In [10]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder, F
from sklearn.compose import ColumnTransformer

from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.dummy import DummyClassifier
from lightgbm import LGBMClassifier
from sklearn.neural_network import MLPClassifier

from imblearn.over_sampling import SMOTE, RandomOverSampler, BorderlineSMOTE, AD
from imblearn.under_sampling import RandomUnderSampler

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_sc
```

Experiments to find best sampling techniques

In [7]:
```
# Load your data
df = pd.read_csv('../data/depression_data.csv')
df = df.drop(columns=['Name'])

# Log scaling for Income (creating this column before splitting features and tar
df['Income'] = df['Income'].apply(lambda x: np.log(x + 1))

# Splitting features and target
X = df.drop(['History of Mental Illness'], axis=1)
y = df['History of Mental Illness'].map({'Yes': 1, 'No': 0})

# Columns setup
categorical_cols = ['Marital Status', 'Education Level', 'Smoking Status', 'Phys
                    'Employment Status', 'Alcohol Consumption', 'Dietary Habits'
                    'History of Substance Abuse', 'Family History of Depression'
numeric_cols = ['Age', 'Number of Children']

# One hot encoding without drop
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
        ('cat', OneHotEncoder(drop=None), categorical_cols)
    ])

# Define models
models = {
    'Dummy': DummyClassifier(strategy='most_frequent'),
    'DecisionTree': DecisionTreeClassifier(),
    'RandomForest': RandomForestClassifier(),
    'LogisticRegression': LogisticRegression(max_iter=1000),
    'LightGBM': LGBMClassifier(),
    'NeuralNetwork': MLPClassifier(hidden_layer_sizes=(64, 32), early_stopping=T
```

```
}

# Sampling techniques
sampling_methods = {
    'None': None,
    'RandomOverSampler': RandomOverSampler(),
    'RandomUnderSampler': RandomUnderSampler(),
    'SMOTE': SMOTE(),
    'BorderlineSMOTE': BorderlineSMOTE(),
    'ADASYN': ADASYN()
}

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

# Apply preprocessing to the training data before resampling
X_train_transformed = preprocessor.fit_transform(X_train)
X_test_transformed = preprocessor.transform(X_test)

# Iterate through each sampling method
for sampling_name, sampler in sampling_methods.items():
    if sampler is not None:
        X_resampled, y_resampled = sampler.fit_resample(X_train_transformed, y_t
    else:
        X_resampled, y_resampled = X_train_transformed, y_train

    print(f"\nSampling Technique: {sampling_name}")

    # Iterate through each model
    for model_name, model in models.items():
        model.fit(X_resampled, y_resampled)
        y_pred = model.predict(X_test_transformed)

        # Print metrics
        print(f"\nModel: {model_name}")
        print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
        print(f"Precision: {precision_score(y_test, y_pred, zero_division=1):.4f
        print(f"Recall: {recall_score(y_test, y_pred, zero_division=1):.4f}")
        print(f"F1 Score: {f1_score(y_test, y_pred, zero_division=1):.4f}")
        print("Classification Report:")
        print(classification_report(y_test, y_pred, zero_division=1))
```

```
Sampling Technique: None

Model: Dummy
Accuracy: 0.6954
Precision: 1.0000
Recall: 0.0000
F1 Score: 0.0000
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       1.00      0.00      0.00     37812

    accuracy                           0.70    124131
   macro avg       0.85      0.50      0.41    124131
weighted avg       0.79      0.70      0.57    124131


Model: DecisionTree
Accuracy: 0.5967
Precision: 0.3252
Recall: 0.3014
F1 Score: 0.3129
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.73      0.71     86319
           1       0.33      0.30      0.31     37812

    accuracy                           0.60    124131
   macro avg       0.51      0.51      0.51    124131
weighted avg       0.59      0.60      0.59    124131


Model: RandomForest
Accuracy: 0.6186
Precision: 0.3323
Recall: 0.2498
F1 Score: 0.2852
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.78      0.74     86319
           1       0.33      0.25      0.29     37812

    accuracy                           0.62    124131
   macro avg       0.52      0.51      0.51    124131
weighted avg       0.59      0.62      0.60    124131


Model: LogisticRegression
Accuracy: 0.6954
Precision: 1.0000
Recall: 0.0000
F1 Score: 0.0000
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       1.00      0.00      0.00     37812
```

```
    accuracy                           0.70    124131
   macro avg       0.85      0.50      0.41    124131
weighted avg       0.79      0.70      0.57    124131

[LightGBM] [Info] Number of positive: 88013, number of negative: 201624
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.006549 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 134
[LightGBM] [Info] Number of data points in the train set: 289637, number of used
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.303873 -> initscore=-0.828920
[LightGBM] [Info] Start training from score -0.828920

Model: LightGBM
Accuracy: 0.6954
Precision: 0.5000
Recall: 0.0001
F1 Score: 0.0002
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       0.50      0.00      0.00     37812

    accuracy                           0.70    124131
   macro avg       0.60      0.50      0.41    124131
weighted avg       0.64      0.70      0.57    124131


Model: NeuralNetwork
Accuracy: 0.6953
Precision: 0.3043
Recall: 0.0002
F1 Score: 0.0004
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       0.30      0.00      0.00     37812

    accuracy                           0.70    124131
   macro avg       0.50      0.50      0.41    124131
weighted avg       0.58      0.70      0.57    124131


Sampling Technique: RandomOverSampler

Model: Dummy
Accuracy: 0.6954
Precision: 1.0000
Recall: 0.0000
F1 Score: 0.0000
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       1.00      0.00      0.00     37812
```

```
    accuracy                           0.70    124131
   macro avg       0.85      0.50      0.41    124131
weighted avg       0.79      0.70      0.57    124131


Model: DecisionTree
Accuracy: 0.5789
Precision: 0.3230
Recall: 0.3489
F1 Score: 0.3354
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.68      0.69     86319
           1       0.32      0.35      0.34     37812

    accuracy                           0.58    124131
   macro avg       0.51      0.51      0.51    124131
weighted avg       0.59      0.58      0.58    124131


Model: RandomForest
Accuracy: 0.5877
Precision: 0.3277
Recall: 0.3362
F1 Score: 0.3319
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.70      0.70     86319
           1       0.33      0.34      0.33     37812

    accuracy                           0.59    124131
   macro avg       0.52      0.52      0.52    124131
weighted avg       0.59      0.59      0.59    124131


Model: LogisticRegression
Accuracy: 0.6168
Precision: 0.3898
Recall: 0.4562
F1 Score: 0.4204
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.69      0.71     86319
           1       0.39      0.46      0.42     37812

    accuracy                           0.62    124131
   macro avg       0.57      0.57      0.57    124131
weighted avg       0.64      0.62      0.62    124131

[LightGBM] [Info] Number of positive: 201624, number of negative: 201624
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.008796 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 134
[LightGBM] [Info] Number of data points in the train set: 403248, number of used
```

```
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

Model: LightGBM
Accuracy: 0.5860
Precision: 0.3740
Recall: 0.5332
F1 Score: 0.4396
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.61      0.67     86319
           1       0.37      0.53      0.44     37812

    accuracy                           0.59    124131
   macro avg       0.56      0.57      0.56    124131
weighted avg       0.63      0.59      0.60    124131


Model: NeuralNetwork
Accuracy: 0.5610
Precision: 0.3621
Recall: 0.5793
F1 Score: 0.4456
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.55      0.64     86319
           1       0.36      0.58      0.45     37812

    accuracy                           0.56    124131
   macro avg       0.56      0.57      0.54    124131
weighted avg       0.63      0.56      0.58    124131


Sampling Technique: RandomUnderSampler

Model: Dummy
Accuracy: 0.6954
Precision: 1.0000
Recall: 0.0000
F1 Score: 0.0000
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       1.00      0.00      0.00     37812

    accuracy                           0.70    124131
   macro avg       0.85      0.50      0.41    124131
weighted avg       0.79      0.70      0.57    124131


Model: DecisionTree
Accuracy: 0.5261
Precision: 0.3192
Recall: 0.4904
F1 Score: 0.3867
Classification Report:
              precision    recall  f1-score   support
```

```
           0       0.71      0.54      0.61     86319
           1       0.32      0.49      0.39     37812

    accuracy                           0.53    124131
   macro avg       0.51      0.52      0.50    124131
weighted avg       0.59      0.53      0.54    124131


Model: RandomForest
Accuracy: 0.5266
Precision: 0.3274
Recall: 0.5254
F1 Score: 0.4034
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.53      0.61     86319
           1       0.33      0.53      0.40     37812

    accuracy                           0.53    124131
   macro avg       0.52      0.53      0.51    124131
weighted avg       0.60      0.53      0.55    124131


Model: LogisticRegression
Accuracy: 0.6168
Precision: 0.3898
Recall: 0.4562
F1 Score: 0.4204
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.69      0.71     86319
           1       0.39      0.46      0.42     37812

    accuracy                           0.62    124131
   macro avg       0.57      0.57      0.57    124131
weighted avg       0.64      0.62      0.62    124131

[LightGBM] [Info] Number of positive: 88013, number of negative: 88013
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.005225 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 134
[LightGBM] [Info] Number of data points in the train set: 176026, number of used
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

Model: LightGBM
Accuracy: 0.5802
Precision: 0.3708
Recall: 0.5426
F1 Score: 0.4405
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.60      0.66     86319
           1       0.37      0.54      0.44     37812
```

```
    accuracy                           0.58    124131
   macro avg       0.56      0.57      0.55    124131
weighted avg       0.63      0.58      0.60    124131


Model: NeuralNetwork
Accuracy: 0.6143
Precision: 0.3880
Recall: 0.4611
F1 Score: 0.4214
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.68      0.71     86319
           1       0.39      0.46      0.42     37812

    accuracy                           0.61    124131
   macro avg       0.57      0.57      0.57    124131
weighted avg       0.63      0.61      0.62    124131


Sampling Technique: SMOTE

Model: Dummy
Accuracy: 0.6954
Precision: 1.0000
Recall: 0.0000
F1 Score: 0.0000
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       1.00      0.00      0.00     37812

    accuracy                           0.70    124131
   macro avg       0.85      0.50      0.41    124131
weighted avg       0.79      0.70      0.57    124131


Model: DecisionTree
Accuracy: 0.5904
Precision: 0.3240
Recall: 0.3174
F1 Score: 0.3207
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.71      0.71     86319
           1       0.32      0.32      0.32     37812

    accuracy                           0.59    124131
   macro avg       0.51      0.51      0.51    124131
weighted avg       0.59      0.59      0.59    124131


Model: RandomForest
Accuracy: 0.5976
Precision: 0.3322
Recall: 0.3177
```

```
F1 Score: 0.3248
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.72      0.71     86319
           1       0.33      0.32      0.32     37812

    accuracy                           0.60    124131
   macro avg       0.52      0.52      0.52    124131
weighted avg       0.59      0.60      0.60    124131


Model: LogisticRegression
Accuracy: 0.6166
Precision: 0.3897
Recall: 0.4565
F1 Score: 0.4205
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.69      0.71     86319
           1       0.39      0.46      0.42     37812

    accuracy                           0.62    124131
   macro avg       0.57      0.57      0.57    124131
weighted avg       0.64      0.62      0.62    124131

[LightGBM] [Info] Number of positive: 201624, number of negative: 201624
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.032741 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 8670
[LightGBM] [Info] Number of data points in the train set: 403248, number of used
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

Model: LightGBM
Accuracy: 0.6805
Precision: 0.4006
Recall: 0.0987
F1 Score: 0.1584
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.94      0.80     86319
           1       0.40      0.10      0.16     37812

    accuracy                           0.68    124131
   macro avg       0.55      0.52      0.48    124131
weighted avg       0.61      0.68      0.61    124131


Model: NeuralNetwork
Accuracy: 0.5729
Precision: 0.3587
Recall: 0.5105
F1 Score: 0.4213
Classification Report:
              precision    recall  f1-score   support
```

```
           0       0.74      0.60      0.66     86319
           1       0.36      0.51      0.42     37812

    accuracy                           0.57    124131
   macro avg       0.55      0.56      0.54    124131
weighted avg       0.62      0.57      0.59    124131


Sampling Technique: BorderlineSMOTE

Model: Dummy
Accuracy: 0.6954
Precision: 1.0000
Recall: 0.0000
F1 Score: 0.0000
Classification Report:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82     86319
           1       1.00      0.00      0.00     37812

    accuracy                           0.70    124131
   macro avg       0.85      0.50      0.41    124131
weighted avg       0.79      0.70      0.57    124131


Model: DecisionTree
Accuracy: 0.5915
Precision: 0.3250
Recall: 0.3164
F1 Score: 0.3206
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.71      0.71     86319
           1       0.32      0.32      0.32     37812

    accuracy                           0.59    124131
   macro avg       0.51      0.51      0.51    124131
weighted avg       0.59      0.59      0.59    124131


Model: RandomForest
Accuracy: 0.5949
Precision: 0.3286
Recall: 0.3163
F1 Score: 0.3223
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.72      0.71     86319
           1       0.33      0.32      0.32     37812

    accuracy                           0.59    124131
   macro avg       0.52      0.52      0.52    124131
weighted avg       0.59      0.59      0.59    124131


Model: LogisticRegression
```

```
Accuracy: 0.6072
Precision: 0.3840
Recall: 0.4792
F1 Score: 0.4264
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.66      0.70     86319
           1       0.38      0.48      0.43     37812

    accuracy                           0.61    124131
   macro avg       0.56      0.57      0.56    124131
weighted avg       0.63      0.61      0.62    124131

[LightGBM] [Info] Number of positive: 201624, number of negative: 201624
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.033673 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 8670
[LightGBM] [Info] Number of data points in the train set: 403248, number of used
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

Model: LightGBM
Accuracy: 0.6818
Precision: 0.3975
Recall: 0.0865
F1 Score: 0.1421
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.94      0.80     86319
           1       0.40      0.09      0.14     37812

    accuracy                           0.68    124131
   macro avg       0.55      0.51      0.47    124131
weighted avg       0.61      0.68      0.60    124131


Model: NeuralNetwork
Accuracy: 0.5599
Precision: 0.3545
Recall: 0.5418
F1 Score: 0.4285
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.57      0.64     86319
           1       0.35      0.54      0.43     37812

    accuracy                           0.56    124131
   macro avg       0.55      0.55      0.54    124131
weighted avg       0.62      0.56      0.58    124131


Sampling Technique: ADASYN

Model: Dummy
Accuracy: 0.3046
```

```
Precision: 0.3046
Recall: 1.0000
F1 Score: 0.4670
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.00      0.00     86319
           1       0.30      1.00      0.47     37812

    accuracy                           0.30    124131
   macro avg       0.65      0.50      0.23    124131
weighted avg       0.79      0.30      0.14    124131


Model: DecisionTree
Accuracy: 0.5906
Precision: 0.3224
Recall: 0.3121
F1 Score: 0.3172
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.71      0.71     86319
           1       0.32      0.31      0.32     37812

    accuracy                           0.59    124131
   macro avg       0.51      0.51      0.51    124131
weighted avg       0.59      0.59      0.59    124131


Model: RandomForest
Accuracy: 0.5921
Precision: 0.3282
Recall: 0.3237
F1 Score: 0.3259
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.71      0.71     86319
           1       0.33      0.32      0.33     37812

    accuracy                           0.59    124131
   macro avg       0.52      0.52      0.52    124131
weighted avg       0.59      0.59      0.59    124131


Model: LogisticRegression
Accuracy: 0.5850
Precision: 0.3734
Recall: 0.5339
F1 Score: 0.4394
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.61      0.67     86319
           1       0.37      0.53      0.44     37812

    accuracy                           0.59    124131
   macro avg       0.56      0.57      0.56    124131
weighted avg       0.63      0.59      0.60    124131
```

```
[LightGBM] [Info] Number of positive: 209568, number of negative: 201624
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.034841 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 8670
[LightGBM] [Info] Number of data points in the train set: 411192, number of used
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.509660 -> initscore=0.038644
[LightGBM] [Info] Start training from score 0.038644

Model: LightGBM
Accuracy: 0.6803
Precision: 0.3993
Recall: 0.0982
F1 Score: 0.1577
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.94      0.80     86319
           1       0.40      0.10      0.16     37812

    accuracy                           0.68    124131
   macro avg       0.55      0.52      0.48    124131
weighted avg       0.61      0.68      0.61    124131


Model: NeuralNetwork
Accuracy: 0.5906
Precision: 0.3645
Recall: 0.4627
F1 Score: 0.4078
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.65      0.69     86319
           1       0.36      0.46      0.41     37812

    accuracy                           0.59    124131
   macro avg       0.55      0.55      0.55    124131
weighted avg       0.62      0.59      0.60    124131
```

Experiment Insights and Takeaways

After conducting multiple experiments with various models and sampling techniques, here is a summary of our findings and the rationale behind our final choice:

1. **Sampling Techniques:**

We experimented with several sampling techniques to address the class imbalance in the dataset, namely:

- No sampling (baseline)
- Random Over-Sampling
- Random Under-Sampling

- SMOTE (Synthetic Minority Over-sampling Technique)
- BorderlineSMOTE
- ADASYN (Adaptive Synthetic Sampling)

Among these, **Random Over-Sampling** and **SMOTE** provided the best results in terms of model performance, especially when paired with Logistic Regression and Random Forest models. **SMOTE** performed slightly better than Random Over-Sampling in recall and F1 scores. Additionally, SMOTE generates synthetic samples of the minority class rather than duplicating data, which reduces the risk of overfitting.

2. **Model Performance Across Techniques:**

- **Dummy Classifier**:
  - As expected, the dummy classifier provided the baseline accuracy (~69%) but did not capture any of the minority class, resulting in an F1 score of 0. This highlighted the importance of balancing techniques to improve predictive power.
- **Decision Tree**:
  - The Decision Tree performed better than the dummy classifier, but its performance was still suboptimal. With **Random Over-Sampling**, it achieved an F1 score of ~0.33. However, its performance improved marginally with SMOTE to an F1 score of ~0.32. The decision tree's performance remained lower compared to other models, as it often struggles with class imbalance.
- **Random Forest**:
  - Random Forest models improved over the Decision Tree, especially with SMOTE and Random Over-Sampling. The best results were achieved with **SMOTE**, where the F1 score reached ~0.32. However, it also exhibited some limitations in handling class imbalance, especially when undersampling was used.
- **Logistic Regression**:
  - Logistic Regression consistently performed well across all sampling techniques. Its performance improved the most with **SMOTE**, reaching an F1 score of ~0.42. This model balanced simplicity with interpretability, making it a strong candidate for the final solution.
- **LightGBM**:
  - LightGBM models performed similarly across sampling methods but exhibited very low recall for the minority class. Despite a high precision, LightGBM struggled to predict the minority class (history of mental illness) effectively. Its best F1 score was observed with **SMOTE** (~0.16).
- **Neural Network**:
  - The neural network model showed promise with F1 scores of ~0.42 using SMOTE and ~0.44 with Random Over-Sampling, but it was computationally more expensive. Given that it did not outperform simpler models like Logistic Regression, it was not chosen as the final model.

3. **Final Model Choice: Logistic Regression + SMOTE**

The Logistic Regression model with **SMOTE** consistently provided the best balance between precision, recall, and F1 scores. Here's a quick breakdown:

- **Accuracy**: ~61.68%
- **Precision**: ~38.97%
- **Recall**: ~45.65%
- **F1 Score**: ~42.05%

While Random Over-Sampling provided comparable results, we opted for **SMOTE** as the final sampling technique due to its ability to generate synthetic samples rather than duplicating data. This choice reduces the risk of overfitting, which can occur when oversampling simply duplicates minority class instances.

4. **Rationale for SMOTE Over Random Over-Sampling**

Though both Random Over-Sampling and SMOTE performed similarly, SMOTE provides a more generalized and robust approach. Random Over-Sampling can lead to overfitting, as it creates exact copies of minority class instances, which can cause the model to memorize the duplicated data points rather than generalize well to unseen data. SMOTE, on the other hand, generates synthetic instances that blend the characteristics of real minority class instances, providing a more balanced and diverse representation of the minority class without overfitting.

Conclusion

For our final solution, we will move forward with **SMOTE**. This gives us a good balance between performance and simplicity, with the added benefit of reducing overfitting risks.

Experiments to find best model and parameters

In [12]:
```
# Load data
df = pd.read_csv('../data/depression_data.csv')
df = df.drop(columns=['Name'])

# Splitting features and target
X = df.drop(['History of Mental Illness'], axis=1)
y = df['History of Mental Illness'].map({'Yes': 1, 'No': 0})

# Columns setup
categorical_cols = ['Marital Status', 'Education Level', 'Smoking Status', 'Phys
                    'Employment Status', 'Alcohol Consumption', 'Dietary Habits'
                    'History of Substance Abuse', 'Family History of Depression'
numeric_cols = ['Age', 'Number of Children']

# Log scaling for Income
df['Income'] = df['Income'].apply(lambda x: np.log(x + 1))

# One hot encoding and scaling
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
```

```
        ('cat', OneHotEncoder(drop=None), categorical_cols)
    ])

# Transform the data
X_transformed = preprocessor.fit_transform(X)

# Train-test split (80-20)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=

# Apply SMOTE to balance the dataset
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

# Define models and hyperparameters for GridSearch
model_params = {
    'DecisionTree': {
        'model': DecisionTreeClassifier(random_state=42),
        'params': {
            'classifier__max_depth': [10, 20, 30, None],
            'classifier__min_samples_split': [2, 5, 10]
        }
    },
    'LogisticRegression': {
        'model': LogisticRegression(random_state=42),
        'params': {
            'classifier__max_iter': [100, 250, 500],
            'classifier__C': [0.01, 0.1, 1, 10],
            'classifier__penalty': ['l1', 'l2'],
            'classifier__solver': ['liblinear', 'saga']
        }
    },
    'LightGBM': {
        'model': LGBMClassifier(random_state=42),
        'params': {
            'classifier__n_estimators': [50, 100, 200],
            'classifier__max_depth': [10, 20, 30, None],
            'classifier__learning_rate': [0.01, 0.05, 0.1]
        }
    },
    'RandomForest': {
        'model': RandomForestClassifier(random_state=42),
        'params': {
            'classifier__n_estimators': [50, 100, 150],
            'classifier__max_depth': [10, 20, 30, None],
            'classifier__min_samples_split': [2, 5, 10]
        }
    }
}


# Iterate through each model, apply GridSearchCV and evaluate
for model_name, mp in model_params.items():
    print(f"\nModel: {model_name}")

    # Create pipeline for model with classifier (preprocessor is already applied
    clf = Pipeline(steps=[('classifier', mp['model'])])

    # Perform Grid Search with Cross-Validation
    grid_search = GridSearchCV(clf, mp['params'], cv=5, scoring='f1', n_jobs=-1)
    grid_search.fit(X_resampled, y_resampled)
```

```python
# Get the best model from grid search
best_model = grid_search.best_estimator_
print("--------------- Best Model: ------------------")
print(best_model)

# Predictions on the test set
y_pred = best_model.predict(X_test)

# Print evaluation metrics
print(f"Best parameters found: {grid_search.best_params_}")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision: {precision_score(y_test, y_pred):.4f}")
print(f"Recall: {recall_score(y_test, y_pred):.4f}")
print(f"F1 Score: {f1_score(y_test, y_pred):.4f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Model: DecisionTree
--------------- Best Model: ------------------
Pipeline(steps=[('classifier', DecisionTreeClassifier(random_state=42))])
Best parameters found: {'classifier__max_depth': None, 'classifier__min_samples_s
plit': 2}
Accuracy: 0.5933
Precision: 0.3287
Recall: 0.3178
F1 Score: 0.3232
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.71      0.71     57471
           1       0.33      0.32      0.32     25283

    accuracy                           0.59     82754
   macro avg       0.52      0.52      0.52     82754
weighted avg       0.59      0.59      0.59     82754


Model: LogisticRegression
--------------- Best Model: ------------------
Pipeline(steps=[('classifier',
                 LogisticRegression(C=0.01, random_state=42,
                                    solver='liblinear'))])
Best parameters found: {'classifier__C': 0.01, 'classifier__max_iter': 100, 'clas
sifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
Accuracy: 0.6172
Precision: 0.3919
Recall: 0.4586
F1 Score: 0.4226
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.69      0.71     57471
           1       0.39      0.46      0.42     25283

    accuracy                           0.62     82754
   macro avg       0.57      0.57      0.57     82754
weighted avg       0.64      0.62      0.62     82754


Model: LightGBM
[LightGBM] [Info] Number of positive: 230472, number of negative: 230472
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.030919 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 8670
[LightGBM] [Info] Number of data points in the train set: 460944, number of used
features: 34
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
--------------- Best Model: ------------------
Pipeline(steps=[('classifier',
                 LGBMClassifier(learning_rate=0.05, max_depth=30,
                                n_estimators=50, random_state=42))])
Best parameters found: {'classifier__learning_rate': 0.05, 'classifier__max_dept
h': 30, 'classifier__n_estimators': 50}
Accuracy: 0.6392
Precision: 0.4000
```

```
Recall: 0.3621
F1 Score: 0.3801
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.76      0.75     57471
           1       0.40      0.36      0.38     25283

    accuracy                           0.64     82754
   macro avg       0.57      0.56      0.56     82754
weighted avg       0.63      0.64      0.63     82754


Model: RandomForest
--------------- Best Model: ------------------
Pipeline(steps=[('classifier',
                 RandomForestClassifier(max_depth=30, min_samples_split=5,
                                        n_estimators=150, random_state=42))])
Best parameters found: {'classifier__max_depth': 30, 'classifier__min_samples_spl
it': 5, 'classifier__n_estimators': 150}
Accuracy: 0.6061
Precision: 0.3403
Recall: 0.3082
F1 Score: 0.3234
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.74      0.72     57471
           1       0.34      0.31      0.32     25283

    accuracy                           0.61     82754
   macro avg       0.52      0.52      0.52     82754
weighted avg       0.60      0.61      0.60     82754
```

Experiment Insights and Key Takeaways

In this experiment, we evaluated four models: Decision Tree, Logistic Regression, LightGBM, and Random Forest. The goal was to determine the best-performing model based on accuracy, precision, recall, and F1 score, while also considering model explainability and robustness.

1. Decision Tree:

- **Best Parameters**: `{'classifier__max_depth': None, 'classifier__min_samples_split': 2}`
- **Performance**:
  - **Accuracy**: 59.33%
  - **Precision**: 32.87%
  - **Recall**: 31.78%
  - **F1 Score**: 32.32%

Decision Tree provided a relatively low performance compared to the other models. Its accuracy and F1 score are below 60%, and it showed limited capability in predicting the minority class (history of mental illness), with a recall of just ~31%. This indicates that the model is struggling to generalize well, especially given the class imbalance, even with SMOTE applied.

2. **Logistic Regression:**

- **Best Parameters**: `{'classifier__C': 0.01, 'classifier__max_iter': 100, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}`
- **Performance**:
  - **Accuracy**: 61.72%
  - **Precision**: 39.19%
  - **Recall**: 45.86%
  - **F1 Score**: 42.26%

Logistic Regression emerged as one of the best-performing models in this experiment. It achieved a good balance between precision and recall, with an F1 score of ~42%. Its simplicity and interpretability make it a strong candidate for the final model. Despite its slightly lower accuracy compared to other models, its performance in terms of recall (ability to identify positive cases of mental illness) is notable, making it a reliable choice for this problem.

3. **LightGBM:**

- **Best Parameters**: `{'classifier__learning_rate': 0.05, 'classifier__max_depth': 30, 'classifier__n_estimators': 50}`
- **Performance**:
  - **Accuracy**: 63.92%
  - **Precision**: 40.00%
  - **Recall**: 36.21%
  - **F1 Score**: 38.01%

LightGBM achieved the highest accuracy (63.92%) among the models, but its recall for the minority class was lower than Logistic Regression. While LightGBM offers powerful predictive performance, it tends to be less interpretable compared to simpler models like Logistic Regression. This makes it less desirable for this task, where model explainability is important.

4. **Random Forest:**

- **Best Parameters**: `{'classifier__max_depth': 30, 'classifier__min_samples_split': 5, 'classifier__n_estimators': 150}`
- **Performance**:
  - **Accuracy**: 60.61%
  - **Precision**: 34.03%
  - **Recall**: 30.82%
  - **F1 Score**: 32.34%

Random Forest provided moderate performance, similar to the Decision Tree model. Despite its slightly higher precision, the recall for the minority class remained relatively low (30.82%), which limits its effectiveness for identifying individuals with a history of mental illness. Its F1 score (32%) also indicates that it struggles to balance precision and recall effectively for this problem.

Final Model Choice: Logistic Regression

Based on the performance of the models, **Logistic Regression** is the best model for our problem. Here are the key reasons for this choice:

- **Balanced Performance**: Logistic Regression provided the best balance between precision and recall, resulting in an F1 score of ~42%. This balance is crucial for the task, where both false positives and false negatives can have significant implications.
- **Explainability**: Logistic Regression is highly interpretable, allowing us to understand how each feature contributes to the predictions. This is important in real-world applications, especially in health-related domains, where decision-making transparency is essential.
- **Simplicity**: Logistic Regression is computationally efficient and straightforward, making it easier to deploy and scale. While models like LightGBM and Random Forest offer marginally higher accuracy, they are more complex and less interpretable.
- **Consistent Results**: Across both experiments (previous and current), Logistic Regression consistently performed well, especially when paired with SMOTE for addressing the class imbalance.

Conclusion

For our final model, we will proceed with **Logistic Regression**, using the best parameters found in this experiment (`C = 0.01, max_iter = 100, penalty = 'l2', solver = 'liblinear'`). This model offers the best combination of performance, simplicity, and explainability for predicting whether an individual is likely to suffer from mental illness.

In [ ]:

```

```