machine model

In [2]:
```
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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [3]:
```
df_1=pd.read_csv(r"C:\Users\arumu\Downloads\cardio_train.csv",delimiter=';')
df_1
```

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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
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 [4]:
```
# Display the first few rows of the dataset
df_1.head()

# Check for missing values
print(df_1.isnull().sum())

# Basic information about the dataset
print(df_1.info())
```

```
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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           70000 non-null  int64  
 1   age          70000 non-null  int64  
 2   gender       70000 non-null  int64  
 3   height       70000 non-null  int64  
 4   weight       70000 non-null  float64
 5   ap_hi        70000 non-null  int64  
 6   ap_lo        70000 non-null  int64  
 7   cholesterol  70000 non-null  int64  
 8   gluc         70000 non-null  int64  
 9   smoke        70000 non-null  int64  
 10  alco         70000 non-null  int64  
 11  active       70000 non-null  int64  
 12  cardio       70000 non-null  int64  
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
None
```

In [5]:
```
# Split dataset into features (X) and target (y)
X = df_1[['age', 'height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']]
y = df_1['cardio']
X
y
```

Out[5]:
```
0        0
1        1
2        1
3        1
4        0
        ..
69995    0
69996    1
69997    1
69998    1
69999    0
Name: cardio, Length: 70000, dtype: int64
```

In [6]:
```
# Ensure there are no categorical columns that need to be encoded
print(X.dtypes)
```

```
age              int64
height           int64
weight         float64
ap_hi            int64
ap_lo            int64
cholesterol      int64
gluc             int64
smoke            int64
alco             int64
active           int64
dtype: object
```

In [7]:
```
# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train, X_test, y_train, y_test
```

```
Out[7]: (         age  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  \
         68681  20417     160    64.0    120     90            3     1      0     0
         19961  22690     167    65.0    120     80            3     3      0     0
         11040  22784     160    66.0    120     90            1     1      0     0
         27673  22648     163    55.0    125     90            3     1      0     0
         22876  21712     158    85.0    150     80            3     1      0     0
         ...      ...     ...     ...    ...    ...          ...   ...    ...   ...
         37194  16001     170    75.0    150     80            1     1      1     0
         6265   23209     162    73.0    160     90            1     1      0     0
         54886  23589     169    74.0    120     80            1     1      0     0
         860    18227     167    70.0    120     80            1     1      0     0
         15795  15114     177    64.0    120     80            1     1      0     0

                active
         68681       1
         19961       0
         11040       1
         27673       1
         22876       1
         ...       ...
         37194       1
         6265        1
         54886       1
         860         0
         15795       1

         [49000 rows x 10 columns],
                  age  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  \
         46730  21770     156    64.0    140     80            2     1      0     0
         48393  21876     170    85.0    160     90            1     1      0     0
         41416  23270     151    90.0    130     80            1     1      0     0
         34506  19741     159    97.0    120     80            1     1      0     0
         43725  18395     164    68.0    120     80            1     1      0     0
         ...      ...     ...     ...    ...    ...          ...   ...    ...   ...
         1216   22392     161    68.0    150    100            2     1      0     0
         19036  14462     168    66.0    130     80            1     1      0     0
         51256  14805     159    81.0    130    100            1     1      0     0
         48198  20519     143    65.0    130     90            1     1      0     0
         2571   16181     156    80.0    180    100            2     1      0     0

                active
         46730       1
         48393       1
         41416       1
         34506       1
         43725       1
         ...       ...
         1216        1
         19036       1
         51256       0
         48198       1
         2571        1

         [21000 rows x 10 columns],
         68681    1
         19961    0
         11040    1
         27673    1
         22876    1
                 ..
         37194    1
         6265     1
         54886    0
         860      0
         15795    0
         Name: cardio, Length: 49000, dtype: int64,
         46730    1
         48393    1
         41416    1
         34506    1
         43725    0
                 ..
         1216     1
         19036    0
         51256    0
         48198    1
         2571     1
         Name: cardio, Length: 21000, dtype: int64)
```

```
In [8]: # Standardize the data (important for Random Forest as well)
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
```

```
X_test
```

Out[8]:

```
array([[ 0.93597822, -1.01890093, -0.70816849, ..., -0.31319072,
        -0.24186407,  0.49466891],
       [ 0.97889556,  0.68916043,  0.75248336, ..., -0.31319072,
        -0.24186407,  0.49466891],
       [ 1.54329899, -1.62892285,  1.10025762, ..., -0.31319072,
        -0.24186407,  0.49466891],
       ...,
       [-1.88401453, -0.65288778,  0.47426396, ..., -0.31319072,
        -0.24186407, -2.02155415],
       [ 0.4294727 , -2.60495792, -0.63861364, ..., -0.31319072,
        -0.24186407,  0.49466891],
       [-1.32689895, -1.01890093,  0.40470911, ..., -0.31319072,
        -0.24186407,  0.49466891]])
```

In [9]:

```
# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Out[9]:

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [10]:

```
# Predict the test set
y_pred = rf_model.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forest Accuracy: {accuracy * 100:.2f}%")

# Print confusion matrix and classification report
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Random Forest Accuracy: 71.34%

Confusion Matrix:
[[7568 2893]
 [3126 7413]]
```

In [11]:

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

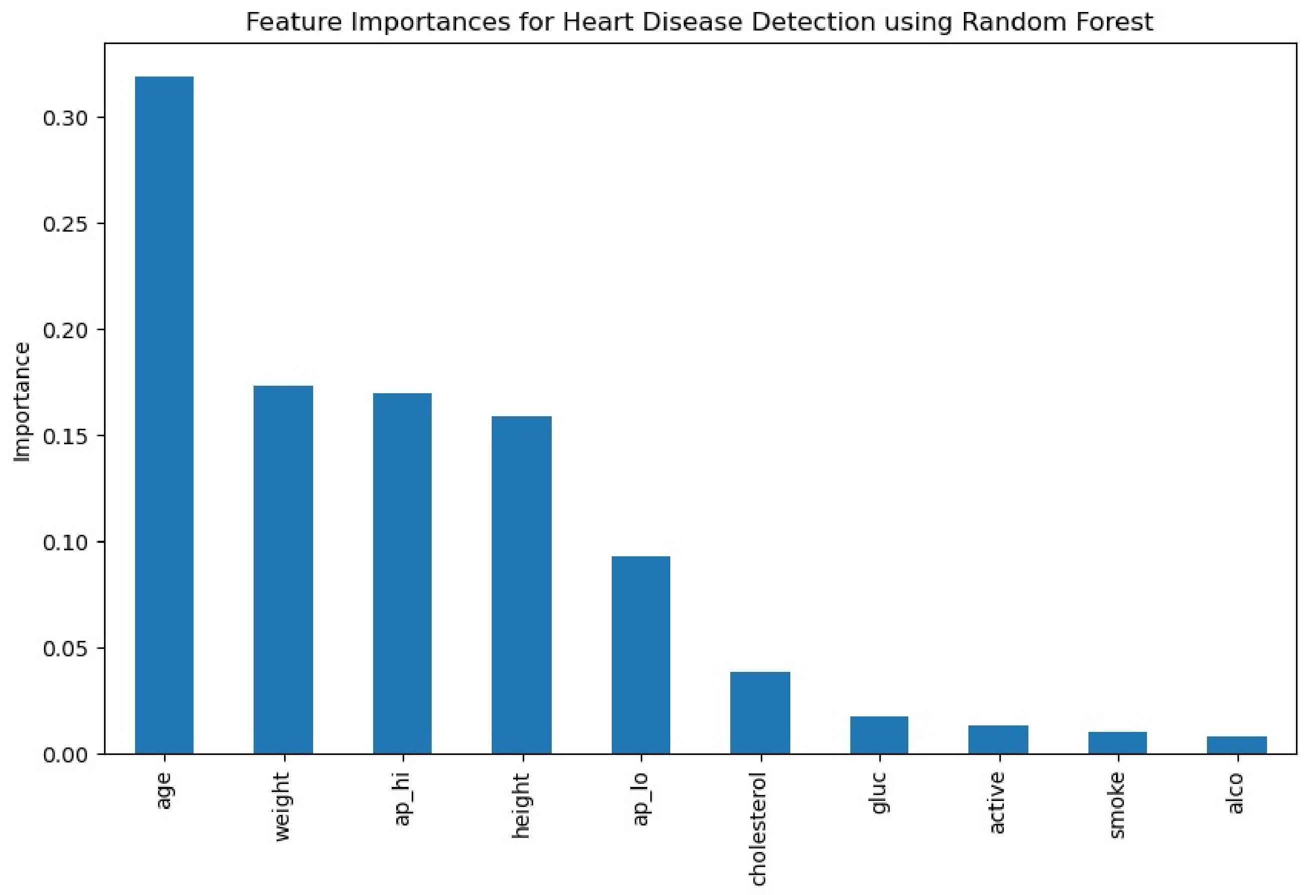
```
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.72      0.72     10461
           1       0.72      0.70      0.71     10539

    accuracy                           0.71     21000
   macro avg       0.71      0.71      0.71     21000
weighted avg       0.71      0.71      0.71     21000
```

In [12]:

```
feature_importances = pd.Series(rf_model.feature_importances_, index=X.columns)
```

In [15]:

```
# Feature importance
plt.figure(figsize=(10,6))
feature_importances.sort_values(ascending=False).plot(kind='bar')
plt.title('Feature Importances for Heart Disease Detection using Random Forest')
plt.ylabel('Importance')
plt.show()
```



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js