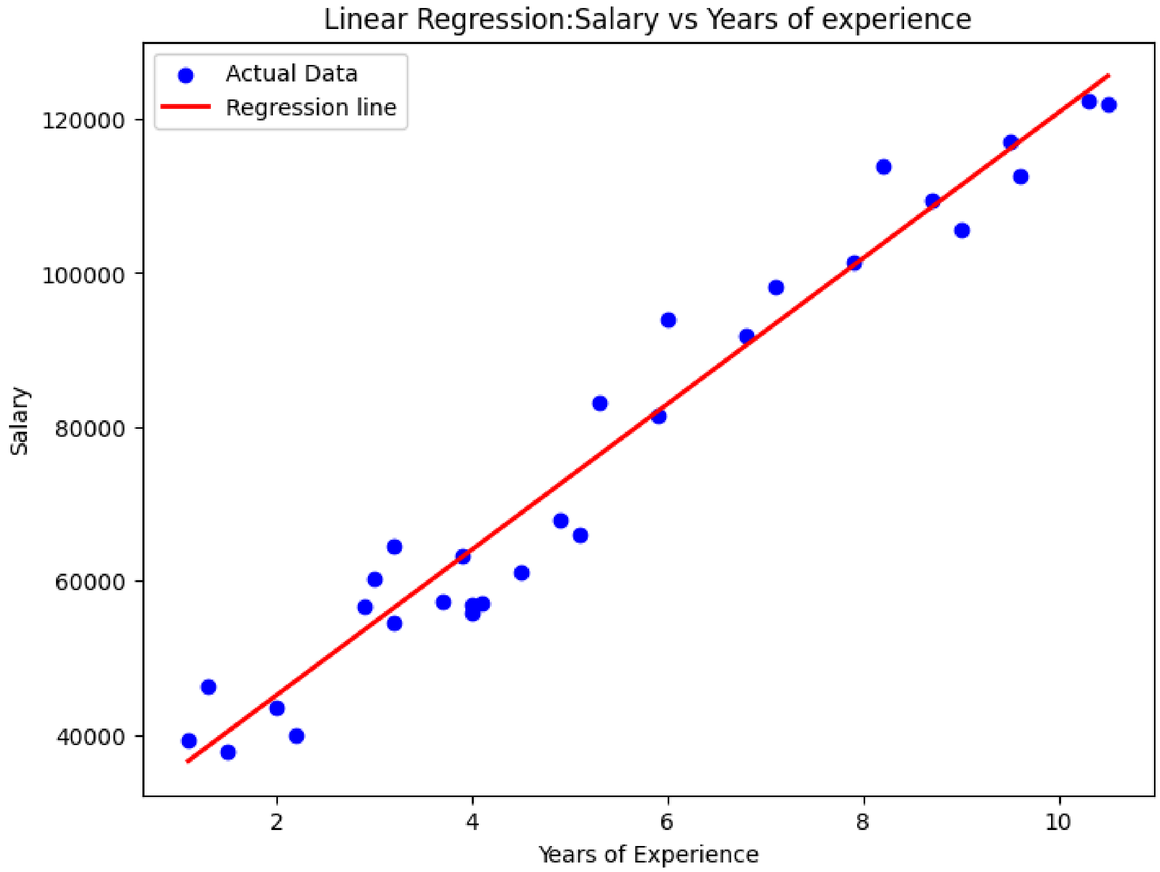
In [3]:
```
#EXPERIMENT 1
#Implement and demonstrate the FIND-S algorithm for finding the most specific hy
#Read the training data from a csv file.
import pandas as pd
import numpy as np
df=pd.read_csv(r"C:\ml datasets\ml1.csv")
df
concepts=np.array(df)[:,:-1]
concepts
target=np.array(df)[:,-1]
target
def train(c,t):
    for i,val in enumerate(t):
        if val=='yes':
            specific_hypothesis=c[i].copy()
            break
    for i,val in enumerate(c):
        if t[i]=='yes':
            for x in range (len(specific_hypothesis)):
                if val[x]!=specific_hypothesis[x]:
                    specific_hypothesis[x]='?'
    return specific_hypothesis
print("\nThe maximally specific find-S hypothesis for the given training example
print("final hypothesis",train(concepts, target))
```

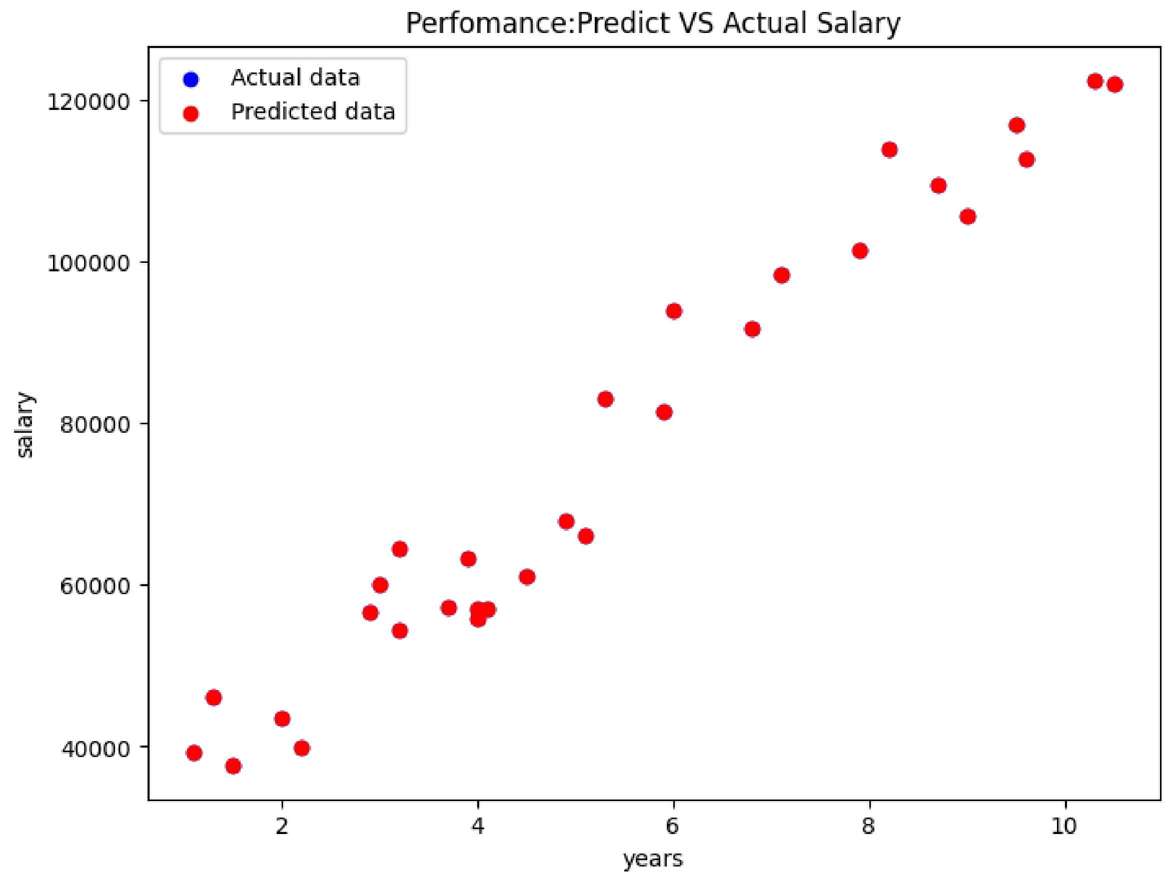
```
The maximally specific find-S hypothesis for the given training examples is:
final hypothesis ['Sunny' 'Warm' '?' 'Strong' '?' '?']
```

In [4]:
```
# EXPERIMENT 2
# implement simple linear regression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
data=pd.read_csv(r"C:\ml datasets\ml2.csv")
print(data.head())
x=data[['Years of Experience']]
y=data['Salary']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=4)
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
print(f"Mean Squared error:{mse}")
print(f"R Squared error:{r2}")
plt.figure(figsize=(8,6))
plt.scatter(x,y,color='blue',label='Actual Data')
plt.plot(x,model.predict(x),color='red',linewidth=2,label='Regression line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Linear Regression:Salary vs Years of experience')
plt.legend()
plt.show()
plt.figure(figsize=(8,6))
plt.scatter(x,y,color='blue',label='Actual data')
plt.scatter(x,y,color='red',label='Predicted data')
```

```
plt.xlabel('years')
plt.ylabel('salary')
plt.title("Perfomance:Predict VS Actual Salary")
plt.legend()
plt.show()
```

```
   Years of Experience   Salary
0                  1.1  39343.0
1                  1.3  46205.0
2                  1.5  37731.0
3                  2.0  43525.0
4                  2.2  39891.0
Mean Squared error:24942746.410306115
R Squared error:0.9504404484884267
```



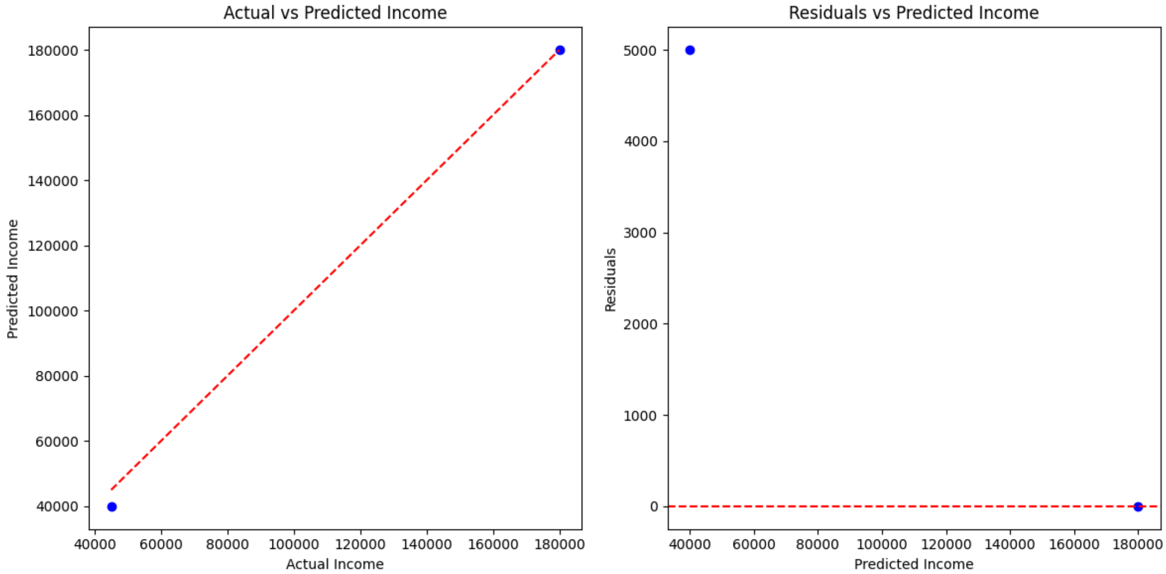


In [7]:

```
# EXPERIMENT 3
# implement multi linear regression
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv(r"C:\MLdatasets\ml3.csv")
data.head()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
x=data[['age','experience']]
y=data['income']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
coefficients = model.coef_
intercept = model.intercept_
print("Mean Squared Error(MSE):",mse)
print ("R2 Score:",r2)
print("Coefficients:",coefficients)
print("Intercept:",intercept)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred, color='blue')
plt.title('Actual vs Predicted Income')
plt.xlabel('Actual Income')
plt.ylabel('Predicted Income')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', li
plt.subplot(1, 2, 2)
residuals = y_test - y_pred
```

```
plt.scatter(y_pred, residuals, color='blue')
plt.title('Residuals vs Predicted Income')
plt.xlabel('Predicted Income')
plt.ylabel('Residuals')
plt.axhline(y=0, color='red', linestyle='--')
plt.tight_layout()
plt.show()
```

```
Mean Squared Error(MSE): 12500000.000000145
R2 Score: 0.9972565157750343
Coefficients: [-6000. 10000.]
Intercept: 169999.99999999994
```

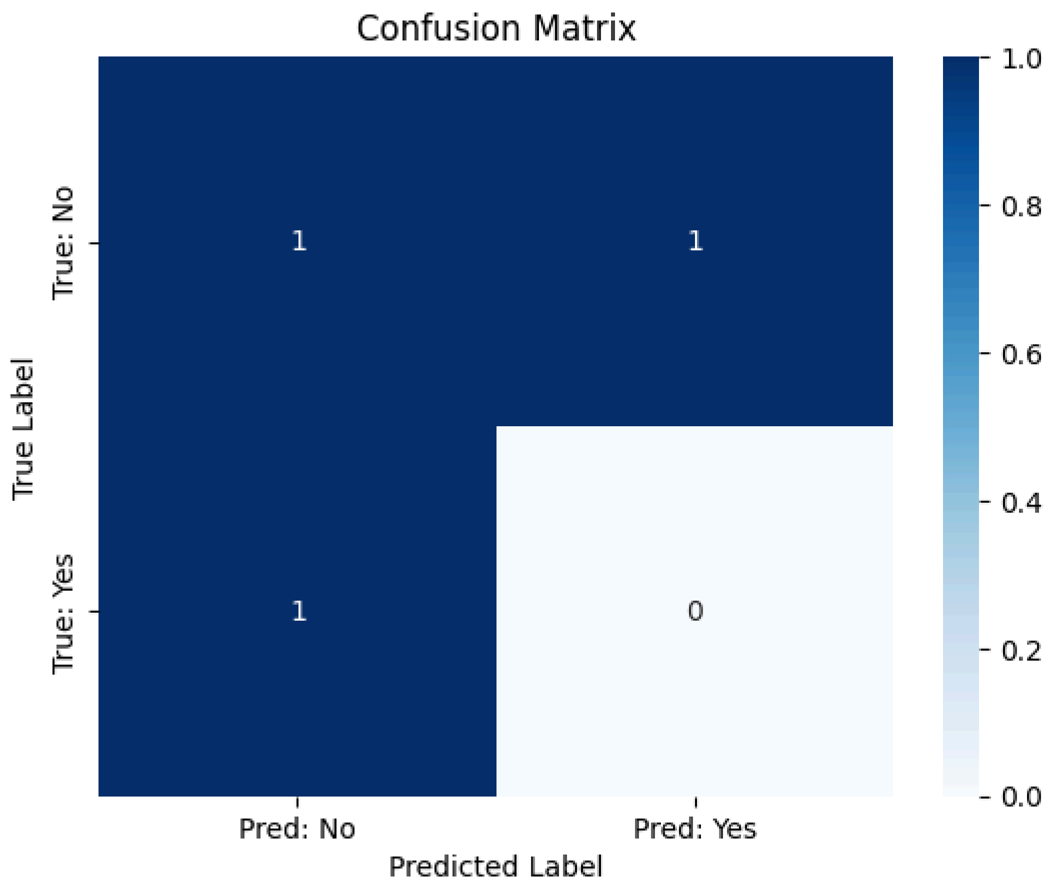


In [10]:

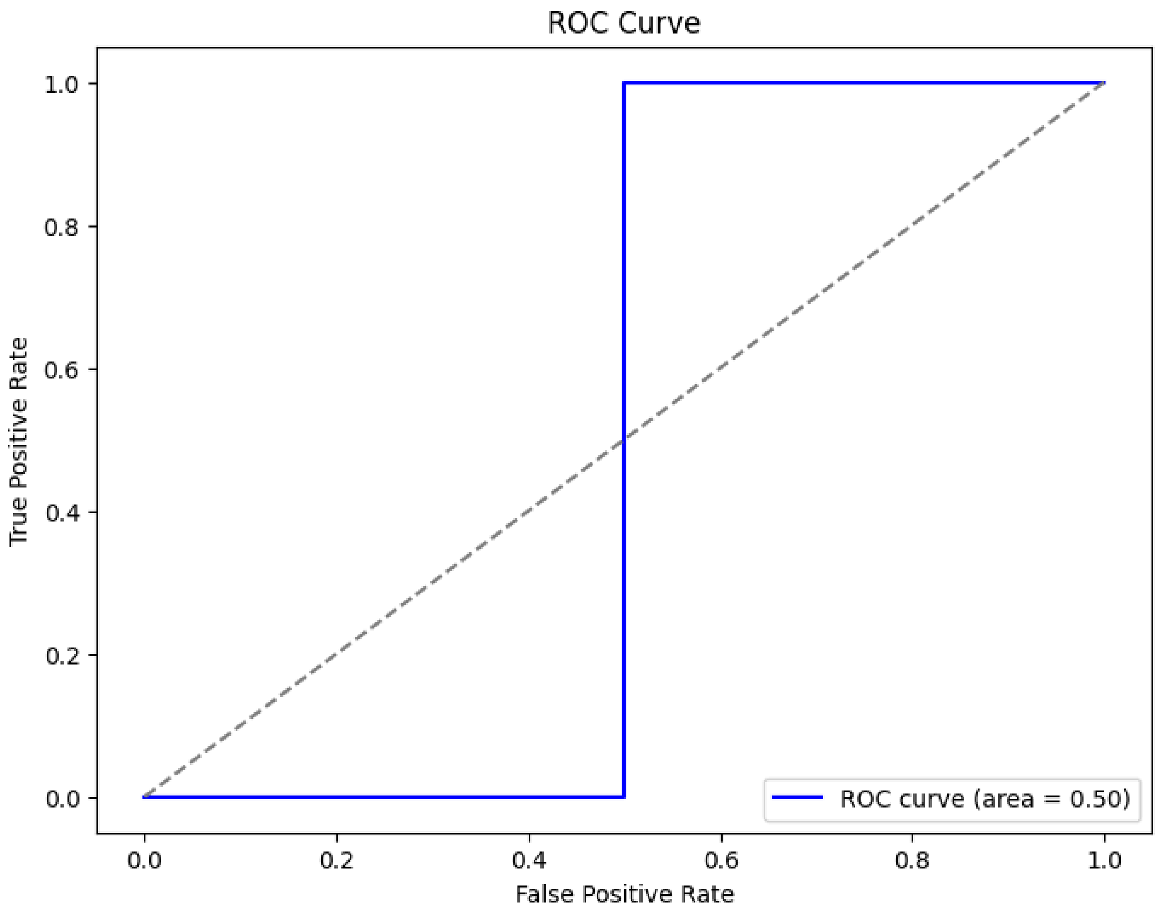
```
# EXPERIMENT 4
# Implement Logistic Regression
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve, auc, cl
df = pd.read_csv(r"C:\MLdatasets\ml4.csv")
print(df.head())
print("Class Distribution:\n", df['target'].value_counts())
x = df.drop(columns='target')
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
st_x = StandardScaler()
x_train = st_x.fit_transform(x_train)
x_test = st_x.transform(x_test)
classifier = LogisticRegression(class_weight='balanced')
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Pred: No', 'Pre
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy * 100, "%")
```

```
classreport = classification_report(y_test, y_pred, zero_division=1)
print("\nClassification Report:\n", classreport)
y_prob = classifier.predict_proba(x_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
Class Distribution:
 target
1    5
0    5
Name: count, dtype: int64
```



```
Accuracy: 33.33333333333333 %

Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.00      0.00      0.00         1

    accuracy                           0.33         3
   macro avg       0.25      0.25      0.25         3
weighted avg       0.33      0.33      0.33         3
```



In [45]:

```
# EXPERIMENT 5
# DATA PREPROCESSING FOR CLASSIFICATION
import numpy as np
import pandas as pd
df=pd.read_csv(r"C:\datasets\l5.csv")
df
x=df.iloc[:,:-1]
x=df.iloc[:,:-1]
x
y=df.iloc[:,-1]
df['AGE']=df["AGE"].fillna(df['AGE'].mean())
df['INCOME']=df['INCOME'].fillna(df['INCOME'].mean())
df
```

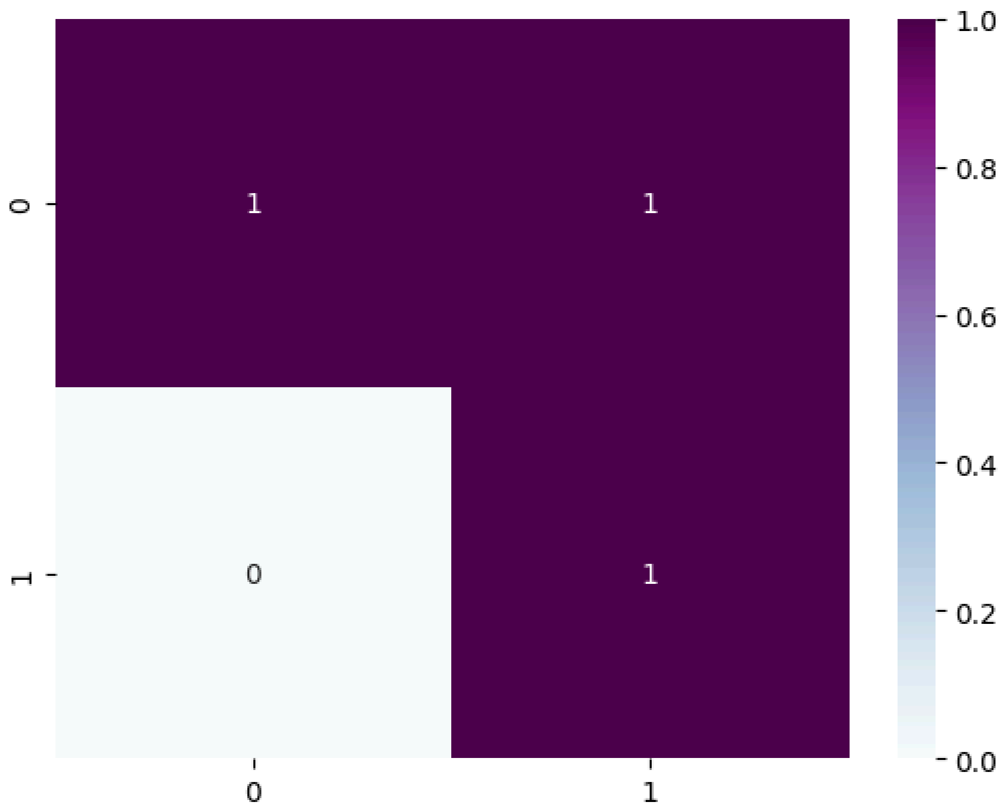
Out[45]:

	AGE	INCOME	EDUCATION	GENDER	TARGET
0	32.10	52000.0	HIGH-SCHOOL	MALE	0
1	28.50	48000.0	BACHELOR	FEMALE	1
2	35.00	55000.0	MASTER	MALE	0
3	31.35	53000.0	PHD	FEMALE	1
4	29.80	52000.0	BACHELOR	MALE	0

In [16]:

```
# EXPERIMENT 6
# CONFUSION MATRIX FOR BINARY CLASSIFIER
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, classification_rep
from sklearn.preprocessing import StandardScaler
df = pd.read_csv(r"C:\datasets\16.csv")
x = df.drop(['target'], axis=1)
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
lgr = LogisticRegression(max_iter=1000)
model = lgr.fit(x_train_scaled, y_train)
prediction = model.predict(x_test_scaled)
cm = confusion_matrix(y_test, prediction)
print("Confusion Matrix:")
print(cm)
sns.heatmap(cm, annot=True, cmap='BuPu')
plt.show()
accuracy = accuracy_score(y_test, prediction)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, prediction))
```

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




```
Accuracy: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.50      1.00      0.67         1

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3
```

In [17]:

```python
# EXPERIMENT 7
# IMPLEMENT SUPPORT VECTOR MACHINE
import numpy as np
import pandas as pd
from sklearn import datasets
cancer = datasets.load_breast_cancer()
print(cancer.target)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(cancer.data,cancer.target,test_si
from sklearn import svm
clf = svm.SVC(kernel='linear')
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
print("Precision:",metrics.precision_score(y_test,y_pred))
print("Recall:",metrics.recall_score(y_test,y_pred))
```

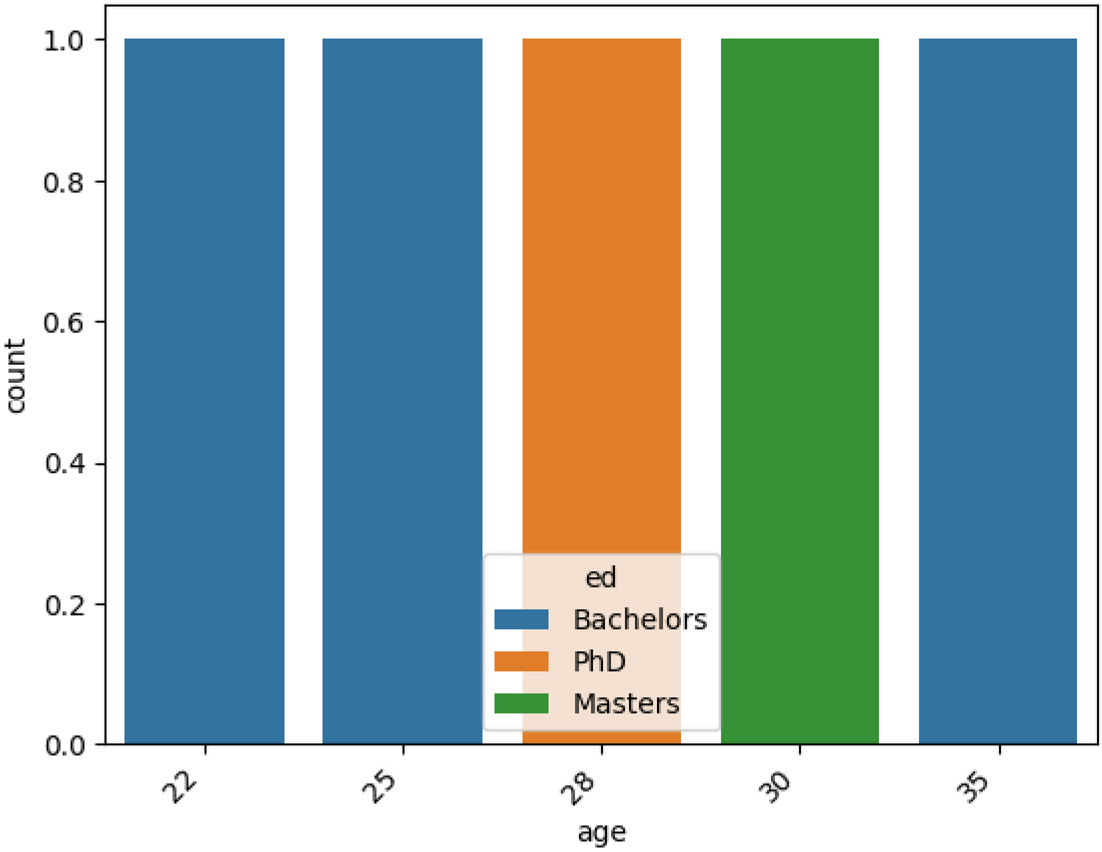
```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
Accuracy: 0.9590643274853801
Precision: 0.9809523809523809
Recall: 0.9537037037037037
```

In [50]:

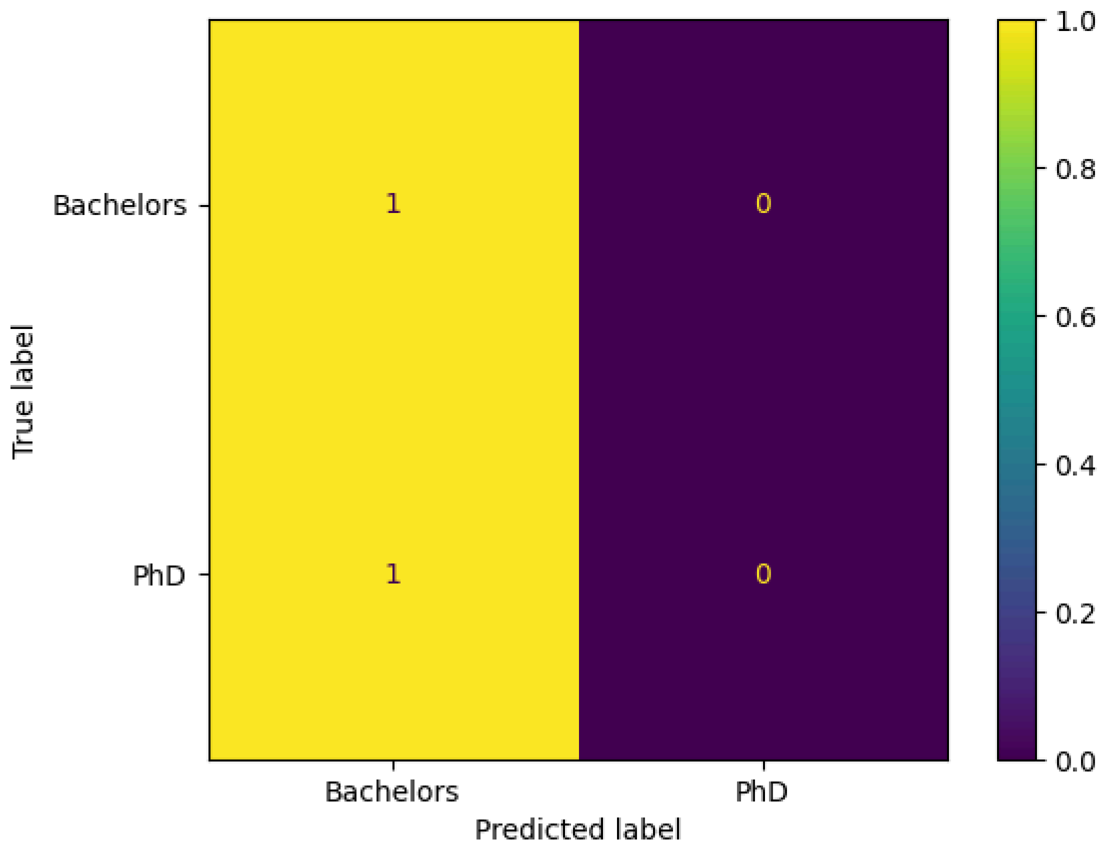
```python
# EXPERIMENT 8
# WRITE A PROGRAM IMPLEMENT THE NAIVE BAYESIAN CLASSIFIER FOR A SAMPLE TRAINING
# OF THE CLASSIFIER, CONSIDERING FEW TEST DATASETS
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
    classification_report
)
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\datasets\l8.csv")
print("Initial Data Types and First Few Rows:")
print(df.dtypes)
print(df.head())

# Step 2: Visualize the data
sns.countplot(data=df, x='age', hue='ed')
plt.xticks(rotation=45, ha='right')
plt.show()
# Step 3: Preprocess the data
pre_df = pd.get_dummies(df, columns=['age'], drop_first=True)
print("\nData after Encoding:")
print(pre_df.head())
# Step 4: Split into features (x) and target (y)
x = pre_df.drop('ed', axis=1)
y = pre_df['ed']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random
model = GaussianNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
accuracy = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted", zero_division=0)
print("Accuracy:", accuracy)
print("F1_score:", f1)
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(conf_matrix, display_labels=model.classes_).plot()
plt.show()
print(classification_report(y_test, y_pred, zero_division=0))
```

```
Initial Data Types and First Few Rows:
age             int64
ed             object
income          int64
savings         int64
loan_amount     int64
dtype: object
   age         ed  income  savings  loan_amount
0   25  Bachelors   50000    10000        20000
1   30    Masters   60000    15000        25000
2   22  Bachelors   40000     8000        18000
3   28        PhD   70000    20000        30000
4   35  Bachelors   80000    80000        25000
```




```
Data after Encoding:
          ed  income  savings  loan_amount  age_25  age_28  age_30  age_35
0  Bachelors   50000    10000        20000    True   False   False   False
1    Masters   60000    15000        25000   False   False    True   False
2  Bachelors   40000     8000        18000   False   False   False   False
3        PhD   70000    20000        30000   False    True   False   False
4  Bachelors   80000    80000        25000   False   False   False    True
Accuracy: 0.5
F1_score: 0.6666666666666666
```



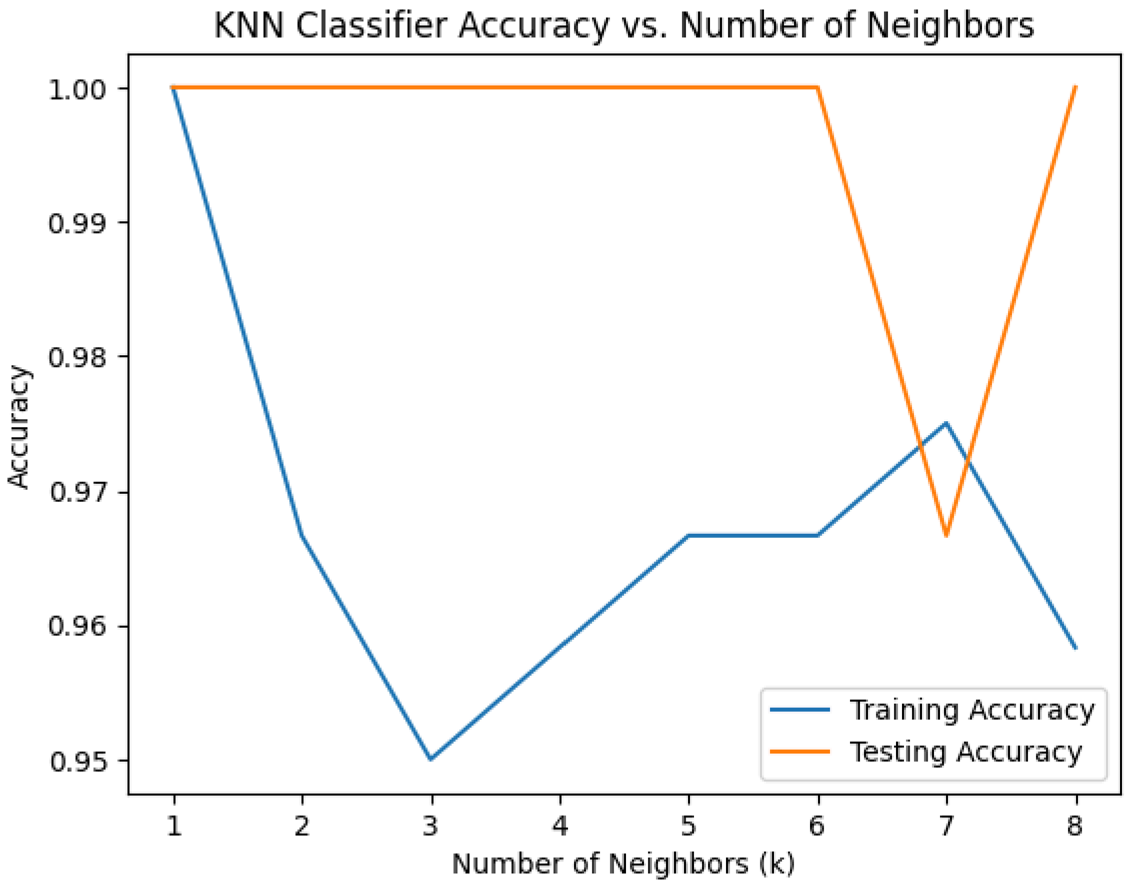

```
              precision    recall  f1-score   support

   Bachelors       0.50      1.00      0.67         1
     Masters       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```

In [31]:

```
# EXPERIMENT 9
# WRITE A PROGRAM TO IMPLEMENT KNEAREST NEIGHBOUR ALGORITHM TO CLASSIFY THE IRIS
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
iris = load_iris()
x = iris.data
y = iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)
    train_accuracy[i] = knn.score(x_train, y_train)
    test_accuracy[i] = knn.score(x_test, y_test)
    y_pred = knn.predict(x_test)
    correct_preds = np.sum(y_pred == y_test)
    incorrect_preds = np.sum(y_pred != y_test)
    print(f"K={k}: Correct Predictions = {correct_preds}, Incorrect Predictions
plt.plot(neighbors, train_accuracy, label='Training Accuracy')
```

```
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Accuracy')
plt.title('KNN Classifier Accuracy vs. Number of Neighbors')
plt.show()
```

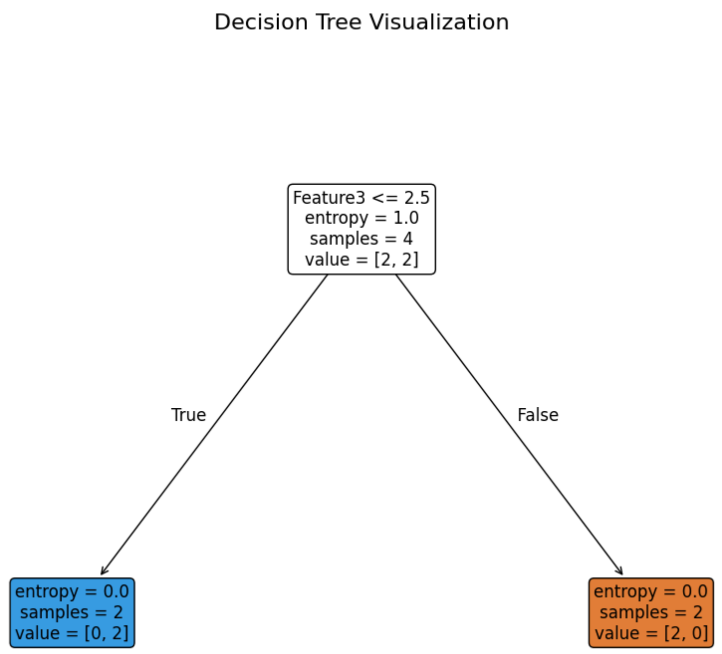
```
K=1: Correct Predictions = 30, Incorrect Predictions = 0
K=2: Correct Predictions = 30, Incorrect Predictions = 0
K=3: Correct Predictions = 30, Incorrect Predictions = 0
K=4: Correct Predictions = 30, Incorrect Predictions = 0
K=5: Correct Predictions = 30, Incorrect Predictions = 0
K=6: Correct Predictions = 30, Incorrect Predictions = 0
K=7: Correct Predictions = 29, Incorrect Predictions = 1
K=8: Correct Predictions = 30, Incorrect Predictions = 0
```



In [44]:
```
# # EXPERIMENT 10
# # WRITE A PROGRAM TO DEMONSTRATE THE WORKING OF THE DECISION TREE BASED ID3 AL
# AN APPROPRIATE DATA THIS KNOWLEDGE TO CLASSIFY A NEW SAMPLE.
import pandas as pd
from sklearn . model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_text , plot_tree
import matplotlib.pyplot as plt
df=pd.read_csv(r"C:\datasets\l10.csv")
df
print("Intial Data Types and First Few Rows:")
print(df.dtypes)
print(df.head())
label_encoders = {}
for column in df.columns:
    if df[column].dtype == 'object' :
        print(f"Encoding column :{column}")
```

```
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        label_encoders[column] =le
print("\n Data after Encoding:")
print(df.head())
x=df.iloc[:, :-1]
y=df.iloc[:,-1]
x
y
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.3, random_state=
id3_tree = DecisionTreeClassifier(criterion="entropy",random_state=42)
id3_tree.fit(x_train,y_train)
accuracy = id3_tree.score (x_test,y_test)
print(f"accuracy :{accuracy:.2f}")
tree_rules = export_text(id3_tree,feature_names=df.columns[:-1].tolist())
print(tree_rules)
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
plt.figure(figsize=(15, 10))
plot_tree(
    id3_tree,
    feature_names=df.columns[:-1],
    filled=True,
    rounded=True,
    fontsize=12,
    precision=2
)
plt.title("Decision Tree Visualization", fontsize=16)
plt.show()
```

```
Intial Data Types and First Few Rows:
Feature1    object
Feature2    object
Feature3     int64
Target      object
dtype: object
  Feature1 Feature2  Feature3 Target
0        A        X         1    Yes
1        B        Y         2     No
2        A        X         1    Yes
3        C        Z         3     No
4        B        Y         2    Yes
Encoding column :Feature1
Encoding column :Feature2
Encoding column :Target

 Data after Encoding:
   Feature1  Feature2  Feature3  Target
0         0         0         1       1
1         1         1         2       0
2         0         0         1       1
3         2         2         3       0
4         1         1         2       1
accuracy :0.50
|--- Feature3 <= 2.50
|   |--- class: 1
|--- Feature3 >  2.50
|   |--- class: 0
```


Decision Tree Visualization
Feature3 <= 2.5
entropy = 1.0
samples = 4
value = [2, 2]
True
False
entropy = 0.0
samples = 2
value = [0, 2]
entropy = 0.0
samples = 2
value = [2, 0]

In [ ]: