In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
df = pd.read_excel("C:\\Users\\KAUSHIK\\OneDrive\\Cardio.xlsx")
```

In [3]:
```python
df = df[df.columns[0]].str.split(';', expand=True)
df.columns = ['id', 'age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol', 'gluc', 'smoke', 'alco', 'a
df.head()
```

Out[3]:
```
   id    age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  cardio
0   0  18393       2     168    62.0    110     80            1     1      0     0       1       0
1   1  20228       1     156    85.0    140     90            3     1      0     0       1       1
2   2  18857       1     165    64.0    130     70            3     1      0     0       0       1
3   3  17623       2     169    82.0    150    100            1     1      0     0       1       1
4   4  17474       1     156    56.0    100     60            1     1      0     0       0       0
```

In [4]:
```python
df
```

Out[4]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69995	99993	19240	2	168	76.0	120	80	1	1	1	0	1	0
69996	99995	22601	1	158	126.0	140	90	2	2	0	0	1	1
69997	99996	19066	2	183	105.0	180	90	3	1	0	1	0	1
69998	99998	22431	1	163	72.0	135	80	1	2	0	0	0	1
69999	99999	20540	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 13 columns

In [5]:

```
df.isnull().sum()
```

Out[5]:

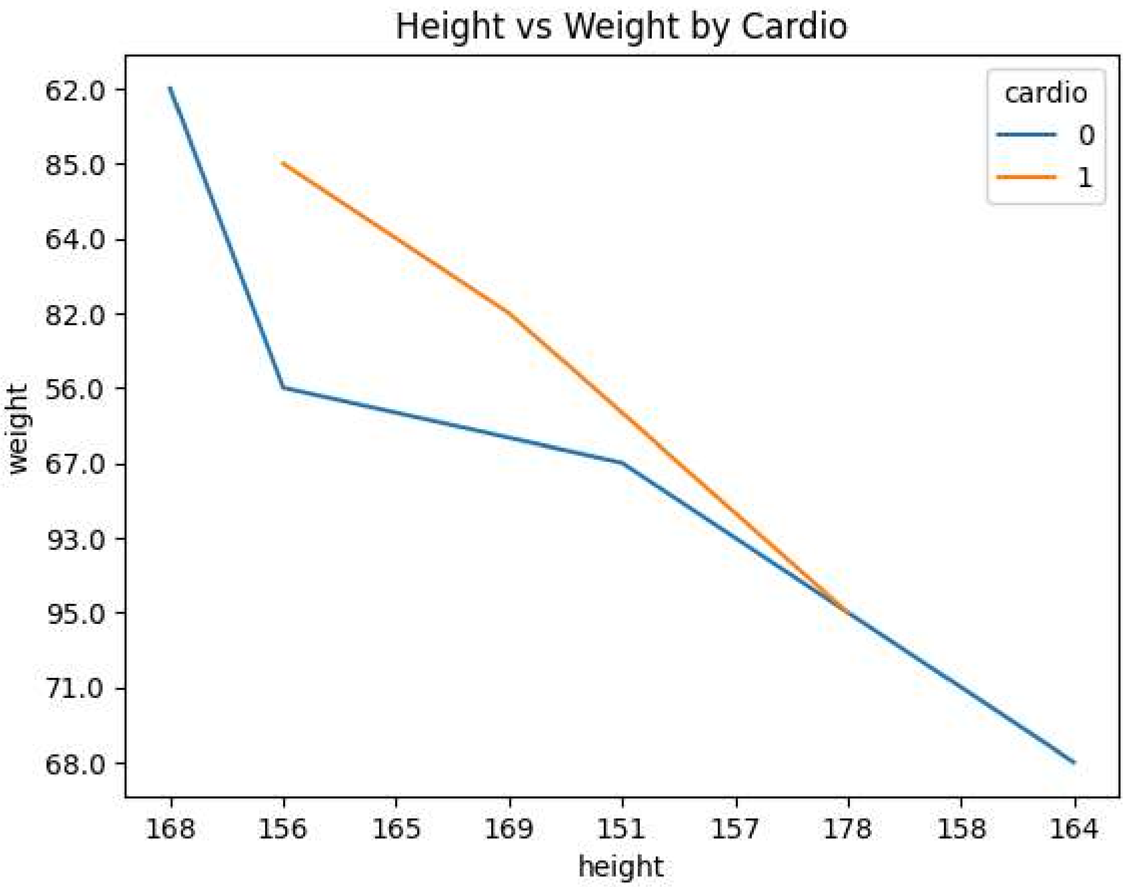
```
id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64
```

In [6]:

```
sns.lineplot(x='height', y='weight', data=df.head(10), hue='cardio')
plt.title('Height vs Weight by Cardio')
```

Out[6]:

```
Text(0.5, 1.0, 'Height vs Weight by Cardio')
```

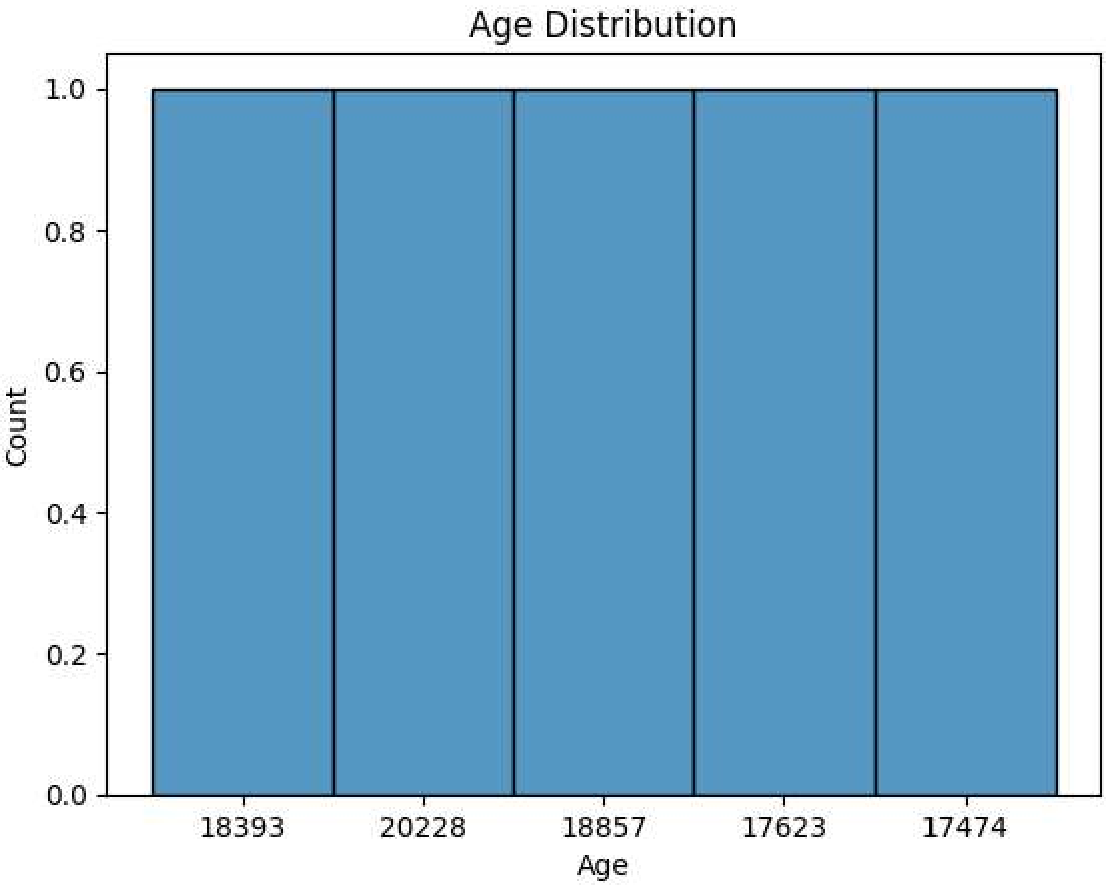


In [7]:

```
sns.histplot(df['age'].head(5))
plt.title('Age Distribution')
plt.xlabel('Age')
```

Out[7]:

```
Text(0.5, 0, 'Age')
```


Age Distribution
1.0
0.8
0.6
0.4
0.2
0.0
Count
18393
20228
18857
17623
17474
Age

In [8]:

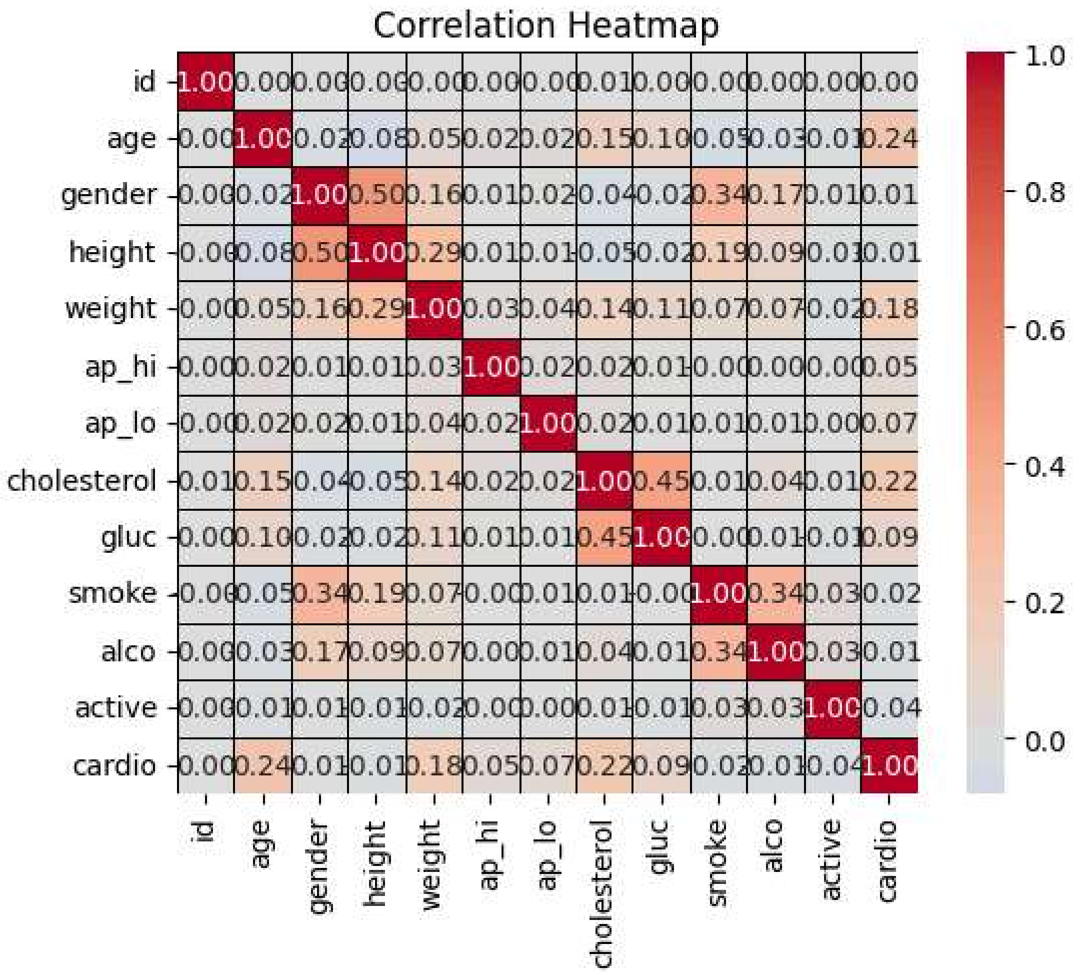
```
df.corr()
```

Out[8]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	al
id	1.000000	0.003457	0.003502	-0.003038	-0.001830	0.003356	-0.002529	0.006106	0.002467	-0.003699	0.0012
age	0.003457	1.000000	-0.022811	-0.081515	0.053684	0.020764	0.017647	0.154424	0.098703	-0.047633	-0.0297
gender	0.003502	-0.022811	1.000000	0.499033	0.155406	0.006005	0.015254	-0.035821	-0.020491	0.338135	0.1709
height	-0.003038	-0.081515	0.499033	1.000000	0.290968	0.005488	0.006150	-0.050226	-0.018595	0.187989	0.0944
weight	-0.001830	0.053684	0.155406	0.290968	1.000000	0.030702	0.043710	0.141768	0.106857	0.067780	0.0671
ap_hi	0.003356	0.020764	0.006005	0.005488	0.030702	1.000000	0.016086	0.023778	0.011841	-0.000922	0.0014
ap_lo	-0.002529	0.017647	0.015254	0.006150	0.043710	0.016086	1.000000	0.024019	0.010806	0.005186	0.0106
cholesterol	0.006106	0.154424	-0.035821	-0.050226	0.141768	0.023778	0.024019	1.000000	0.451578	0.010354	0.0357
gluc	0.002467	0.098703	-0.020491	-0.018595	0.106857	0.011841	0.010806	0.451578	1.000000	-0.004756	0.0112
smoke	-0.003699	-0.047633	0.338135	0.187989	0.067780	-0.000922	0.005186	0.010354	-0.004756	1.000000	0.3400
alco	0.001210	-0.029723	0.170966	0.094419	0.067113	0.001408	0.010601	0.035760	0.011246	0.340094	1.0000
active	0.003755	-0.009927	0.005866	-0.006570	-0.016867	-0.000033	0.004780	0.009911	-0.006770	0.025858	0.0254
cardio	0.003799	0.238159	0.008109	-0.010821	0.181660	0.054475	0.065719	0.221147	0.089307	-0.015486	-0.0073

In [9]:

```
sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm',center=0, square=True, linewidths=0.5, linecolor='blacl
plt.title('Correlation Heatmap')
plt.figure(figsize=(18, 15))
plt.show()
```

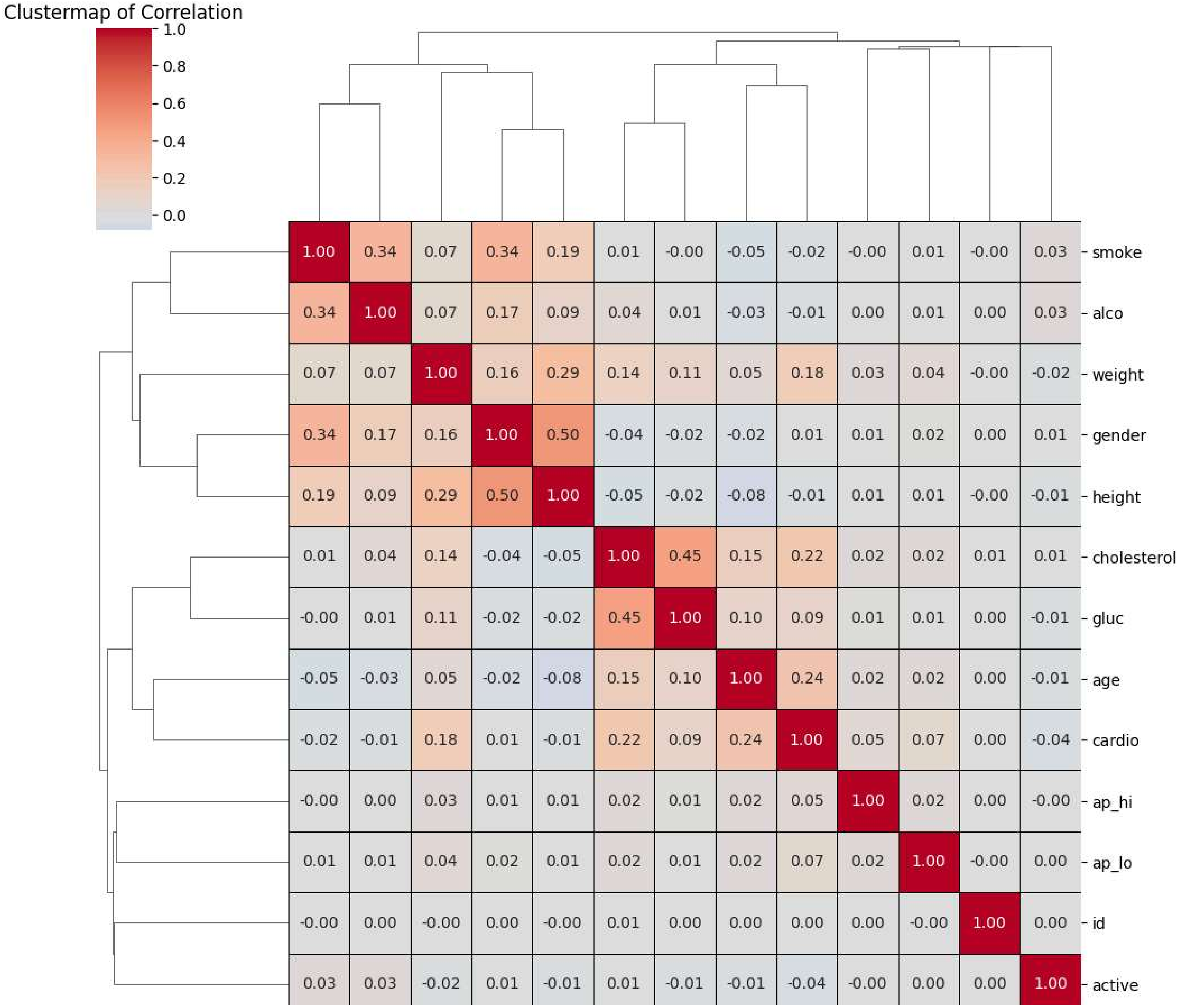


```
<Figure size 1800x1500 with 0 Axes>
```

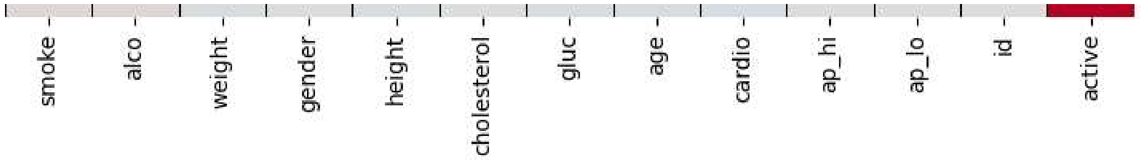
In [10]:
```
sns.clustermap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm', center=0, square=True, linewidths=0.5, linecolor='b
plt.title('Clustermap of Correlation')
```

```
c:\Users\KAUSHIK\AppData\Local\Programs\Python\Python312\Lib\site-packages\seaborn\matrix.py:1124: UserWarning: ``squ
are=True`` ignored in clustermap
  warnings.warn(msg)
```

Out[10]:
```
Text(0.5, 1.0, 'Clustermap of Correlation')
```



1.00	0.34	0.07	0.34	0.19	0.01	-0.00	-0.05	-0.02	-0.00	0.01	-0.00	0.03	smoke
0.34	1.00	0.07	0.17	0.09	0.04	0.01	-0.03	-0.01	0.00	0.01	0.00	0.03	alco
0.07	0.07	1.00	0.16	0.29	0.14	0.11	0.05	0.18	0.03	0.04	-0.00	-0.02	weight
0.34	0.17	0.16	1.00	0.50	-0.04	-0.02	-0.02	0.01	0.01	0.02	0.00	0.01	gender
0.19	0.09	0.29	0.50	1.00	-0.05	-0.02	-0.08	-0.01	0.01	0.01	-0.00	-0.01	height
0.01	0.04	0.14	-0.04	-0.05	1.00	0.45	0.15	0.22	0.02	0.02	0.01	0.01	cholesterol
-0.00	0.01	0.11	-0.02	-0.02	0.45	1.00	0.10	0.09	0.01	0.01	0.00	-0.01	gluc
-0.05	-0.03	0.05	-0.02	-0.08	0.15	0.10	1.00	0.24	0.02	0.02	0.00	-0.01	age
-0.02	-0.01	0.18	0.01	-0.01	0.22	0.09	0.24	1.00	0.05	0.07	0.00	-0.04	cardio
-0.00	0.00	0.03	0.01	0.01	0.02	0.01	0.02	0.05	1.00	0.02	0.00	-0.00	ap_hi
0.01	0.01	0.04	0.02	0.01	0.02	0.01	0.02	0.07	0.02	1.00	-0.00	0.00	ap_lo
-0.00	0.00	-0.00	0.00	-0.00	0.01	0.00	0.00	0.00	0.00	-0.00	1.00	0.00	id
0.03	0.03	-0.02	0.01	-0.01	0.01	-0.01	-0.01	-0.04	-0.00	0.00	0.00	1.00	active



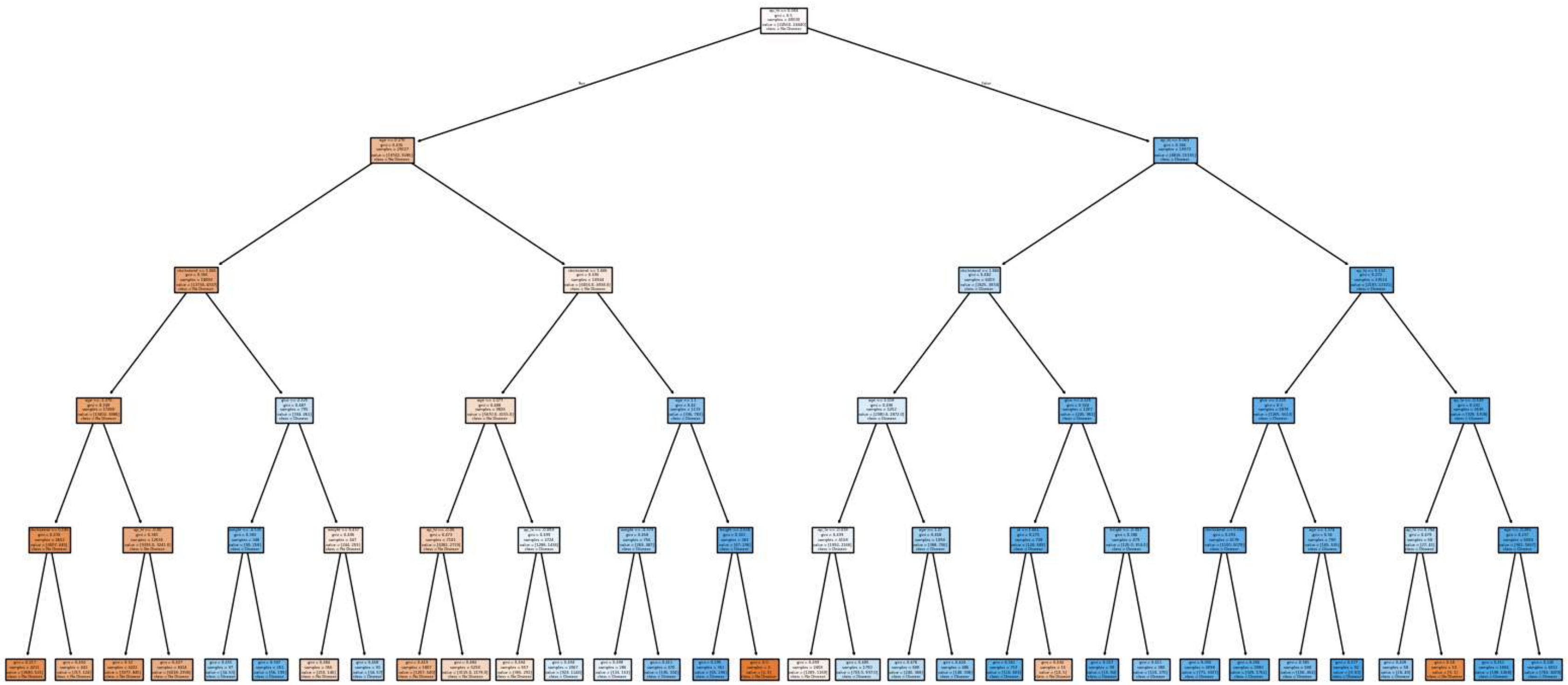
In [11]:

```python
# Import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler

 # Replace with your actual CSV file

# Preview data

# Optional: clean/rename columns if needed
# df.columns = [col.strip().lower().replace(" ", "_") for col in df.columns]

# Features and target
X = df.drop("cardio", axis=1)  # 'cardio' is usually the target column (0 = no disease, 1 = disease)
y = df["cardio"]

# Optional: normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Initialize and train the model
clf = DecisionTreeClassifier(max_depth=5, random_state=42)
clf.fit(X_train, y_train)

# Predict and evaluate
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7341428571428571

Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.80      0.75     10461
           1       0.77      0.67      0.72     10539

    accuracy                           0.73     21000
   macro avg       0.74      0.73      0.73     21000
weighted avg       0.74      0.73      0.73     21000
```

In [12]:

```python
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20, 10))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=["No Disease", "Disease"])
plt.show()
```



In [13]:

```python
from sklearn.model_selection import cross_val_score
# Perform cross-validation
cv_scores = cross_val_score(clf, X_scaled, y, cv=5)
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())
# Feature importance
importances = clf.feature_importances_
feature_names = X.columns
```

```
Cross-validation scores: [0.54164286 0.73485714 0.72935714 0.72857143 0.73028571]
Mean cross-validation score: 0.6929428571428572
```

In [14]:

```python
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the Random Forest model
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
```

```
rf_clf.fit(X_train, y_train)

# Predict and evaluate
rf_y_pred = rf_clf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_y_pred))
print("\nRandom Forest Classification Report:\n", classification_report(y_test, rf_y_pred))
```

```
Random Forest Accuracy: 0.7341904761904762

Random Forest Classification Report:
               precision    recall  f1-score   support

           0       0.71      0.79      0.75     10461
           1       0.77      0.68      0.72     10539

    accuracy                           0.73     21000
   macro avg       0.74      0.73      0.73     21000
weighted avg       0.74      0.73      0.73     21000
```

In [15]:

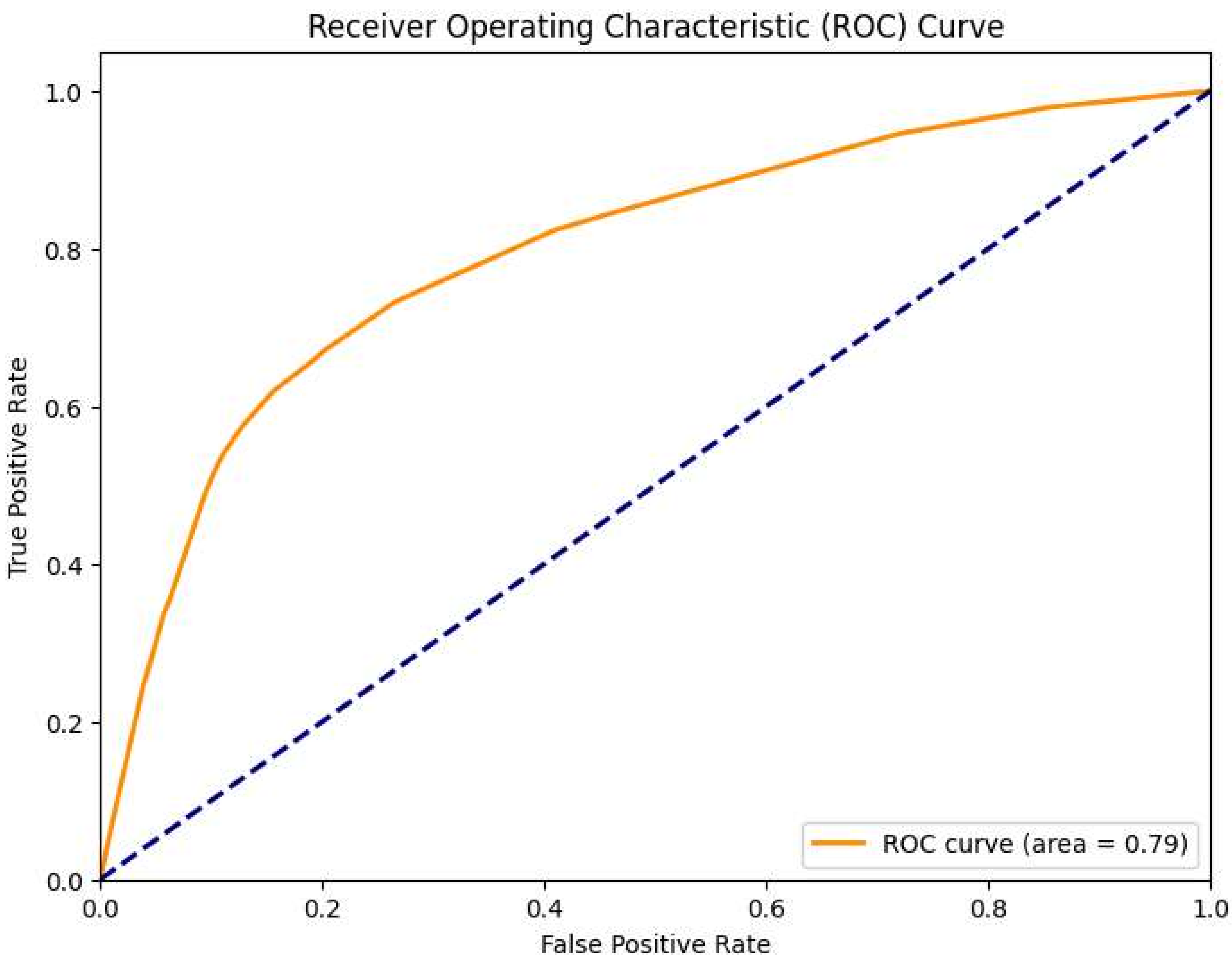
```
from sklearn.model_selection import KFold, cross_val_score

# Set up KFold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Evaluate Decision Tree with KFold
dt_kfold_scores = cross_val_score(clf, X_scaled, y, cv=kf)
print("Decision Tree KFold scores:", dt_kfold_scores)
print("Decision Tree KFold mean score:", dt_kfold_scores.mean())

# Evaluate Random Forest with KFold
rf_kfold_scores = cross_val_score(rf_clf, X_scaled, y, cv=kf)
print("Random Forest KFold scores:", rf_kfold_scores)
print("Random Forest KFold mean score:", rf_kfold_scores.mean())
```

```
Decision Tree KFold scores: [0.73442857 0.7315     0.73042857 0.73407143 0.72621429]
Decision Tree KFold mean score: 0.7313285714285714
Random Forest KFold scores: [0.73278571 0.73014286 0.72971429 0.72514286 0.72514286]
Random Forest KFold mean score: 0.7285857142857143
```

In [16]:

```
from sklearn.metrics import roc_curve, auc

# Get predicted probabilities for the positive class
```

```python
y_proba = clf.predict_proba(X_test)[:, 1]

# Compute ROC curve and ROC area
fpr, tpr, thresholds = roc_curve(y_test.astype(int), y_proba.astype(float))
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



In [17]:

```
from sklearn.neighbors import KNeighborsClassifier
# Initialize and train the KNN model
knn_clf = KNeighborsClassifier(n_neighbors=5)
knn_clf.fit(X_train, y_train)
# Predict and evaluate
knn_y_pred = knn_clf.predict(X_test)
print("KNN Accuracy:", accuracy_score(y_test, knn_y_pred))
print("\nKNN Classification Report:\n", classification_report(y_test, knn_y_pred))
```

```
KNN Accuracy: 0.6248095238095238

KNN Classification Report:
               precision    recall  f1-score   support

           0       0.62      0.64      0.63     10461
           1       0.63      0.61      0.62     10539

    accuracy                           0.62     21000
   macro avg       0.62      0.62      0.62     21000
weighted avg       0.62      0.62      0.62     21000
```

In [18]:

```
# Import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler

 # Replace with your actual CSV file

# Preview data

# Optional: clean/rename columns if needed
# df.columns = [col.strip().lower().replace(" ", "_") for col in df.columns]

# Features and target
X = df.drop("cardio", axis=1)  # 'cardio' is usually the target column (0 = no disease, 1 = disease)
y = df["cardio"]

# Optional: normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Initialize and train the model
```

```python
svm_clf = SVC(kernel='linear', random_state=42)
svm_clf.fit(X_train, y_train)

# Predict and evaluate
svm_y_pred = svm_clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, svm_y_pred))
print("\nClassification Report:\n", classification_report(y_test, svm_y_pred))
```

```
Accuracy: 0.7261428571428571

Classification Report:
               precision    recall  f1-score   support

           0       0.69      0.81      0.75     10461
           1       0.77      0.64      0.70     10539

    accuracy                           0.73     21000
   macro avg       0.73      0.73      0.72     21000
weighted avg       0.73      0.73      0.72     21000
```

In [19]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Initialize Logistic Regression model
logreg = LogisticRegression(max_iter=1000, random_state=42)

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

# Predict on test set
logreg_y_pred = logreg.predict(X_test)

# Evaluate performance
print("Logistic Regression Accuracy:", accuracy_score(y_test, logreg_y_pred))
print("\nClassification Report:\n", classification_report(y_test, logreg_y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, logreg_y_pred))
```

```
Logistic Regression Accuracy: 0.7198571428571429

Classification Report:
               precision    recall  f1-score   support

           0       0.70      0.76      0.73     10461
           1       0.74      0.68      0.71     10539

    accuracy                           0.72     21000
   macro avg       0.72      0.72      0.72     21000
weighted avg       0.72      0.72      0.72     21000


Confusion Matrix:
 [[7966 2495]
 [3388 7151]]
```

In [20]:

```
#Decision Tree Classifier
y_pred
```

Out[20]:

```
array(['1', '1', '1', ..., '1', '1', '1'], shape=(21000,), dtype=object)
```

In [21]:

```
rf_y_pred
```

Out[21]:

```
array(['1', '1', '1', ..., '1', '1', '1'], shape=(21000,), dtype=object)
```

In [22]:

```
knn_y_pred
```

Out[22]:

```
array(['1', '1', '1', ..., '1', '1', '1'], shape=(21000,), dtype=object)
```

In [23]:

```
svm_y_pred
```

Out[23]:

```
array(['1', '1', '1', ..., '0', '1', '1'], shape=(21000,), dtype=object)
```

In [24]:

```
logreg_y_pred
```

Out[24]:

```
array(['1', '1', '1', ..., '0', '1', '1'], shape=(21000,), dtype=object)
```