction

y_pred = model.predict(x_test)

# step 7  # accuracy

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

# step 8   # Visualize the decision tree

plt.figure(figsize=(12, 8))
plot_tree(model, feature_names=['size', 'sweetness'], class_names=['0', '1'], filled=True, rounded=True)
plt.show()

# step 9   # Confusion Matrix

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], yticklabels=['0', '1'])
```

```
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 1.00
```

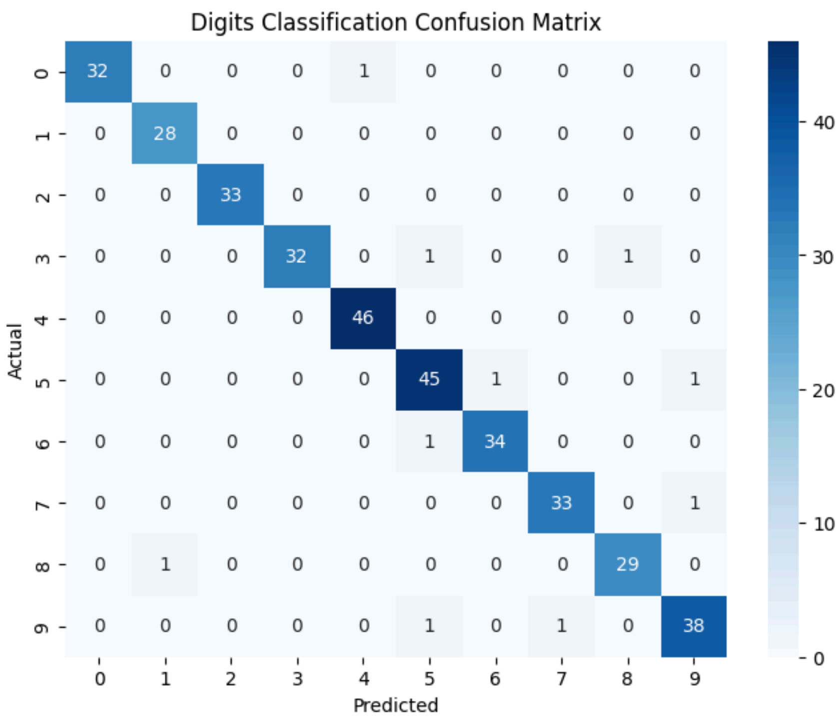





```
# random forest

from sklearn.datasets import load_digits
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load the digits dataset
digits = load_digits()
X = digits.data
y = digits.target

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the Random Forest model
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

```
# Predict
y_pred = clf.predict(X_test)

# Evaluate
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Confusion matrix heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Digits Classification Confusion Matrix')
plt.show()
```

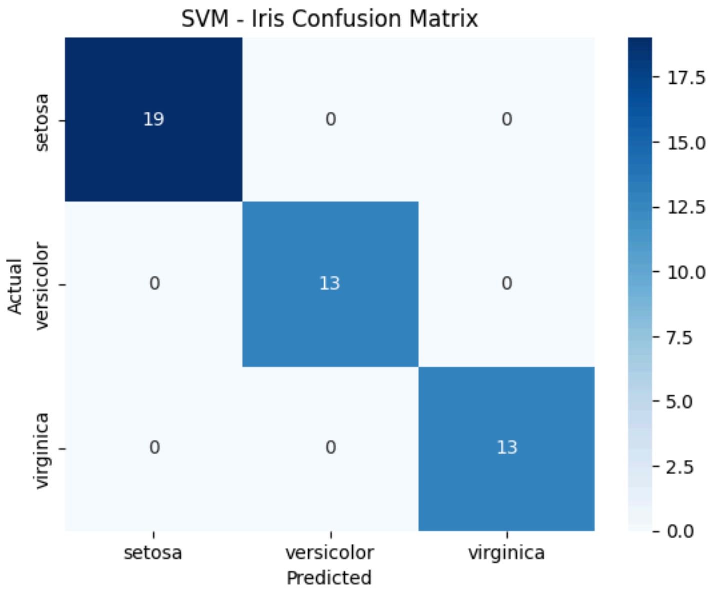
```
0.9722222222222222
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        33
           1       0.97      1.00      0.98        28
           2       1.00      1.00      1.00        33
           3       1.00      0.94      0.97        34
           4       0.98      1.00      0.99        46
           5       0.94      0.96      0.95        47
           6       0.97      0.97      0.97        35
           7       0.97      0.97      0.97        34
           8       0.97      0.97      0.97        30
           9       0.95      0.95      0.95        40

    accuracy                           0.97       360
   macro avg       0.97      0.97      0.97       360
weighted avg       0.97      0.97      0.97       360

[[32  0  0  0  1  0  0  0  0  0]
 [ 0 28  0  0  0  0  0  0  0  0]
 [ 0  0 33  0  0  0  0  0  0  0]
 [ 0  0  0 32  0  1  0  0  1  0]
 [ 0  0  0  0 46  0  0  0  0  0]
 [ 0  0  0  0  0 45  1  0  0  1]
 [ 0  0  0  0  0  1 34  0  0  0]
 [ 0  0  0  0  0  0  0 33  0  1]
 [ 0  1  0  0  0  0  0  0 29  0]
 [ 0  0  0  0  0  1  0  1  0 38]]
```




```
# svm (support vector machine)
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
```

```
data = load_iris()
X = data.data
y = data.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and train SVM model
model = SVC(kernel='linear')
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")

print(classification_report(y_test, y_pred, target_names=data.target_names))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap='Blues', xticklabels=data.target_names, yticklabels=data.target_names)
plt.title("SVM - Iris Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Accuracy: 1.00
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```




```
# k-nearest neighbours knn

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# sample data                                              # 1=Fit, 0=Unfit

data = {
    'Age': [25, 30, 45, 35, 50, 23, 40, 60, 55, 33],
    'ExerciseMins': [60, 45, 30, 50, 20, 70, 40, 15, 10, 55],
    'Fit': [1, 1, 0, 1, 0, 1, 0, 0, 0, 1]
    }

df = pd.DataFrame(data)


# split x,y
```

```
x = df[['Age', 'ExerciseMins']]
y = df['Fit']

# train split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

# model

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# predict
y_pred = knn.predict(X_test)

# model evaluation
print("Test Data:\n", X_test)
print("\nPredictions:", y_pred)
print("\nActual:", y_test.values)
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Test Data:
    Age  ExerciseMins
8   55           10
1   30           45
5   23           70

Predictions: [0 1 1]

Actual: [0 1 1]

Accuracy: 1.0

Confusion Matrix:
 [[1 0]
 [0 2]]
```