```
import pandas as pd
import numpy as np
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, confusion_matrix,
classification_report
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

Project 1 : Cardiovascular Disease Prediction

Heart disease, also known as cardiovascular disease, is one of the most serious illnesses in both India and the rest of the globe. According to estimates, cardiac illnesses account for 28.1% of fatalities. More than 17.6 million fatalities, or a large portion of all deaths in 2016, were caused by it in 2016. Therefore, a system that can predict with exact precision and dependability is required for the appropriate and prompt diagnosis as well as the treatment of such diseases. Numerous academics do extensive research utilising a variety of machine learning algorithms to predict heart illness using different datasets that contain numerous factors that lead to heart attacks. Now it is your turn to do a analysis with the given dataset.

cardio_train.csv.zip

Project Output Instructions :

- Perform data pre-processing operations.F
- As a part of data analysis and visualizations draw all the possible plots to provide essential informations and to derive some meaningful insights.
- Show your correlation matrix of features according to the datasets.
- Find out accuracy levels of various machine learning techniques such as Support Vector Machines (SVM), K-Nearest Neighbor (KNN), Decision Trees (DT) , Logistic Regression (LR) and Random Forest (RF).
- Build your Machine learning model for heart disease detection according to the result.

```
df = pd.read_csv('/kaggle/input/cardio/cardio_train.csv', sep=';')

print(df.head())
```

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

   alco  active  cardio
0     0       1       0
1     0       1       1
2     0       0       1
3     0       1       1
4     0       0       0
```

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

```
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
```

```
print(df.dtypes)
```

```
id               int64
age              int64
gender           int64
height           int64
weight         float64
ap_hi            int64
ap_lo            int64
cholesterol      int64
```

```
gluc             int64
smoke            int64
alco             int64
active           int64
cardio           int64
dtype: object
```

DATA PRE PROCESSING

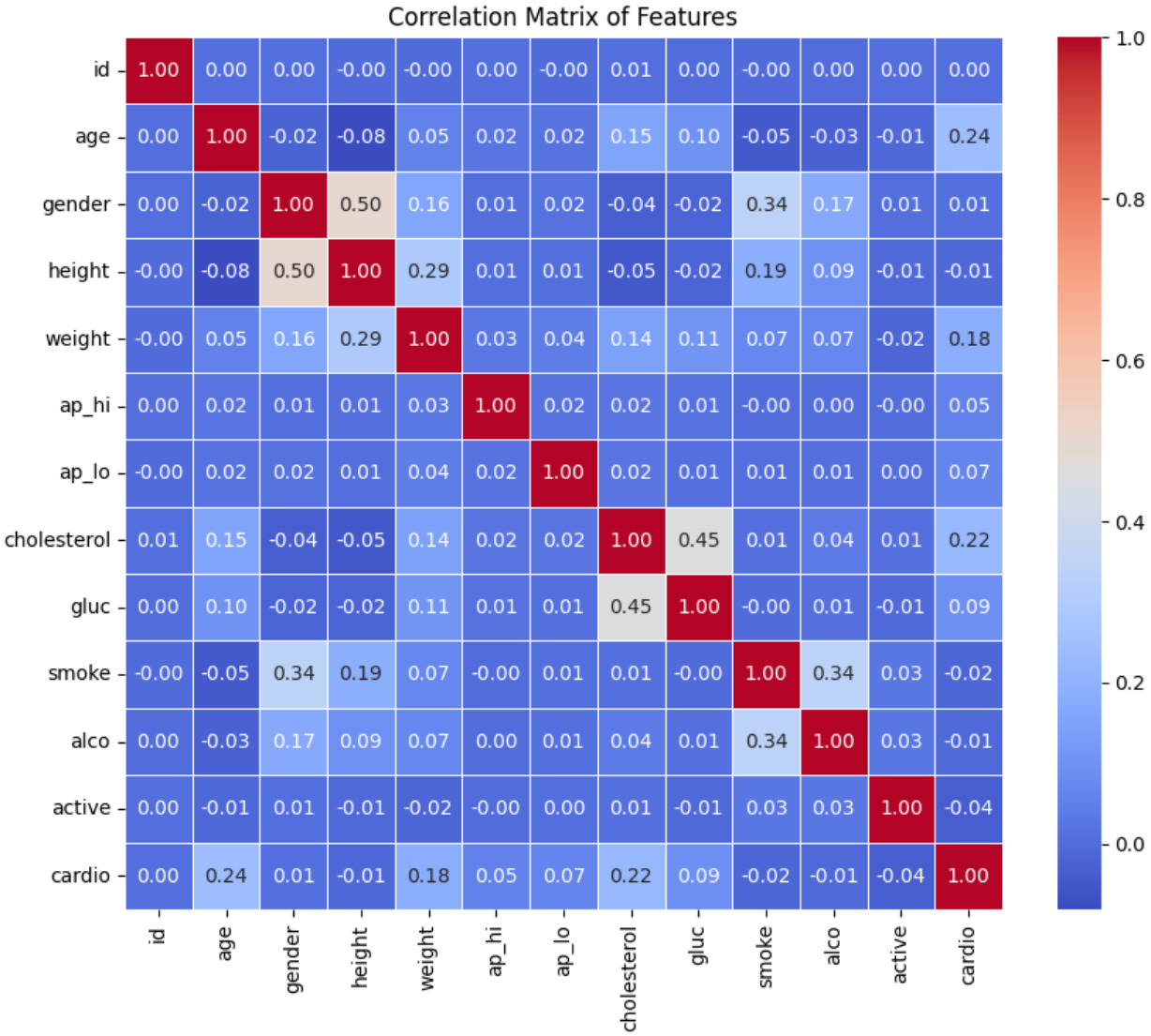
```
df.drop_duplicates(inplace=True)

df['age'] = df['age'] / 365.25

print(df.describe())
```

```
                 id           age        gender        height
weight  \
count  70000.000000  70000.000000  70000.000000  70000.000000
70000.000000
mean   49972.419900     53.302850      1.349571    164.359229
74.205690
std    28851.302323      6.754967      0.476838      8.210126
14.395757
min        0.000000     29.563313      1.000000     55.000000
10.000000
25%    25006.750000     48.361396      1.000000    159.000000
65.000000
50%    50001.500000     53.943874      1.000000    165.000000
72.000000
75%    74889.250000     58.390144      2.000000    170.000000
82.000000
max    99999.000000     64.922656      2.000000    250.000000
200.000000

               ap_hi         ap_lo   cholesterol          gluc
smoke  \
count  70000.000000  70000.000000  70000.000000  70000.000000
70000.000000
mean     128.817286     96.630414      1.366871      1.226457
0.088129
std      154.011419    188.472530      0.680250      0.572270
0.283484
min     -150.000000    -70.000000      1.000000      1.000000
0.000000
25%      120.000000     80.000000      1.000000      1.000000
0.000000
50%      120.000000     80.000000      1.000000      1.000000
0.000000
```

```
75%      140.000000     90.000000      2.000000      1.000000
0.000000
max    16020.000000  11000.000000      3.000000      3.000000
1.000000

                alco        active        cardio
count  70000.000000  70000.000000  70000.000000
mean       0.053771      0.803729      0.499700
std        0.225568      0.397179      0.500003
min        0.000000      0.000000      0.000000
25%        0.000000      1.000000      0.000000
50%        0.000000      1.000000      0.000000
75%        0.000000      1.000000      1.000000
max        1.000000      1.000000      1.000000
```

EDA

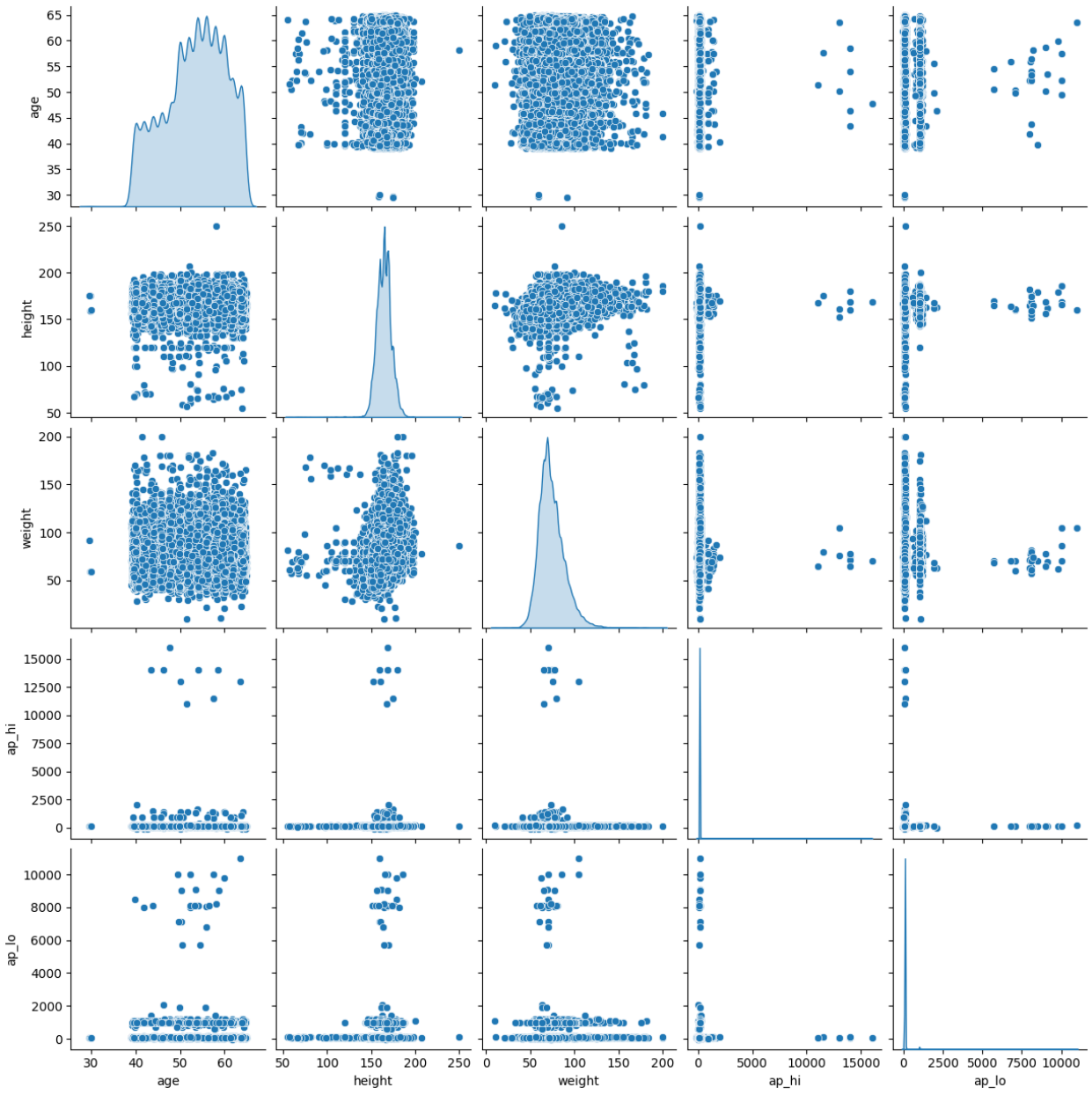
```
# Compute the correlation matrix
correlation_matrix = df.corr()


plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix of Features')
plt.show()
```



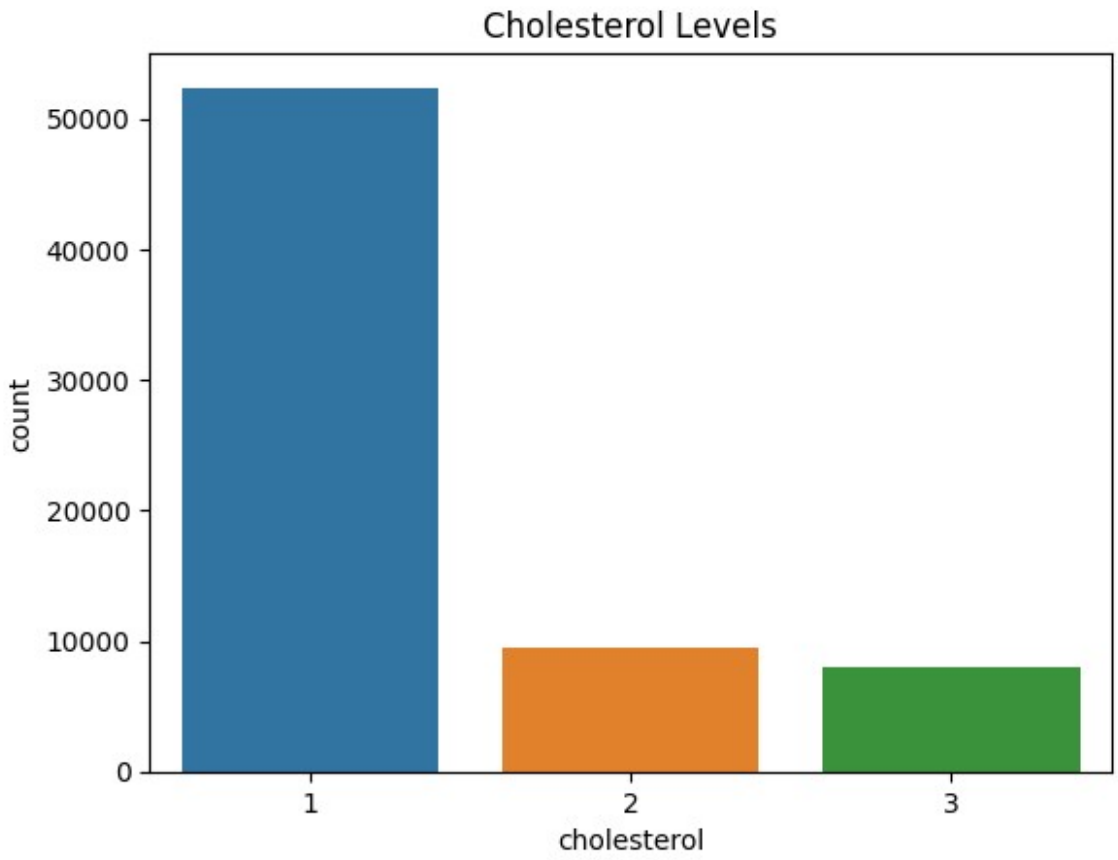

```
sns.pairplot(df[['age', 'height', 'weight', 'ap_hi', 'ap_lo']],
diag_kind='kde')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
```

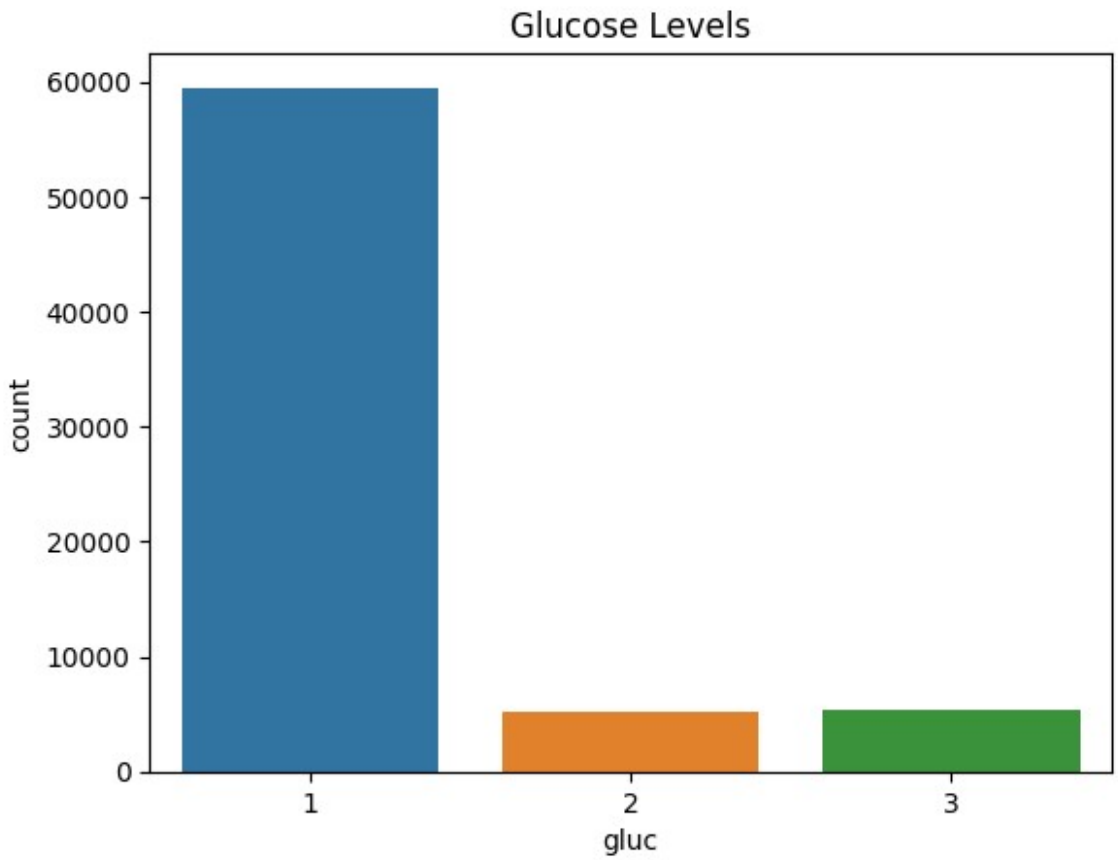
```
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



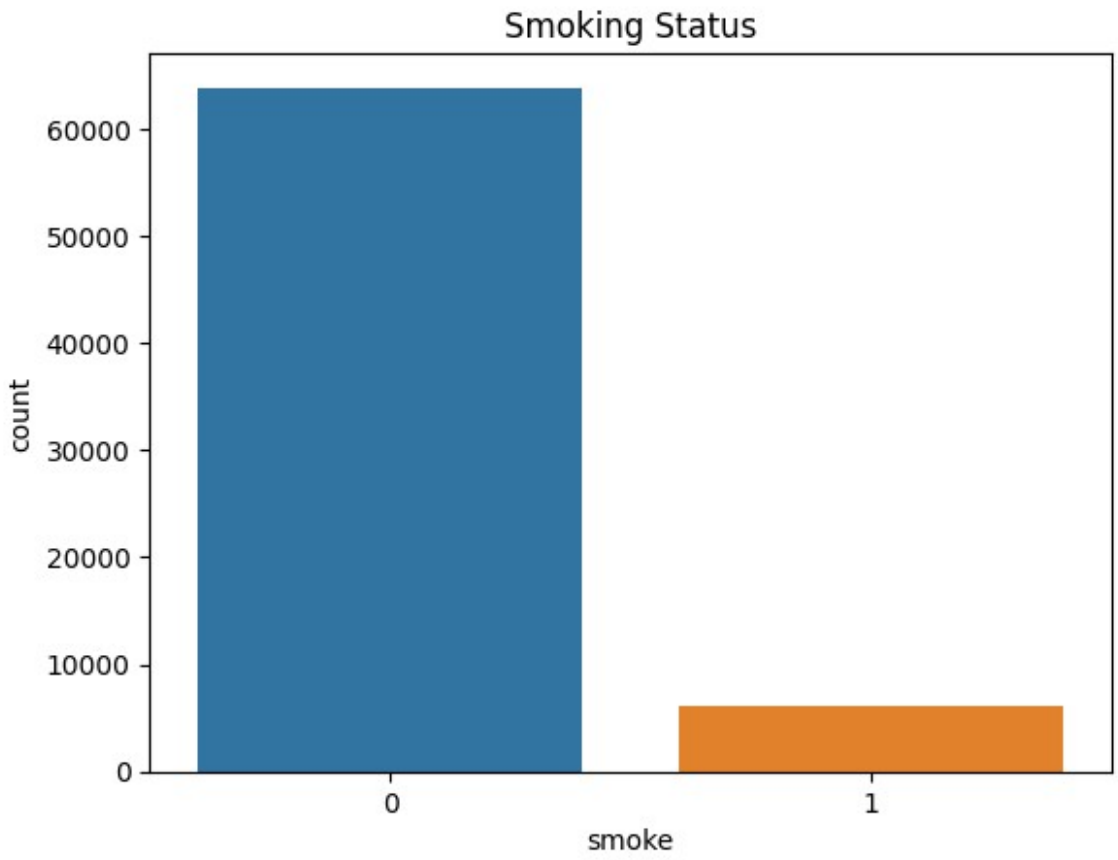

```
sns.countplot(x='cholesterol', data=df)
plt.title('Cholesterol Levels')
plt.show()
```



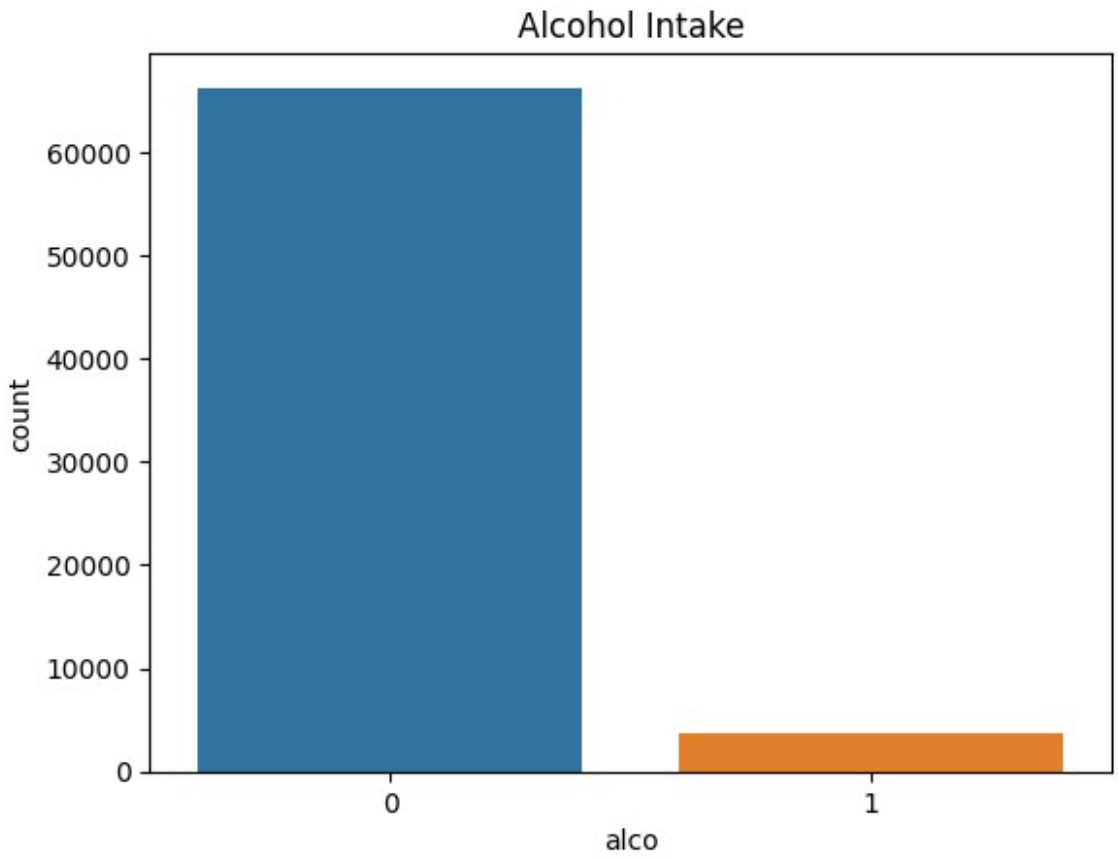

```
sns.countplot(x='gluc', data=df)
plt.title('Glucose Levels')
plt.show()
```



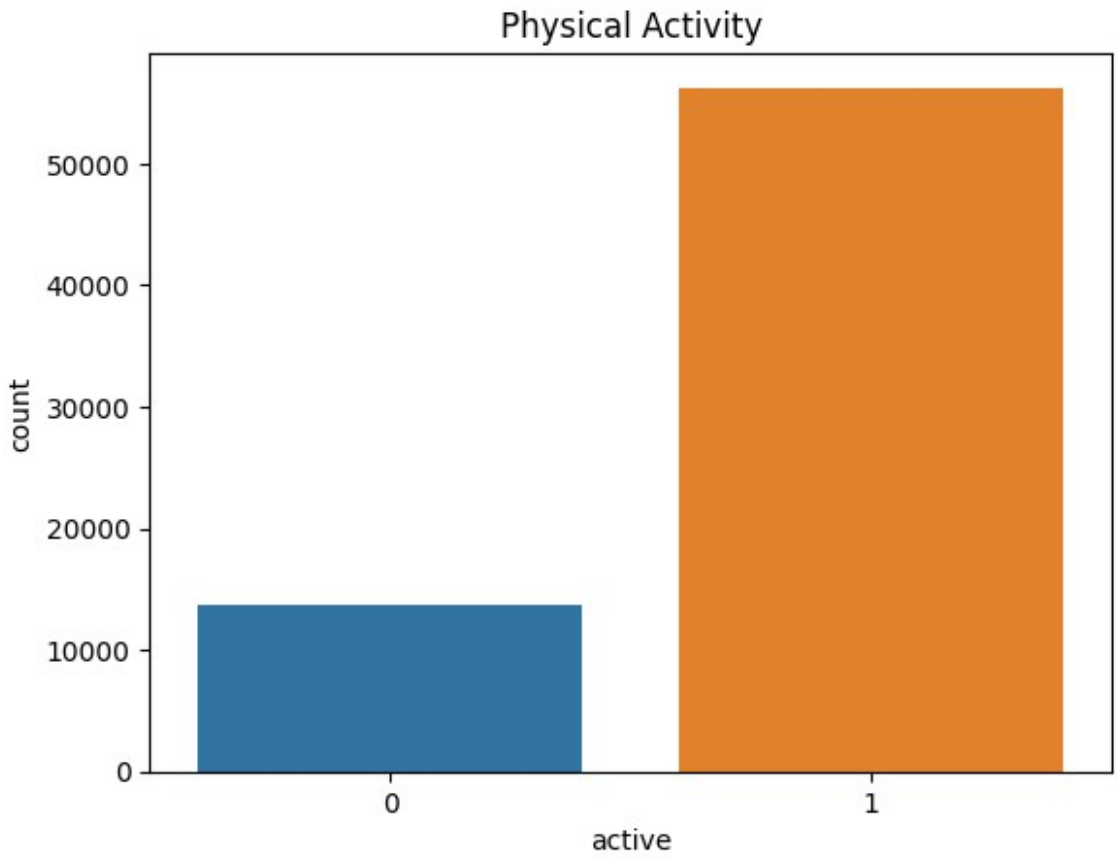

```
sns.countplot(x='smoke', data=df)
plt.title('Smoking Status')
plt.show()
```



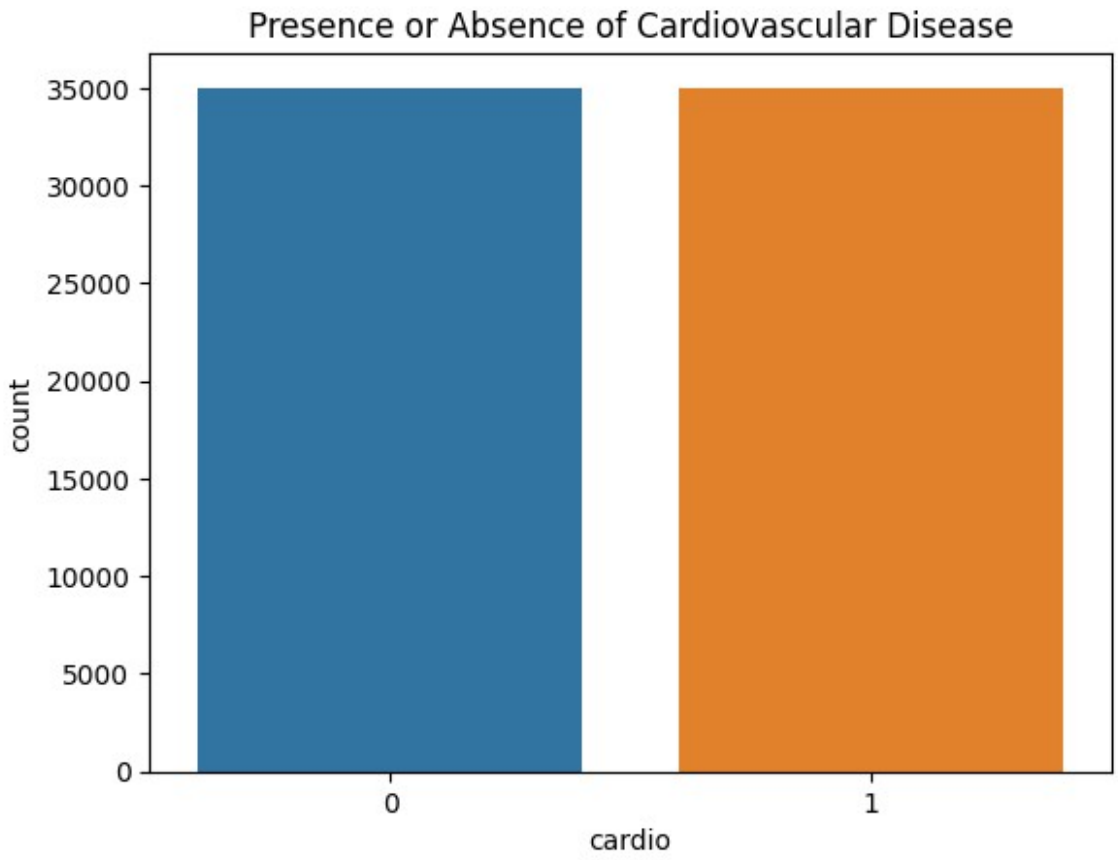

```
sns.countplot(x='alco', data=df)
plt.title('Alcohol Intake')
plt.show()
```




```
sns.countplot(x='active', data=df)
plt.title('Physical Activity')
plt.show()
```




```
sns.countplot(x='cardio', data=df)
plt.title('Presence or Absence of Cardiovascular Disease')
plt.show()
```



Comparing Machine Learning Models

```
df.drop(columns=['id'], inplace=True)

X = df.drop(columns=['cardio'])
y = df['cardio']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm_clf = SVC(kernel='linear')
knn_clf = KNeighborsClassifier(n_neighbors=5)
dt_clf = DecisionTreeClassifier(random_state=42)
lr_clf = LogisticRegression(max_iter=1000)
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)

svm_clf.fit(X_train_scaled, y_train)
knn_clf.fit(X_train_scaled, y_train)
dt_clf.fit(X_train_scaled, y_train)
```

```
lr_clf.fit(X_train_scaled, y_train)
rf_clf.fit(X_train_scaled, y_train)
```

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

```
svm_pred = svm_clf.predict(X_test_scaled)
knn_pred = knn_clf.predict(X_test_scaled)
dt_pred = dt_clf.predict(X_test_scaled)
lr_pred = lr_clf.predict(X_test_scaled)
rf_pred = rf_clf.predict(X_test_scaled)
```

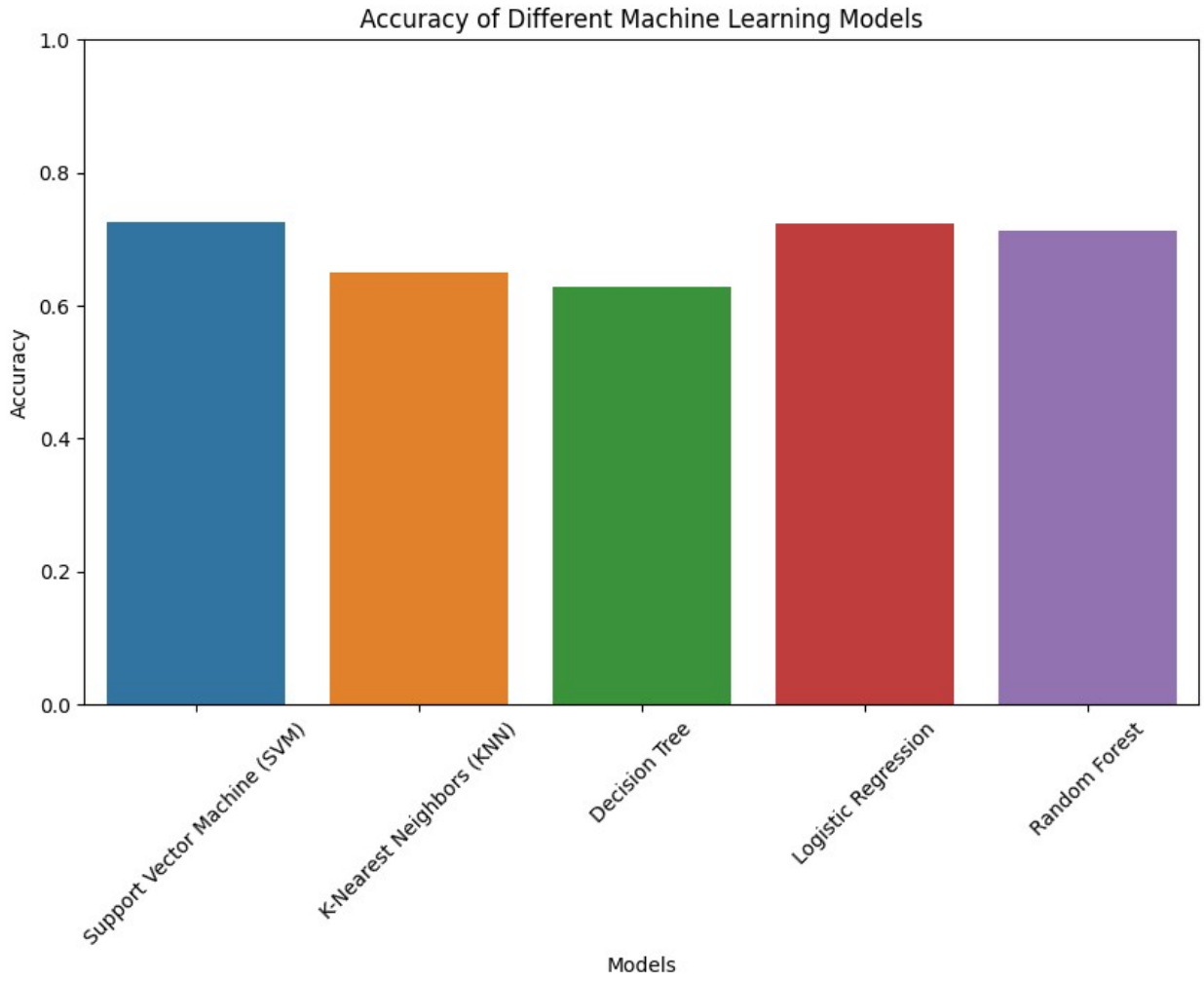
```
print("Support Vector Machine (SVM) Accuracy:", accuracy_score(y_test,
svm_pred))
print("K-Nearest Neighbors (KNN) Accuracy:", accuracy_score(y_test,
knn_pred))
print("Decision Tree Accuracy:", accuracy_score(y_test, dt_pred))
print("Logistic Regression Accuracy:", accuracy_score(y_test,
lr_pred))
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
```

```
Support Vector Machine (SVM) Accuracy: 0.7264285714285714
K-Nearest Neighbors (KNN) Accuracy: 0.6499285714285714
Decision Tree Accuracy: 0.6284285714285714
Logistic Regression Accuracy: 0.7236428571428571
Random Forest Accuracy: 0.7129285714285715
```

```
accuracy_scores = {
    "Support Vector Machine (SVM)": accuracy_score(y_test, svm_pred),
    "K-Nearest Neighbors (KNN)": accuracy_score(y_test, knn_pred),
    "Decision Tree": accuracy_score(y_test, dt_pred),
    "Logistic Regression": accuracy_score(y_test, lr_pred),
    "Random Forest": accuracy_score(y_test, rf_pred)
}
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x=list(accuracy_scores.keys()),
y=list(accuracy_scores.values()))
plt.title('Accuracy of Different Machine Learning Models')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.ylim(0.0, 1.0)
plt.xticks(rotation=45)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1765:
FutureWarning: unique with argument that is not not a Series, Index,
ExtensionArray, or np.ndarray is deprecated and will raise in a future
version.
  order = pd.unique(vector)
```



BULDING A CLASSIFICATION MODEL

By taking user inputs

```
def predict_heart_disease(age, gender, height, weight, ap_hi, ap_lo,
cholesterol, gluc, smoke, alco, active):
    # Create a DataFrame with user inputs
    user_data = pd.DataFrame({
        'age': [age],
        'gender': [gender],
        'height': [height],
        'weight': [weight],
        'ap_hi': [ap_hi],
        'ap_lo': [ap_lo],
        'cholesterol': [cholesterol],
        'gluc': [gluc],
        'smoke': [smoke],
```

```
        'alco': [alco],
        'active': [active]
    })


    user_data_scaled = scaler.transform(user_data)


    prediction = lr_clf.predict(user_data_scaled)[0]
    probability = lr_clf.predict_proba(user_data_scaled)[0]

    if prediction == 1:
        print("Prediction: You have a high probability of having
cardiovascular disease.")
    else:
        print("Prediction: You have a low probability of having
cardiovascular disease.")

    print("Probability of having cardiovascular disease:",
probability[1])



age = 50
gender = 1
height = 170
weight = 70
ap_hi = 120
ap_lo = 80
cholesterol = 1
gluc = 1
smoke = 0
alco = 0
active = 1

predict_heart_disease(age, gender, height, weight, ap_hi, ap_lo,
cholesterol, gluc, smoke, alco, active)
```

```
Prediction: You have a low probability of having cardiovascular
disease.
Probability of having cardiovascular disease: 0.33256907726948637
```