EXPERIMENT 1A:

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train a Random Forest classifier

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Generate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Plot the confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', xticklabels=iris.target\_names, yticklabels=iris.target\_names)

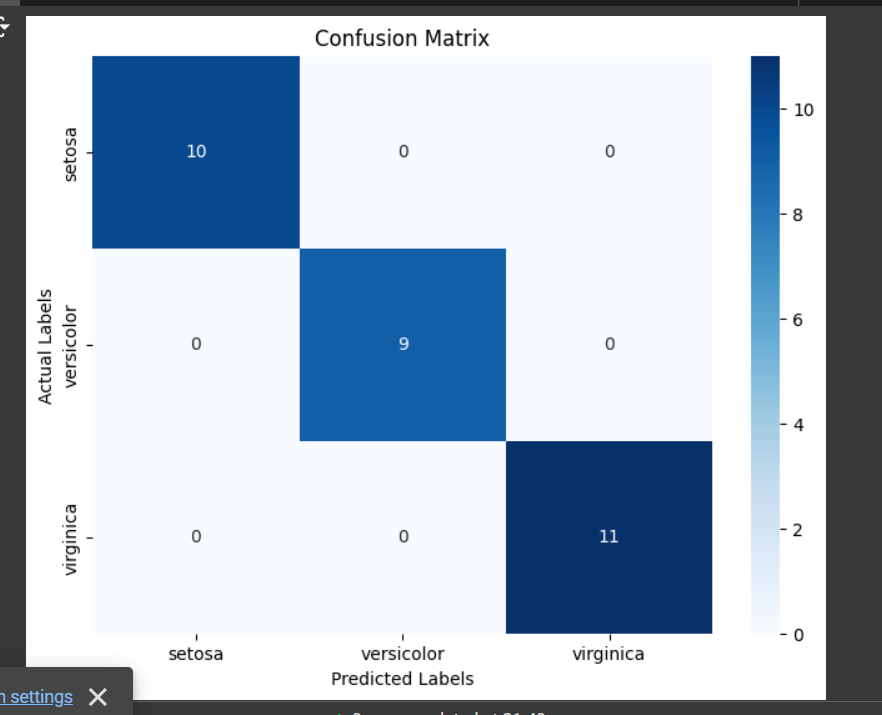
plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

plt.title('Confusion Matrix')

plt.show()

OUTPUT:



EXPERIMENT 1B:

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

import seaborn as sns

# Load the Breast Cancer dataset

data = load\_breast\_cancer()

X = data.data

y = data.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train a Logistic Regression model

model = LogisticRegression(max\_iter=10000)

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Generate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Plot the confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', xticklabels=data.target\_names, yticklabels=data.target\_names)

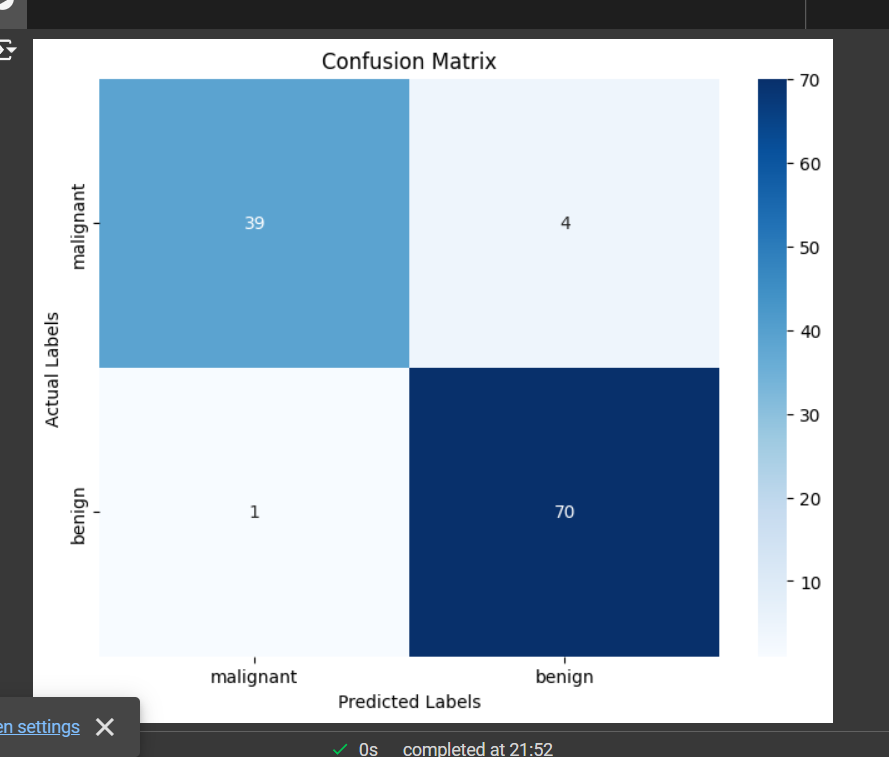
plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

plt.title('Confusion Matrix')

plt.show()

OUTPUT;



EXPERIMENT 2::

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

import seaborn as sns

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train a Random Forest classifier

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Generate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Plot the confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', xticklabels=iris.target\_names, yticklabels=iris.target\_names)

plt.xlabel('Predicted Labels')

plt.ylabel('Actual Labels')

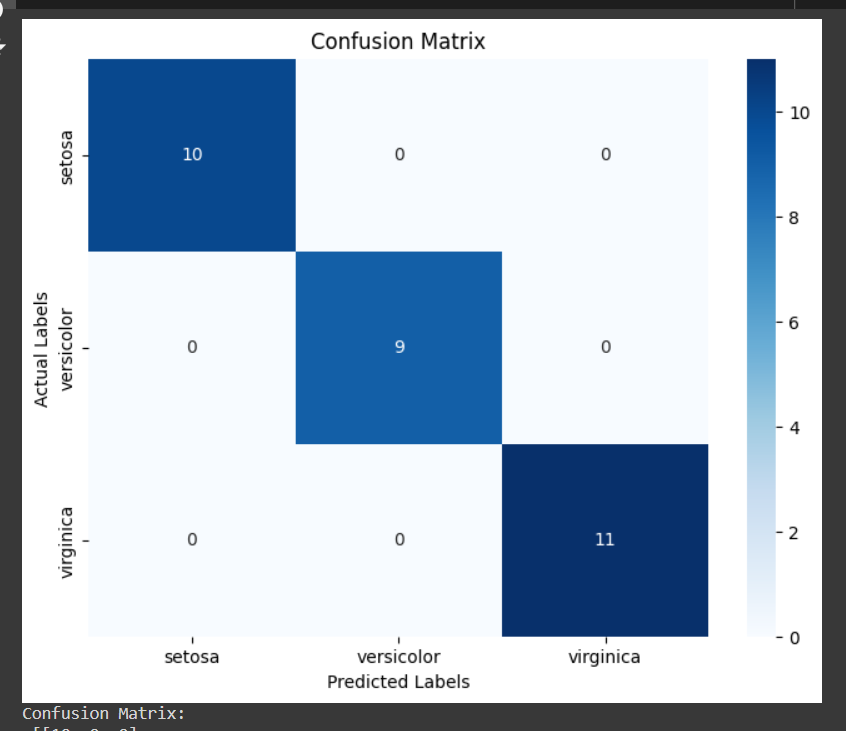
plt.title('Confusion Matrix')

plt.show()

# Print the confusion matrix for further verification

print("Confusion Matrix:\n", cm)

OUTPUT:



EXPERIMENT 3::

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train a Random Forest classifier

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions on the training set

y\_train\_pred = model.predict(X\_train)

# Make predictions on the test set

y\_test\_pred = model.predict(X\_test)

# Evaluate the model's accuracy on the training set

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

# Evaluate the model's accuracy on the test set

test\_accuracy = accuracy\_score(y\_test, y\_test\_pred)

# Print the accuracies to check for overfitting

print(f"Training Accuracy: {train\_accuracy}")

print(f"Testing Accuracy: {test\_accuracy}")

OUTPUT:

Training Accuracy: 1.0

Testing Accuracy: 1.0