In [31]:

```
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, roc_auc_score
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.metrics import roc_curve
import numpy as np

%matplotlib inline
pd.set_option('display.max_columns', None)

# Set directory for data
DATA_DIR = './data/'
IMAGES_DIR = './images/'

# Ensure the necessary directories exist
os.makedirs(DATA_DIR, exist_ok=True)
os.makedirs(IMAGES_DIR, exist_ok=True)
```

In [32]:

```
def load_data(path=os.path.join(DATA_DIR, 'depression_data.csv')):
    try:
        data = pd.read_csv(path)
        print(f"Data successfully loaded from {path}")
    except FileNotFoundError:
        print(f"File not found at {path}.")
        return None
    return data

def clean_data(data):
    if data.isnull().sum().any():
        print("Missing values detected. Dropping missing values...")
        data_cleaned = data.dropna()
    else:
        print("No missing values detected.")
        data_cleaned = data
    return data_cleaned

# Load the data
raw_data_path = os.path.join(DATA_DIR, "depression_data.csv")
data = load_data(raw_data_path)
cleaned_data = clean_data(data)

# Save cleaned data
cleaned_data_path = os.path.join(DATA_DIR, "cleaned_data.csv")
cleaned_data.to_csv(cleaned_data_path, index=False)
print(f"Cleaned data saved to '{cleaned_data_path}'")
```

```
Data successfully loaded from ./data/depression_data.csv
No missing values detected.
Cleaned data saved to './data/cleaned_data.csv'
```

In [39]:

```
count=cleaned_data['Marital Status'].value_counts()
percent=cleaned_data['Marital Status'].value_counts(normalize=True)*100
FreqTable=pd.DataFrame({'Frequency': count,'Percentage': percent})
FreqTable
```

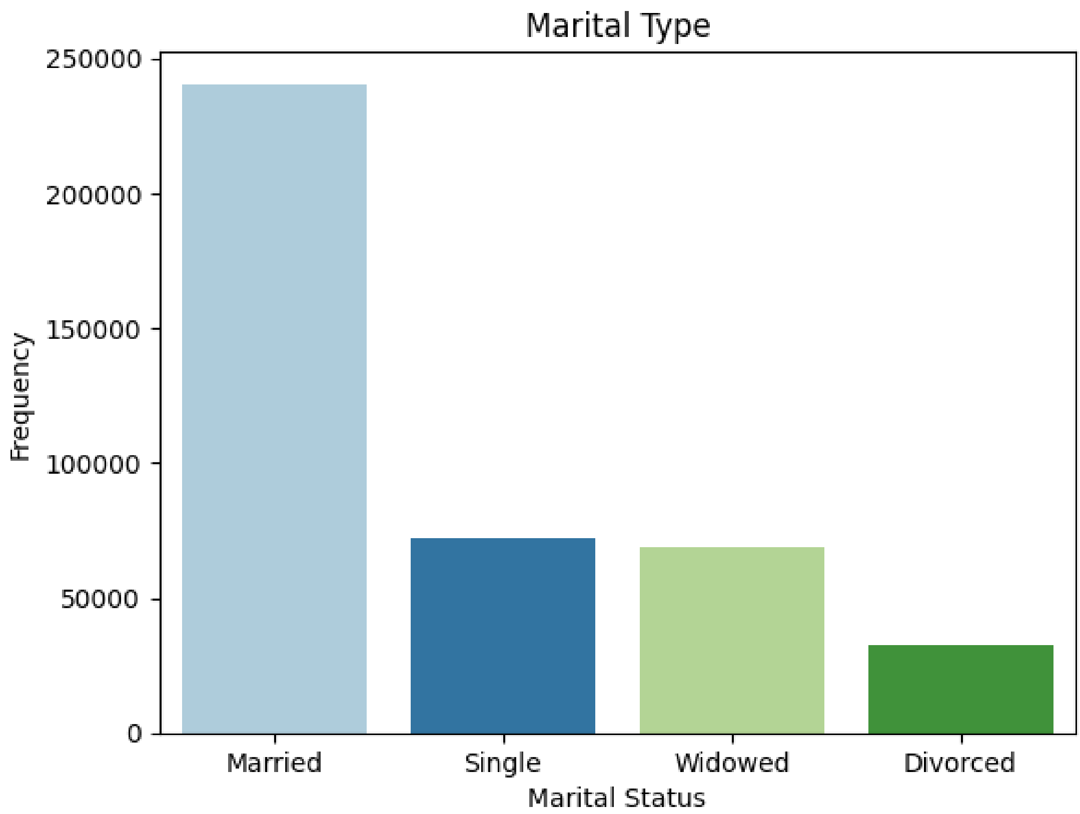
Out[39]:

Marital Status	Frequency	Percentage
1	240444	58.110825
2	72110	17.427641
3	68485	16.551546
0	32729	7.909988

In [40]:

```
sns.barplot(x=['Married','Single','Widowed','Divorced'],y=count, palette=sns.color_palet
te("Paired"))
plt.title('Marital Type')
plt.xlabel('Marital Status')
plt.ylabel('Frequency')
plt.show()
```

```
<ipython-input-40-fb6c43ee07a3>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=['Married','Single','Widowed','Divorced'],y=count, palette=sns.color_pale
tte("Paired"))
<ipython-input-40-fb6c43ee07a3>:1: UserWarning: The palette list has more values (12) tha
n needed (4), which may not be intended.
  sns.barplot(x=['Married','Single','Widowed','Divorced'],y=count, palette=sns.color_pale
tte("Paired"))
```



Most of the People who have depression are Married

In [41]:

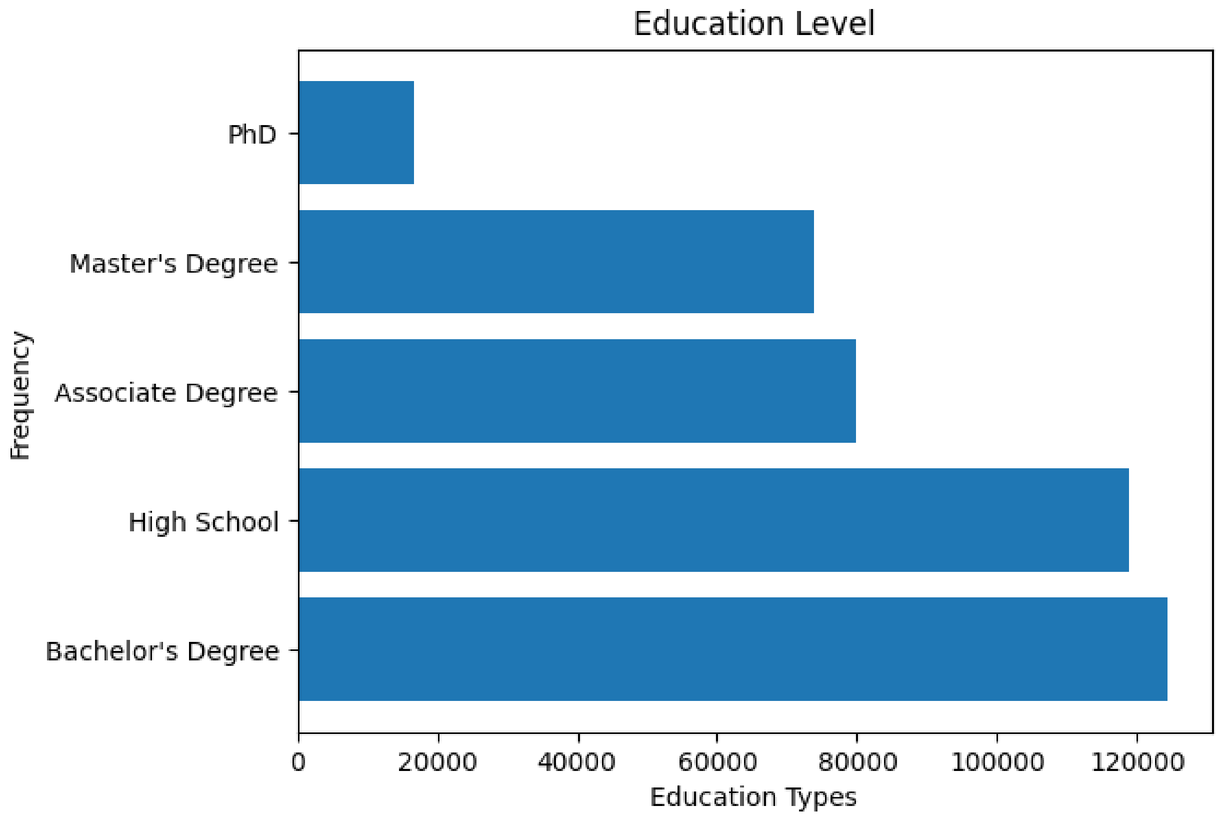
```
count=count=cleaned_data['Education Level'].value_counts()
percent=cleaned_data['Education Level'].value_counts(normalize=True)*100
FreqTable=pd.DataFrame({'Frequency': count,'Percentage': percent})
FreqTable
```

Out[41]:

Education Level	Frequency	Percentage
1	124329	30.047998
2	118927	28.742435
0	79999	19.334265
3	73768	17.828348
4	16745	4.046954

In [42]:

```
plt.barh(["Bachelor's Degree","High School","Associate Degree","Master's Degree", "PhD"],
count,align='center')
plt.title('Education Level')
plt.xlabel('Education Types')
plt.ylabel('Frequency')
plt.show()
```



The people with Bachelor's Degree was the most people who have depression

In [43]:

```
count=cleaned_data['Number of Children'].value_counts()
percent=cleaned_data['Number of Children'].value_counts(normalize=True)*100
FreqTable=pd.DataFrame({'Frequency': count,'Percentage': percent})
FreqTable
```

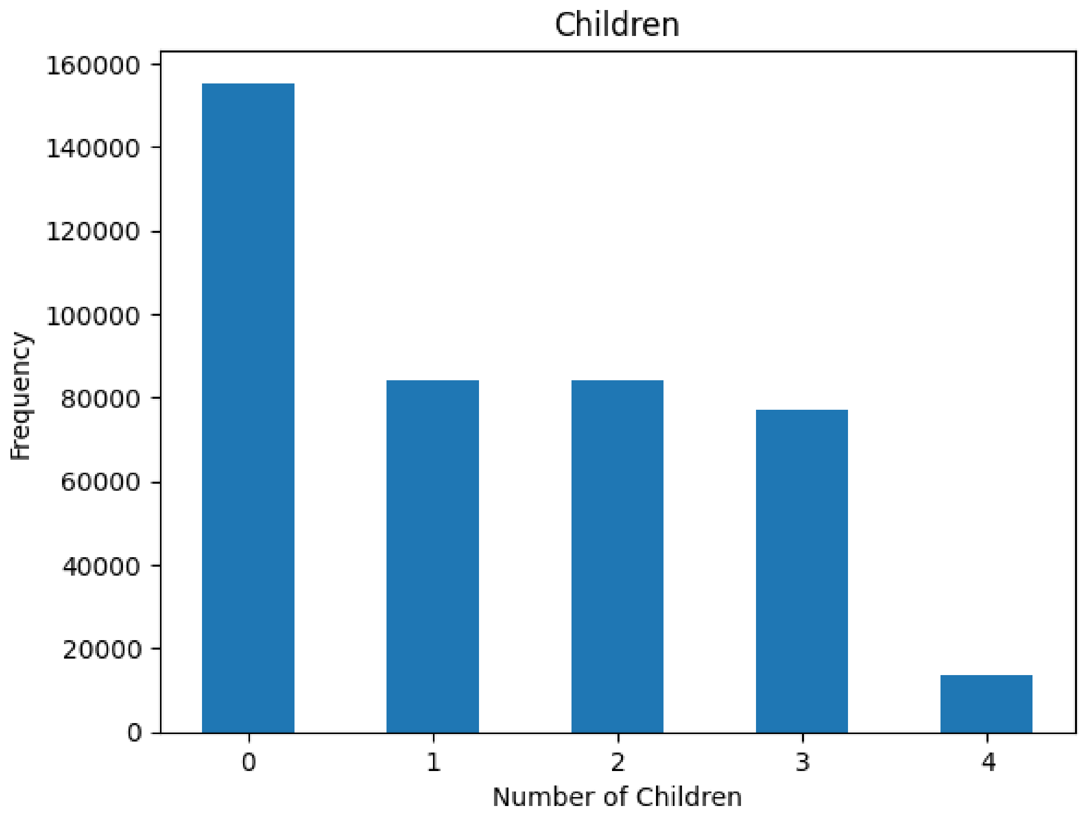
Out[43]:

Number of Children	Frequency	Percentage
0	155232	37.516676
2	83961	20.291806
1	83925	20.283106

Number of Children	Frequency	Percentage
3	76974	18.603179
4	13676	3.305234

In [44]:

```
plt.bar(["0","1","2","3", "4"],count,align='center',width=0.5)
plt.title('Children')
plt.xlabel('Number of Children')
plt.ylabel('Frequency')
plt.show()
```



Most parents who have depression have 0 number of children

In [45]:

```
count=cleaned_data['Smoking Status'].value_counts()
percent=cleaned_data['Smoking Status'].value_counts(normalize=True)*100
FreqTable=pd.DataFrame({'Frequency': count,'Percentage': percent})
FreqTable
```

Out[45]:

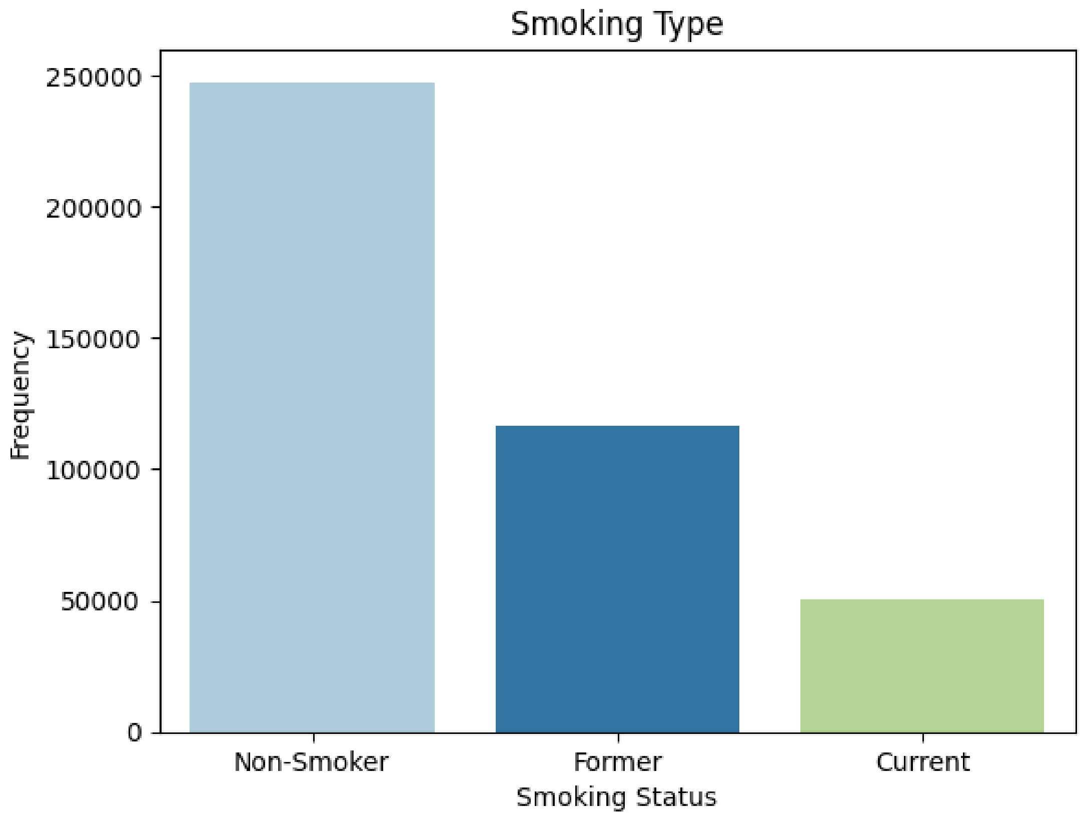
Smoking Status	Frequency	Percentage
2	247416	59.795828
1	116184	28.079503
0	50168	12.124669

In [46]:

```
sns.barplot(x=['Non-Smoker','Former','Current'],y=count, palette=sns.color_palette("Pair
ed"))
plt.title('Smoking Type')
plt.xlabel('Smoking Status')
plt.ylabel('Frequency')
plt.show()
```

```
<ipython-input-46-21925b214013>:1: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=['Non-Smoker','Former','Current'],y=count, palette=sns.color_palette("Paired"))
<ipython-input-46-21925b214013>:1: UserWarning: The palette list has more values (12) than needed (3), which may not be intended.
  sns.barplot(x=['Non-Smoker','Former','Current'],y=count, palette=sns.color_palette("Paired"))
```



Most of the People who have depression are Non-Smokers

In [47]:

```python
count=cleaned_data['Physical Activity Level'].value_counts()
percent=cleaned_data['Physical Activity Level'].value_counts(normalize=True)*100
FreqTable=pd.DataFrame({'Frequency': count,'Percentage': percent})
FreqTable
```

Out[47]:

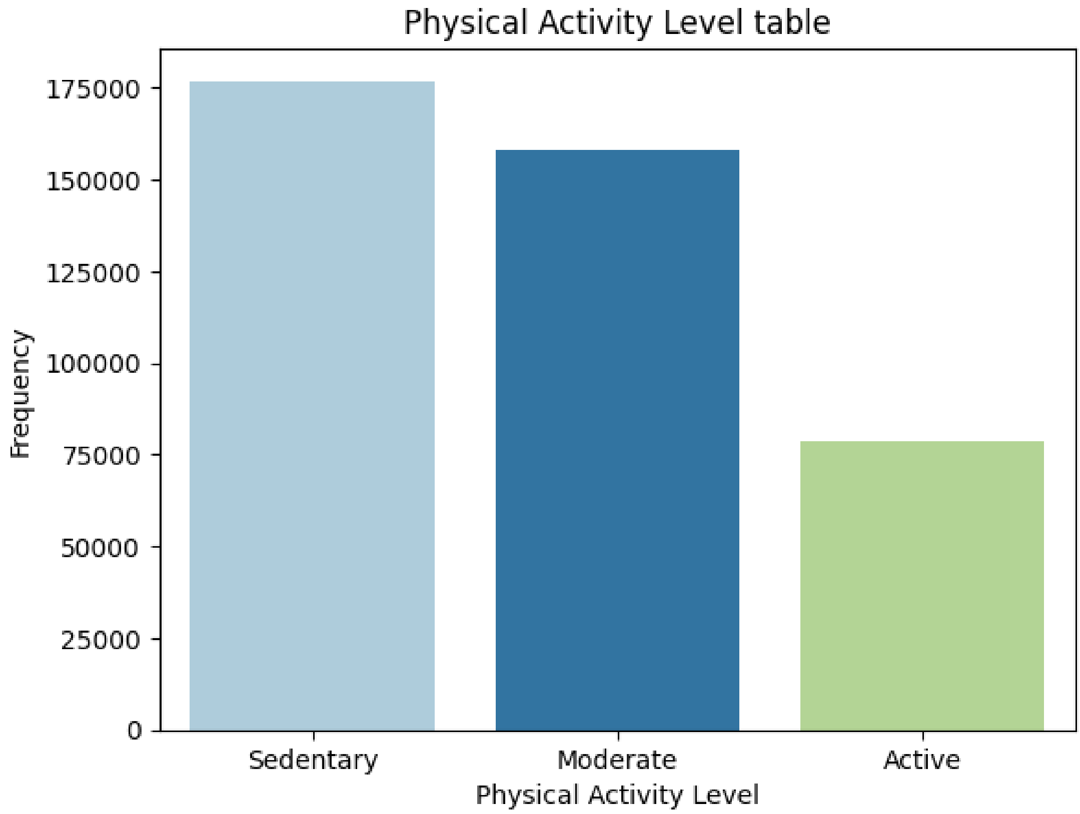
Physical Activity Level	Frequency	Percentage
Sedentary	176850	42.741343
Moderate	158013	38.188792
Active	78905	19.069865

In [48]:

```python
sns.barplot(x=['Sedentary','Moderate','Active'],y=count,palette=sns.color_palette("Paired"))
plt.title("Physical Activity Level table")
plt.xlabel("Physical Activity Level")
plt.ylabel("Frequency")
plt.show()
```

```
<ipython-input-48-984843667e01>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
  sns.barplot(x=['Sedentary','Moderate','Active'],y=count,palette=sns.color_palette("Pair
ed"))
<ipython-input-48-984843667e01>:1: UserWarning: The palette list has more values (12) tha
n needed (3), which may not be intended.
  sns.barplot(x=['Sedentary','Moderate','Active'],y=count,palette=sns.color_palette("Pair
ed"))
```



Most of the people who have depression are Sedentary

In [49]:

```
cleaned_data['Income'].describe()
```

Out[49]:

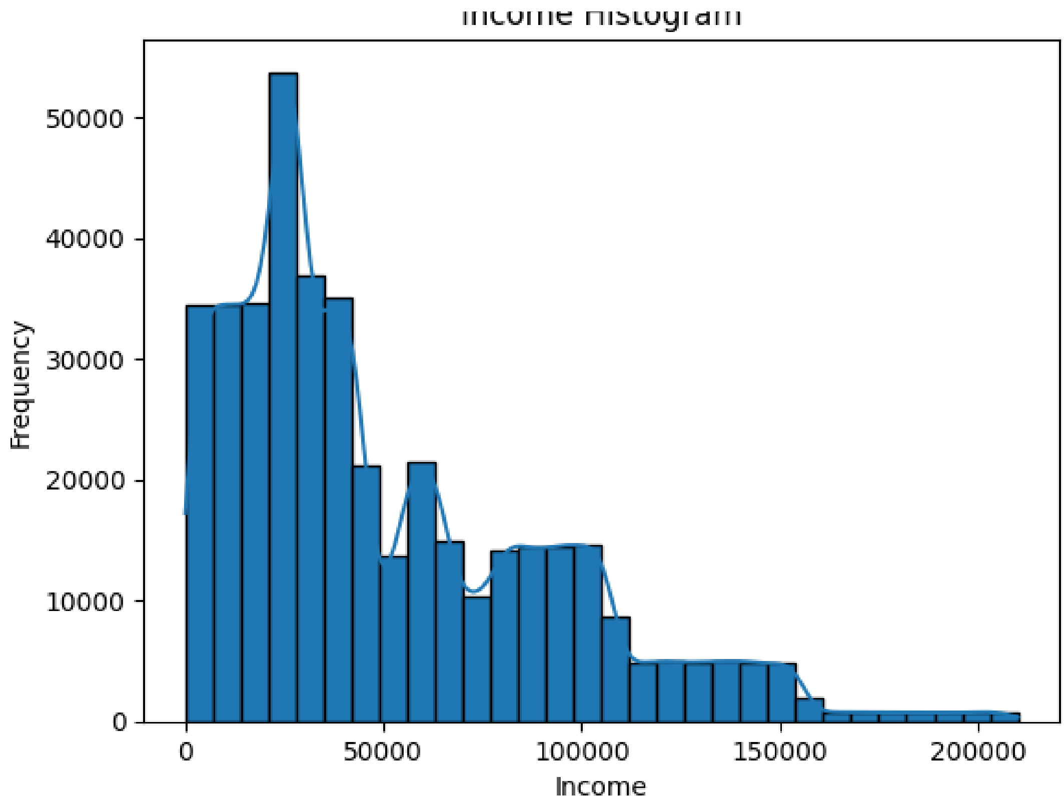
	Income
count	413768.000000
mean	50661.707971
std	40624.100565
min	0.410000
25%	21001.030000
50%	37520.135000
75%	76616.300000
max	209995.220000

dtype: float64

In [50]:

```
sns.histplot(cleaned_data['Income'],bins=30,kde=True,linewidth=1,alpha=1, fill=True)
plt.title('Income Histogram')
plt.xlabel("Income")
plt.ylabel('Frequency')
plt.show()
```

Income Histogram



Most people who have depression have Income that is less than 50000

In [33]:

```
def plot_correlation_matrix(X):
    correlation_matrix = X.corr()
    plt.figure(figsize=(12, 10))
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0
.5)
    plt.title("Correlation Matrix of Features")
    plt.savefig(os.path.join(IMAGES_DIR, "correlation_matrix.png"))
    plt.close()

    # Print highly correlated pairs
    high_corr_pairs = [(feature1, feature2) for feature1 in correlation_matrix.columns
                       for feature2 in correlation_matrix.columns
                       if feature1 != feature2 and abs(correlation_matrix.loc[feature1,
feature2]) > 0.8]
    print("Highly correlated feature pairs (correlation > 0.8):")
    for pair in high_corr_pairs:
        print(pair)

# Exploratory Analysis
def exploratory_analysis(data):
    print("Performing exploratory data analysis...")
    print("Basic Statistics:")
    print(data.describe(include='all'))  # Include all data types for summary
    plot_correlation_matrix(data.select_dtypes(include=[np.number]))  # Only plot numeric
al features

# Perform EDA
exploratory_analysis(cleaned_data)
```

```
Performing exploratory data analysis...
Basic Statistics:
                 Name            Age Marital Status     Education Level  \
count          413768  413768.000000         413768              413768
unique         196851            NaN              4                   5
top     Michael Smith            NaN        Married  Bachelor's Degree
freq              198            NaN         240444              124329
mean              NaN      49.000713            NaN                 NaN
std               NaN      18.158759            NaN                 NaN
min               NaN      18.000000            NaN                 NaN
25%               NaN      33.000000            NaN                 NaN
```

```
50%                NaN      49.000000                NaN                 NaN
75%                NaN      65.000000                NaN                 NaN
max                NaN      80.000000                NaN                 NaN

        Number of Children Smoking Status Physical Activity Level  \
count        413768.000000         413768                  413768
unique                 NaN              3                       3
top                    NaN     Non-smoker               Sedentary
freq                   NaN         247416                  176850
mean              1.298972            NaN                     NaN
std               1.237054            NaN                     NaN
min               0.000000            NaN                     NaN
25%               0.000000            NaN                     NaN
50%               1.000000            NaN                     NaN
75%               2.000000            NaN                     NaN
max               4.000000            NaN                     NaN

       Employment Status         Income Alcohol Consumption Dietary Habits  \
count             413768  413768.000000              413768         413768
unique                 2            NaN                   3              3
top             Employed            NaN            Moderate      Unhealthy
freq              265659            NaN              173440         170817
mean                 NaN   50661.707971                 NaN            NaN
std                  NaN   40624.100565                 NaN            NaN
min                  NaN       0.410000                 NaN            NaN
25%                  NaN   21001.030000                 NaN            NaN
50%                  NaN   37520.135000                 NaN            NaN
75%                  NaN   76616.300000                 NaN            NaN
max                  NaN  209995.220000                 NaN            NaN

       Sleep Patterns History of Mental Illness History of Substance Abuse  \
count          413768                    413768                     413768
unique              3                         2                          2
top              Fair                        No                         No
freq           196789                    287943                     284880
mean              NaN                       NaN                        NaN
std               NaN                       NaN                        NaN
min               NaN                       NaN                        NaN
25%               NaN                       NaN                        NaN
50%               NaN                       NaN                        NaN
75%               NaN                       NaN                        NaN
max               NaN                       NaN                        NaN

       Family History of Depression Chronic Medical Conditions
count                        413768                     413768
unique                            2                          2
top                              No                         No
freq                         302515                     277561
mean                            NaN                        NaN
std                             NaN                        NaN
min                             NaN                        NaN
25%                             NaN                        NaN
50%                             NaN                        NaN
75%                             NaN                        NaN
max                             NaN                        NaN
Highly correlated feature pairs (correlation > 0.8):
```

In [34]:

```python
def encode_features(data):
    categorical_columns = ['Marital Status', 'Education Level', 'Smoking Status',
                           'Employment Status', 'Sleep Patterns', 'Dietary Habits']
    for col in categorical_columns:
        if col in data.columns:
            data[col] = data[col].astype('category').cat.codes
    return data

def select_features(data):
    data = encode_features(data)
    features = ['Age', 'Income', 'Education Level', 'Family History of Depression', 'Sle
ep Patterns']
```

```
    target = 'History of Mental Illness'
    X = data[features]
    y = data[target].map({'No': 0, 'Yes': 1})
    return X, y

# Select features
X, y = select_features(cleaned_data)
```

In [35]:

```
# Convert categorical target variable to numeric
data['History of Mental Illness'] = data['History of Mental Illness'].map({'No': 0, 'Yes'
: 1})

# Drop the Name column
data = data.drop(columns=['Name'], errors='ignore')

# Split features and target variable
X = data.drop('History of Mental Illness', axis=1)
y = data['History of Mental Illness']  # This should now be numeric

# Check if the dataset is empty
if X.empty or y.empty:
    print("Feature set or target variable is empty after cleaning. Exiting.")
    exit()

# Encode categorical variables in X
categorical_cols_to_encode = X.select_dtypes(include=['object']).columns.tolist()
X_encoded = pd.get_dummies(X[categorical_cols_to_encode], drop_first=True)
X = X.drop(categorical_cols_to_encode, axis=1).join(X_encoded)

# Ensure all columns are numeric
X = X.apply(pd.to_numeric, errors='coerce')
X.dropna(inplace=True)
y = y[X.index]  # Ensure y matches the index of X

# Class distribution before SMOTE
print("Class distribution before SMOTE:")
print(y.value_counts())

# Resampling using SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Class distribution after SMOTE
print("Class distribution after SMOTE:")
print(pd.Series(y_resampled).value_counts())

# Feature scaling
scaler = StandardScaler()
X_resampled = scaler.fit_transform(X_resampled)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=
0.2, random_state=42)
```

```
Class distribution before SMOTE:
History of Mental Illness
0    287943
1    125825
Name: count, dtype: int64
Class distribution after SMOTE:
History of Mental Illness
1    287943
0    287943
Name: count, dtype: int64
```

In [36]:

```
# Define the function to train and evaluate models
def train_and_evaluate_models(X_train, y_train, X_test, y_test):
```

```
    lr_model = LogisticRegression(solver='liblinear')
    lr_model.fit(X_train, y_train)

    rf_model = RandomForestClassifier()
    rf_model.fit(X_train, y_train)

    xgb_model = XGBClassifier(eval_metric='logloss', use_label_encoder=False)
    xgb_model.fit(X_train, y_train)

    return lr_model, rf_model, xgb_model

# Train and evaluate models
print("Training and evaluating models...")
lr_model, rf_model, xgb_model = train_and_evaluate_models(X_train, y_train, X_test, y_te
st)
```

```
Training and evaluating models...
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [10:18:39] WARN
ING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

In [37]:

```
def plot_roc_curve(model, X_test, y_test, model_name):
    y_prob = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.figure()
    plt.plot(fpr, tpr, label='ROC Curve')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve for {model_name}')
    plt.savefig(os.path.join(IMAGES_DIR, f'roc_curve_{model_name}.png'))
    plt.close()

# Plot ROC curves
plot_roc_curve(lr_model, X_test, y_test, 'Logistic Regression')
plot_roc_curve(rf_model, X_test, y_test, 'Random Forest')
plot_roc_curve(xgb_model, X_test, y_test, 'XGBoost')
```

In [38]:

```
def print_classification_report(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    print(f"Classification report for {model_name}:")
    print(classification_report(y_test, y_pred))

# Print classification reports
print_classification_report(lr_model, X_test, y_test, 'Logistic Regression')
print_classification_report(rf_model, X_test, y_test, 'Random Forest')
print_classification_report(xgb_model, X_test, y_test, 'XGBoost')
```

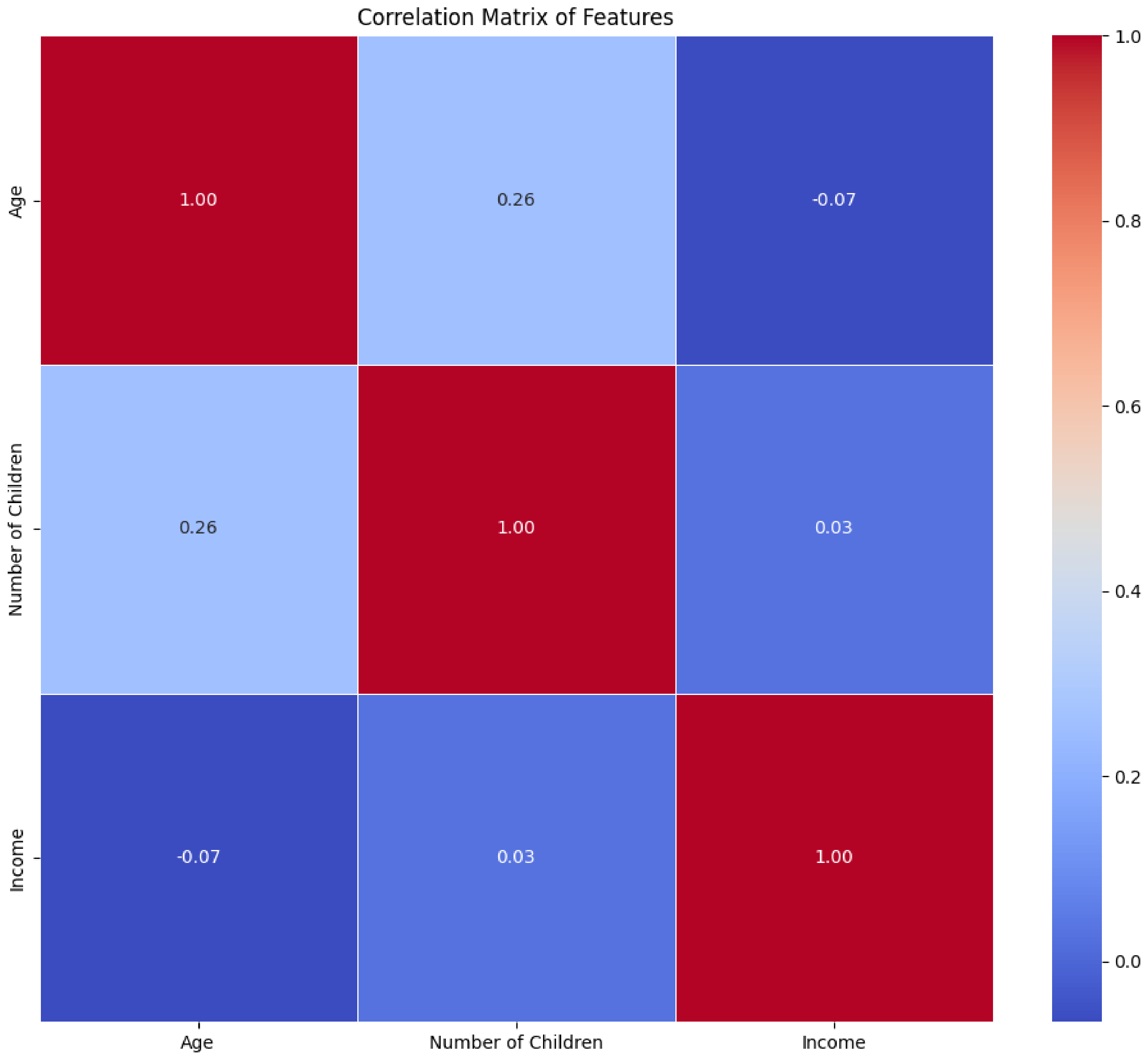
```
Classification report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.67      0.71      0.69     57430
           1       0.69      0.65      0.67     57748

    accuracy                           0.68    115178
   macro avg       0.68      0.68      0.68    115178
weighted avg       0.68      0.68      0.68    115178

Classification report for Random Forest:
              precision    recall  f1-score   support

           0       0.68      0.78      0.73     57430
           1       0.74      0.64      0.69     57748

    accuracy                           0.71    115178
   macro avg       0.71      0.71      0.71    115178
```

```
weighted avg       0.71      0.71      0.71    115178

Classification report for XGBoost:
              precision    recall  f1-score   support

           0       0.67      0.89      0.76     57430
           1       0.83      0.57      0.67     57748

    accuracy                           0.73    115178
   macro avg       0.75      0.73      0.72    115178
weighted avg       0.75      0.73      0.72    115178
```

In [30]:

```
Data successfully loaded from ./data/depression_data.csv
No missing values detected.
Cleaned data saved to './data/cleaned_data.csv'
Performing exploratory data analysis...
Basic Statistics:
                 Name            Age Marital Status    Education Level  \
count          413768  413768.000000         413768             413768
unique         196851            NaN              4                  5
top     Michael Smith            NaN        Married  Bachelor's Degree
freq              198            NaN         240444             124329
mean              NaN      49.000713            NaN                NaN
std               NaN      18.158759            NaN                NaN
min               NaN      18.000000            NaN                NaN
25%               NaN      33.000000            NaN                NaN
50%               NaN      49.000000            NaN                NaN
75%               NaN      65.000000            NaN                NaN
max               NaN      80.000000            NaN                NaN

        Number of Children Smoking Status Physical Activity Level  \
count        413768.000000         413768                  413768
unique                 NaN              3                       3
top                    NaN     Non-smoker               Sedentary
freq                   NaN         247416                  176850
mean              1.298972            NaN                     NaN
std               1.237054            NaN                     NaN
min               0.000000            NaN                     NaN
25%               0.000000            NaN                     NaN
50%               1.000000            NaN                     NaN
75%               2.000000            NaN                     NaN
max               4.000000            NaN                     NaN

       Employment Status         Income Alcohol Consumption Dietary Habits  \
count             413768  413768.000000              413768         413768
unique                 2            NaN                   3              3
top             Employed            NaN            Moderate      Unhealthy
freq              265659            NaN              173440         170817
mean                 NaN   50661.707971                 NaN            NaN
std                  NaN   40624.100565                 NaN            NaN
min                  NaN       0.410000                 NaN            NaN
25%                  NaN   21001.030000                 NaN            NaN
50%                  NaN   37520.135000                 NaN            NaN
75%                  NaN   76616.300000                 NaN            NaN
max                  NaN  209995.220000                 NaN            NaN

       Sleep Patterns History of Mental Illness History of Substance Abuse  \
count          413768                    413768                     413768
unique              3                         2                          2
top              Fair                        No                         No
freq           196789                    287943                     284880
mean              NaN                       NaN                        NaN
std               NaN                       NaN                        NaN
min               NaN                       NaN                        NaN
25%               NaN                       NaN                        NaN
50%               NaN                       NaN                        NaN
75%               NaN                       NaN                        NaN
```

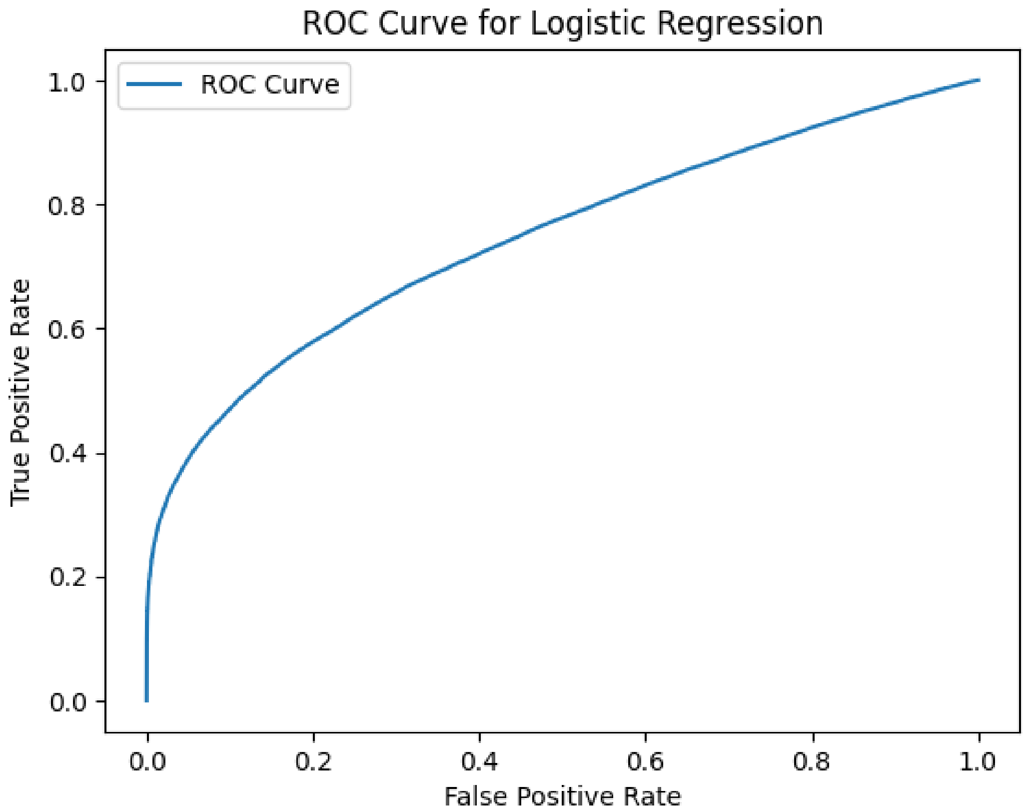
```
max              NaN                        NaN                        NaN

       Family History of Depression Chronic Medical Conditions
count                        413768                     413768
unique                            2                          2
top                              No                         No
freq                         302515                     277561
mean                            NaN                        NaN
std                             NaN                        NaN
min                             NaN                        NaN
25%                             NaN                        NaN
50%                             NaN                        NaN
75%                             NaN                        NaN
max                             NaN                        NaN
```

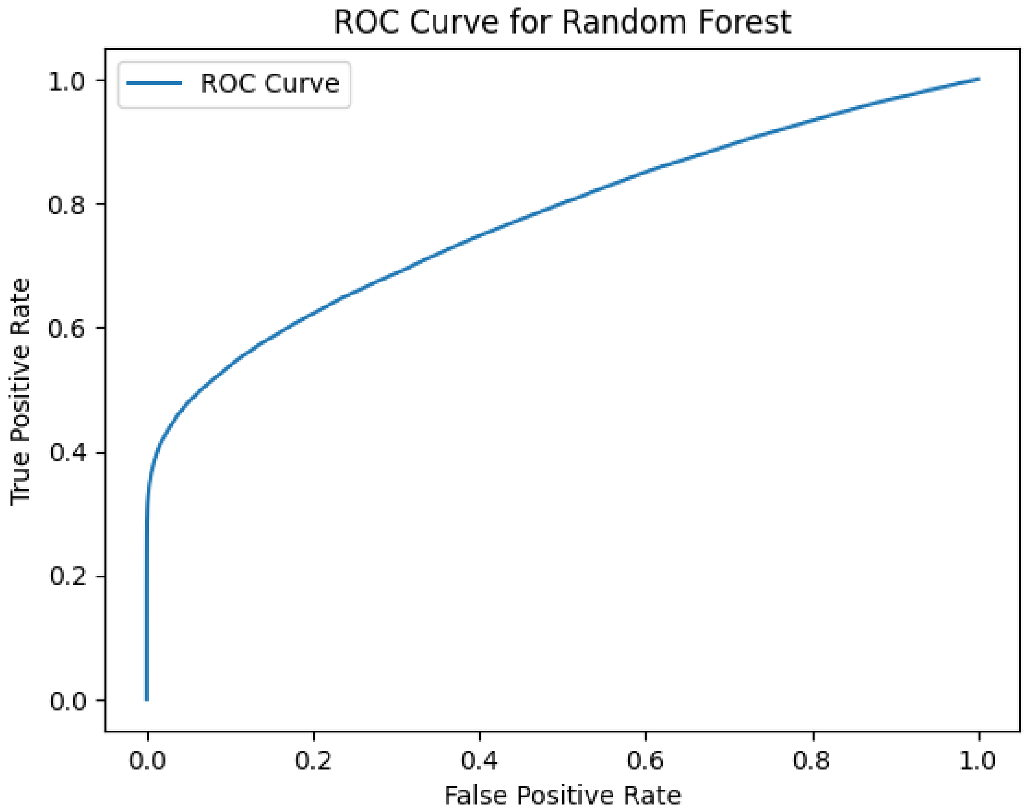


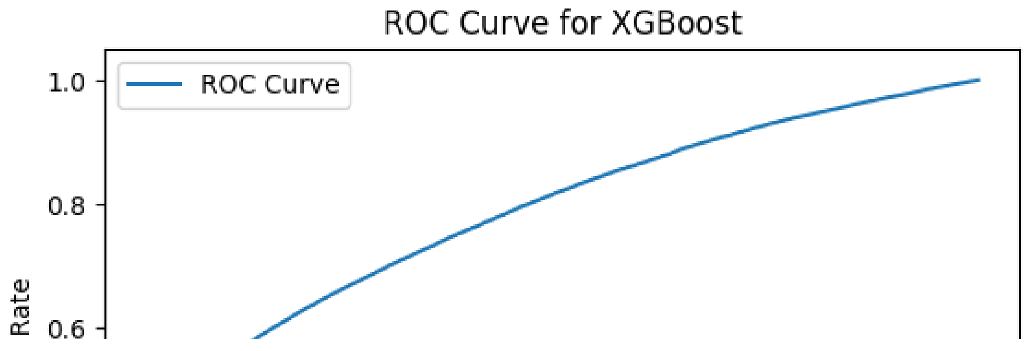

```
Highly correlated feature pairs (correlation > 0.8):
Class distribution before SMOTE:
History of Mental Illness
0    287943
1    125825
Name: count, dtype: int64
Class distribution after SMOTE:
History of Mental Illness
1    287943
0    287943
Name: count, dtype: int64
Training and evaluating models...
```

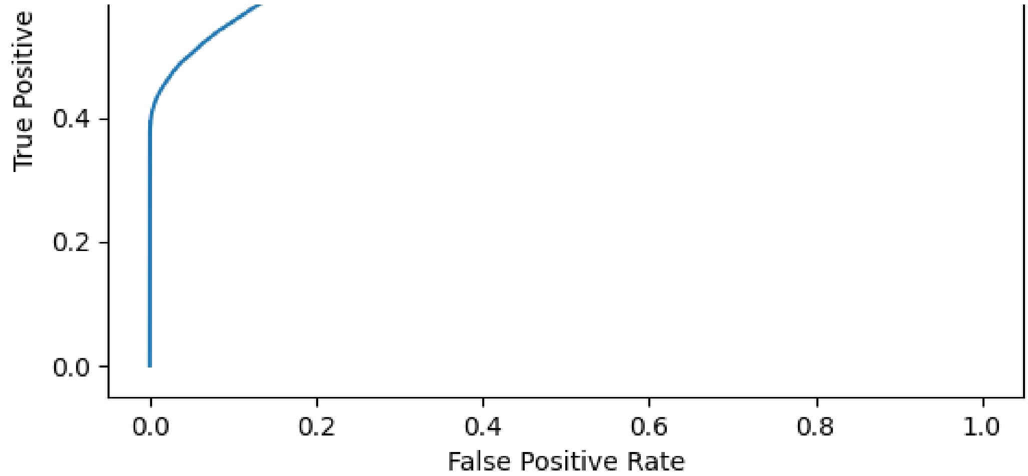
```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [10:14:38] WARN
```

```
ING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```










```
Classification report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.67      0.71      0.69     57430
           1       0.69      0.65      0.67     57748

    accuracy                           0.68    115178
   macro avg       0.68      0.68      0.68    115178
weighted avg       0.68      0.68      0.68    115178

Classification report for Random Forest:
              precision    recall  f1-score   support

           0       0.68      0.78      0.73     57430
           1       0.74      0.64      0.69     57748

    accuracy                           0.71    115178
   macro avg       0.71      0.71      0.71    115178
weighted avg       0.71      0.71      0.71    115178

Classification report for XGBoost:
              precision    recall  f1-score   support

           0       0.67      0.89      0.76     57430
           1       0.83      0.57      0.67     57748

    accuracy                           0.73    115178
   macro avg       0.75      0.73      0.72    115178
weighted avg       0.75      0.73      0.72    115178
```

- **Logistic Regression: Achieved 68% accuracy with balanced precision and recall across classes, indicating moderate performance in predicting both classes.**
- **Random Forest: Improved accuracy at 71%, with better recall for class 0, suggesting it effectively identifies non-ill individuals, while still managing reasonable performance for class 1.**
- **XGBoost: Best accuracy at 73%, with high recall for class 0 (89%) but lower recall for class 1 (57%), indicating potential issues in identifying positive cases.**

Overall, XGBoost performed best overall.