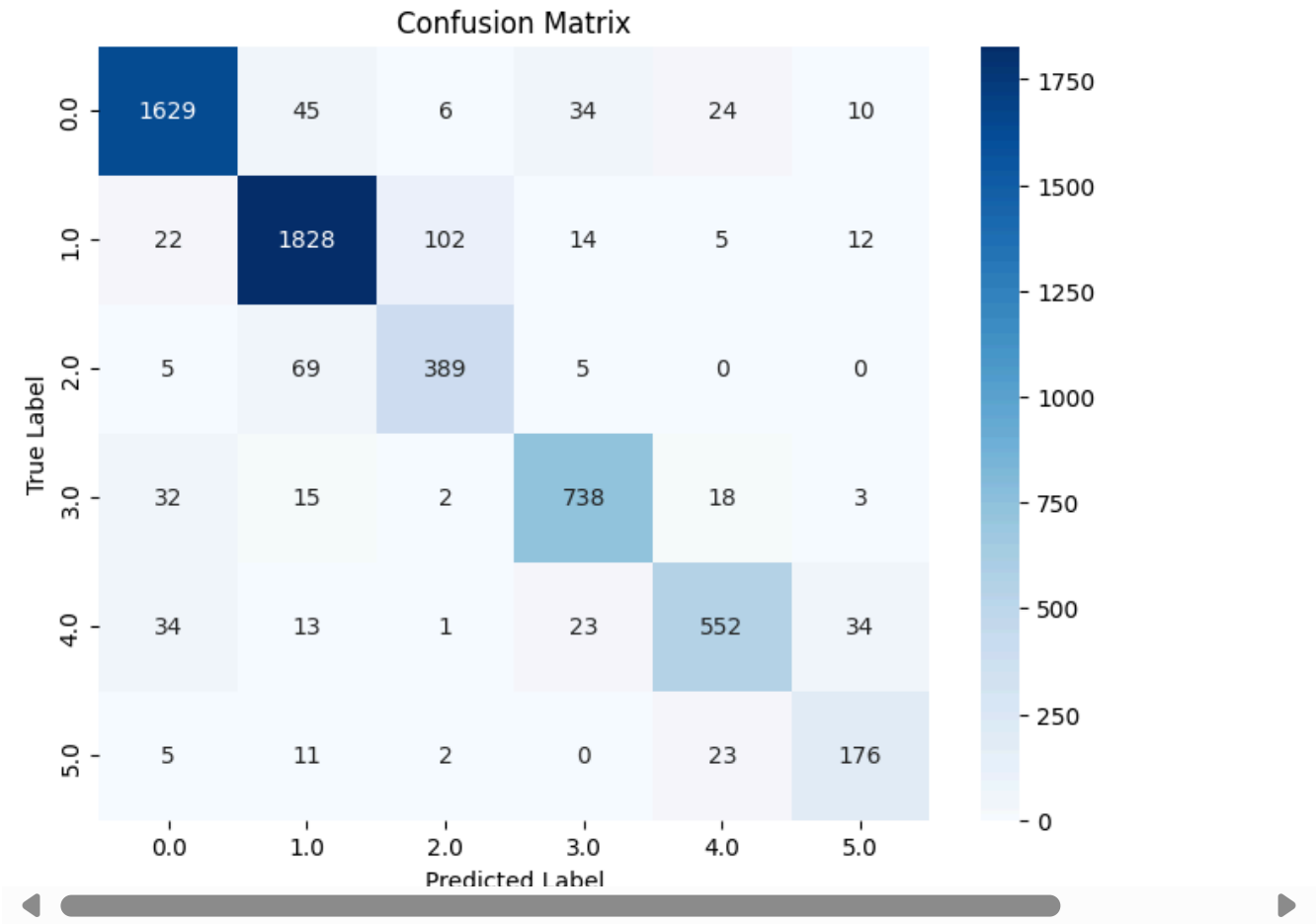
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics.pairwise import cosine_similarity
import seaborn as sns
import matplotlib.pyplot as plt
import pickle


# Load and preprocess data
data = pd.read_csv('text.csv')
df = pd.DataFrame(data)


df = df.dropna(subset=['text', 'label'])


# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    df["text"], df["label"], test_size=0.2, random_state=42, stratify=df["label"]
)



# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=10000, stop_words='english', ngram_range=(1, 3))
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)



# Model Training
param_grid = {
    'C': [0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs'],
    'max_iter': [100, 200]
}

lr = LogisticRegression(class_weight='balanced', random_state=42)
grid_search = GridSearchCV(lr, param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train_vec, y_train)

model = grid_search.best_estimator_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbf
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbf
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbf
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
        https://scikit-learn.org/stable/modules/preprocessing.html
      Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
        n_iter_i = _check_optimize_result(
```

```
# Save model and vectorizer
pickle.dump(model, open('emotion_model.sav', 'wb'))
pickle.dump(vectorizer, open('vectorizer.sav', 'wb'))


# Evaluate the model
y_pred = model.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(
    cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=df['label'].astype('category').cat.categories,
    yticklabels=df['label'].astype('category').cat.categories
)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.9032477469818058

Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.93      0.94      1748
         1.0       0.92      0.92      0.92      1983
         2.0       0.77      0.83      0.80       468
         3.0       0.91      0.91      0.91       808
         4.0       0.89      0.84      0.86       657
         5.0       0.75      0.81      0.78       217

    accuracy                           0.90      5881
   macro avg       0.86      0.87      0.87      5881
weighted avg       0.90      0.90      0.90      5881
```




```
# Prediction with similarity-based fallback
input_data = "i just feel really helpless and heavy hearted"
input_data_vec = vectorizer.transform([input_data])
prediction = model.predict(input_data_vec)
probabilities = model.predict_proba(input_data_vec)
```

```
    # Define emotion labels
    emotions = {
        0: "Neutral",
        1: "Happiness",
        2: "Excitement",
        3: "Calm",
        4: "Sadness",
        5: "Fatigue"
    }

    predicted_emotion = emotions.get(prediction[0], "Unknown")
```