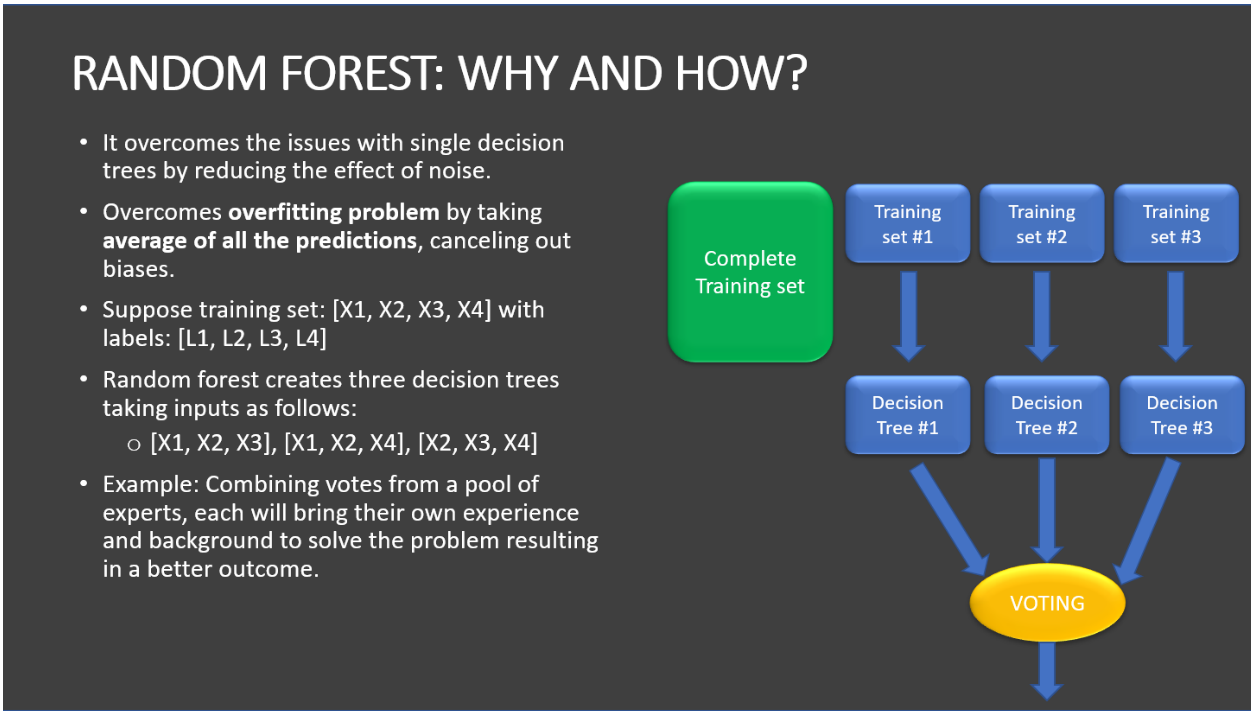
TASK 1: UNDERSTAND THE PROBLEM STATEMENT



TASK #2: IMPORT LIBRARIES AND DATASETS

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
data=pd.read_csv("kyphosis.csv")
```

```python
data.head()
```

	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

```python
data.tail()
```

```
   Kyphosis  Age  Number  Start
76  present  157       3     13
77   absent   26       7     13
78   absent  120       2     13
79  present   42       7      6
80   absent   36       4     13
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Kyphosis  81 non-null     object
 1   Age       81 non-null     int64 
 2   Number    81 non-null     int64 
 3   Start     81 non-null     int64 
dtypes: int64(3), object(1)
memory usage: 2.7+ KB
```

PRACTICE OPPORTUNITY #1 [OPTIONAL]:

- **List the average, minimum and maximum age (in years) considered in this study using two different methods**

```
#Method 1 - Using pandas
# Calculate statistics
average_age = data['Age'].mean()
min_age = data['Age'].min()
max_age = data['Age'].max()

print(f"Method 1 - Using pandas:")
print(f"Average Age: {average_age}")
print(f"Minimum Age: {min_age}")
print(f"Maximum Age: {max_age}")
```

```
Method 1 - Using pandas:
Average Age: 83.65432098765432
Minimum Age: 1
Maximum Age: 206
```

```
#Method 2 - Using built-in functions
ages = data['Age'].tolist()

average_age = sum(ages) / len(ages)
min_age = min(ages)
max_age = max(ages)

print(f"\nMethod 2 - Using built-in functions:")
print(f"Average Age: {average_age}")
```

```
print(f"Minimum Age: {min_age}")
print(f"Maximum Age: {max_age}")
```

```
Method 2 - Using built-in functions:
Average Age: 83.65432098765432
Minimum Age: 1
Maximum Age: 206
```

TASK #3: PERFORM DATA VISUALIZATION

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
LabelEncoder_y = LabelEncoder()
data['Kyphosis'] = LabelEncoder_y.fit_transform(data['Kyphosis'])

data
```

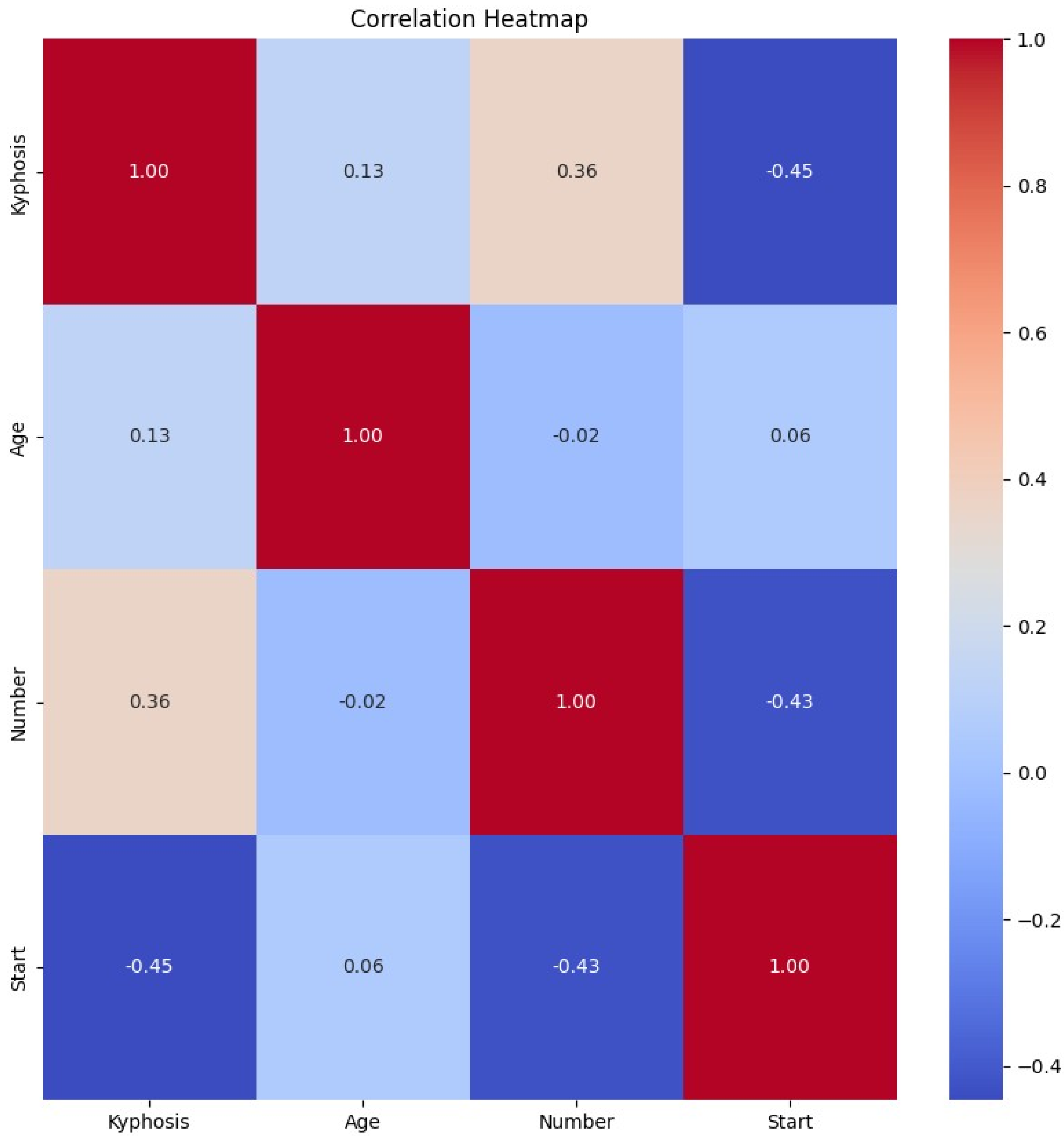
```
    Kyphosis  Age  Number  Start
0          0   71       3      5
1          0  158       3     14
2          1  128       4      5
3          0    2       5      1
4          0    1       4     15
..       ...  ...     ...    ...
76         1  157       3     13
77         0   26       7     13
78         0  120       2     13
79         1   42       7      6
80         0   36       4     13

[81 rows x 4 columns]
```

```
Kyphosis_df = pd.DataFrame(data)

Kyphosis_True = Kyphosis_df[Kyphosis_df['Kyphosis']==1]

Kyphosis_False = Kyphosis_df[Kyphosis_df['Kyphosis']==0]

print( 'Disease present after operation percentage =',
(len(Kyphosis_True) / len(Kyphosis_df) )*100,"%")
```

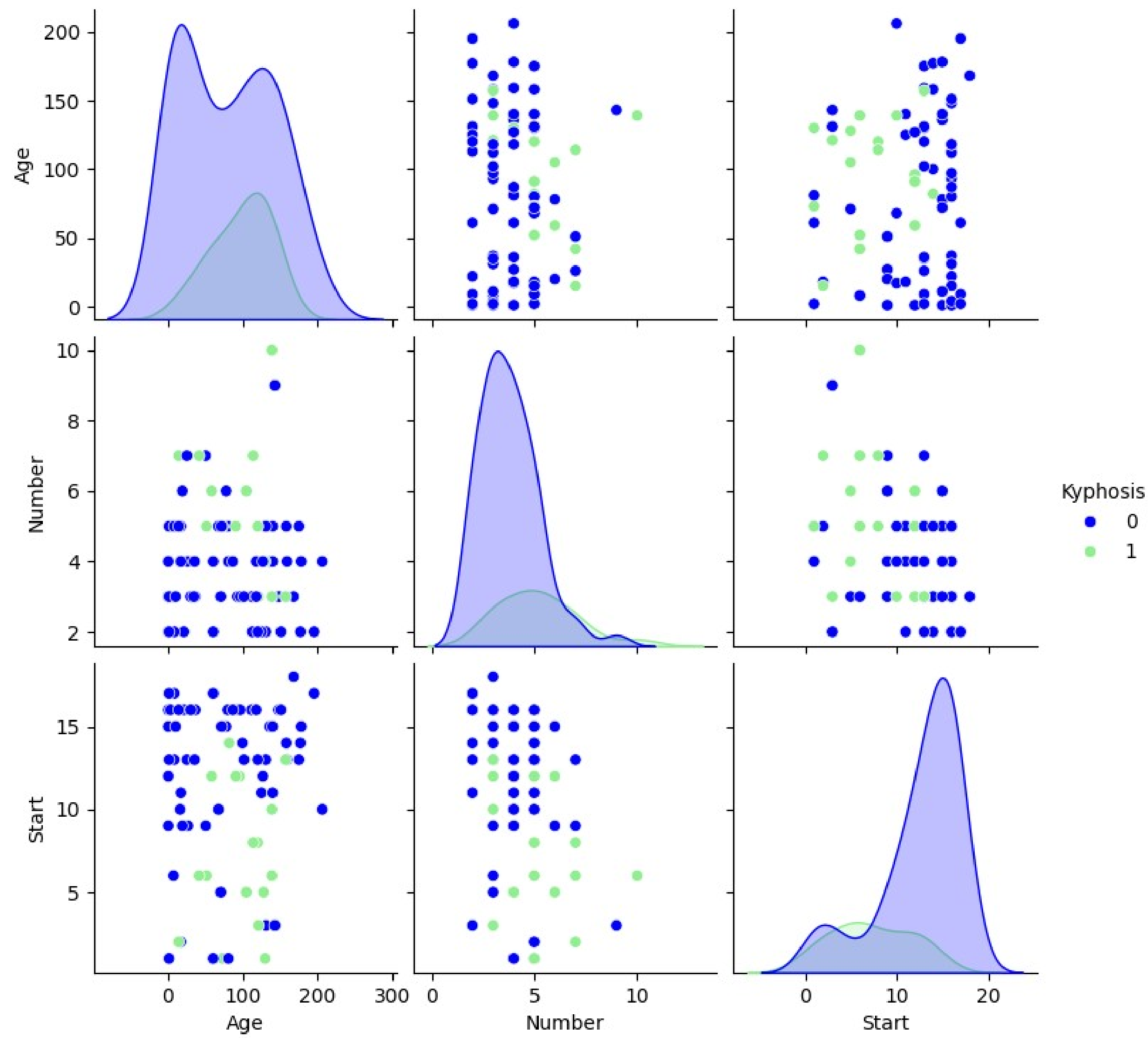
```
Disease present after operation percentage = 20.98765432098765 %
```

```
# Calculate the correlation matrix
corr = Kyphosis_df.corr()

# Plot the heatmap
plt.figure(figsize=(10,10))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
```

```python
plt.title('Correlation Heatmap')
plt.show()
```



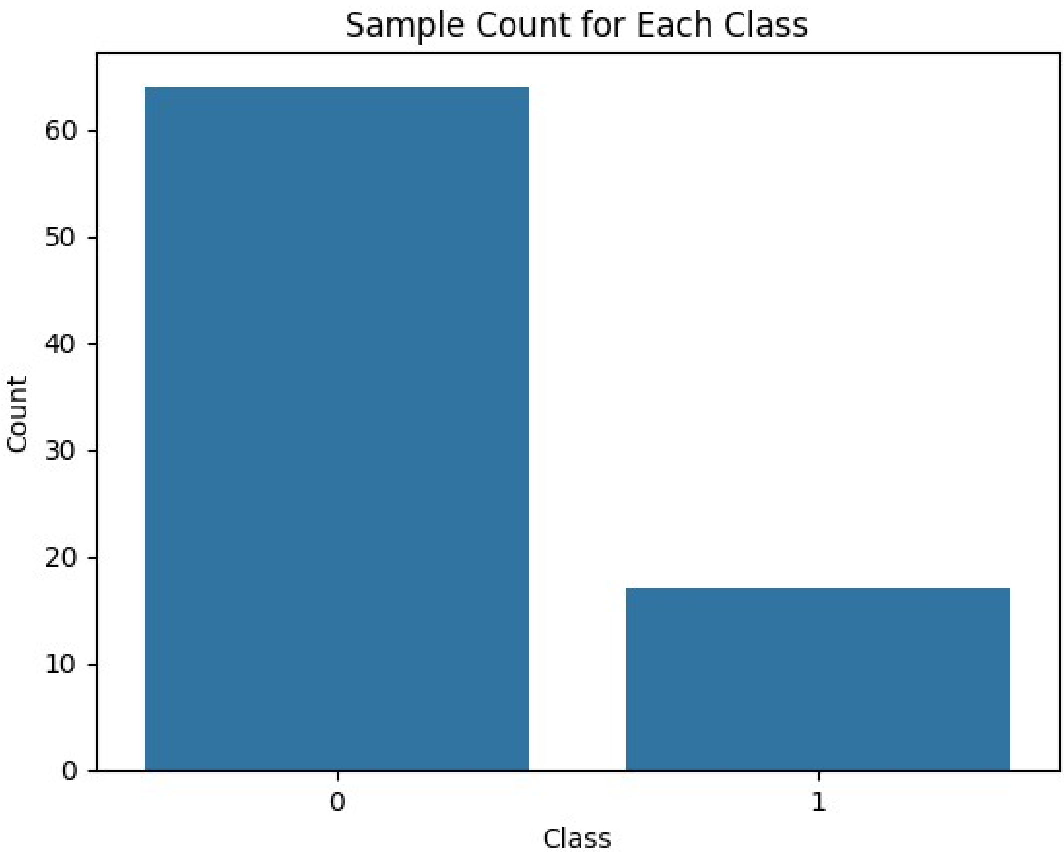

```python
# Create a pairplot with hue based on 'Kyphosis'
sns.pairplot(Kyphosis_df, hue='Kyphosis', palette={0: 'blue', 1:
'lightgreen'})

plt.show()
```



PRACTICE OPPORTUNITY #2 [OPTIONAL]:

- **Plot the data countplot showing how many samples belong to each class**

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot the countplot for the 'Kyphosis' column
sns.countplot(x='Kyphosis', data=Kyphosis_df)
plt.title('Sample Count for Each Class')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```



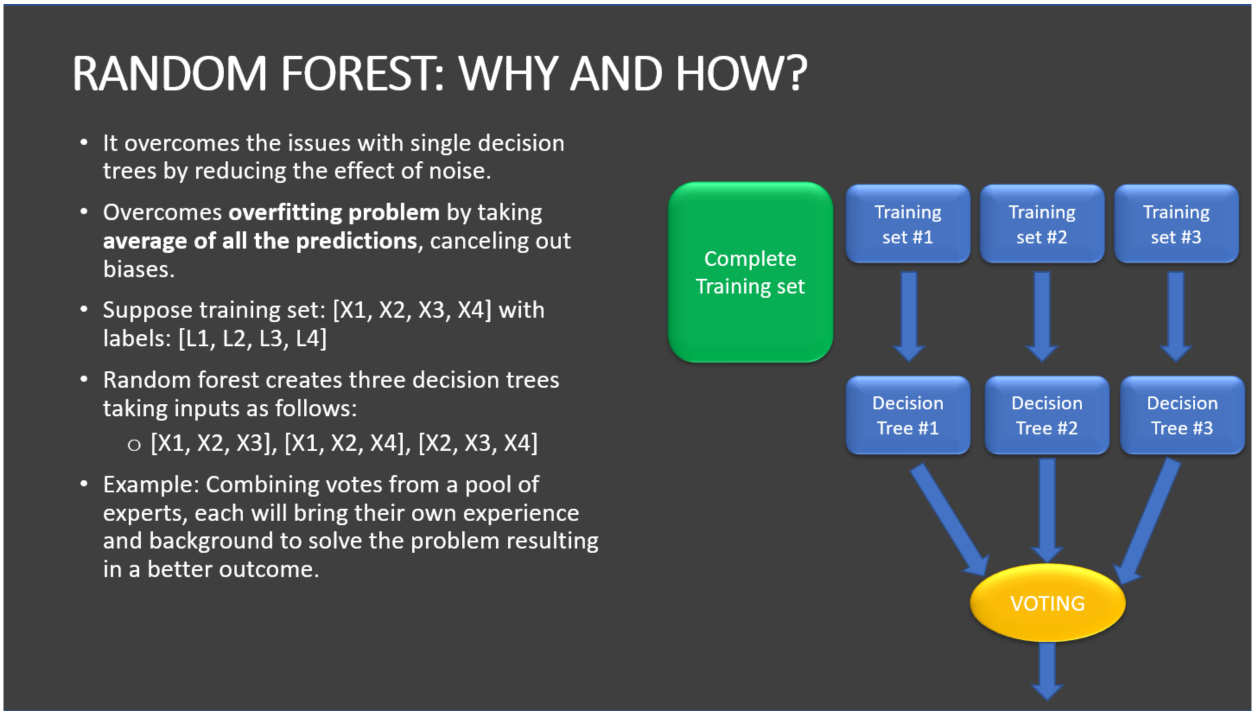
TASK #4: CREATE TESTING AND TRAINING DATASET/DATA CLEANING

```
# Let's drop the target label coloumns
X = Kyphosis_df.drop('Kyphosis', axis=1)

# Features (all columns except the target)
X
```

```
    Age  Number  Start
0    71       3      5
1   158       3     14
2   128       4      5
3     2       5      1
4     1       4     15
..  ...     ...    ...
76  157       3     13
77   26       7     13
78  120       2     13
79   42       7      6
80   36       4     13
```

```
[81 rows x 3 columns]

# Target label
y = Kyphosis_df['Kyphosis']

y

0     0
1     0
2     1
3     0
4     0
     ..
76    1
77    0
78    0
79    1
80    0
Name: Kyphosis, Length: 81, dtype: int32

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

TASK #5: TRAIN A LOGISTIC REGRESSION CLASSIFIER MODEL




```
X_train.shape
```

```
(64, 3)
```

```
y_train.shape
```

```
(64,)
```

```
X_test.shape
```

```
(17, 3)
```

```
y_test.shape
```

```
(17,)
```

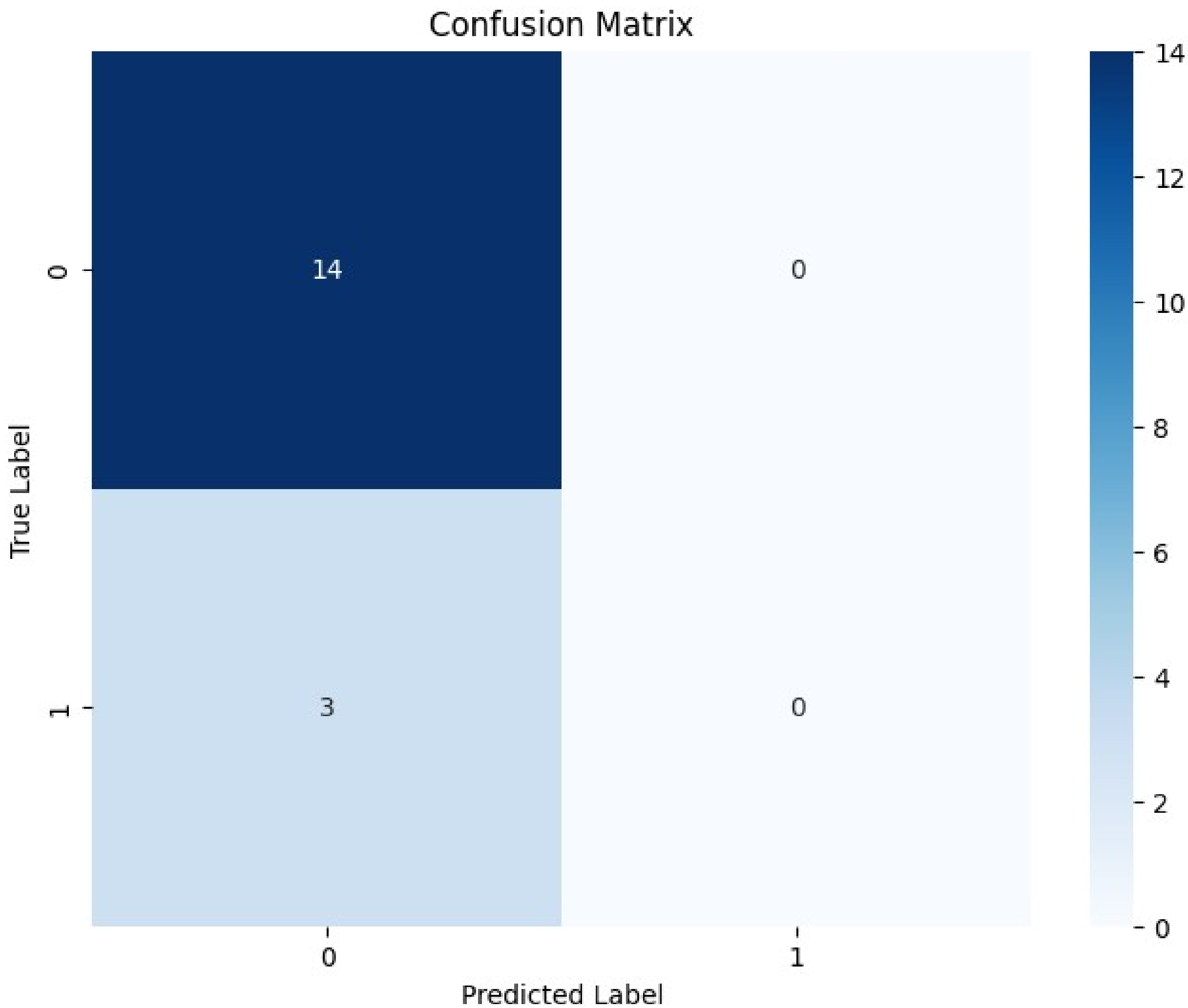
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Initialize the Logistic Regression classifier
lr = LogisticRegression(random_state=42)

# Train the model
lr.fit(X_train, y_train)
```

```python
# Predict on training data
y_pred_train = lr.predict(X_train)
accuracy_train = accuracy_score(y_train, y_pred_train)
print(f"Train Accuracy: {accuracy_train:.2f}")

# Predict on test data
y_pred_test = lr.predict(X_test)
accuracy_test = accuracy_score(y_test, y_pred_test)
print(f"Test Accuracy: {accuracy_test:.2f}")
```

```
Train Accuracy: 0.84
Test Accuracy: 0.82
```

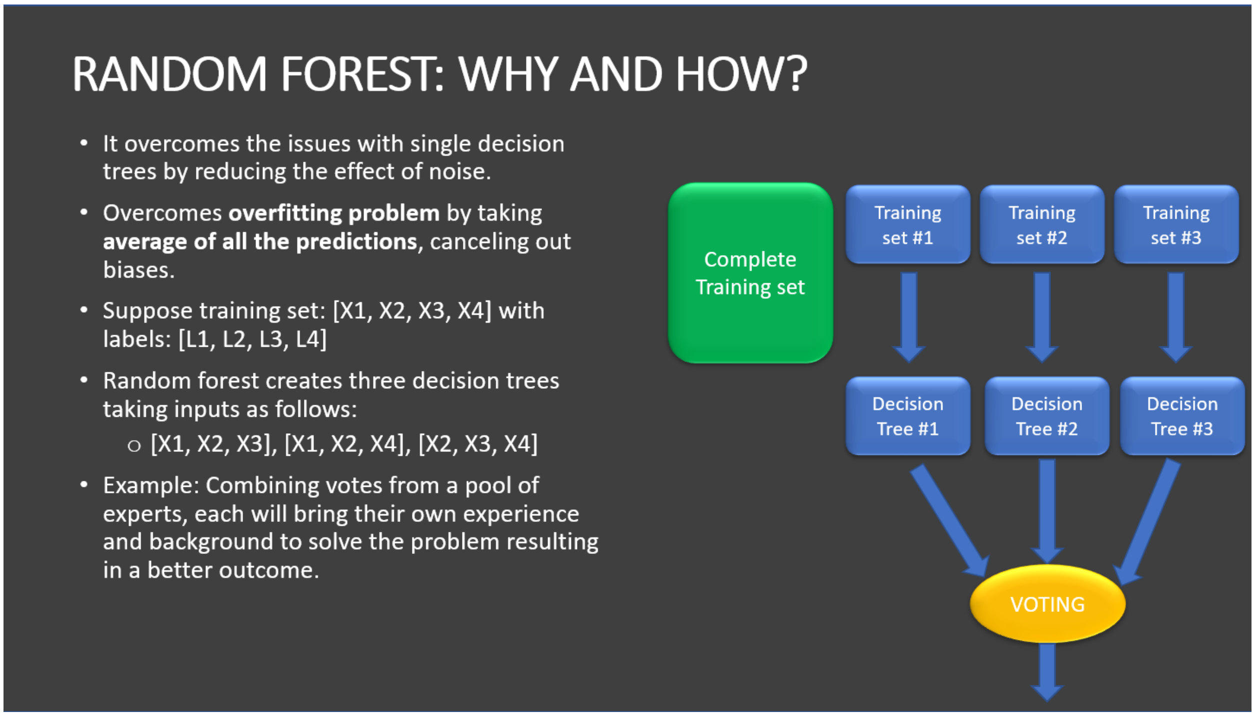
TASK #6: EVALUATE TRAINED MODEL PERFORMANCE

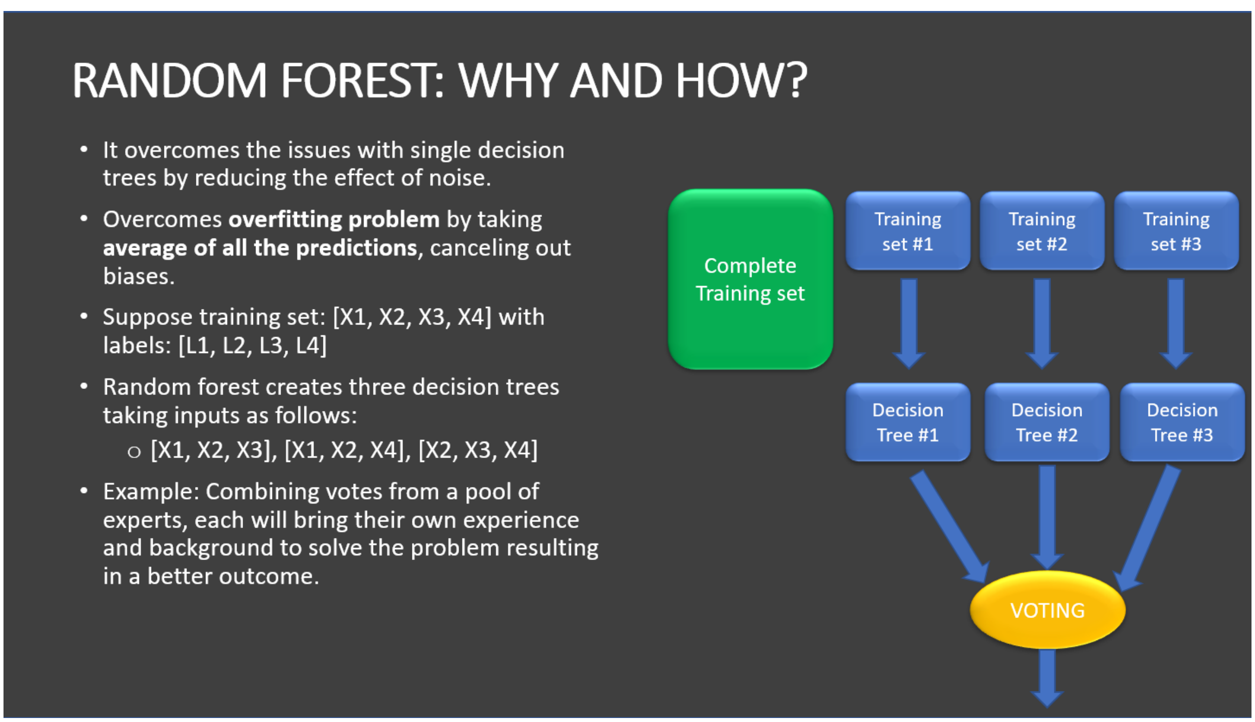
```python
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Defining class names
class_names = ['0', '1']

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred_test)

# Plotting the confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()

# Generate classification report
report = classification_report(y_test, y_pred_test,
target_names=class_names,zero_division=0)
print("Classification Report:\n", report)
```




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

           0       0.82      1.00      0.90        14
           1       0.00      0.00      0.00         3

    accuracy                           0.82        17
   macro avg       0.41      0.50      0.45        17
weighted avg       0.68      0.82      0.74        17
```

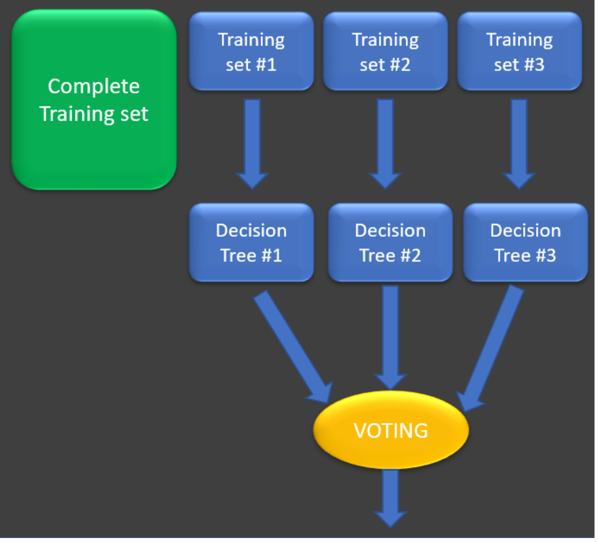
TASK #7: UNDERSTAND THE THEORY AND INTUITION BEHIND DECISION TREES AND RANDOM FOREST CLASSIFIER MODELS





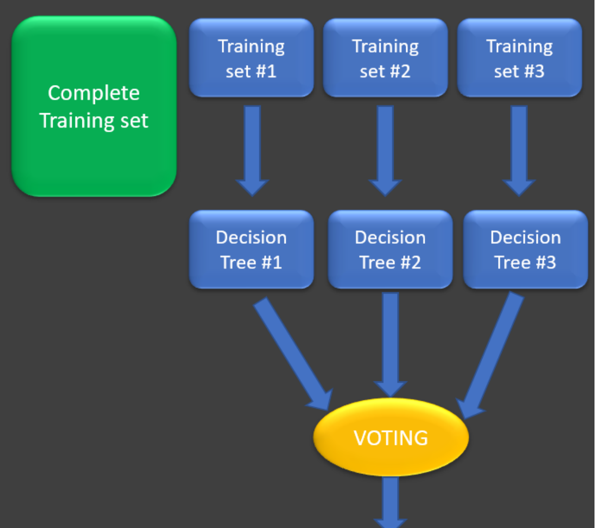
RANDOM FOREST: WHY AND HOW?

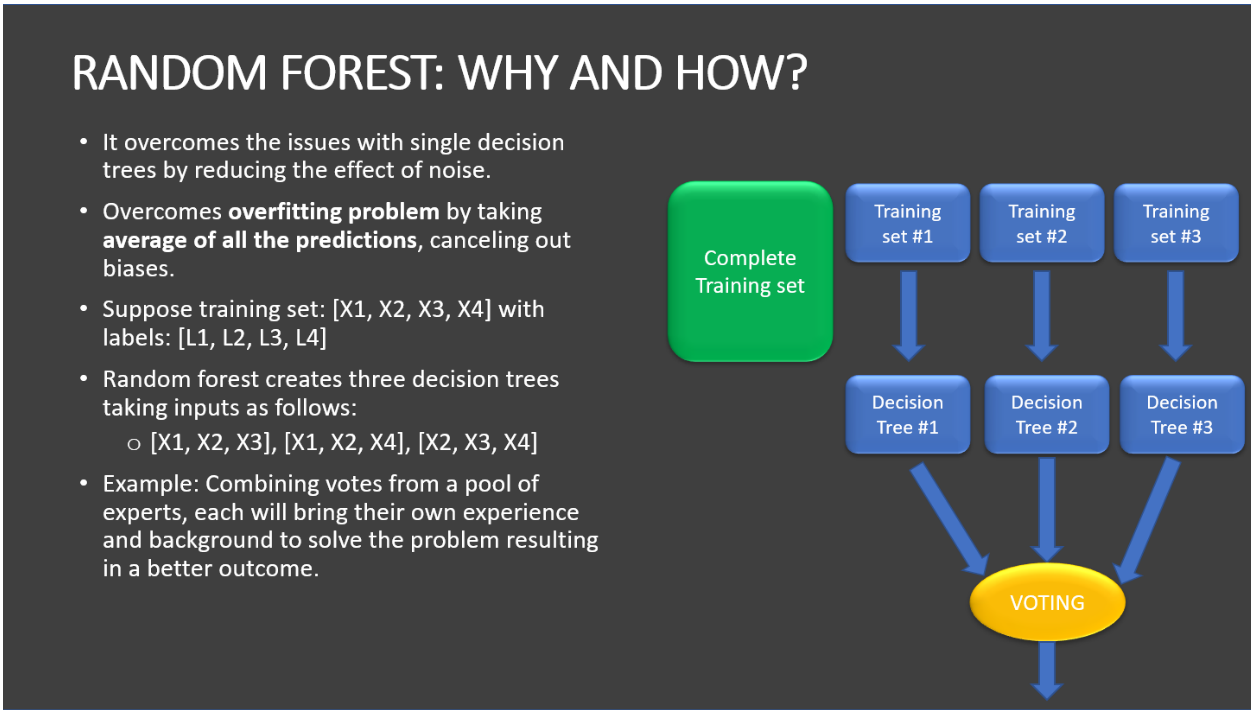
- It overcomes the issues with single decision trees by reducing the effect of noise.
- Overcomes **overfitting problem** by taking **average of all the predictions**, canceling out biases.
- Suppose training set: [X1, X2, X3, X4] with labels: [L1, L2, L3, L4]
- Random forest creates three decision trees taking inputs as follows:
  - [X1, X2, X3], [X1, X2, X4], [X2, X3, X4]
- Example: Combining votes from a pool of experts, each will bring their own experience and background to solve the problem resulting in a better outcome.



RANDOM FOREST: WHY AND HOW?

- It overcomes the issues with single decision trees by reducing the effect of noise.
- Overcomes **overfitting problem** by taking **average of all the predictions**, canceling out biases.
- Suppose training set: [X1, X2, X3, X4] with labels: [L1, L2, L3, L4]
- Random forest creates three decision trees taking inputs as follows:
  - [X1, X2, X3], [X1, X2, X4], [X2, X3, X4]
- Example: Combining votes from a pool of experts, each will bring their own experience and background to solve the problem resulting in a better outcome.

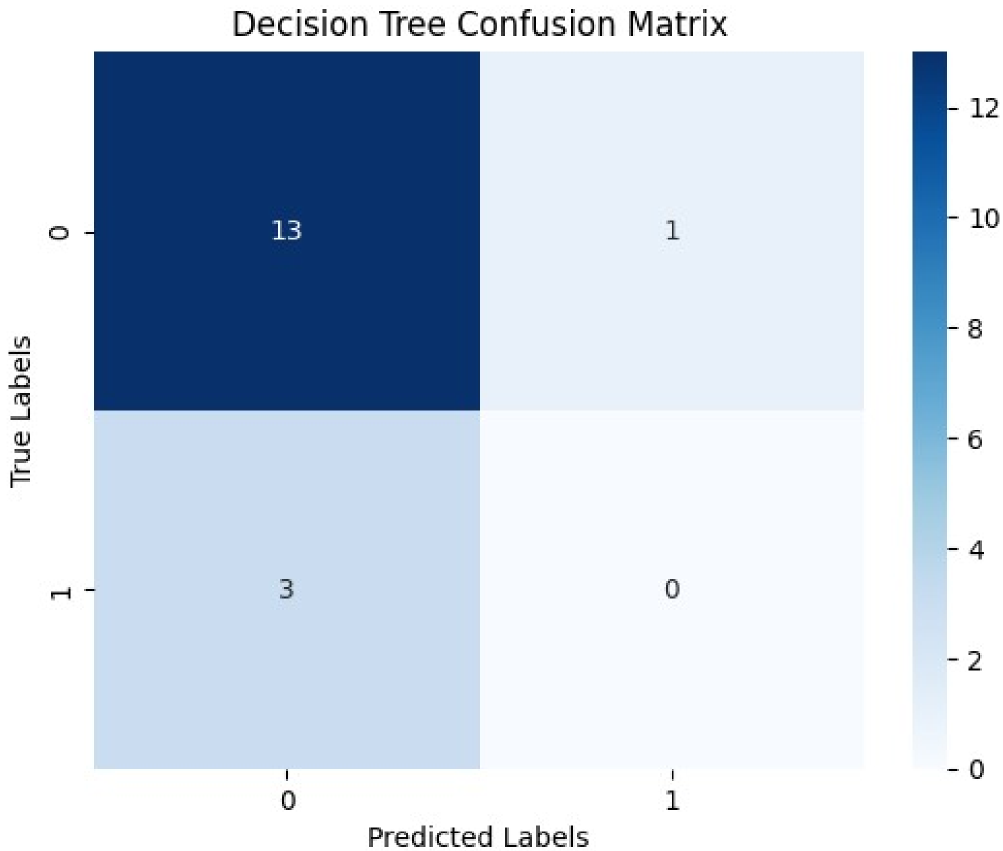




TASK #8: IMPROVE THE MODEL

```
from sklearn.tree import DecisionTreeClassifier
# Initialize the classifier
dt = DecisionTreeClassifier()
dt = DecisionTreeClassifier(
    criterion='gini',       # or 'entropy'
    max_depth=5,            # limit depth of the tree
    min_samples_split=10,   # minimum samples to split
    min_samples_leaf=4     # minimum samples per leaf
)

# Assuming you already have your decision tree classifier 'dt' 
initialized
# and your training data: X_train, y_train

# Fit the decision tree classifier
dt.fit(X_train, y_train)

# Now predict on test data
y_predict_test = dt.predict(X_test)

# Generate the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predict_test)

# Plot the confusion matrix as a heatmap
```

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```




```python
print(classification_report(y_test, y_predict_test,zero_division=0))
```

```
              precision    recall  f1-score   support

           0       0.81      0.93      0.87        14
           1       0.00      0.00      0.00         3

    accuracy                           0.76        17
   macro avg       0.41      0.46      0.43        17
weighted avg       0.67      0.76      0.71        17
```

```python
import pandas as pd
```

```
feature_names = ['Age', 'Number', 'Start']

feature_importances = pd.DataFrame(
    dt.feature_importances_,
    index=feature_names,
    columns=['importance']
).sort_values('importance', ascending=False)

print(feature_importances)
```

```
        importance
Start     0.877361
Age       0.122639
Number    0.000000
```

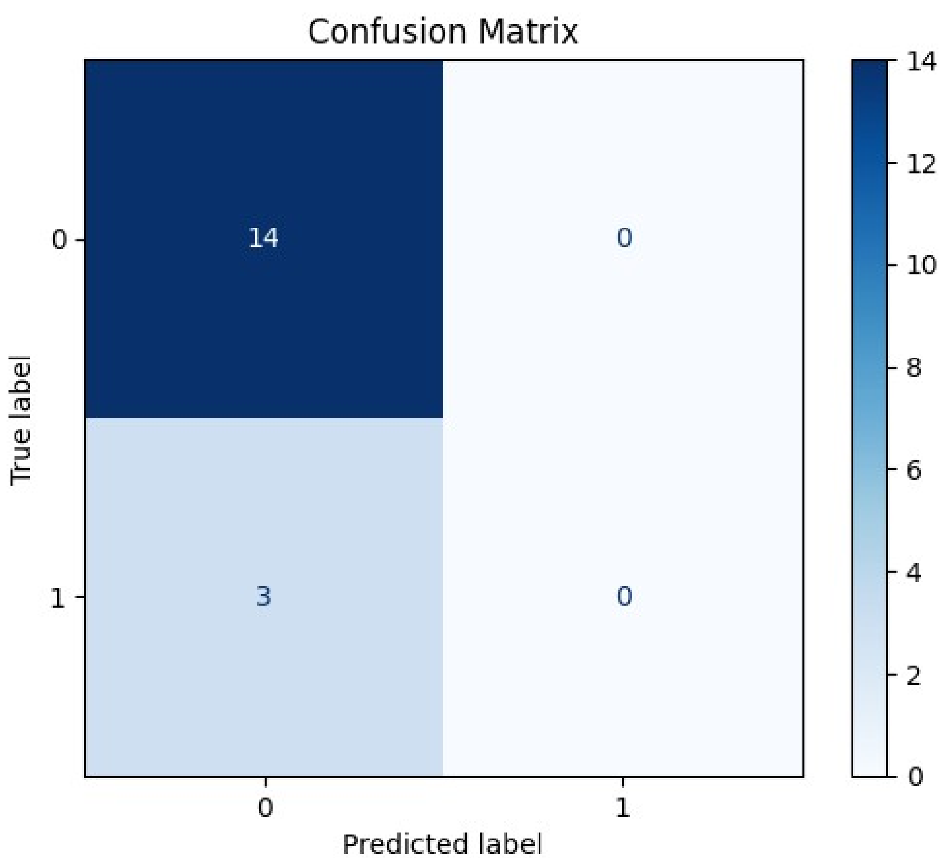
PRACTICE OPPORTUNITY #3 [OPTIONAL]:

- **Train a random forest classifier model and assess its performance**
- **Plot the confusion matrix**
- **Print the classification Report**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report, accuracy_score
import matplotlib.pyplot as plt

# Replace these with your feature matrix and target variable
# For example:
# X = df[['Age', 'Number', 'Start']]
# y = df['Kyphosis']

# Split 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)

# Initialize the Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf.fit(X_train, y_train)

# Predict on train and test sets
y_train_pred = rf.predict(X_train)
y_test_pred = rf.predict(X_test)

# Calculate accuracy
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)
```

```
# Print accuracies
print(f'Training Accuracy: {train_accuracy:.2f}')
print(f'Testing Accuracy: {test_accuracy:.2f}')

# Plot confusion matrix for test data
cm = confusion_matrix(y_test, y_test_pred, labels=rf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=rf.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()

# Print classification report
report = classification_report(y_test, y_test_pred,zero_division=0)
print('Classification Report:')
print(report)
```

```
Training Accuracy: 1.00
Testing Accuracy: 0.82
```




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

```
              0       0.82      1.00      0.90        14
              1       0.00      0.00      0.00         3

       accuracy                           0.82        17
      macro avg       0.41      0.50      0.45        17
   weighted avg       0.68      0.82      0.74        17
```

GREAT JOB!

PRACTICE OPPORTUNITIES SOLUTIONS

PRACTICE OPPORTUNITY #1 SOLUTION:

- **List the average, minimum and maximum age (in years) considered in this study using two different methods**

```
Kyphosis_df.describe()
```

	Kyphosis	Age	Number	Start
count	81.000000	81.000000	81.000000	81.000000
mean	0.209877	83.654321	4.049383	11.493827
std	0.409758	58.104251	1.619423	4.883962
min	0.000000	1.000000	2.000000	1.000000
25%	0.000000	26.000000	3.000000	9.000000
50%	0.000000	87.000000	4.000000	13.000000
75%	0.000000	130.000000	5.000000	16.000000
max	1.000000	206.000000	10.000000	18.000000

```
Kyphosis_df['Age'].mean()/12
```

```
6.97119341563786
```

```
Kyphosis_df['Age'].min()/12
```

```
0.08333333333333333
```

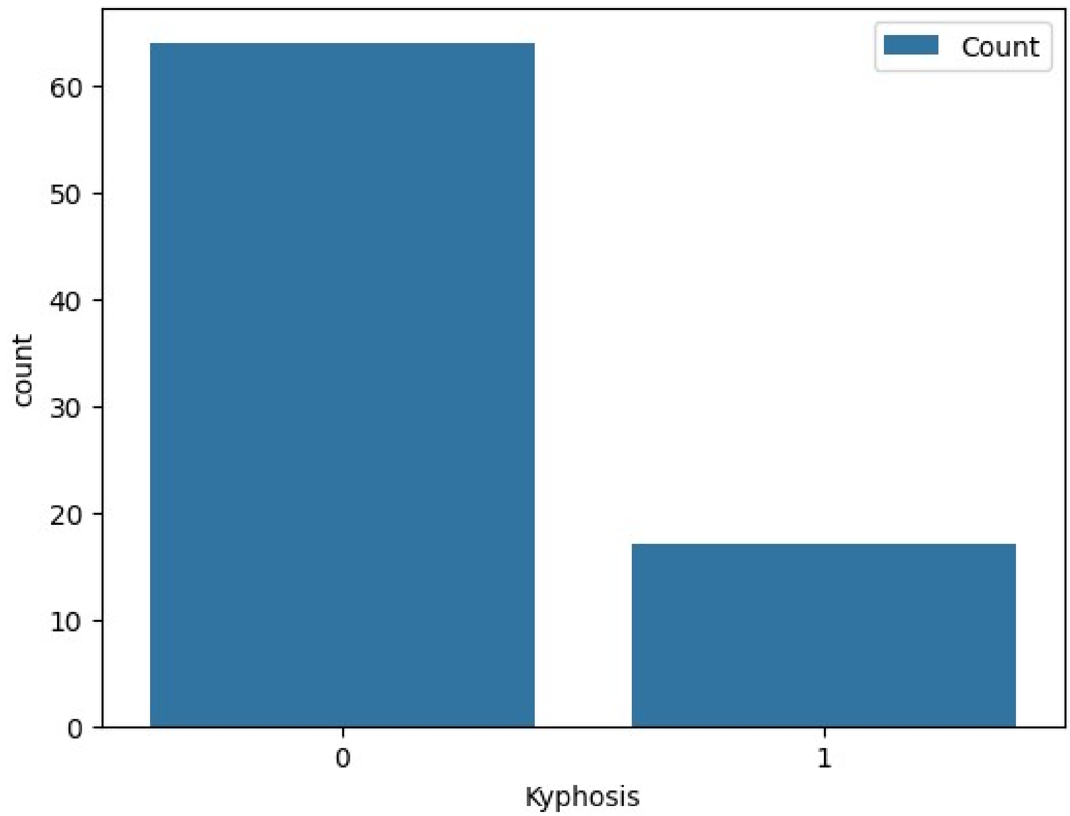
```
Kyphosis_df['Age'].max()/12
```

```
17.166666666666668
```

PRACTICE OPPORTUNITY #2 SOLUTION:

- **Plot the data countplot showing how many samples belong to each class**

```
sns.countplot(x = Kyphosis_df['Kyphosis'], label = "Count");
```



PRACTICE OPPORTUNITY #3 SOLUTION:

- **Train a random forest classifier model and assess its performance**
- **Plot the confusion matrix**
- **Print the classification Report**

```
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest.fit(X_train, y_train)

# Predicting the Test set results
y_predict_test = RandomForest.predict(X_test)
cm = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm, annot=True)

print(classification_report(y_test, y_predict_test,zero_division=0))

              precision    recall  f1-score   support

           0       0.82      1.00      0.90        14
           1       0.00      0.00      0.00         3

    accuracy                           0.82        17
   macro avg       0.41      0.50      0.45        17
```

```
weighted avg       0.68      0.82      0.74        17
```