1.Problem Statement:

Develop a model to predict hospital readmissions within 30 days post-discharge using patient data.

2.Data Collection and Description:

The dataset collected from the UCI Machine Learning Repository encapsulates ten years (1999–2008) of clinical care data across 130 U.S. hospitals and integrated delivery networks. The primary focus is on patients diagnosed with diabetes, their hospitalizations, laboratory medications, and related care patterns. Below are the details about the dataset:

Dataset Description

Attributes

The dataset includes **50 features** grouped into the following categories:

1. **Patient Demographics:**
    - race, gender, age, etc.
2. **Hospital Admission Information:**
    - admission_type_id, discharge_disposition_id, admission_source_id.
3. **Medical Specialties:**
    - medical_specialty, representing the department overseeing the patient's care.
4. **Lab and Medication Indicators:**
    - Features indicating whether specific lab tests or medications were performed/administered.
5. **Outcome Variables:**
    - **Readmission** (readmitted): Whether the patient was readmitted to the hospital after discharge.Possible values: NO, >30 (readmission after 30 days), <30 (readmission within 30 days).

Dataset: : https://archive.ics.uci.edu/dataset/296/diabetes+130-us+hospitals+for+years+1999-2008

3. Data Preprocessing:

Handling Missing Values:

- Calculated missing value percentage of each and every column and dropped the columns which are having more than 50% missing values.
- For the columns with missing value percentage near to 50% like ***medical_specialty(49% missing) and payer_code(40% missing)***, verified the relation with target variable and removed the columns(payer_code) which doesn't have much relation with target variable.
- For all the remaining columns, replaced the missing values in categorical features with mode and replaced the missing values in numerical features with mean

Feature Engineering:

- Reduced the cardinality of the highly cardinal features like medical_speciality by grouping infrequent categories with 'other' categories.
- Encoded the highly cardinal features like **medical_speciality and age** using frequency encoding
- Encoded the remaining categorical features using one hot encoding and removed dummy variable to avoid dummy variable trap

Feature Selection:

Chi-Square Test for Categorical Variables

- Used the Chi-Square test to determine which categorical features had significant associations with the target (readmitted).
- Features with p-values < 0.05 were retained for the model.

Correlation Analysis for Numerical Variables

- Calculated the correlation matrix for numerical features and removed highly correlated features (correlation >=0.5) to avoid multicollinearity.

Addressing Class Imbalance:

- **Class Weights:** During model training (e.g., Random Forest, Logistic Regression), used class weights proportional to the inverse of the class frequencies.
- **Sample Weights:** Applied during LightGBM and XGBoost training to account for imbalance.

Data Splitting:

- **Stratified Train-Test Split:**Ensured the target variable (readmitted) had the same class distribution in training and testing datasets to prevent bias.

4. Model Selection:

Multiple models were tested to identify the best-performing one for this multi-class classification problem:

a. Logistic Regression

- **Why Chosen:**
    - Serves as a baseline model due to its simplicity and interpretability.
- **Implementation:**
    - Regularization (L2) was applied to prevent overfitting.
    - Class weights were incorporated to handle imbalance.
- **Results:**
    - Performed reasonably but was outperformed by tree-based models.

b. Random Forest

- **Why Chosen:**
    - Robust to missing values and outliers.
    - Handles multi-class classification effectively.
- **Implementation:**
    - Hyperparameters tuned:
        - n_estimators: 200 (number of trees).
        - max_depth: 22 (to control overfitting).

        - class_weight: Automatically balanced using the inverse of class frequencies.
- **Results**:
    - Achieved good accuracy and F1-scores across all classes.
    - Performance was consistent and reliable.

c. LightGBM

- **Why Chosen:**
    - Gradient boosting framework optimized for speed and efficiency.
    - Excellent performance on structured/tabular data.
- **Implementation**:
    - Trained using weighted samples to account for class imbalance.
    - Hyperparameters tuned:
        - max_depth: 20
        - num_leaves: 30
        - learning_rate: 0.05
        - num_boost_round: 100
- **Results**:
    - Provided comparable performance to Random Forest but faster training.

d. **XGBoost**

- **Why Chosen:**
    - Highly effective gradient boosting algorithm for classification tasks.
    - Known for its ability to handle complex interactions between features.
- **Implementation:**
    - Used the multi:softmax objective for multi-class classification.
    - Hyperparameters tuned:
        - n_estimators: 250
        - max_depth: 25
        - learning_rate: 0.05
- **Results**:
    - Performed slightly better than LightGBM, achieving high accuracy and precision across all classes.

e. Artificial Neural Network (ANN)

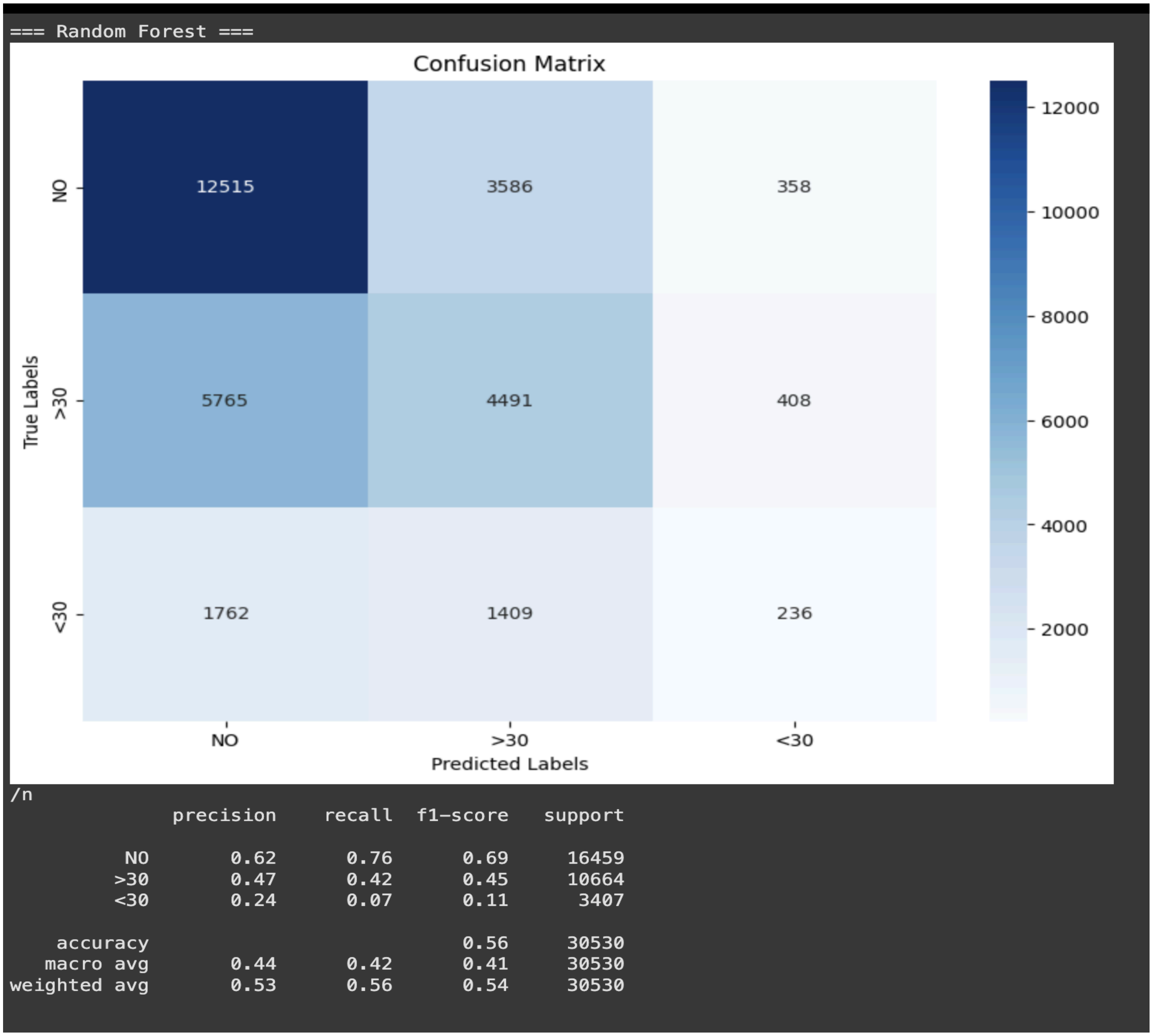
- **Why Chosen:**
    - ANN has the capacity to capture non-linear relationships in the data.
- **Architecture:**
    - Three fully connected layers:
        - Input Layer: Matching the number of features.
        - Hidden Layers: Two layers with 256 and 128 neurons, using ELU (Exponential Linear Unit) activation.
        - Output Layer: Three neurons (one for each class) with softmax activation.
    - Loss Function: Cross-entropy loss for multi-class classification.
    - Optimizer: Adam optimizer with a learning rate of 0.001.
- **Training**:
    - Batch size: 32
    - Epochs: 50
    - Weighted loss to handle class imbalance.
- **Results**:
    - Performance was comparable to tree-based models but required more computation and careful tuning.

5. Model Evaluation

Metrics Used:

- **Accuracy:** Overall correctness of predictions.
- **Precision, Recall, and F1-Score:** To evaluate performance per class.
- **Confusion Matrix:** To visualize class-level performance and identify misclassifications.

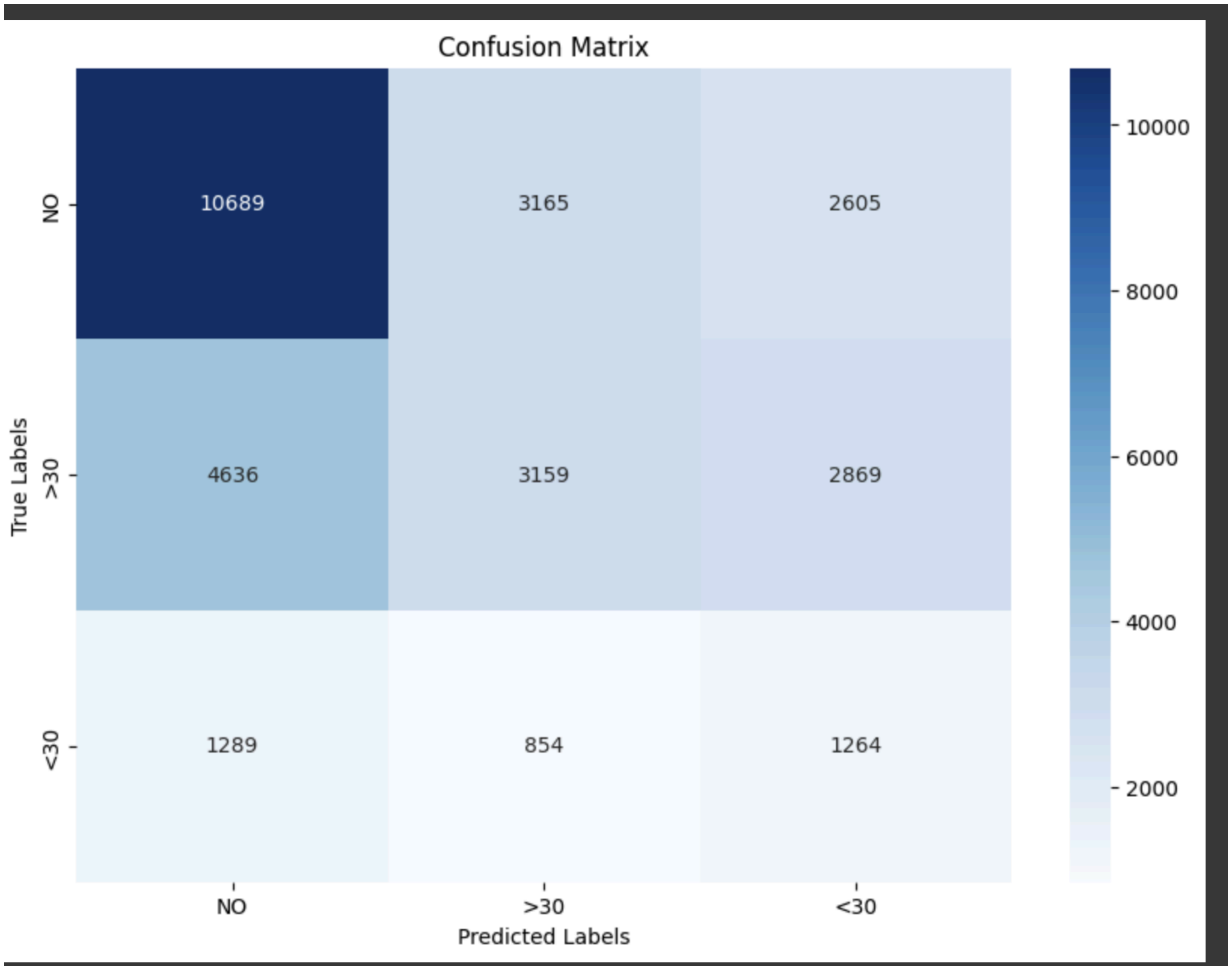
Random Forest:



```
=== Random Forest ===
```

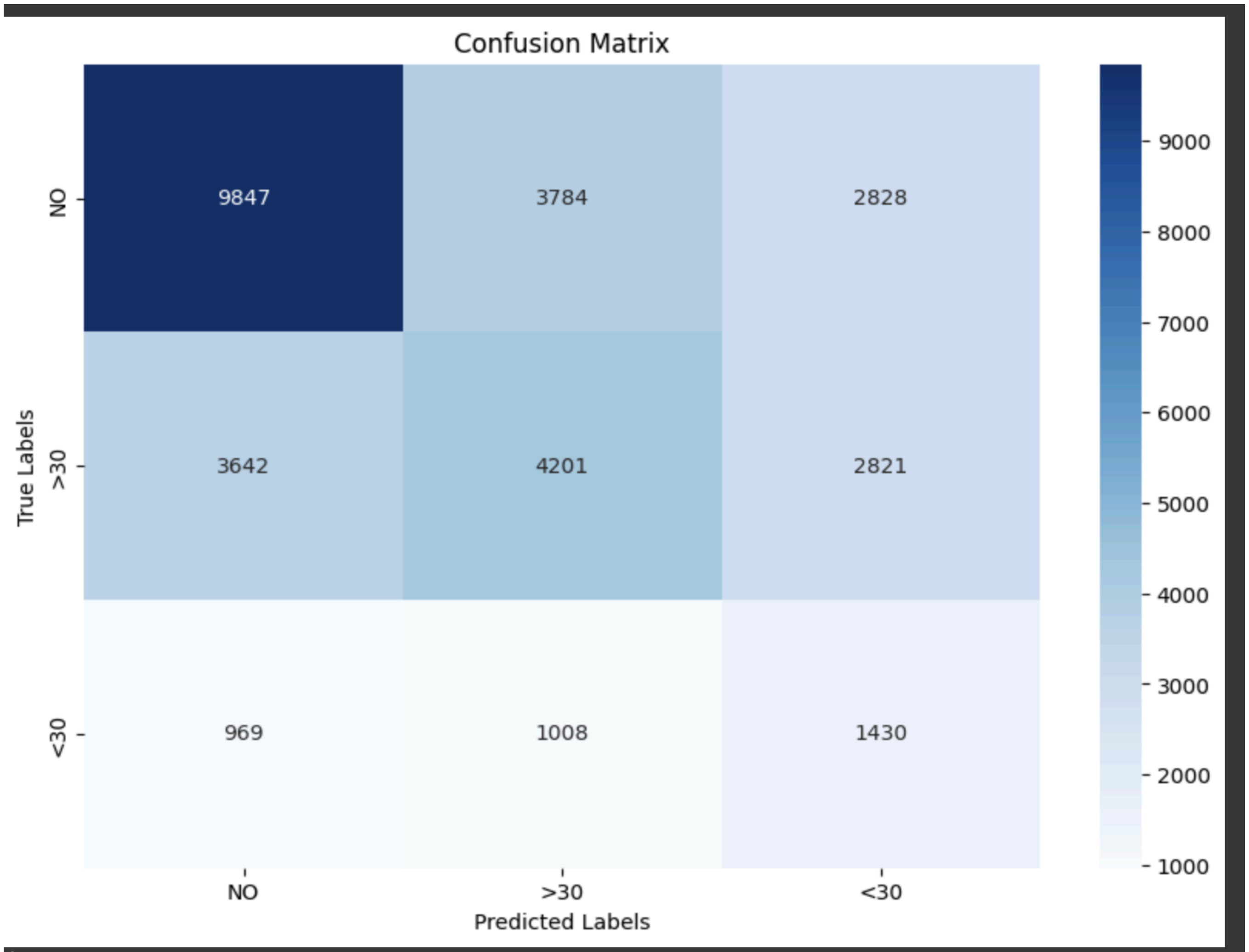
```
/n
              precision    recall  f1-score   support

          NO       0.62      0.76      0.69     16459
         >30       0.47      0.42      0.45     10664
         <30       0.24      0.07      0.11      3407

    accuracy                           0.56     30530
   macro avg       0.44      0.42      0.41     30530
weighted avg       0.53      0.56      0.54     30530
```

Logistic Regression:



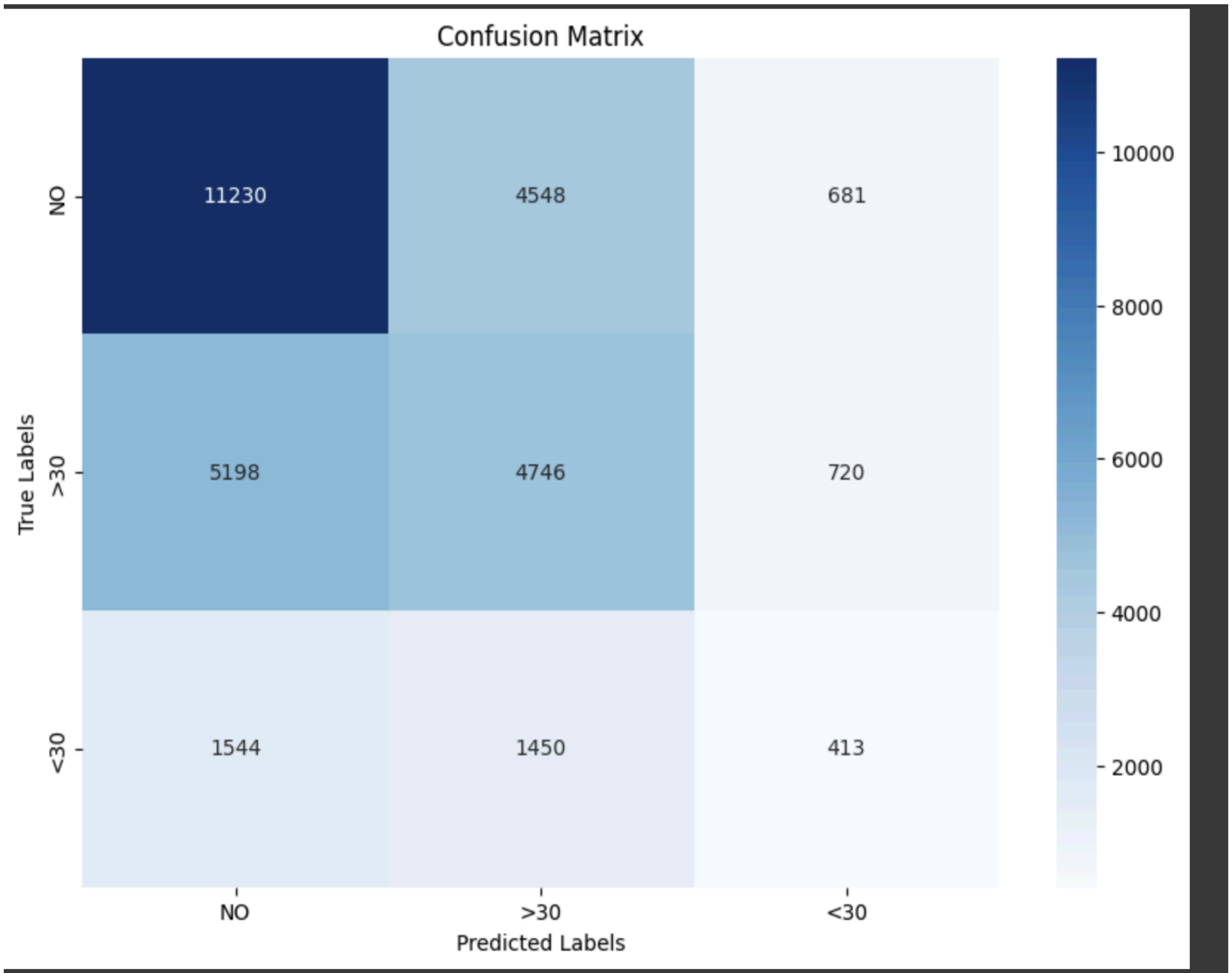
```
'n
              precision    recall  f1-score   support

          NO       0.64      0.65      0.65     16459
         >30       0.44      0.30      0.35     10664
         <30       0.19      0.37      0.25      3407

    accuracy                           0.49     30530
   macro avg       0.42      0.44      0.42     30530
eighted avg       0.52      0.49      0.50     30530
```

Light GBM:



```
/n
              precision    recall  f1-score   support

          NO       0.68      0.60      0.64     16459
         >30       0.47      0.39      0.43     10664
         <30       0.20      0.42      0.27      3407

    accuracy                           0.51     30530
   macro avg       0.45      0.47      0.45     30530
weighted avg       0.55      0.51      0.52     30530
```

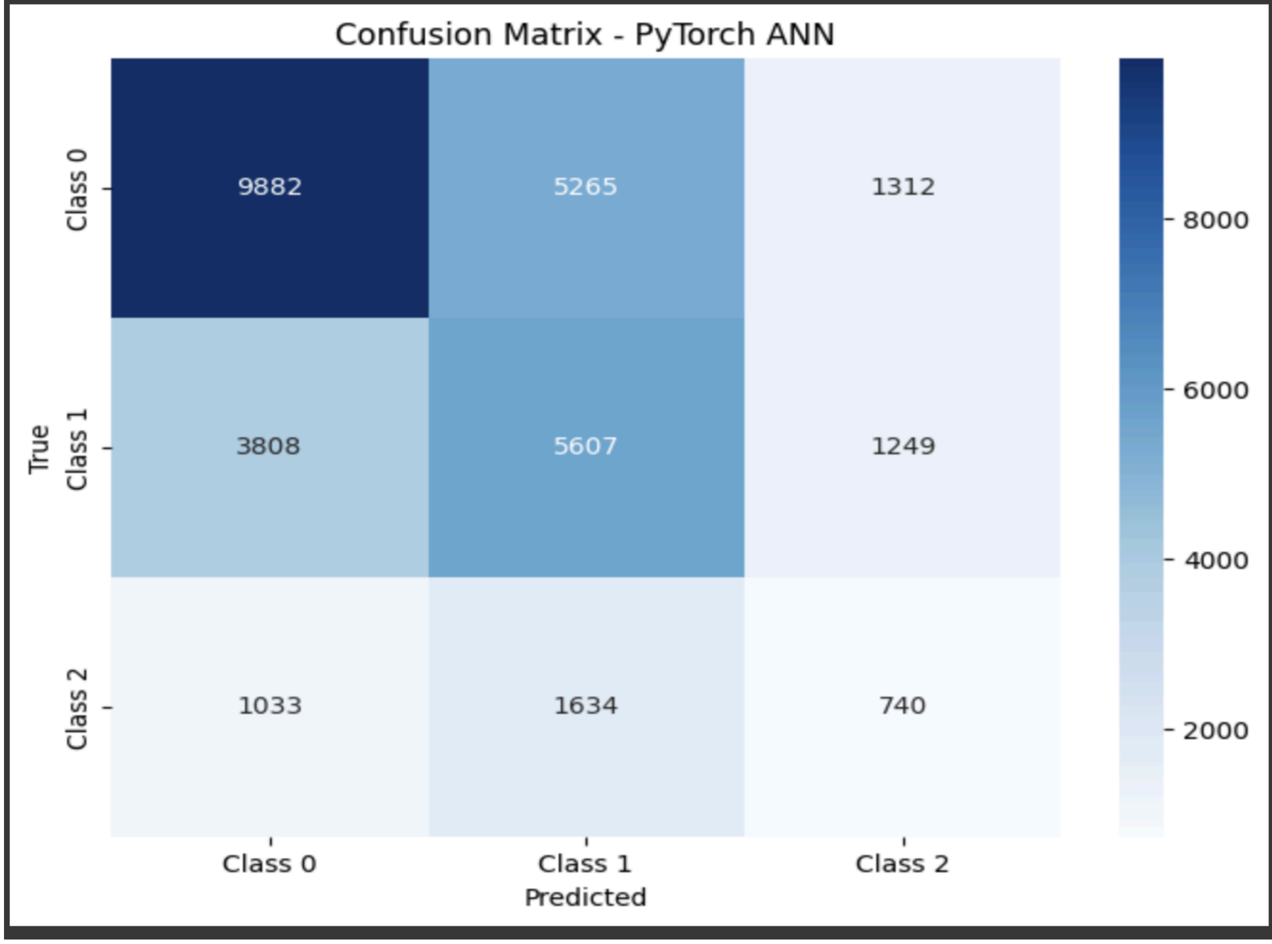
XG BOOST:



```
/n
              precision    recall  f1-score   support

          NO       0.62      0.68      0.65     16459
         >30       0.44      0.45      0.44     10664
         <30       0.23      0.12      0.16      3407

    accuracy                           0.54     30530
   macro avg       0.43      0.42      0.42     30530
weighted avg       0.52      0.54      0.52     30530
```

Artificial Neural Network (ANN):

```
              precision    recall  f1-score   support

           0       0.67      0.60      0.63     16459
           1       0.45      0.53      0.48     10664
           2       0.22      0.22      0.22      3407

    accuracy                           0.53     30530
   macro avg       0.45      0.45      0.45     30530
weighted avg       0.54      0.53      0.54     30530
```



6.Results

- Random Forest, XGBoost, and LightGBM outperformed Logistic Regression and ANN in terms of accuracy and F1-scores.
- **Best Model**: XGBoost showed slightly better performance due to its ability to capture feature interactions.