In [1]:
```
import pandas as pd
!pip install imblearn
```

```
Requirement already satisfied: imblearn in /opt/anaconda3/lib/python3.12/site-p
ackages (0.0)
Requirement already satisfied: imbalanced-learn in /opt/anaconda3/lib/python3.1
2/site-packages (from imblearn) (0.12.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/anaconda3/lib/python3.12/s
ite-packages (from imbalanced-learn->imblearn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /opt/anaconda3/lib/python3.12/si
te-packages (from imbalanced-learn->imblearn) (1.13.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /opt/anaconda3/lib/python
3.12/site-packages (from imbalanced-learn->imblearn) (1.5.1)
Requirement already satisfied: joblib>=1.1.1 in /opt/anaconda3/lib/python3.12/s
ite-packages (from imbalanced-learn->imblearn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from imbalanced-learn->imblearn) (3.5.0)
```

In [2]:
```
heart=pd.read_csv("/Users/ronak/Library/Containers/com.microsoft.Excel/Data/Dc
```

In [3]:
```
heart.head()
```

Out[3]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cance
0	Poor	Within the past 2 years	No	No	No	N
1	Very Good	Within the past year	No	Yes	No	N
2	Very Good	Within the past year	Yes	No	No	N
3	Poor	Within the past year	Yes	Yes	No	N
4	Good	Within the past year	No	No	No	N

In [4]:
```
heart.shape
```

Out[4]:
```
(308854, 19)
```

In [5]:
```
heart.isnull().sum()
```

Out[5]:
```
General_Health                  0
Checkup                         0
Exercise                        0
Heart_Disease                   0
Skin_Cancer                     0
Other_Cancer                    0
Depression                      0
Diabetes                        0
Arthritis                       0
Sex                             0
Age_Category                    0
Height_(cm)                     0
Weight_(kg)                     0
BMI                             0
Smoking_History                 0
Alcohol_Consumption             0
Fruit_Consumption               0
Green_Vegetables_Consumption    0
FriedPotato_Consumption         0
dtype: int64
```

Data Preprocessing

In [6]:
```
# Convering the column names into lower case and replacing the space with an u
heart.columns = heart.columns.str.lower().str.replace(" ", "_")

#Changing the name of a big column

heart.rename(columns = {'height_(cm)' : 'height', 'weight_(kg)' : 'weight', 'g
```

In [7]:
```
heart.head()
```

Out[7]:

	general_health	checkup	exercise	heart_disease	skin_cancer	other_cancer
0	Poor	Within the past 2 years	No	No	No	No
1	Very Good	Within the past year	No	Yes	No	No
2	Very Good	Within the past year	Yes	No	No	No
3	Poor	Within the past year	Yes	Yes	No	No
4	Good	Within the past year	No	No	No	No

In [8]:
```
heart['checkup'] = heart['checkup'].replace('Within the past 2 years', 'Past 2
heart['checkup'] = heart['checkup'].replace('Within the past year', 'Past 1 ye
heart['checkup'] = heart['checkup'].replace('Within the past 5 years', 'Past 5
heart['checkup'] = heart['checkup'].replace('5 or more years ago', 'More than 


heart['diabetes'] = heart['diabetes'].replace('No, pre-diabetes or borderline 
heart['diabetes'] = heart['diabetes'].replace('Yes, but female told only durin

heart['age_category'] = heart['age_category'].replace('18-24', 'Young')
heart['age_category'] = heart['age_category'].replace('25-29', 'Adult')
heart['age_category'] = heart['age_category'].replace('30-34', 'Adult')
heart['age_category'] = heart['age_category'].replace('35-39', 'Adult')
heart['age_category'] = heart['age_category'].replace('40-44', 'Mid-Aged')
heart['age_category'] = heart['age_category'].replace('45-49', 'Mid-Aged')
heart['age_category'] = heart['age_category'].replace('50-54', 'Mid-Aged')
heart['age_category'] = heart['age_category'].replace('55-59', 'Senior-Adult')
heart['age_category'] = heart['age_category'].replace('60-64', 'Senior-Adult')
heart['age_category'] = heart['age_category'].replace('65-69', 'Elderly')
heart['age_category'] = heart['age_category'].replace('70-74', 'Elderly')
heart['age_category'] = heart['age_category'].replace('75-79', 'Elderly')
heart['age_category'] = heart['age_category'].replace('80+', 'Elderly')
```

In [9]:
```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 19 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   general_health          308854 non-null  object 
 1   checkup                 308854 non-null  object 
 2   exercise                308854 non-null  object 
 3   heart_disease           308854 non-null  object 
 4   skin_cancer             308854 non-null  object 
 5   other_cancer            308854 non-null  object 
 6   depression              308854 non-null  object 
 7   diabetes                308854 non-null  object 
 8   arthritis               308854 non-null  object 
 9   sex                     308854 non-null  object 
 10  age_category            308854 non-null  object 
 11  height                  308854 non-null  float64
 12  weight                  308854 non-null  float64
 13  bmi                     308854 non-null  float64
 14  smoking_history         308854 non-null  object 
 15  alcohol_consumption     308854 non-null  float64
 16  fruit_consumption       308854 non-null  float64
 17  vegetables_consumption  308854 non-null  float64
 18  potato_consumption      308854 non-null  float64
dtypes: float64(7), object(12)
memory usage: 44.8+ MB
```

In [10]:
```
# Visualization
import plotly.express as px
import plotly.subplots as sp
import plotly.graph_objs as go
import matplotlib.pyplot as plt
colors = px.colors.sequential.Plasma_r
```

```
In [11]: fig1 = px.histogram(heart, x="general_health", color = 'general_health', color
fig1.update_layout(plot_bgcolor='white')
fig1.show()
print('\n',"="*90,'\n')

fig2 = px.histogram(heart, x="general_health", color = 'heart_disease', color_
fig2.update_layout(plot_bgcolor='white')
fig2.show()
print('\n',"="*90,'\n')
```

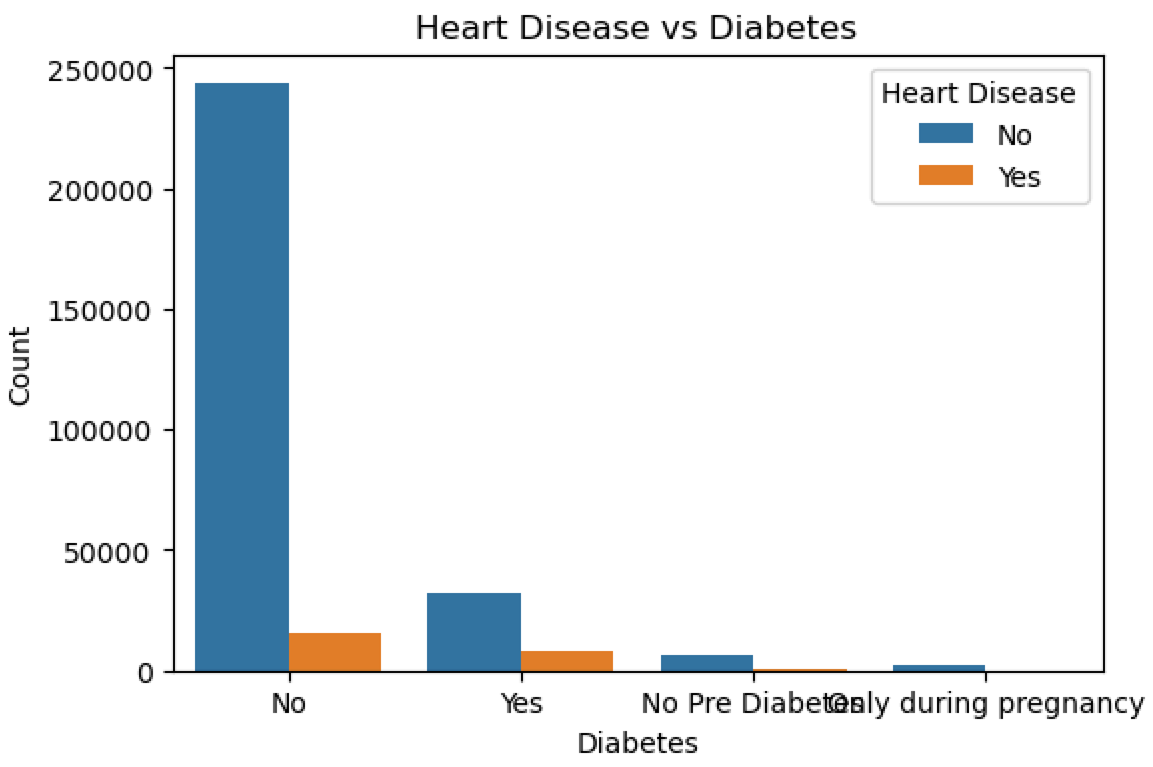
```
================================================================================
==========
================================================================================
===========
```

```
In [12]: age_category_counts = heart['age_category'].value_counts()
fig_1 = px.bar(x=age_category_counts.index, y=age_category_counts.values, colo
fig_1.update_layout(title="1.Distribution of Age Categories in the Dataset", x
fig_1.show()
print('\n', "="*80, '\n')

fig_2 = px.histogram(heart, x="age_category", color='heart_disease', barmode='
fig_2.update_layout(xaxis_title="age_category", yaxis_title="Count", legend_ti
fig_2.show()
print('\n', "="*80, '\n')

grouped_data = heart.groupby(['age_category', 'heart_disease'], as_index=False
fig_3 = px.bar(grouped_data, x='age_category', y='bmi', color='heart_disease',
fig_3.update_layout(xaxis_title="Age Group", yaxis_title="Average BMI", legend
fig_3.show()
```

==
=

==
=

In [13]:
```
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(data=heart, x="diabetes", hue="heart_disease")
plt.title("Heart Disease vs Diabetes")
plt.xlabel("Diabetes")
plt.ylabel("Count")
plt.legend(title="Heart Disease", labels=["No", "Yes"])
plt.show()
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [14]: col = ['alcohol_consumption', 'fruit_consumption', 'vegetables_consumption', '

         for i in col:
             heart[i] = heart[i].astype(int)
```

```
In [15]: # Define BMI ranges and labels for each group
         bmi_bins = [12.02, 18.3, 26.85, 31.58, 37.8, 100]
         bmi_labels = ['Underweight', 'Normal weight', 'Overweight', 'Obese I', 'Obese
         heart['bmi_group'] = pd.cut(heart['bmi'], bins=bmi_bins, labels=bmi_labels, ri
```

```
In [16]: column_to_move = heart.pop('bmi_group')
         heart.insert(14, 'bmi_group', column_to_move)
```

```
In [17]: heart['bmi_group'] = heart['bmi_group'].astype('object')
```

```
In [18]: # Import the OneHotEncoder class from scikit-learn
         from sklearn.preprocessing import OneHotEncoder
         heart['heart_disease'] = heart['heart_disease'].map({'Yes':1, 'No':0})
         cat=['sex', 'smoking_history']

         OH_Encoder = OneHotEncoder(handle_unknown='ignore',  sparse_output=False)
```

```
OH = OH_Encoder.fit_transform(heart[cat])
cols = OH_Encoder.get_feature_names_out(cat)
OH = pd.DataFrame(OH, columns=cols)
heart = heart.drop(cat,axis=1)
heart = pd.concat([heart, OH], axis =1)
```

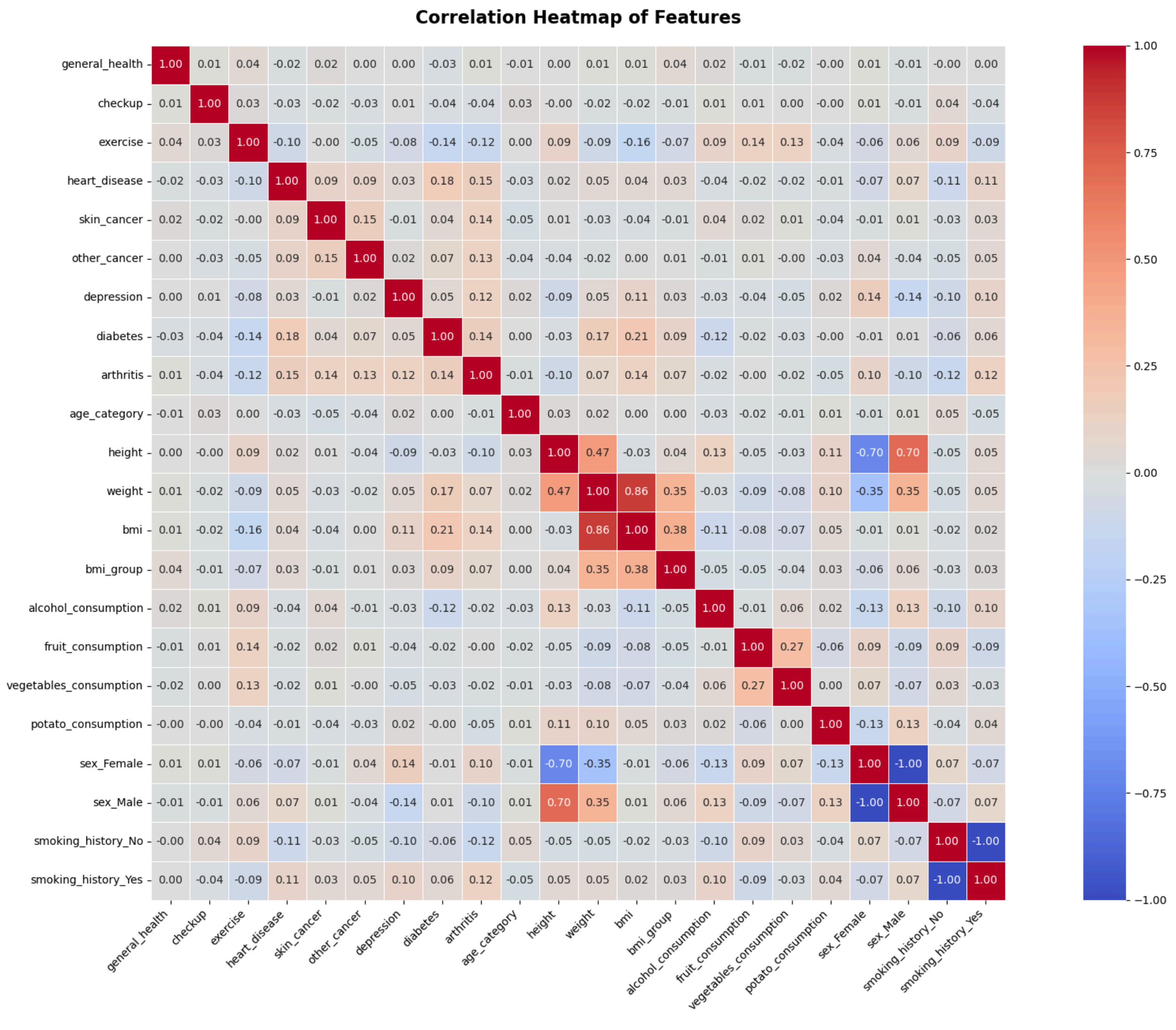
In [19]:
```
from sklearn.preprocessing import LabelEncoder
categorical_columns = ['general_health', 'checkup', 'exercise', 'skin_cancer',

# Initialize LabelEncoder

label_encoder = LabelEncoder()

# Apply label encoding to each ordinal categorical column

for col in categorical_columns:
    heart[col] = label_encoder.fit_transform(heart[col])
```

In [20]:
```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 22 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   general_health          308854 non-null  int64  
 1   checkup                 308854 non-null  int64  
 2   exercise                308854 non-null  int64  
 3   heart_disease           308854 non-null  int64  
 4   skin_cancer             308854 non-null  int64  
 5   other_cancer            308854 non-null  int64  
 6   depression              308854 non-null  int64  
 7   diabetes                308854 non-null  int64  
 8   arthritis               308854 non-null  int64  
 9   age_category            308854 non-null  int64  
 10  height                  308854 non-null  float64
 11  weight                  308854 non-null  float64
 12  bmi                     308854 non-null  float64
 13  bmi_group               308854 non-null  int64  
 14  alcohol_consumption     308854 non-null  int64  
 15  fruit_consumption       308854 non-null  int64  
 16  vegetables_consumption  308854 non-null  int64  
 17  potato_consumption      308854 non-null  int64  
 18  sex_Female              308854 non-null  float64
 19  sex_Male                308854 non-null  float64
 20  smoking_history_No      308854 non-null  float64
 21  smoking_history_Yes     308854 non-null  float64
dtypes: float64(7), int64(15)
memory usage: 51.8 MB
```

In [21]:
```
# Compute correlation only for numerical features
corr = heart.corr(numeric_only=True)
```

```
# Set figure size
plt.figure(figsize=(20,13))

# Draw heatmap
sns.heatmap(
    corr,
    annot=True,            # Show correlation values
    fmt=".2f",             # Format decimals
    cmap="coolwarm",       # Color scheme
    cbar=True,             # Show color bar
    square=True,           # Make cells square
    linewidths=0.5         # Add cell borders
)

plt.title("Correlation Heatmap of Features", fontsize=16, fontweight="bold", p
plt.xticks(rotation=45, ha="right")
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



```
In [22]: heart["heart_disease"].value_counts()
```

Out[22]:
```
heart_disease
0    283883
1     24971
Name: count, dtype: int64
```

In [23]:
```python
X = heart.drop("heart_disease", axis = 1)
y = heart['heart_disease']
```

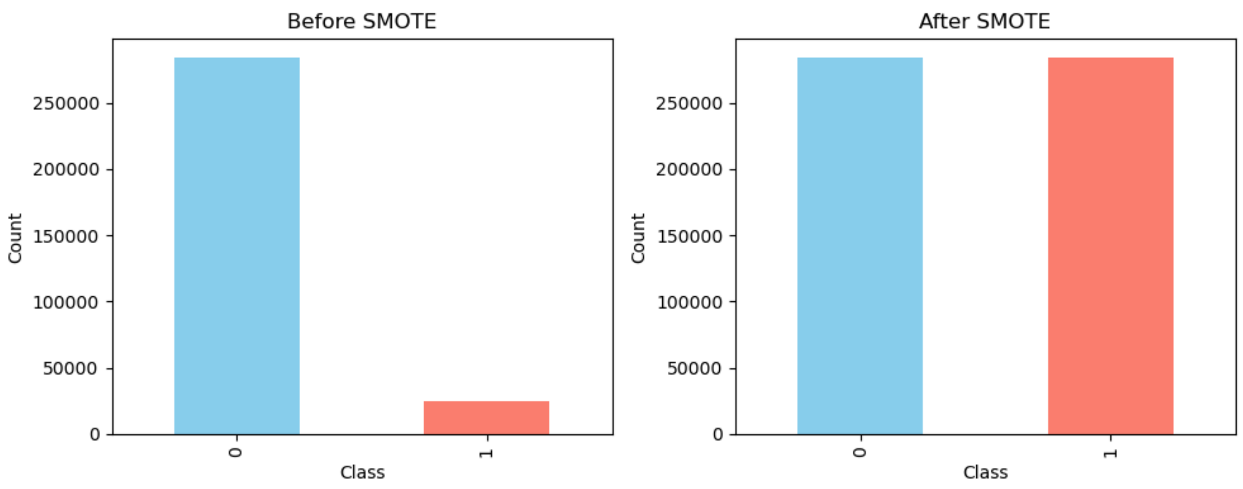
In [24]:
```python
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)
```

In [25]:
```python
from collections import Counter
print("Before SMOTE:", Counter(y))

# After SMOTE
print("After SMOTE:", Counter(y_balanced))

# Convert to DataFrame for better visualization
before = pd.Series(y).value_counts()
after = pd.Series(y_balanced).value_counts()

print("\nClass distribution before SMOTE:\n", before)
print("\nClass distribution after SMOTE:\n", after)
```

```
Before SMOTE: Counter({0: 283883, 1: 24971})
After SMOTE: Counter({0: 283883, 1: 283883})

Class distribution before SMOTE:
 heart_disease
0    283883
1     24971
Name: count, dtype: int64

Class distribution after SMOTE:
 heart_disease
0    283883
1    283883
Name: count, dtype: int64
```

In [26]:
```python
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(10,4))

# Before SMOTE
before.plot(kind="bar", ax=axes[0], color=["skyblue", "salmon"])
axes[0].set_title("Before SMOTE")
axes[0].set_xlabel("Class")
axes[0].set_ylabel("Count")

# After SMOTE
after.plot(kind="bar", ax=axes[1], color=["skyblue", "salmon"])
axes[1].set_title("After SMOTE")
```

```
axes[1].set_xlabel("Class")
axes[1].set_ylabel("Count")

plt.tight_layout()
plt.show()
```



In [27]:
```
from sklearn.model_selection import train_test_split
# Splitting the data into training and testing sets for diabetes balanced

X_train, X_test, y_train, y_test = train_test_split(X_balanced, y_balanced, te
```

In [28]:
```
from sklearn.preprocessing import StandardScaler
scaler_d = StandardScaler()
X_train_scaled = scaler_d.fit_transform(X_train)
X_test_scaled = scaler_d.transform(X_test)
```

In [29]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifie
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, classific
from xgboost import XGBClassifier


# Define models in a dictionary
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42)
    "Gradient Boosting": GradientBoostingClassifier(),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', r

}
```

In [30]:
```
# Dictionary to store results
results = {}

# Train and evaluate each model
```

```python
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    # Calculate metrics
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, model.predict_proba(X_test_scaled)[:,1])

    results[name] = {"Accuracy": acc, "F1-score": f1, "ROC-AUC": auc}

    print(f"=== {name} ===")
    print(classification_report(y_test, y_pred))
    print("\n")
```

```
=== Logistic Regression ===
              precision    recall  f1-score   support

           0       0.72      0.72      0.72     85071
           1       0.72      0.72      0.72     85259

    accuracy                           0.72    170330
   macro avg       0.72      0.72      0.72    170330
weighted avg       0.72      0.72      0.72    170330



=== Random Forest ===
              precision    recall  f1-score   support

           0       0.92      0.95      0.94     85071
           1       0.95      0.92      0.93     85259

    accuracy                           0.93    170330
   macro avg       0.94      0.93      0.93    170330
weighted avg       0.94      0.93      0.93    170330



=== Gradient Boosting ===
              precision    recall  f1-score   support

           0       0.87      0.89      0.88     85071
           1       0.89      0.86      0.88     85259

    accuracy                           0.88    170330
   macro avg       0.88      0.88      0.88    170330
weighted avg       0.88      0.88      0.88    170330
```

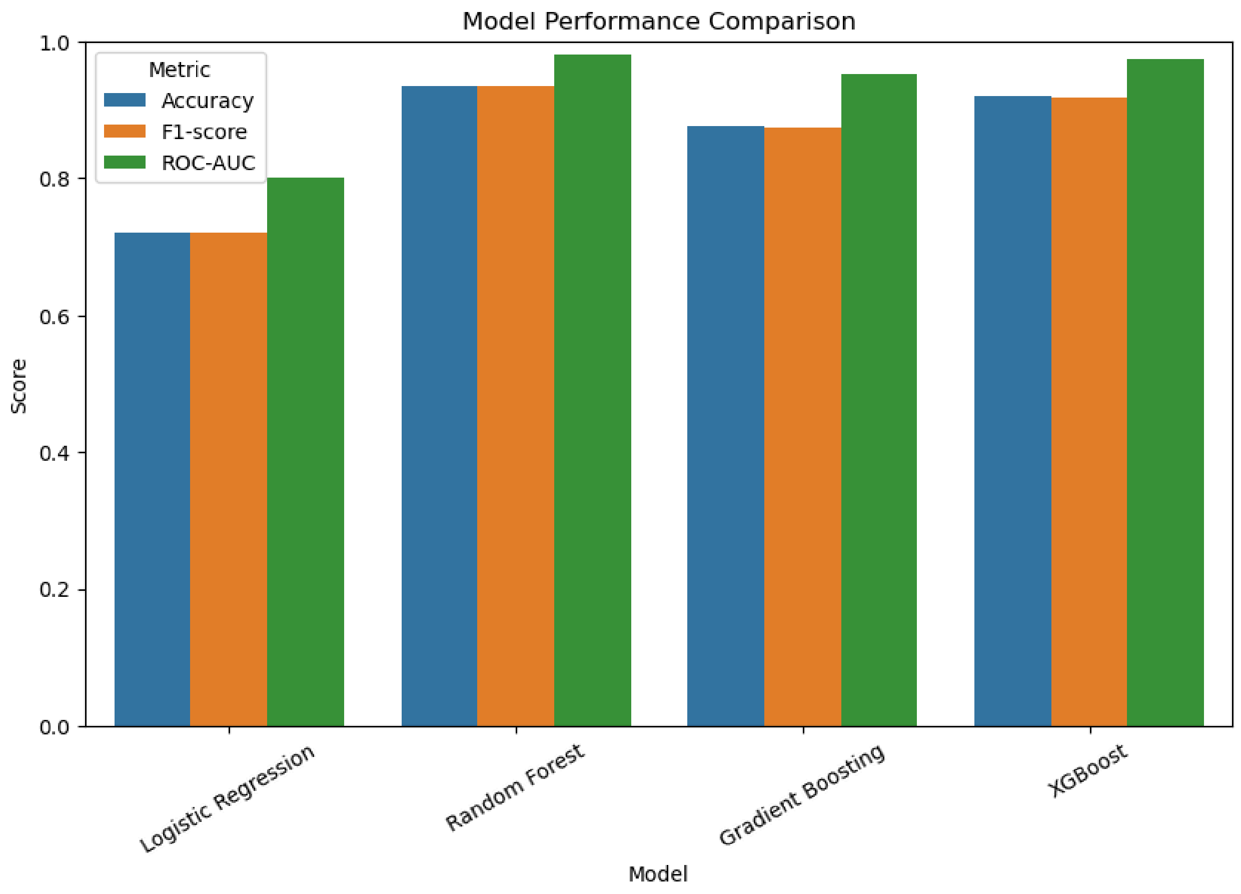
```
/opt/anaconda3/lib/python3.12/site-packages/xgboost/training.py:183: UserWarnin
g:

[19:19:59] WARNING: /Users/runner/work/xgboost/xgboost/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.
```

```
=== XGBoost ===
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     85071
           1       0.94      0.90      0.92     85259

    accuracy                           0.92    170330
   macro avg       0.92      0.92      0.92    170330
weighted avg       0.92      0.92      0.92    170330
```

In [32]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Store results
results = []

# Train & evaluate sklearn models
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    y_pred_prob = model.predict_proba(X_test_scaled)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_pred_prob)

    results.append({"Model": name, "Accuracy": acc, "F1-score": f1, "ROC-AUC":



# Convert results to DataFrame
df_results = pd.DataFrame(results)
print(df_results)

# === Visualization ===
plt.figure(figsize=(10,6))
df_melted = df_results.melt(id_vars="Model", var_name="Metric", value_name="Sc

sns.barplot(data=df_melted, x="Model", y="Score", hue="Metric")
plt.title("Model Performance Comparison")
plt.xticks(rotation=30)
plt.ylim(0,1)
plt.show()
```

```
/opt/anaconda3/lib/python3.12/site-packages/xgboost/training.py:183: UserWarnin
g:

[19:27:50] WARNING: /Users/runner/work/xgboost/xgboost/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.
```

```
                 Model  Accuracy  F1-score   ROC-AUC
0  Logistic Regression  0.721893  0.721821  0.800895
1        Random Forest  0.934873  0.934060  0.981420
2    Gradient Boosting  0.876939  0.875495  0.952148
3              XGBoost  0.919832  0.917980  0.974887
```



```
In [ ]:
```