In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Specify the full file path
file_path = r'C:\Users\chris\Downloads\PCOS_cardiovascular_data.csv'
```

In [3]:
```
# Load the CSV file
df = pd.read_csv(file_path)
```

In [4]:
```
# Display the first few rows to verify the data
print(df.head())
```

```
            Study                                            Results  \
0      De Jong [25]                    No difference in blood pressure
1         Johan [14]  Increased incidence rate of hypertension indep...
2       Khomami [18]     Higher risk of hypertension in non-obese women
3       Ka?u?na [26]  Higher blood pressure level (but within the no...
4  Mellembakken [12]  Higher blood pressure within the normal values...

   Type of Study_Longitudinal  Type of Study_Prospective  \
0                           0                          0
1                           0                          1
2                           0                          1
3                           0                          0
4                           0                          0

   Type of Study_Retrospective  Assessment Method_Aortic PWV  \
0                            0                             0
1                            0                             0
2                            0                             0
3                            0                             0
4                            0                             0

   Assessment Method_Augmentation index  \
0                                     0
1                                     0
2                                     0
3                                     0
4                                     0

   Assessment Method_Augmentation index and PWV  Assessment Method_CAVI  \
0                                             0                       0
1                                             0                       0
2                                             0                       0
3                                             0                       0
4                                             0                       0

   Assessment Method_PWV  Assessment Method_PWV brachial  \
0                      1                               0
1                      1                               0
2                      1                               0
3                      1                               0
4                      1                               0

   Assessment Method_baPWV    PCOS n  Control n  Mean Age  Mean Follow-Up  \
0                        0 -0.162836  -0.256319  1.163710        0.002785
1                        0 -0.134071   0.027619 -0.119572        1.133646
2                        0 -0.142346  -0.055374  0.085207       -5.328415
3                        0 -0.152985  -0.254202 -0.679302        0.002785
4                        0 -0.141470  -0.246795 -0.256092        0.002785

   PCOS cIMT (mm)  Control cIMT (mm)
0       -0.139904           0.087901
1       -0.139904           0.087901
2       -0.139904           0.087901
3       -0.139904           0.087901
4       -0.139904           0.087901
```

In [5]:

```
# Get a summary of the dataset
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58 entries, 0 to 57
Data columns (total 18 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   Study                                      58 non-null     object 
 1   Results                                    55 non-null     object 
 2   Type of Study_Longitudinal                 58 non-null     int64  
 3   Type of Study_Prospective                  58 non-null     int64  
 4   Type of Study_Retrospective                58 non-null     int64  
 5   Assessment Method_Aortic PWV               58 non-null     int64  
 6   Assessment Method_Augmentation index       58 non-null     int64  
 7   Assessment Method_Augmentation index and PWV  58 non-null     int64  
 8   Assessment Method_CAVI                     58 non-null     int64  
 9   Assessment Method_PWV                      58 non-null     int64  
 10  Assessment Method_PWV brachial             58 non-null     int64  
 11  Assessment Method_baPWV                    58 non-null     int64  
 12  PCOS n                                     58 non-null     float64
 13  Control n                                  58 non-null     float64
 14  Mean Age                                   58 non-null     float64
 15  Mean Follow-Up                             58 non-null     float64
 16  PCOS cIMT (mm)                             58 non-null     float64
 17  Control cIMT (mm)                          58 non-null     float64
dtypes: float64(6), int64(10), object(2)
memory usage: 8.3+ KB
None
```

In [6]:

```
# Display summary statistics for numerical columns
print(df.describe())
```

```
       Type of Study_Longitudinal  Type of Study_Prospective  \
count                   58.000000                  58.000000
mean                     0.017241                   0.120690
std                      0.131306                   0.328611
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      1.000000                   1.000000

       Type of Study_Retrospective  Assessment Method_Aortic PWV  \
count                    58.000000                     58.000000
mean                      0.051724                      0.017241
std                       0.223404                      0.131306
min                       0.000000                      0.000000
25%                       0.000000                      0.000000
50%                       0.000000                      0.000000
75%                       0.000000                      0.000000
max                       1.000000                      1.000000

       Assessment Method_Augmentation index  \
count                             58.000000
mean                               0.017241
std                                0.131306
min                                0.000000
25%                                0.000000
50%                                0.000000
75%                                0.000000
max                                1.000000

       Assessment Method_Augmentation index and PWV  Assessment Method_CAVI  \
count                                     58.000000               58.000000
mean                                       0.034483                0.017241
std                                        0.184059                0.131306
min                                        0.000000                0.000000
25%                                        0.000000                0.000000
50%                                        0.000000                0.000000
75%                                        0.000000                0.000000
max                                        1.000000                1.000000

       Assessment Method_PWV  Assessment Method_PWV brachial  \
count              58.000000                       58.000000
mean                0.810345                        0.017241
std                 0.395452                        0.131306
min                 0.000000                        0.000000
25%                 1.000000                        0.000000
50%                 1.000000                        0.000000
75%                 1.000000                        0.000000
max                 1.000000                        1.000000

       Assessment Method_baPWV        PCOS n     Control n      Mean Age  \
count                58.000000  5.800000e+01  5.800000e+01  5.800000e+01
mean                  0.017241  5.172415e-11 -1.724139e-11 -1.724141e-11
std                   0.131306  1.008734e+00  1.008734e+00  1.008734e+00
min                   0.000000 -1.632742e-01 -2.569991e-01 -2.058148e+00
25%                   0.000000 -1.621358e-01 -2.559220e-01 -4.847617e-01
50%                   0.000000 -1.607129e-01 -2.551850e-01 -1.195722e-01
75%                   0.000000 -1.560281e-01 -2.486279e-01  9.544591e-02
max                   1.000000  7.483244e+00  6.341230e+00  4.467481e+00
```

```
       Mean Follow-Up  PCOS cIMT (mm)  Control cIMT (mm)
count     5.800000e+01    5.800000e+01       5.800000e+01
mean     -2.241379e-10    6.896552e-11       2.241379e-10
std       1.008734e+00    1.008734e+00       1.008734e+00
min      -5.328415e+00   -2.367389e+00      -4.681414e+00
25%       2.785371e-03   -1.399036e-01       8.790056e-02
50%       2.785371e-03   -1.399036e-01       8.790056e-02
75%       2.785371e-03   -1.399036e-01       8.790056e-02
max       4.903182e+00    5.190150e+00       3.541542e+00
```

In [7]:

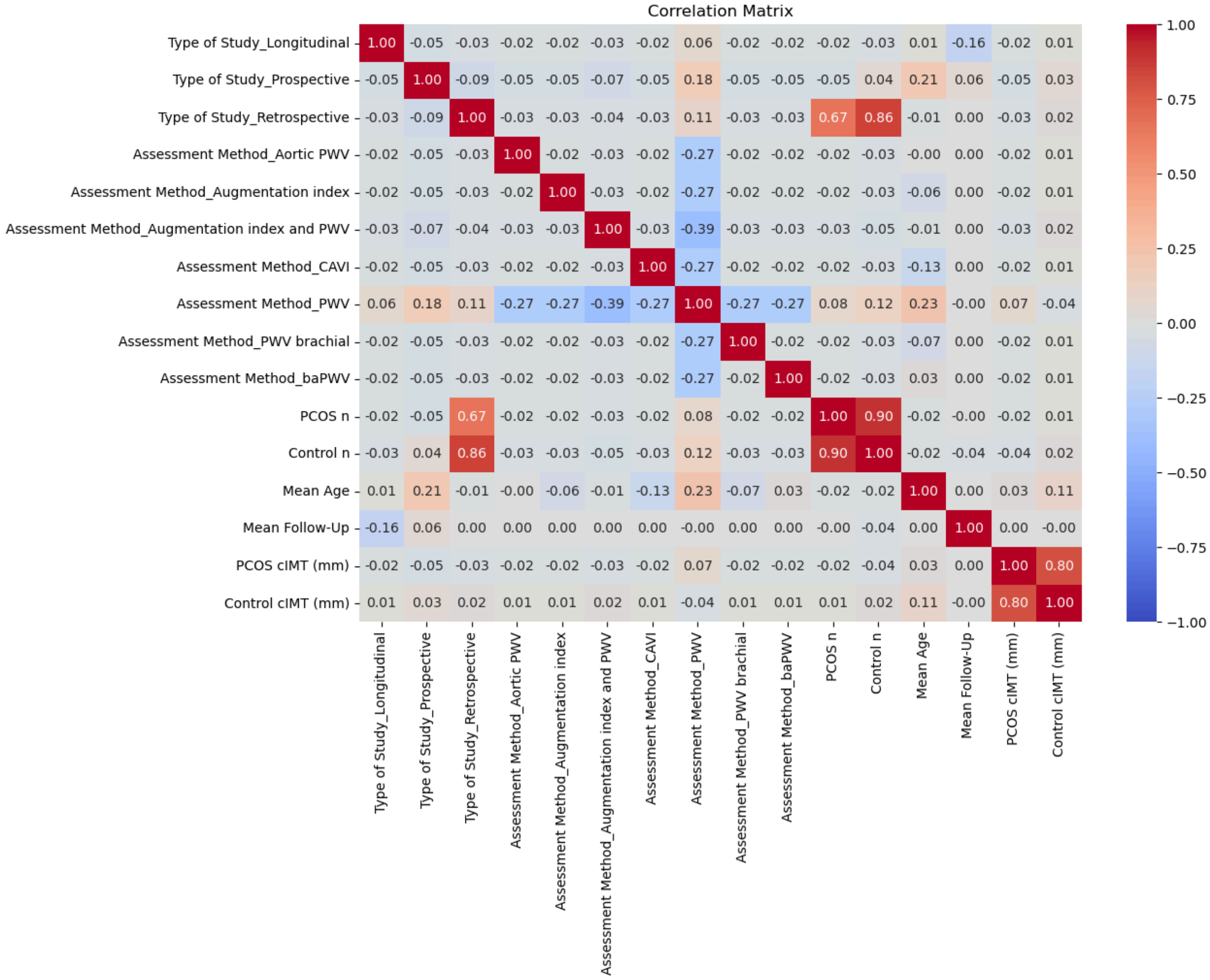
```
# Check for missing values
print(df.isnull().sum())
```

```
Study                                            0
Results                                          3
Type of Study_Longitudinal                       0
Type of Study_Prospective                        0
Type of Study_Retrospective                      0
Assessment Method_Aortic PWV                     0
Assessment Method_Augmentation index             0
Assessment Method_Augmentation index and PWV     0
Assessment Method_CAVI                           0
Assessment Method_PWV                            0
Assessment Method_PWV brachial                   0
Assessment Method_baPWV                          0
PCOS n                                           0
Control n                                        0
Mean Age                                         0
Mean Follow-Up                                   0
PCOS cIMT (mm)                                   0
Control cIMT (mm)                                0
dtype: int64
```

In [8]:

```
# Compute the correlation matrix
correlation_matrix = df.corr()

# Display the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', vmin=-1, vmax=
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\chris\AppData\Local\Temp\ipykernel_47004\2181533435.py:2: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  correlation_matrix = df.corr()
```



In [29]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc, accuracy_score, precision_score, recall_sc
from sklearn.model_selection import StratifiedKFold
```

In [81]:

```
print(data.columns)
```

```
Index(['Study', 'Results', 'Type of Study_Longitudinal',
       'Type of Study_Prospective', 'Type of Study_Retrospective',
       'Assessment Method_Aortic PWV', 'Assessment Method_Augmentation index',
       'Assessment Method_Augmentation index and PWV',
       'Assessment Method_CAVI', 'Assessment Method_PWV',
       'Assessment Method_PWV brachial', 'Assessment Method_baPWV', 'PCOS n',
       'Control n', 'Mean Age', 'Mean Follow-Up', 'PCOS cIMT (mm)',
       'Control cIMT (mm)'],
      dtype='object')
```

In [83]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

```python
# Load the dataset
data = pd.read_csv("C:\\Users\\chris\\Downloads\\PCOS_cardiovascular_data.csv")

# Handle missing values (if any)
data.fillna(data.median(), inplace=True)

# Define predictors and target
predictors = ['Mean Age', 'PCOS cIMT (mm)', 'Control cIMT (mm)', 'PCOS n', 'Control n'
target = 'Results'  # Assuming 'Results' is the target variable

# If 'Results' is categorical, we need to encode it
label_encoder = LabelEncoder()
data[target] = label_encoder.fit_transform(data[target])

# Splitting the data into training and test sets
X = data[predictors]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

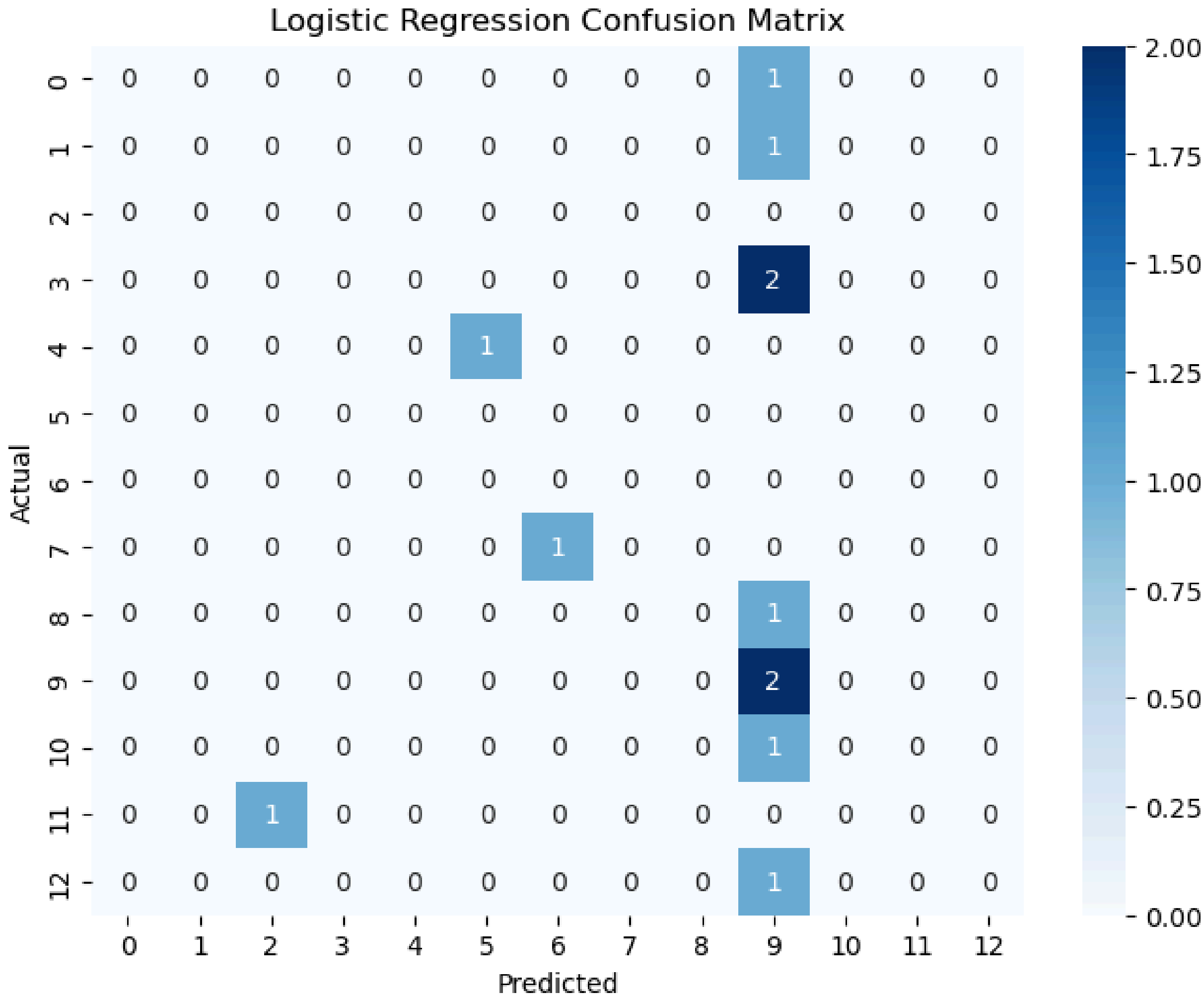
```
C:\Users\chris\AppData\Local\Temp\ipykernel_47004\1556139857.py:11: FutureWarning: Th
e default value of numeric_only in DataFrame.median is deprecated. In a future versio
n, it will default to False. In addition, specifying 'numeric_only=None' is deprecate
d. Select only valid columns or specify the value of numeric_only to silence this war
ning.
  data.fillna(data.median(), inplace=True)
```

In [117…

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import LabelEncoder

# Load the dataset
data = pd.read_csv("C:\\Users\\chris\\Downloads\\PCOS_cardiovascular_data.csv")

# Handle missing values (if any)
data.fillna(data.median(), inplace=True)

# Define predictors and target
predictors = ['Mean Age', 'PCOS cIMT (mm)', 'Control cIMT (mm)', 'PCOS n', 'Control n'
target = 'Results'  # Assuming 'Results' is the target variable

# If 'Results' is categorical, we need to encode it
label_encoder = LabelEncoder()
data[target] = label_encoder.fit_transform(data[target])

# Splitting the data into training and test sets
X = data[predictors]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Logistic Regression Model
model = LogisticRegression(max_iter=1000)

# Fitting the model
model.fit(X_train, y_train)
```

```python
# Predicting on test data
y_pred = model.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n{cm}')

# Plot the confusion matrix using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=[str(i) for i in range(
            yticklabels=[str(i) for i in range(len(cm))])
plt.title('Logistic Regression Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
C:\Users\chris\AppData\Local\Temp\ipykernel_47004\1061873663.py:14: FutureWarning: Th
e default value of numeric_only in DataFrame.median is deprecated. In a future versio
n, it will default to False. In addition, specifying 'numeric_only=None' is deprecate
d. Select only valid columns or specify the value of numeric_only to silence this war
ning.
  data.fillna(data.median(), inplace=True)
Accuracy: 16.67%
Confusion Matrix:
[[0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 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 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0]]
```



In [121…

```
print(X_train.shape)
print(y_bin.shape)
```

```
(46, 5)
(58, 3)
```

In [89]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=0.2, random_st
```

In [92]:

```
print(X_train.shape)  # Shape of features
print(y_bin.shape)    # Shape of target labels
```

```
(46, 5)
(58, 3)
```

In [93]:

```
from sklearn.model_selection import train_test_split

# Ensure features and labels are split together
X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=0.2, random_st

print(X_train.shape)
print(y_train.shape)
```

```
(46, 5)
(46, 3)
```

In [94]:
```python
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression

# Create the model
model = OneVsRestClassifier(LogisticRegression(max_iter=1000))

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)
```

In [95]:
```python
from sklearn.metrics import accuracy_score, classification_report

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Detailed classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9166666666666666
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0

   micro avg       0.00      0.00      0.00         1
   macro avg       0.00      0.00      0.00         1
weighted avg       0.00      0.00      0.00         1
 samples avg       0.00      0.00      0.00         1
```

```
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in lab
els with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 due
to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
samples with no predicted labels. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in sam
ples with no true labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

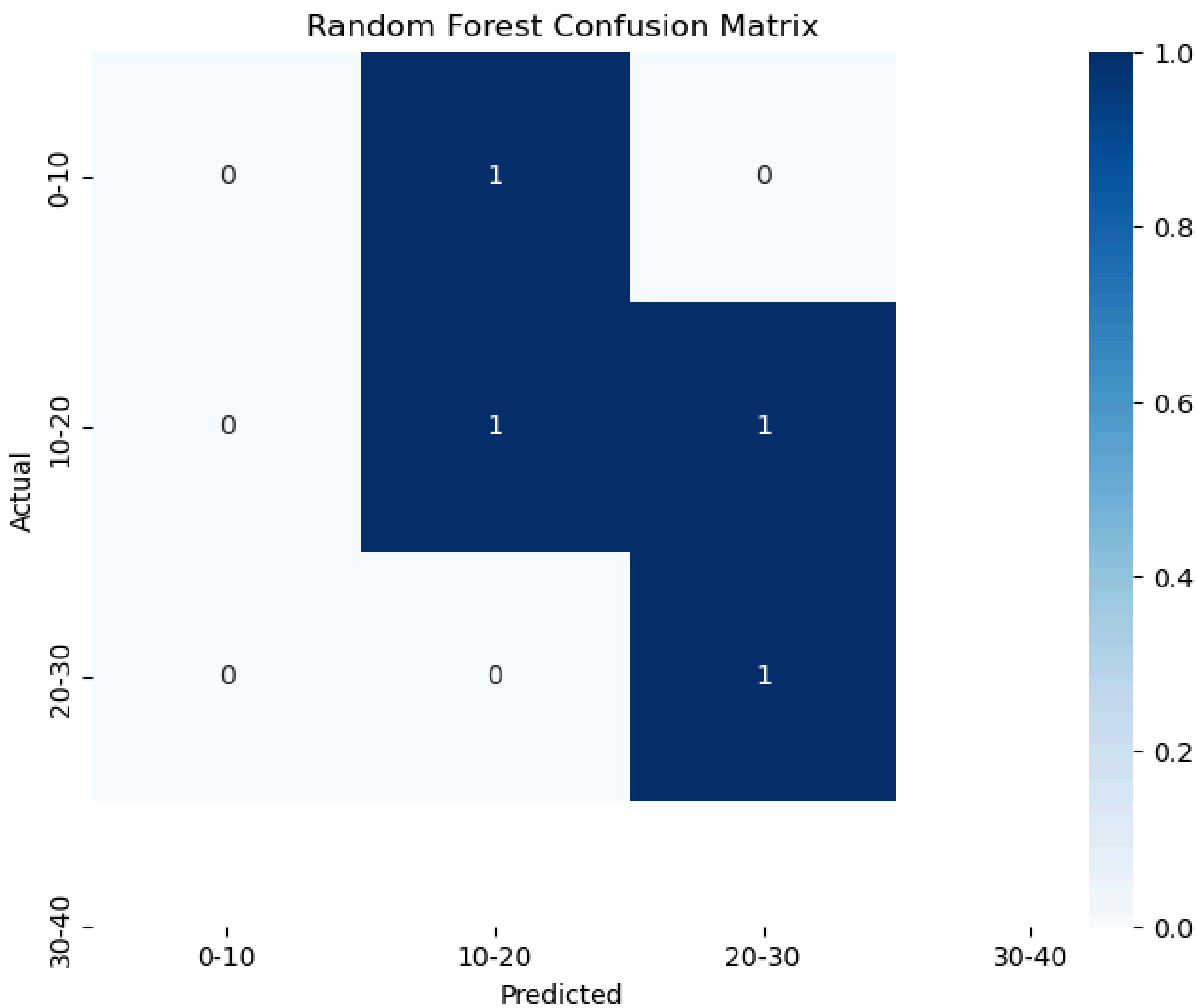
In [102…
```python
from sklearn.metrics import confusion_matrix
```

In [103…

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import pandas as pd

# Assuming rf_predictions_binned and y_test_binned are already defined as binned categ
# Convert both to strings or NumPy arrays (if they are categorical, otherwise you can
```

In [104…

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Example continuous data
# Replace with your own dataset
y = [12, 25, 5, 18, 34, 27, 12, 20, 10, 15, 30, 22]  # This is just an example
X = [[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [13, 14], [15, 16], [17, 18],

# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state

# Train RandomForest model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predict using the trained model
rf_predictions = model.predict(X_test)

# Bin the continuous labels for both actual (y_test) and predicted (rf_predictions)
bins = [0, 10, 20, 30, 40]  # Define your bin edges
y_test_binned = pd.cut(y_test, bins=bins, labels=['0-10', '10-20', '20-30', '30-40'])
rf_predictions_binned = pd.cut(rf_predictions, bins=bins, labels=['0-10', '10-20', '20

# Generate confusion matrix
rf_cm = confusion_matrix(y_test_binned, rf_predictions_binned)

# Plot the heatmap using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['0-10', '10-20', '20-30', '30-40'],
            yticklabels=['0-10', '10-20', '20-30', '30-40'])

# Add labels and title
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [133…
```
print(X_train.isnull().sum())  # Check for missing values in your features
print(y_train.isnull().sum())  # Check for missing values in the target
```

```
Mean Age             0
PCOS cIMT (mm)       0
Control cIMT (mm)    0
PCOS n               0
Control n            0
dtype: int64
0
```

In [134…
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [135…
```
rf_model = RandomForestClassifier(n_estimators=100, class_weight='balanced')
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

In [136…
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
```

```
    'class_weight': ['balanced', None]
}
grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best parameters found:", grid_search.best_params_)
rf_model = grid_search.best_estimator_
rf_predictions = rf_model.predict(X_test)
```

```
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\model_selection\_split.py:725: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=5.
  * Generate test sets such that all contain the same distribution of
Best parameters found: {'class_weight': 'balanced', 'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
```

In [137…

```
from sklearn.metrics import accuracy_score

# After making predictions
rf_predictions = rf_model.predict(X_test)

# Calculate and print the accuracy score
accuracy = accuracy_score(y_test, rf_predictions)
print("Accuracy Score:", accuracy)
```

```
Accuracy Score: 0.25
```

In [143…

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)

# Predict on the test set
rf_predictions = rf_model.predict(X_test)

# Print the accuracy score
accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {accuracy}")

# Print the classification report for Random Forest
print("Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions))
```

```
Random Forest Accuracy: 0.25
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.20      1.00      0.33         1
           3       0.00      0.00      0.00         1
           8       0.00      0.00      0.00         0
          11       1.00      0.50      0.67         2
          13       0.00      0.00      0.00         1
          21       0.00      0.00      0.00         0
          23       0.00      0.00      0.00         1
          24       0.00      0.00      0.00         1
          26       0.00      0.00      0.00         2
          27       0.00      0.00      0.00         1
          28       0.00      0.00      0.00         1
          31       1.00      1.00      1.00         1

    accuracy                           0.25        12
   macro avg       0.18      0.21      0.17        12
weighted avg       0.27      0.25      0.22        12
```

```
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in lab
els with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in lab
els with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in lab
els with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [97]:

```python
# Get feature importance
importances = model.feature_importances_

# Sort features by importance
indices = importances.argsort()

# Display feature importance
for i in indices:
    print(f"Feature {i}: {importances[i]}")
```

```
Feature 2: 0.030317074700738647
Feature 1: 0.030667008719042467
Feature 3: 0.23035936018188696
Feature 0: 0.34610538366337834
Feature 4: 0.36255117273495346
```

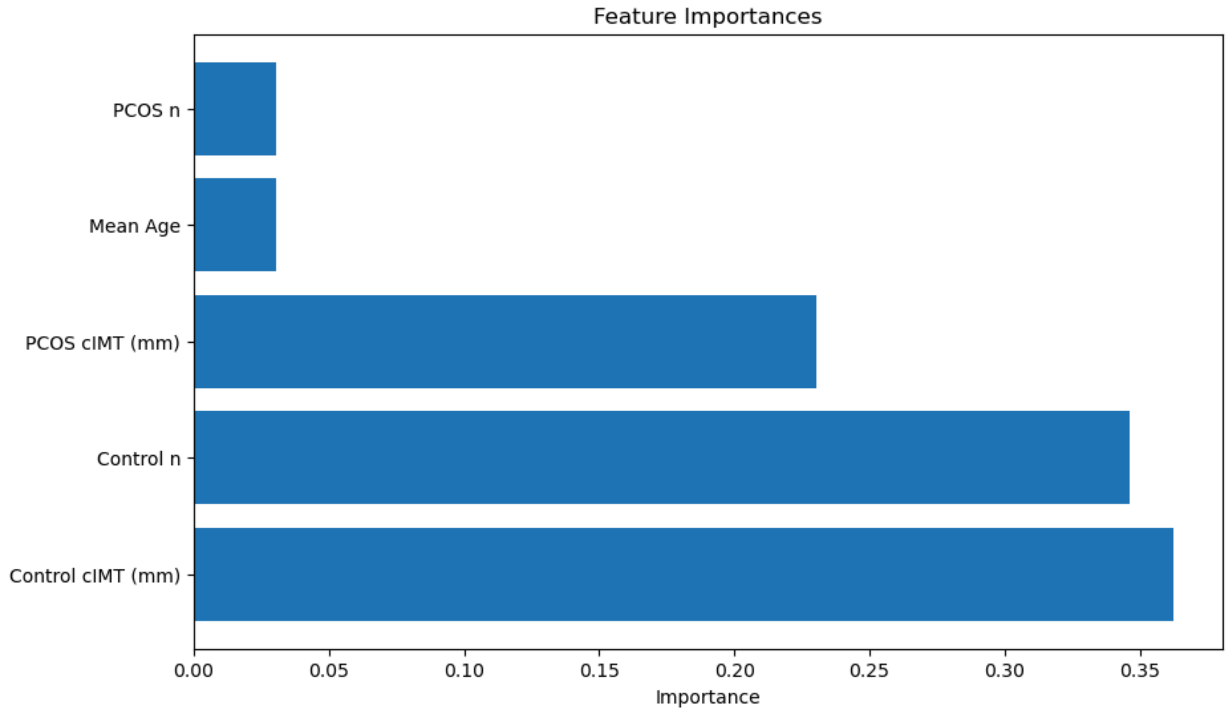
In [100…

```python
# Assuming X is your features dataframe and model is your trained model
# Display the feature importance scores
import pandas as pd

# Feature names from your dataset (replace 'X.columns' with your actual dataset if nec
feature_names = X.columns

# Feature importances from the model
importances = model.feature_importances_

# Create a DataFrame to combine feature names with importance scores
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
})

# Sort the features by importance
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=F

# Display the feature importance ranking
print(feature_importance_df)
```

```
             Feature  Importance
4          Control n    0.362551
0           Mean Age    0.346105
3             PCOS n    0.230359
1     PCOS cIMT (mm)    0.030667
2  Control cIMT (mm)    0.030317
```

In [101…

```python
import matplotlib.pyplot as plt
import numpy as np

# Assuming these are your feature names
feature_names = ['Control n', 'Mean Age', 'PCOS n', 'PCOS cIMT (mm)', 'Control cIMT (m

# Get the feature importances from the model
importances = model.feature_importances_

# Sort the feature importances in descending order
indices = np.argsort(importances)[::-1]

# Plot the feature importances
plt.figure(figsize=(10, 6))
plt.barh(range(len(feature_names)), importances[indices], align='center')
plt.yticks(range(len(feature_names)), np.array(feature_names)[indices])
plt.xlabel('Importance')
plt.title('Feature Importances')
plt.show()
```


Feature Importances
PCOS n
Mean Age
PCOS cIMT (mm)
Control n
Control cIMT (mm)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Importance

In [ ]: