```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.compose import ColumnTransformer
```

Load Dataset

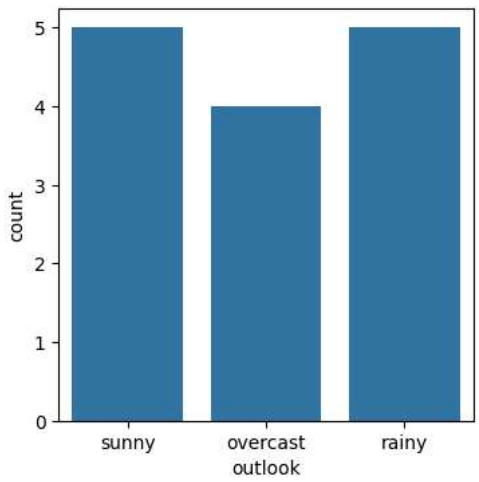
```
# Assuming the dataset is in a CSV file named 'PlayTennis.csv'
from google.colab import drive
drive.mount("/content/drive")
path = "/content/drive/MyDrive/Dataset/PlayTennis.csv"
df = pd.read_csv(path)

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

```
Mounted at /content/drive
     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
```

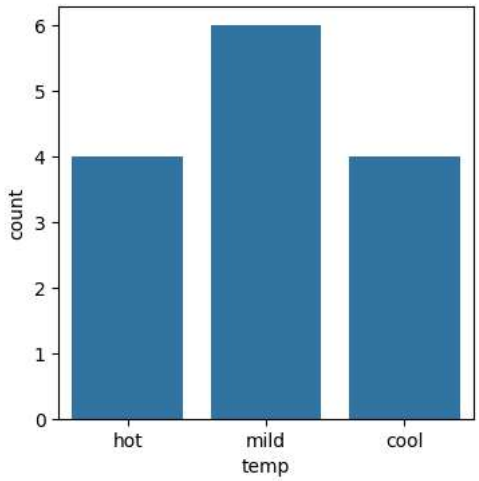
```
#dataset visualization
df.head()
df.shape
```

```
(14, 5)
```

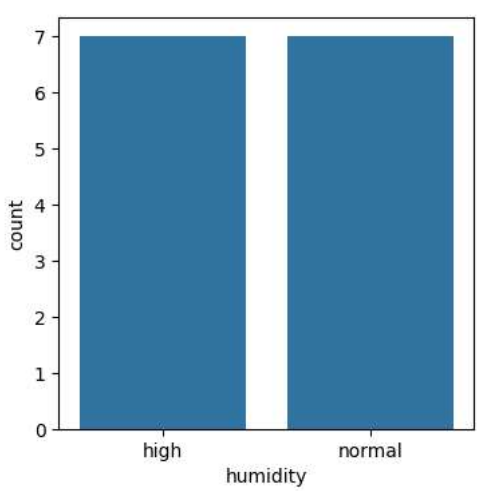
```
ax = plt.subplots(figsize = (4,4))
ax = sns.countplot(x=df['outlook'])
plt.show()
```




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

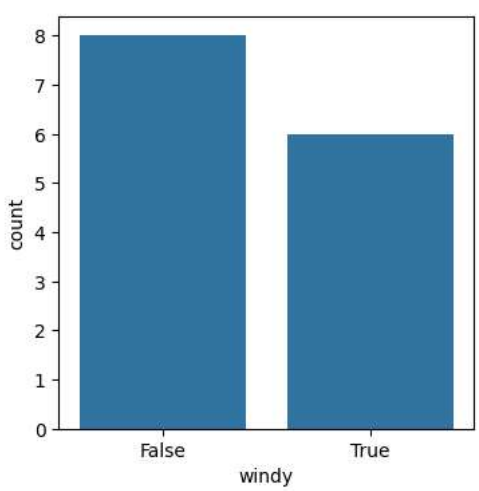



```
ax = plt.subplots(figsize = (4,4))
ax = sns.countplot(x=df['humidity'])
plt.show()
```



Start coding or generate with AI.

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



```
#feature Extraction
# 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

```
categorical_features = ['outlook', 'temp', 'humidity', 'windy']

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

X_encoded = preprocessor.fit_transform(X)
```

Use LabelEncoder for the target variable

```
#use LabelEncoder for target variable
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

Split the dataset into training and testing sets

```
#split the dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.2, random_state=42)
```

Create a Random Forest classifier

```
#create decision tree classifier
rf_classifier = RandomForestClassifier(n_estimators=50)
```

Fit the model to the training data

```
rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=50)
```

Make predictions on the test set

```
y_pred = rf_classifier.predict(X_test)
```

Evaluate the performance of the classifier

```
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

```
print(f'RF Accuracy: {accuracy}')
print(f'RF 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'RF Classification Report:\n{classification_report_str}')
```

```
RF Accuracy: 0.3333333333333333
RF Confusion Matrix:
[[0 1]
 [1 1]]
RF Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.50      0.50      0.50         2

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