Import necessary libraries

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import LabelEncoder, OneHotEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
        from sklearn.compose import ColumnTransformer
```

Load the dataset

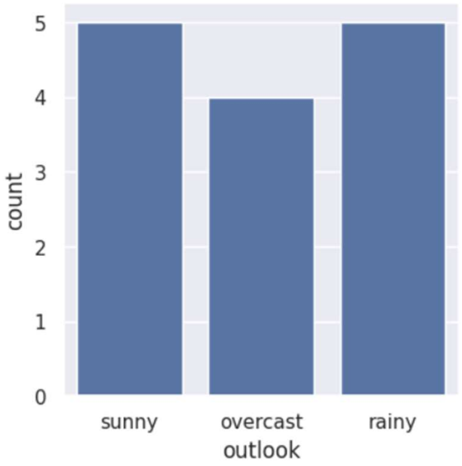
```
In [2]: ##Read  the data set
        from google.colab import drive
        drive.mount("/content/drive")
        path = "/content/drive/MyDrive/Dataset/PlayTennis.csv"
        df = pd.read_csv(path)

        value=['Outlook','Temprature','Humidity','Wind']
        print(df)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
     outlook  temp humidity  windy play
0      sunny   hot     high  False   no
1      sunny   hot     high   True   no
2   overcast   hot     high  False  yes
3      rainy  mild     high  False  yes
4      rainy  cool   normal  False  yes
5      rainy  cool   normal   True   no
6   overcast  cool   normal   True  yes
7      sunny  mild     high  False   no
8      sunny  cool   normal  False  yes
9      rainy  mild   normal  False  yes
10     sunny  mild   normal   True  yes
11  overcast  mild     high   True  yes
12  overcast   hot   normal  False  yes
13     rainy  mild     high   True   no
```

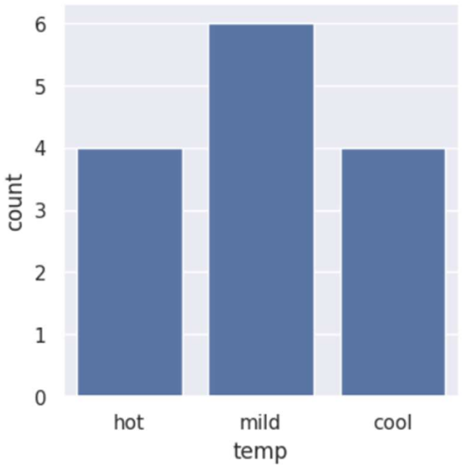
```
In [4]: df.head()
        df.shape

Out[4]: (14, 5)
```

EDA

```
In [5]: ax = plt.subplots(figsize = (4,4))
        ax = sns.countplot(x=df['Outlook'])
        plt.show()
```

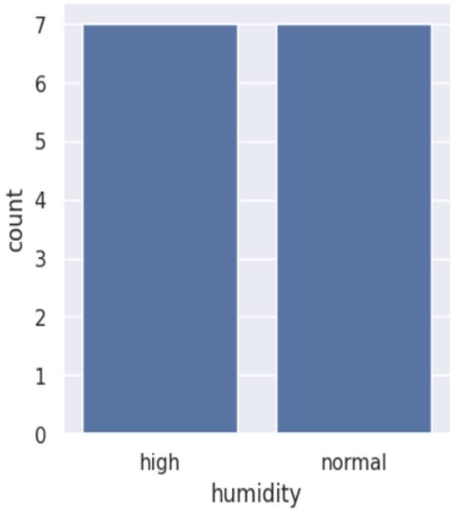


```
In [6]: ax = plt.subplots(figsize = (4,4))
        ax = sns.countplot(x=df['Temperature'])
        plt.show()
```



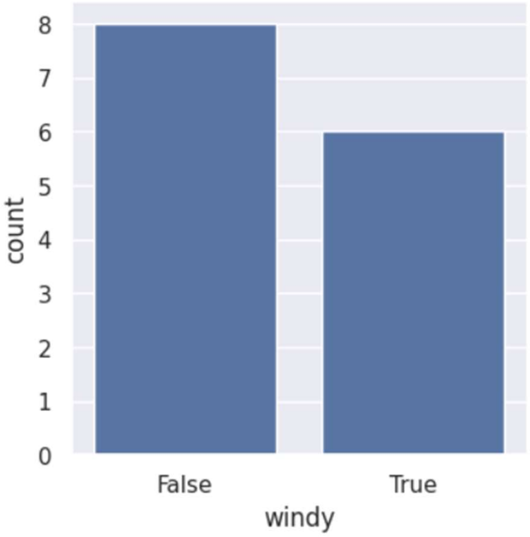
In [7]:

```
ax  =  plt.subplots(figsize  =  (4,4))
ax = sns.countplot(x=df['Humidity'])
plt.show()
```



In [8]:

```
ax = plt.subplots(figsize = (4,4))
ax = sns.countplot(x=df['Wind'])
plt.show()
```



Feature Extraction

In [9]:
```
# Separate features (X) and target variable (y)
X = df.iloc[:, :-1]  # All columns except the last one
y = df.iloc[:, -1]   # The last column 'PlayTennis'
```

Use OneHotEncoder for categorical variables in features

In [10]:
```
categorical_features = ['Outlook', 'Temperature', 'Humidity', 'Wind']

preprocessor = ColumnTransformer(
    transformers=[
        ('encoder', OneHotEncoder(), categorical_features)
    ],
    remainder='passthrough'
)

X_encoded = preprocessor.fit_transform(X)
```

Use LabelEncoder for the target variable

In [11]:
```
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

Split the dataset into training and testing sets

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.2, random_state=42)
```

Create a Decision Tree classifier

In [13]:
```
dt_classifier = DecisionTreeClassifier(random_state=42)
```

Fit the model to the training data

In [14]:
```
dt_classifier.fit(X_train, y_train)
```

Out[14]:
```
▾          DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

Make predictions on the test set

In [15]:
```
y_pred = dt_classifier.predict(X_test)
```

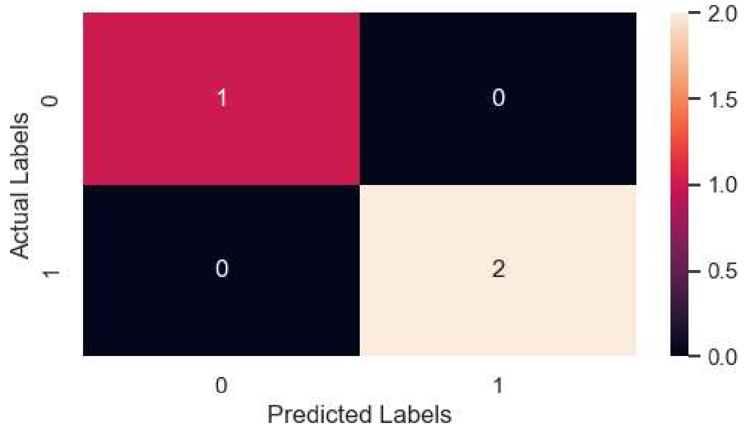
Evaluate the performance of the classifier

In [16]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_report_str = classification_report(y_test, y_pred)
```

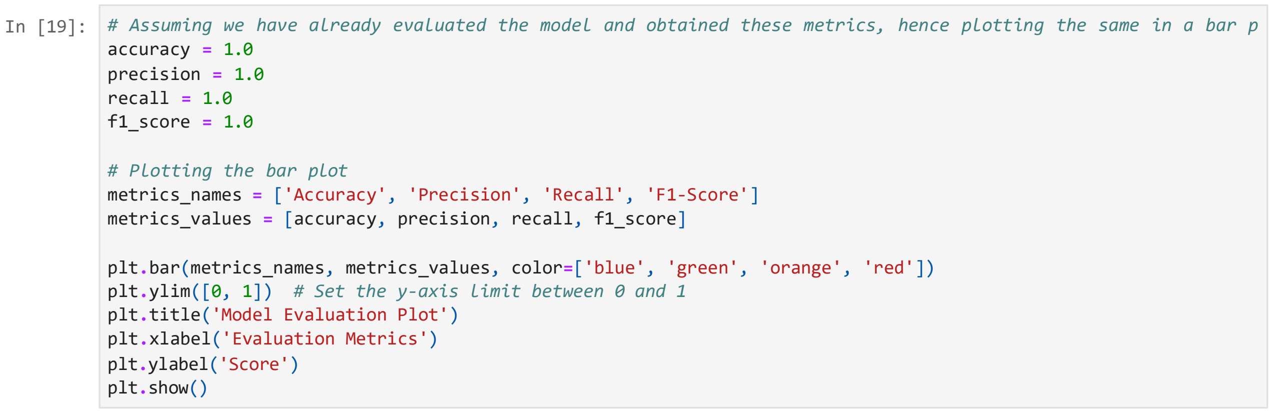
Print the results

In [17]:
```
print(f'DT Accuracy: {accuracy}')
print(f'DT Confusion Matrix:\n{conf_matrix}')
sns.set(rc={'figure.figsize':(6,3)})
sns.heatmap(confusion_matrix(y_test,y_pred),annot = True,fmt = 'd')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
print(f'DT Classification Report:\n{classification_report_str}')
```

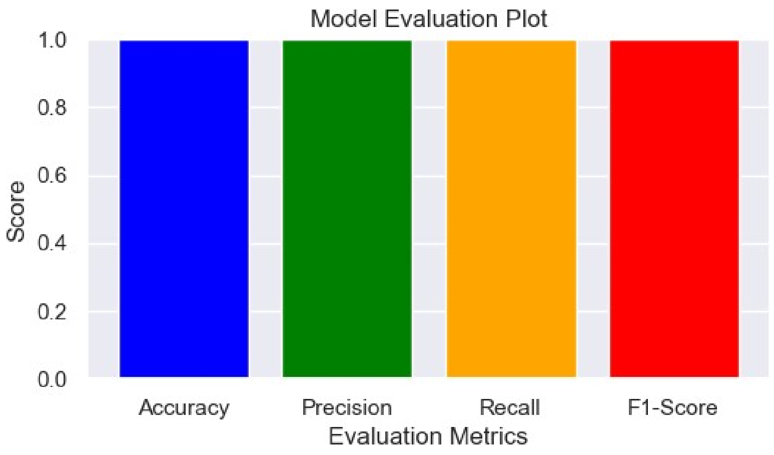
```
DT Accuracy: 1.0
DT Confusion Matrix:
[[1 0]
 [0 2]]
DT Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```



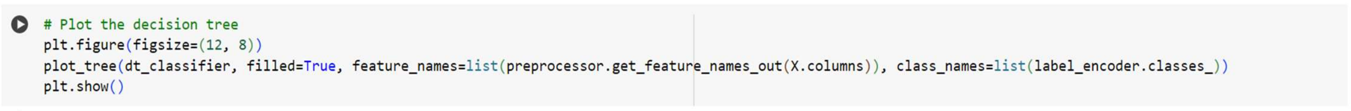
Output Visualization using Bar Plot



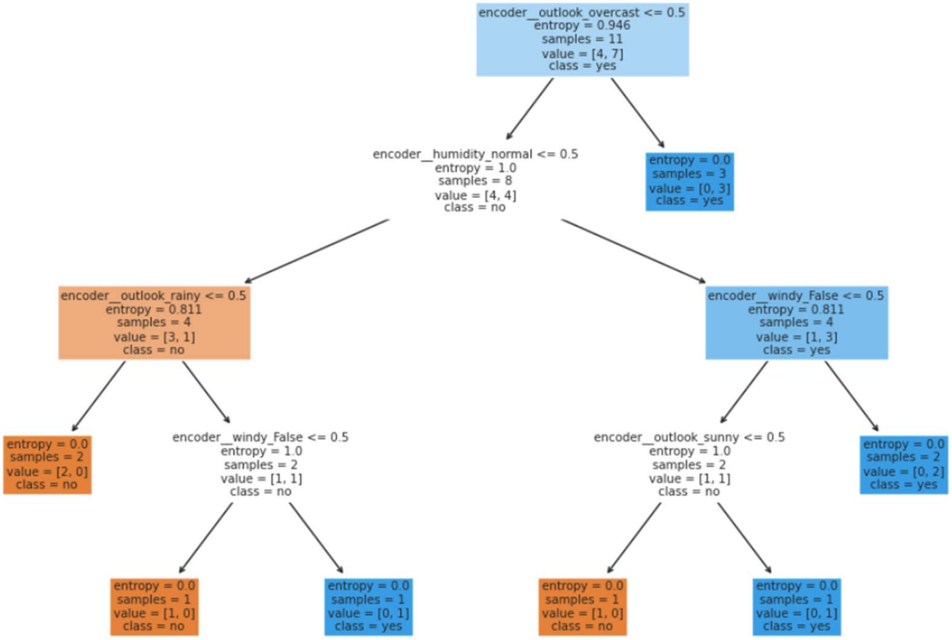
```
In [19]: # Assuming we have already evaluated the model and obtained these metrics, hence plotting the same in a bar p
         accuracy = 1.0
         precision = 1.0
         recall = 1.0
         f1_score = 1.0

         # Plotting the bar plot
         metrics_names = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
         metrics_values = [accuracy, precision, recall, f1_score]

         plt.bar(metrics_names, metrics_values, color=['blue', 'green', 'orange', 'red'])
         plt.ylim([0, 1])  # Set the y-axis limit between 0 and 1
         plt.title('Model Evaluation Plot')
         plt.xlabel('Evaluation Metrics')
         plt.ylabel('Score')
         plt.show()
```

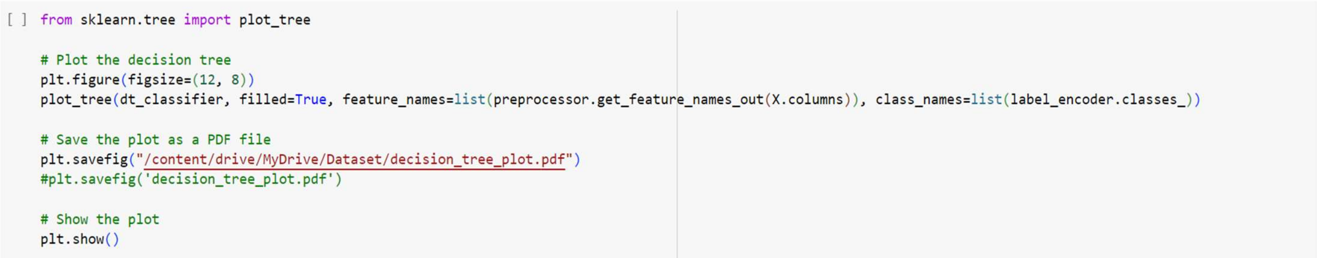


Plot the decision tree

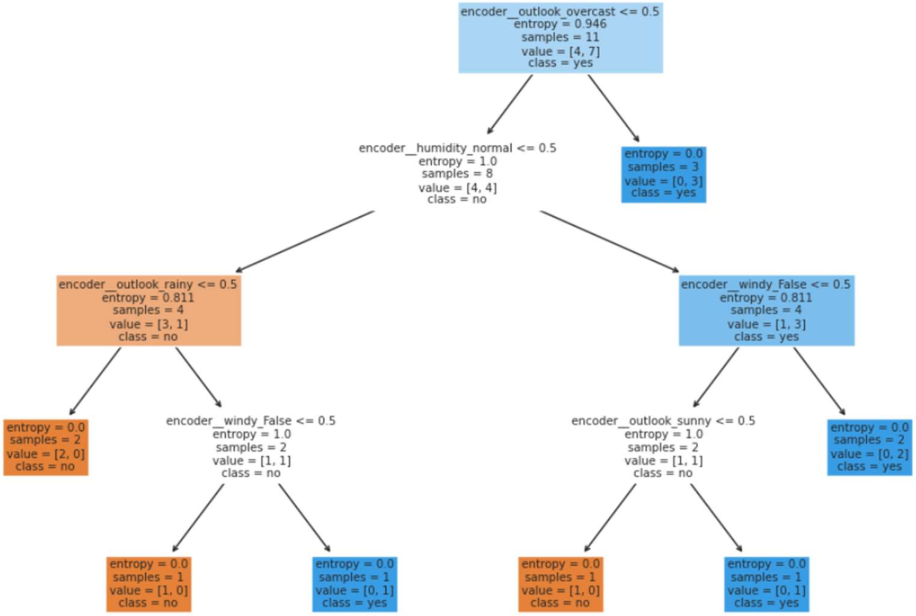


```
# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(dt_classifier, filled=True, feature_names=list(preprocessor.get_feature_names_out(X.columns)), class_names=list(label_encoder.classes_))
plt.show()
```





```
[ ] from sklearn.tree import plot_tree

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(dt_classifier, filled=True, feature_names=list(preprocessor.get_feature_names_out(X.columns)), class_names=list(label_encoder.classes_))

    # Save the plot as a PDF file
    plt.savefig("/content/drive/MyDrive/Dataset/decision_tree_plot.pdf")
    #plt.savefig('decision_tree_plot.pdf')

    # Show the plot
    plt.show()
```


encoder__outlook_overcast <= 0.5
entropy = 0.946
samples = 11
value = [4, 7]
class = yes
encoder__humidity_normal <= 0.5
entropy = 1.0
samples = 8
value = [4, 4]
class = no
entropy = 0.0
samples = 3
value = [0, 3]
class = yes
encoder__outlook_rainy <= 0.5
entropy = 0.811
samples = 4
value = [3, 1]
class = no
encoder__windy_False <= 0.5
entropy = 0.811
samples = 4
value = [1, 3]
class = yes
entropy = 0.0
samples = 2
value = [2, 0]
class = no
encoder__windy_False <= 0.5
entropy = 1.0
samples = 2
value = [1, 1]
class = no
encoder__outlook_sunny <= 0.5
entropy = 1.0
samples = 2
value = [1, 1]
class = no
entropy = 0.0
samples = 2
value = [0, 2]
class = yes
entropy = 0.0
samples = 1
value = [1, 0]
class = no
entropy = 0.0
samples = 1
value = [0, 1]
class = yes
entropy = 0.0
samples = 1
value = [1, 0]
class = no
entropy = 0.0
samples = 1
value = [0, 1]
class = yes