| **CO3** | **Implement various algorithms of Regression and Classification.** |
| --- | --- |
| **Task5**: | Build a model using any of the classification algorithms that can effectively analyze and classify a new sample. Calculate the classification rate, accuracy, precision, and recall for your data set.  **Platform: Alteryx Google co-lab, Language: Python** |

**Use Case: Titanic Dataset**

**Objective:**

The objective of this use case is to develop a machine learning model that predicts the direction of Titanic data set.

**Dataset:**

The dataset contains the following fields: Pclass, Sex, Age, SibSp, Parch, Fare, Embarked.

Algorithm:   
1. Load the dataset from the specified file path.

2. Create a new column 'Survive'.

3. Split the dataset into training and testing sets.

4. Initialize a different Classifiers.

5. Train the classifiers using the training data.

6. Make predictions on the test set.

7. Calculate accuracy, precision, recall, and confusion matrix.

8. Print the results.

9. Plot the Confusion Matrix Heatmap.

10. Plot the Precision-Recall Curve.

12. Plot a Bar plot of Accuracy, Precision, and Recall.

**Program:**

import pandas as pd

import seaborn as sns

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score, precision\_score, recall\_score, confusion\_matrix, roc\_curve, auc, precision\_recall\_curve

import matplotlib.pyplot as plt

# Load Titanic dataset

titanic = sns.load\_dataset("titanic")

# Preprocess the data

titanic.dropna(subset=['age', 'embarked'], inplace=True)

X = titanic[['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']]

X = pd.get\_dummies(X, columns=['sex', 'embarked'], drop\_first=True)

y = titanic['survived']

# 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)

# Decision Tree Classifier

dt\_classifier = DecisionTreeClassifier(random\_state=42)

dt\_classifier.fit(X\_train, y\_train)

y\_pred\_dt = dt\_classifier.predict(X\_test)

# Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_classifier.fit(X\_train, y\_train)

y\_pred\_rf = rf\_classifier.predict(X\_test)

# Calculate metrics for Decision Tree

classification\_rep\_dt = classification\_report(y\_test, y\_pred\_dt)

accuracy\_dt = accuracy\_score(y\_test, y\_pred\_dt)

precision\_dt = precision\_score(y\_test, y\_pred\_dt)

recall\_dt = recall\_score(y\_test, y\_pred\_dt)

print(f'Accuracy: {accuracy\_dt}')

print(f'Precision: {precision\_dt}')

print(f'Recall: {recall\_dt}')

# Confusion Matrix for Decision Tree

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

sns.heatmap(cm\_dt, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix - Decision Tree')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print('Confusion Matrix:\n', cm\_dt)

# Precision-Recall Curve for Decision Tree

precision\_dt, recall\_dt, \_ = precision\_recall\_curve(y\_test, dt\_classifier.predict\_proba(X\_test)[:, 1])

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

plt.plot(recall\_dt, precision\_dt, color='blue', label='Decision Tree')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve - Decision Tree')

plt.legend()

plt.show()

# Calculate metrics for Random Forest

classification\_rep\_rf = classification\_report(y\_test, y\_pred\_rf)

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)

precision\_rf = precision\_score(y\_test, y\_pred\_rf)

recall\_rf = recall\_score(y\_test, y\_pred\_rf)

print(f'Accuracy: {accuracy\_rf}')

print(f'Precision: {precision\_rf}')

print(f'Recall: {recall\_rf}')

# Confusion Matrix for Random Forest

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

sns.heatmap(cm\_rf, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix - Random Forest')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print('Confusion Matrix:\n', cm\_rf)

# Precision-Recall Curve for Random Forest

precision\_rf, recall\_rf, \_ = precision\_recall\_curve(y\_test, rf\_classifier.predict\_proba(X\_test)[:, 1])

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

plt.plot(recall\_rf, precision\_rf, color='green', label='Random Forest')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve - Random Forest')

plt.legend()

plt.show()

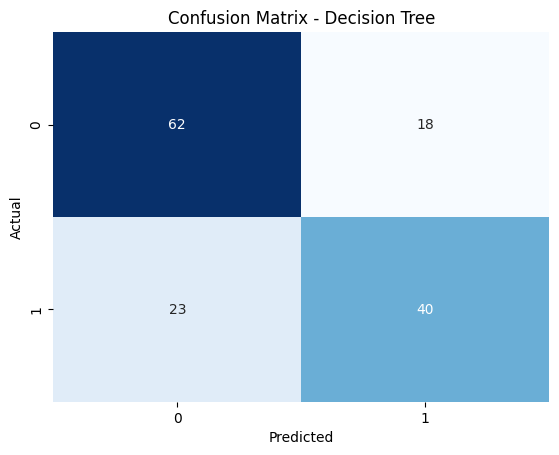
print()

**Output:**

Accuracy: 0.7132867132867133

Precision: 0.6896551724137931

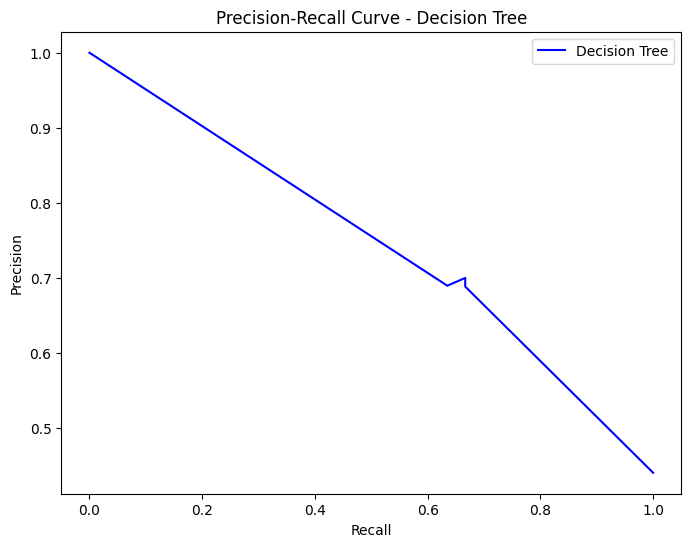
Recall: 0.6349206349206349



Confusion Matrix:

[[62 18]

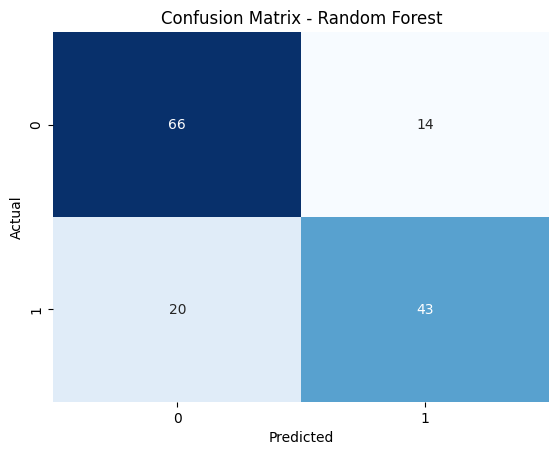
[23 40]]



Accuracy: 0.7622377622377622

Precision: 0.7543859649122807

Recall: 0.6825396825396826



Confusion Matrix:

[[66 14]

[20 43]]

