```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import re

# 1. Load the dataset
# The dataset MH_Dataset is assumed to have the following columns:
# 'clinical_notes': Text data from clinical notes.
# 'PHQ_score': Patient Health Questionnaire scores (numerical).
# 'GAD_score': Generalized Anxiety Disorder scores (numerical).
# 'mental_health_challenge': Binary target column indicating presence (1) or absence (0) of a challenge.

# Creating an imaginary dataset for demonstration purposes.
data = {
  'clinical_notes': [
    "Patient reports feeling down and hopeless.",
    "No significant mental health complaints.",
    "Experiencing frequent anxiety and nervousness.",
    "Patient states they have trouble concentrating and sleeping.",
    "Denies feelings of depression or anxiety."
  ],
  'PHQ_score': [15, 2, 7, 13, 1],
```

```
    'GAD_score': [10, 1, 8, 9, 0],
    'mental_health_challenge': [1, 0, 1, 1, 0]
}

# Converting the data dictionary into a pandas DataFrame
MH_Dataset = pd.DataFrame(data)

# 2. Preprocess the data
# Clean the text data in 'clinical_notes'
def preprocess_text(text):
    """
    Preprocesses clinical notes by removing special characters, converting to lowercase, and removing extra spaces.
    """
    text = re.sub(r'[^a-zA-Z ]', '', text)  # Remove non-alphabetic characters
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'\s+', ' ', text)  # Remove extra spaces
    return text

MH_Dataset['cleaned_notes'] = MH_Dataset['clinical_notes'].apply(preprocess_text)

# Combine features (PHQ, GAD scores, and TF-IDF vectorized notes)
vectorizer = TfidfVectorizer(max_features=100)
X_text = vectorizer.fit_transform(MH_Dataset['cleaned_notes']).toarray()
X_scores = MH_Dataset[['PHQ_score', 'GAD_score']].values
X = np.hstack((X_text, X_scores))  # Combine text and numerical features
```

```
y = MH_Dataset['mental_health_challenge']

# 3. Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Train the Logistic Regression Model
model = LogisticRegression()
model.fit(X_train, y_train)

# 5. Evaluate the Model
predictions = model.predict(X_test)
print("Classification Report:")
print(classification_report(y_test, predictions))

# 6. Function to reuse the model for new data
def predict_mental_health(new_notes, new_PHQ, new_GAD):
    """
    Predict mental health challenges based on new clinical notes, PHQ, and GAD scores.

    Parameters:
        new_notes (list): List of clinical notes as free text.
        new_PHQ (list): List of PHQ scores.
        new_GAD (list): List of GAD scores.

    Returns:
        List of predictions (1 for challenge, 0 for no challenge).
```

```
    """
    cleaned_notes = [preprocess_text(note) for note in new_notes]
    text_features = vectorizer.transform(cleaned_notes).toarray()
    score_features = np.array(list(zip(new_PHQ, new_GAD)))
    features = np.hstack((text_features, score_features))
    return model.predict(features)

# 7. Example Usage of the Reusable Function
new_data_notes = [
    "Patient is experiencing severe anxiety and frequent panic attacks.",
    "No signs of mental distress or significant emotional concerns."
]
new_data_PHQ = [12, 3]
new_data_GAD = [10, 2]

predictions = predict_mental_health(new_data_notes, new_data_PHQ, new_data_GAD)
print("Predictions for new data:", predictions)

# Each part of the code has been extensively explained to ensure reusability and
understanding.

# Save the model and vectorizer for reusability
import joblib
joblib.dump(model, 'logistic_model.pkl')
joblib.dump(vectorizer, 'tfidf_vectorizer.pkl')
```