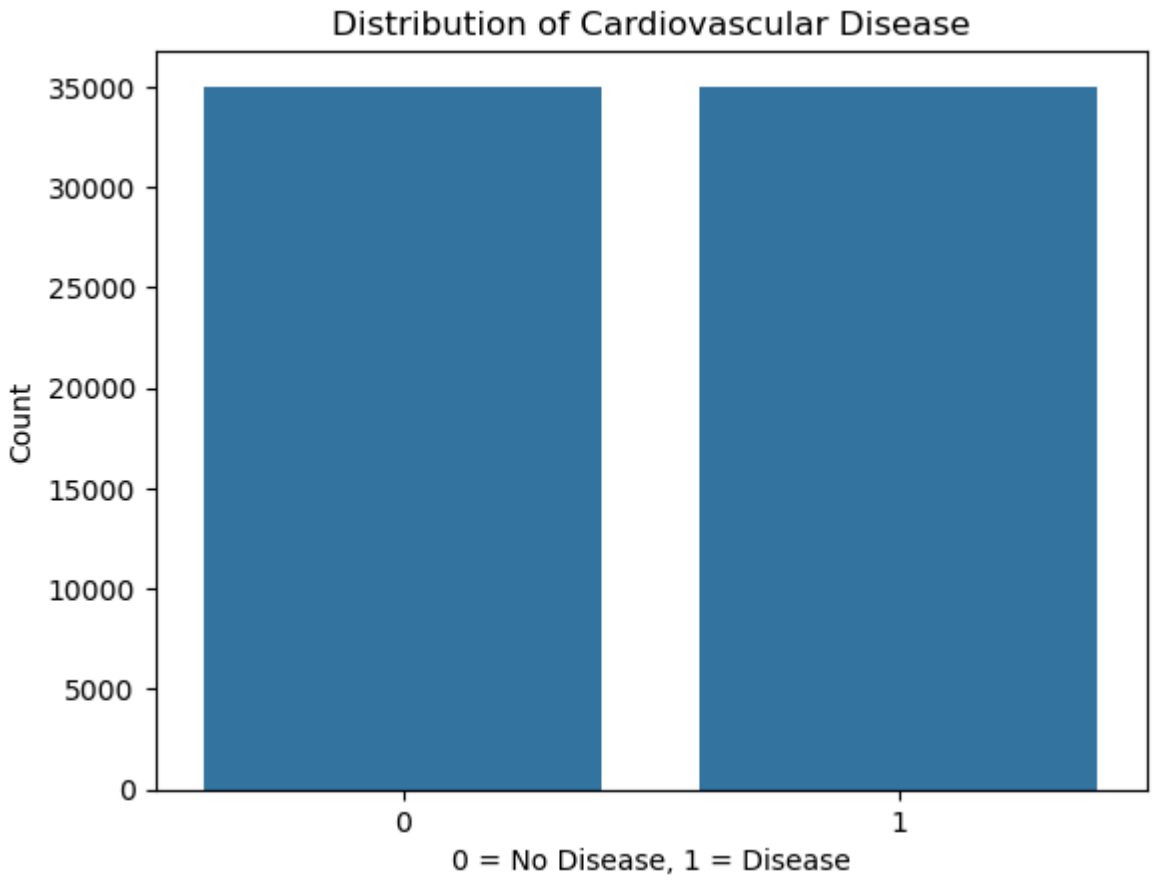
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# Load dataset
df = pd.read_csv("cardio_pred.csv", delimiter=';')

# Convert age from days to years
df['age'] = (df['age'] / 365).astype(int)

# Drop the 'id' column
df.drop('id', axis=1, inplace=True)
df
```

Out[1]:

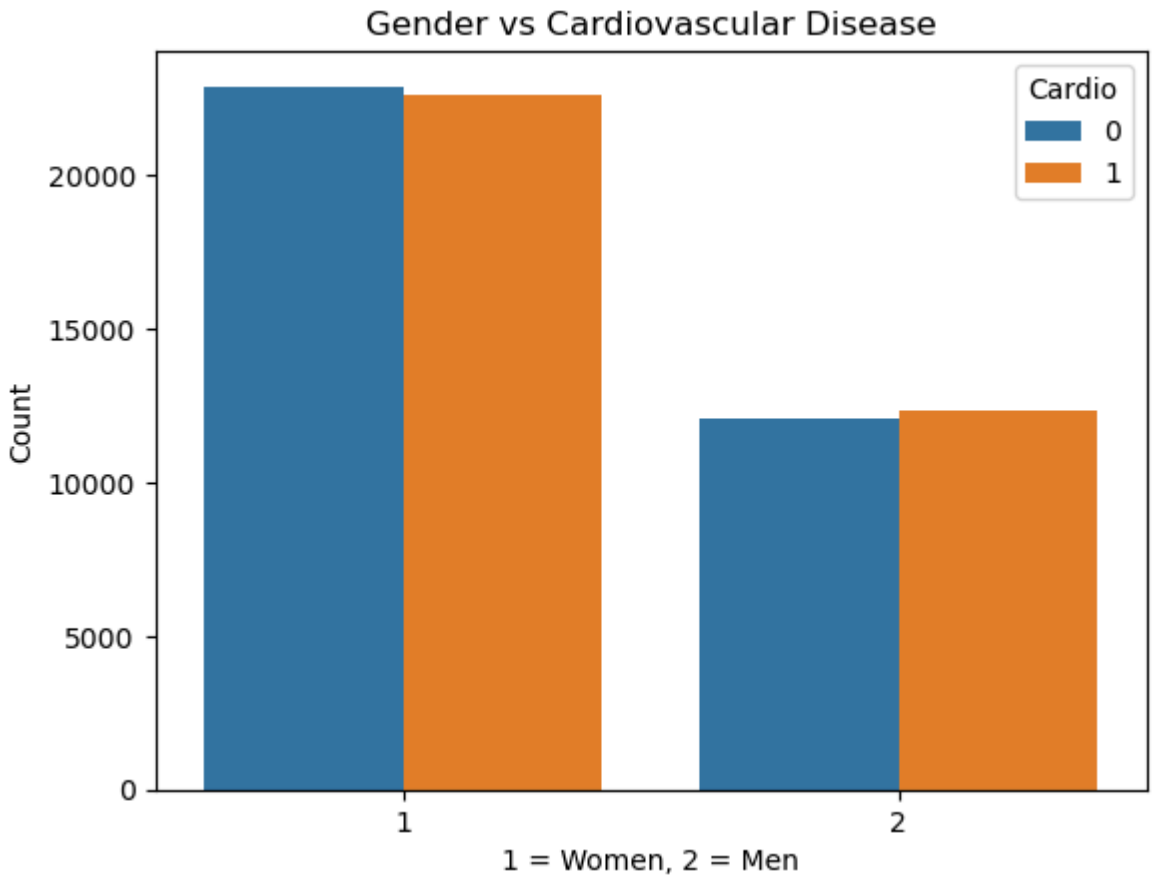
	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	acti
0	50	2	168	62.0	110	80	1	1	0	0	
1	55	1	156	85.0	140	90	3	1	0	0	
2	51	1	165	64.0	130	70	3	1	0	0	
3	48	2	169	82.0	150	100	1	1	0	0	
4	47	1	156	56.0	100	60	1	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
69995	52	2	168	76.0	120	80	1	1	1	0	
69996	61	1	158	126.0	140	90	2	2	0	0	
69997	52	2	183	105.0	180	90	3	1	0	1	
69998	61	1	163	72.0	135	80	1	2	0	0	
69999	56	1	170	72.0	120	80	2	1	0	0	

70000 rows × 12 columns

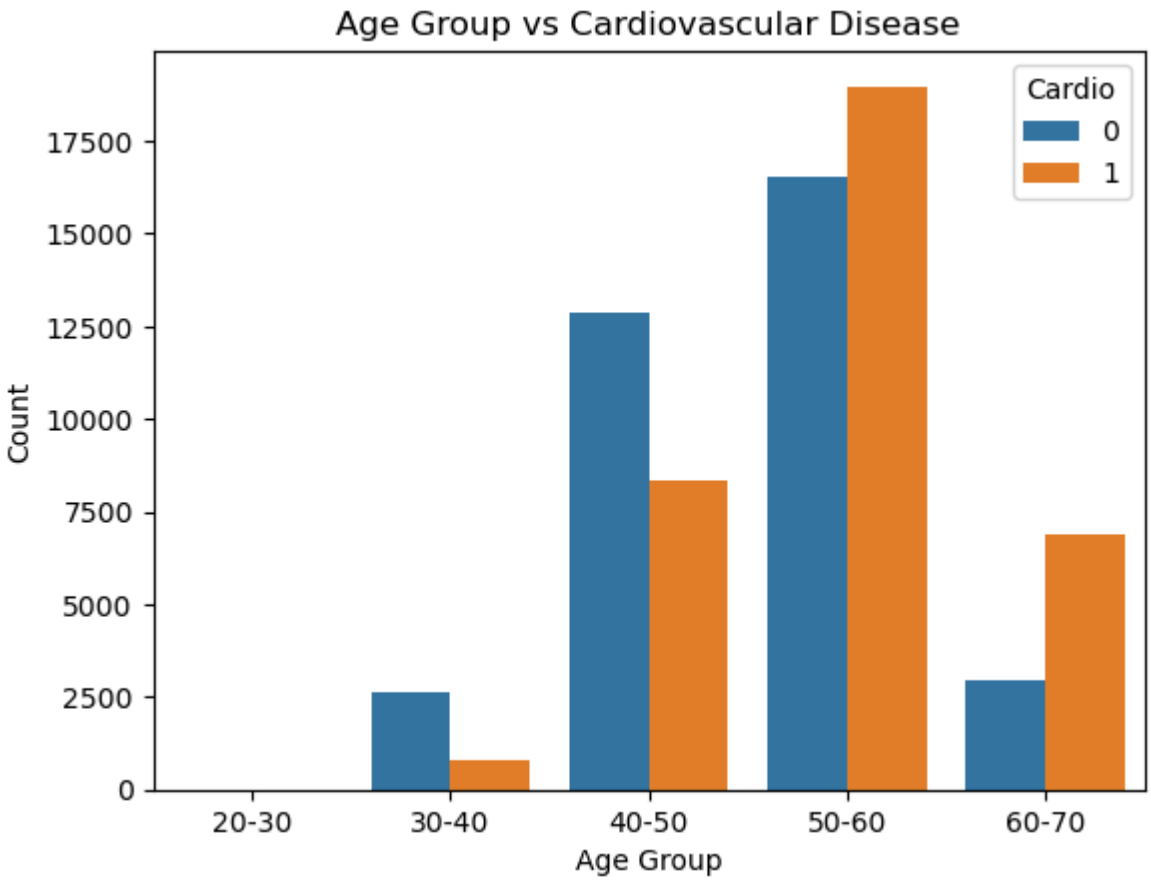
In [2]:
```
sns.countplot(x='cardio', data=df)
plt.title('Distribution of Cardiovascular Disease')
plt.xlabel('0 = No Disease, 1 = Disease')
plt.ylabel('Count')
plt.show()
```



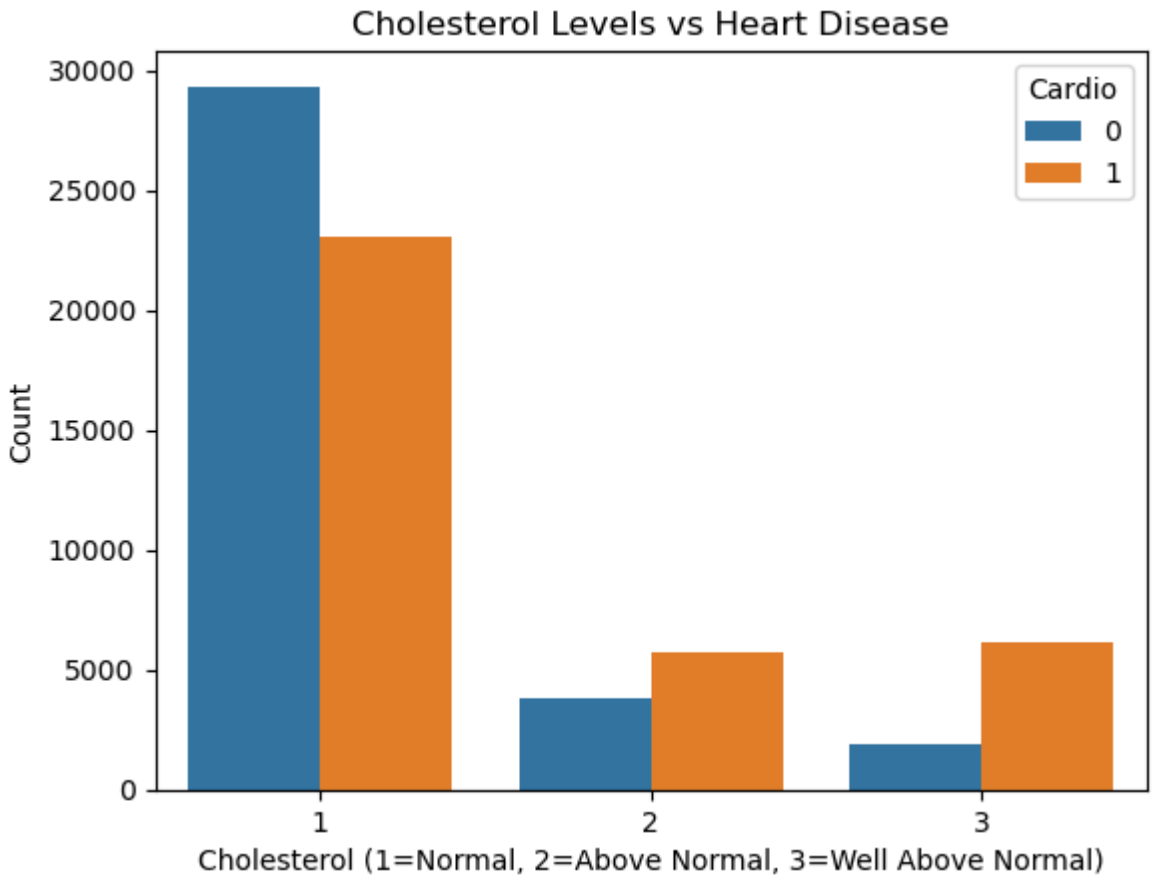
In [3]:
```
sns.countplot(x='gender', hue='cardio', data=df)
plt.title('Gender vs Cardiovascular Disease')
plt.xlabel('1 = Women, 2 = Men')
plt.ylabel('Count')
plt.legend(title='Cardio')
plt.show()
```



In [4]:
```
df['age_group'] = pd.cut(df['age'], bins=[20, 30, 40, 50, 60, 70], labels=['20-3
sns.countplot(x='age_group', hue='cardio', data=df)
plt.title('Age Group vs Cardiovascular Disease')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Cardio')
plt.show()
```

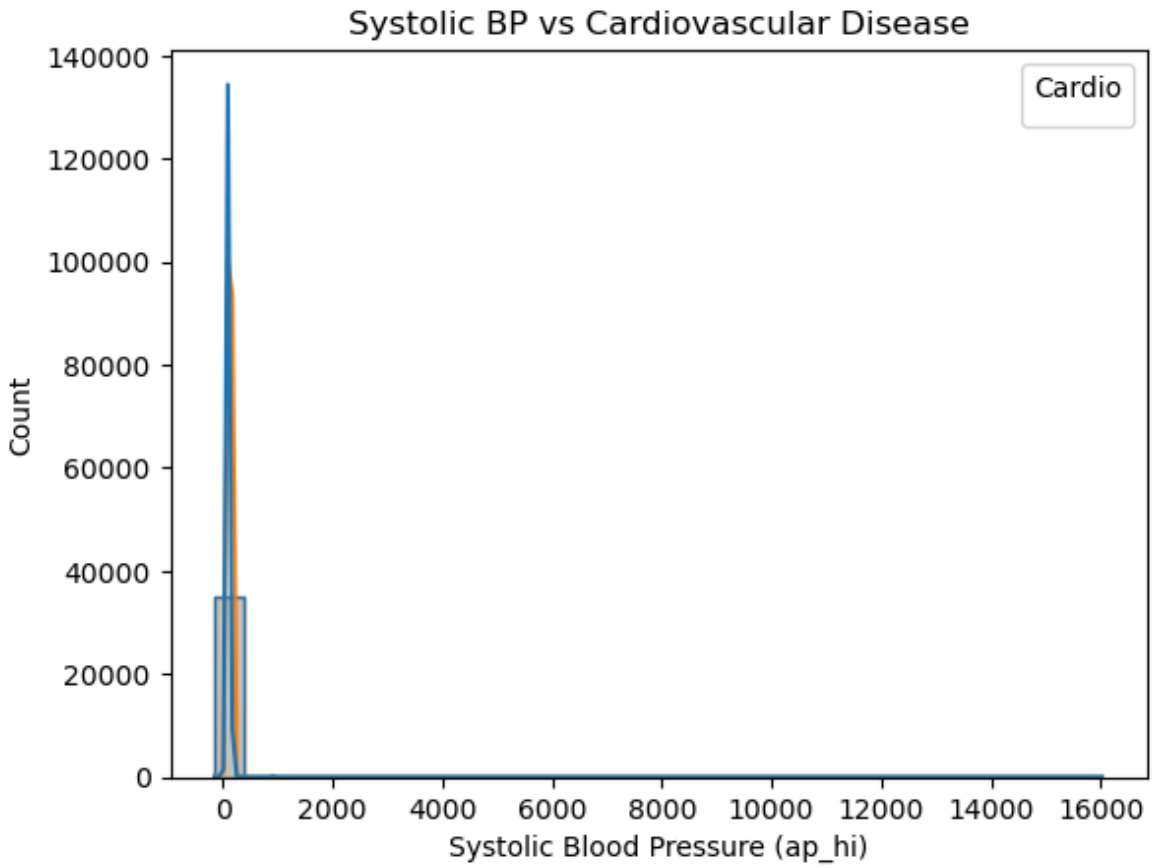


In [5]:
```
sns.countplot(x='cholesterol', hue='cardio', data=df)
plt.title('Cholesterol Levels vs Heart Disease')
plt.xlabel('Cholesterol (1=Normal, 2=Above Normal, 3=Well Above Normal)')
plt.ylabel('Count')
plt.legend(title='Cardio')
plt.show()
```

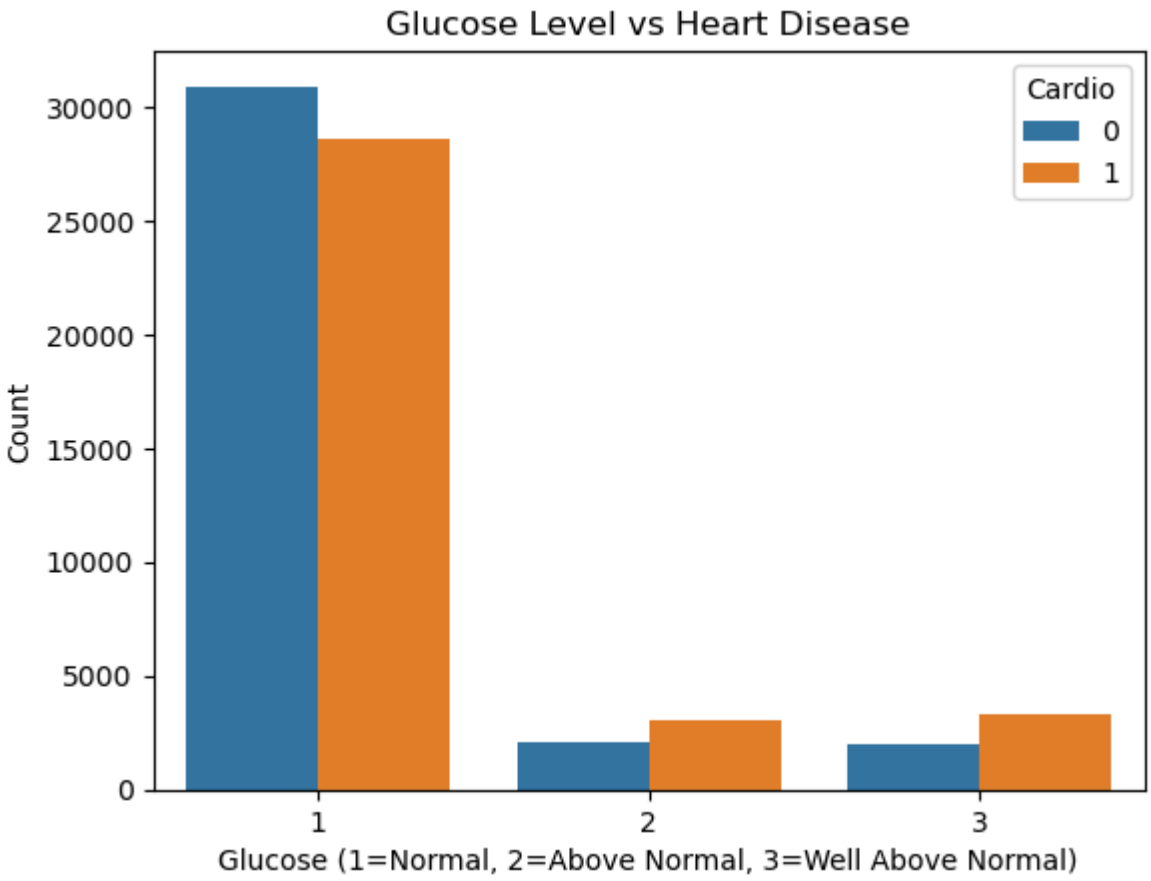


In [6]:
```
sns.histplot(data=df, x='ap_hi', hue='cardio', bins=30, kde=True, element='step'
plt.title('Systolic BP vs Cardiovascular Disease')
plt.xlabel('Systolic Blood Pressure (ap_hi)')
plt.ylabel('Count')
plt.legend(title='Cardio')
plt.show()
```

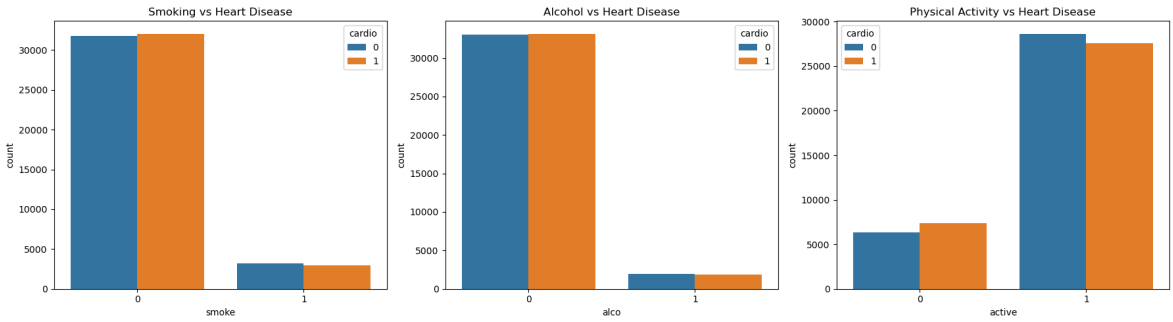
```
C:\Users\shash\AppData\Local\Temp\ipykernel_21544\2904920865.py:5: UserWarning: N
o artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
  plt.legend(title='Cardio')
```



In [7]:
```
sns.countplot(x='gluc', hue='cardio', data=df)
plt.title('Glucose Level vs Heart Disease')
plt.xlabel('Glucose (1=Normal, 2=Above Normal, 3=Well Above Normal)')
plt.ylabel('Count')
plt.legend(title='Cardio')
plt.show()
```

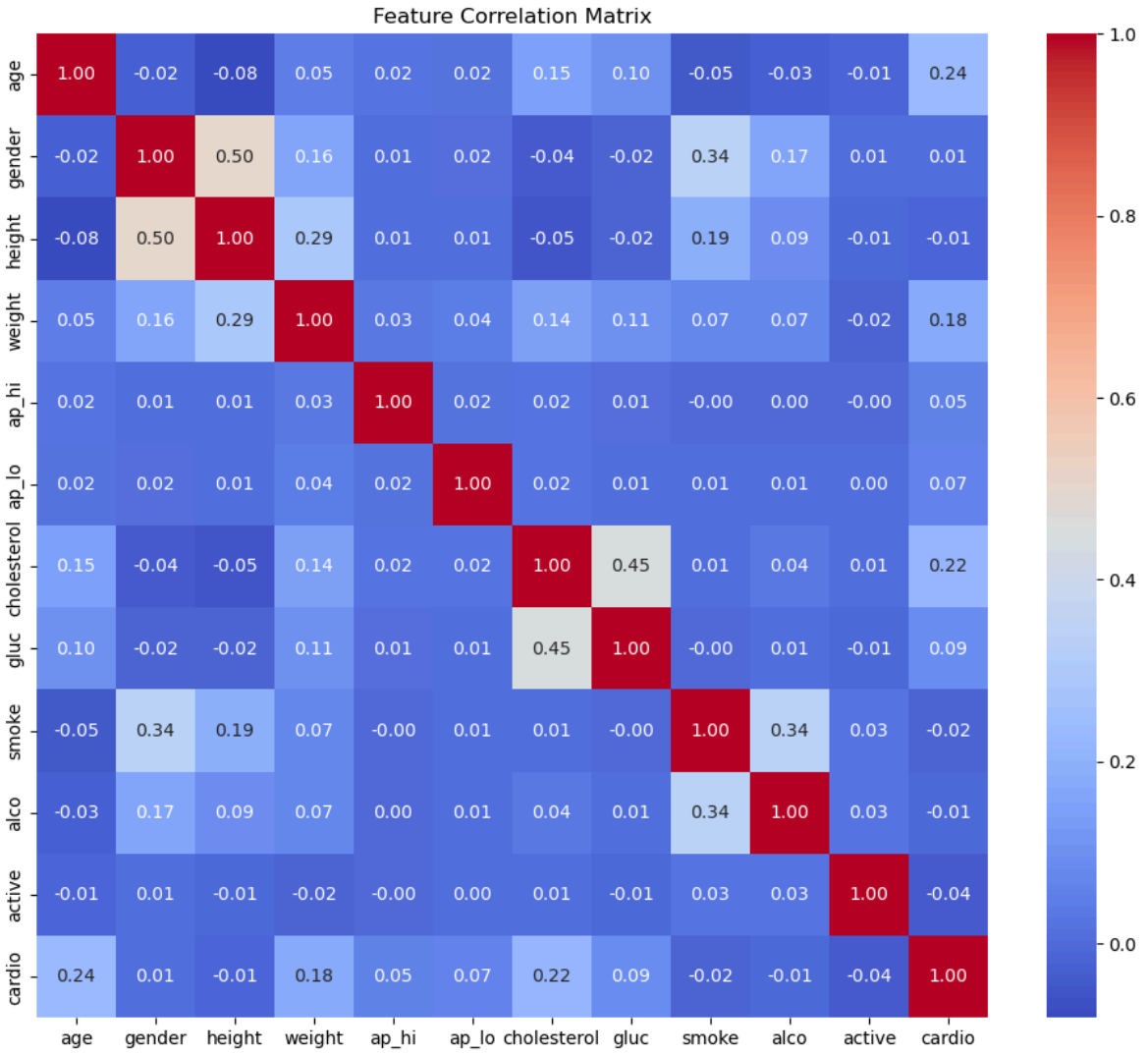


In [8]:
```
fig, axs = plt.subplots(1, 3, figsize=(18, 5))
sns.countplot(x='smoke', hue='cardio', data=df, ax=axs[0])
axs[0].set_title('Smoking vs Heart Disease')
sns.countplot(x='alco', hue='cardio', data=df, ax=axs[1])
axs[1].set_title('Alcohol vs Heart Disease')
sns.countplot(x='active', hue='cardio', data=df, ax=axs[2])
axs[2].set_title('Physical Activity vs Heart Disease')
plt.tight_layout()
plt.show()
```



Positive and stronger Correlation with the Target (cardio)

In [10]:
```
corr_data = df.drop(columns=['age_group'])
plt.figure(figsize=(12,10))                          # Create the correlation m
sns.heatmap(corr_data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Feature Correlation Matrix")
plt.show()
```



Accuracy levels of various machine learning techniques

Train-Test Split

In [13]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
x = df.drop('cardio', axis=1)
y = df['cardio']

label_encoder = LabelEncoder()
x['age_group'] = label_encoder.fit_transform(x['age_group'])  #converted non num

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
```

In [14]:
```
x_train
```

Out[14]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	acti
68681	55	1	160	64.0	120	90	3	1	0	0	
19961	62	2	167	65.0	120	80	3	3	0	0	
11040	62	1	160	66.0	120	90	1	1	0	0	
27673	62	1	163	55.0	125	90	3	1	0	0	
22876	59	1	158	85.0	150	80	3	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
37194	43	2	170	75.0	150	80	1	1	1	0	
6265	63	2	162	73.0	160	90	1	1	0	0	
54886	64	1	169	74.0	120	80	1	1	0	0	
860	49	1	167	70.0	120	80	1	1	0	0	
15795	41	2	177	64.0	120	80	1	1	0	0	

49000 rows × 12 columns

In [15]:
```
y_train
```

Out[15]:
```
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
```

In [16]:
```
x_test
```

Out[16]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	acti
46730	59	1	156	64.0	140	80	2	1	0	0	
48393	59	1	170	85.0	160	90	1	1	0	0	
41416	63	1	151	90.0	130	80	1	1	0	0	
34506	54	1	159	97.0	120	80	1	1	0	0	
43725	50	1	164	68.0	120	80	1	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
1216	61	1	161	68.0	150	100	2	1	0	0	
19036	39	1	168	66.0	130	80	1	1	0	0	
51256	40	1	159	81.0	130	100	1	1	0	0	
48198	56	1	143	65.0	130	90	1	1	0	0	
2571	44	1	156	80.0	180	100	2	1	0	0	

21000 rows × 12 columns

In [17]:
```
y_test
```

Out[17]:
```
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
```

Logistic Regression

In [19]:
```
log_model1=LogisticRegression()
log_model1.fit(x_train,y_train)
```

```
C:\Users\shash\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[19]:
```
LogisticRegression
LogisticRegression()
```

In [20]:
```
ans=log_model1.predict(x_test)
ans
```

Out[20]:
```
array([1, 1, 1, ..., 0, 1, 1], dtype=int64)
```

In [21]:
```
log_model1.score(x_test,y_test)
```

Out[21]:
```
0.7109047619047619
```

Support Vector Machines(SVM)

In [88]:
```
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

scaler = StandardScaler()          # Standardize the data
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

svc_model = SVC()
svc_model.fit(x_train_scaled, y_train)
y_pred_svc = svc_model.predict(x_test_scaled)
svc_accuracy = accuracy_score(y_test, y_pred_svc)

print(f"SVC Accuracy: {svc_accuracy:.4f}")
```

```
SVC Accuracy: 0.7262
```

Decision Tree Classifier

In [83]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)

dt_pred = dt_model.predict(x_test)
print(f"Decision Tree Accuracy: {accuracy_score(y_test, dt_pred):.4f}")
```

```
Decision Tree Accuracy: 0.6350
```

Random Forest Classifier

In [81]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)
rf_pred = rf_model.predict(x_test)
```

```
rf_accuracy = accuracy_score(y_test, rf_pred)
print(f"Random Forest Accuracy: {rf_accuracy:.4f}")
```

```
Random Forest Accuracy: 0.7088
```

K-Neighbors Classifier

In [79]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

knn_model = KNeighborsClassifier(n_neighbors=5)  # can change k=5 to other value
knn_model.fit(x_train_scaled, y_train)
knn_pred = knn_model.predict(x_test_scaled)
knn_accuracy = accuracy_score(y_test, knn_pred)
print(f"KNN Accuracy: {knn_accuracy:.4f}")
```

```
KNN Accuracy: 0.6460
```

Final Model Result

Best accuracy Machine learning model for heart disease detection : Support Vector Machines(SVM)

The Accuracy : 0.73%