```
!pip install -q pandas scikit-learn nltk datasets
```

```
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.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score, confu
import pickle
import re
import nltk
from datasets import load_dataset

nltk.download('stopwords')
from nltk.corpus import stopwords

print("✅ All libraries imported successfully!")
```

```
✅ All libraries imported successfully!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
print("Loading Davidson hate speech dataset...")
dataset = load_dataset("tdavidson/hate_speech_offensive")
df = pd.DataFrame(dataset['train'])

# Dataset info
print(f"Dataset shape: {df.shape}")
print(f"Columns: {df.columns.tolist()}")
print(f"\nClass distribution:")
print(df['class'].value_counts())
```

```
Loading Davidson hate speech dataset...
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:9
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settin
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to acc
  warnings.warn(
README.md: 5.92k/? [00:00<00:00, 538kB/s]
data/train-00000-of-00001.parquet: 100% 1.63M/1.63M [00:01<00:00, 1.24MB/s]
Generating train split: 100% 24783/24783 [00:00<00:00, 367456.05 examples/s]
Dataset shape: (24783, 6)
Columns: ['count', 'hate_speech_count', 'offensive_language_count', 'nei

Class distribution:
class
1    19190
2     4163
0     1430
Name: count, dtype: int64
```

```
print(f"Dataset shape: {df.shape}")
print(f"Columns: {df.columns.tolist()}")
print(f"\nClass distribution:")
print(df['class'].value_counts())
```

```
Dataset shape: (24783, 6)
Columns: ['count', 'hate_speech_count', 'offensive_language_count', 'nei

Class distribution:
class
1    19190
2     4163
0     1430
Name: count, dtype: int64
```

```
df['label'] = df['class']
```

```
def preprocess_text(text):
    """Clean and preprocess text"""
    # Convert to lowercase
    text = str(text).lower()
    # Remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTIL
    # Remove user mentions
    text = re.sub(r'@\w+', '', text)
    # Remove hashtags
    text = re.sub(r'#\w+', '', text)
    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Remove extra whitespace
    text = ' '.join(text.split())
    return text

print("Preprocessing text...")
df['clean_text'] = df['tweet'].apply(preprocess_text)
```

```
Preprocessing text...
```

```
df = df[df['clean_text'].str.strip() != '']
print(f"Dataset shape after cleaning: {df.shape}")

# ========== Cell 5: Split data ==========
X = df['clean_text']
y = df['label']
```

```
Dataset shape after cleaning: (24766, 8)
```

```
X = df['clean_text']
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")
```

```
Training set size: 19812
Test set size: 4954
```

```
print("Extracting TF-IDF features...")
vectorizer = TfidfVectorizer(
    max_features=5000,
    min_df=5,
    max_df=0.8,
    ngram_range=(1, 2),
    stop_words='english'
)
```

```
Extracting TF-IDF features...
```

```
print("Extracting TF-IDF features...")
vectorizer = TfidfVectorizer(
    max_features=5000,
    min_df=5,
    max_df=0.8,
    ngram_range=(1, 2),
    stop_words='english'
)

X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

```
print(f"Feature matrix shape: {X_train_vec.shape}")

# ========== Cell 7: Train MLP Classifier ==========
print("Training MLP Classifier...")
print("This may take 5-10 minutes... ⏳")

clf = MLPClassifier(
    hidden_layer_sizes=(128, 64, 32),
    activation='relu',
    solver='adam',
    alpha=0.0001,
    batch_size=256,
    learning_rate='adaptive',
    max_iter=50,
    random_state=42,
    verbose=True,
    early_stopping=True,
    validation_fraction=0.1
)

clf.fit(X_train_vec, y_train)
print("\n✅ Training complete!")
```

```
Extracting TF-IDF features...
Feature matrix shape: (19812, 5000)
Training MLP Classifier...
This may take 5-10 minutes... ⏳
Iteration 1, loss = 0.68894543
Validation score: 0.763875
Iteration 2, loss = 0.35319695
Validation score: 0.874874
Iteration 3, loss = 0.22883325
Validation score: 0.892028
Iteration 4, loss = 0.16447861
Validation score: 0.878406
Iteration 5, loss = 0.12318773
Validation score: 0.874369
Iteration 6, loss = 0.09253814
Validation score: 0.872856
Iteration 7, loss = 0.06900776
Validation score: 0.864783
Iteration 8, loss = 0.05083560
Validation score: 0.871342
Iteration 9, loss = 0.04034878
Validation score: 0.861251
Iteration 10, loss = 0.03246286
Validation score: 0.863774
Iteration 11, loss = 0.02911272
Validation score: 0.862260
```

```
Iteration 12, loss = 0.02544957
Validation score: 0.865288
Iteration 13, loss = 0.02366003
Validation score: 0.866801
Iteration 14, loss = 0.02148547
Validation score: 0.863774
Validation score did not improve more than tol=0.000100 for 10 consecuti

✅ Training complete!
```

```
print("\n=== Model Evaluation ===")
y_pred = clf.predict(X_test_vec)
accuracy = accuracy_score(y_test, y_pred)

print(f"\nAccuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred,
                           target_names=['hate_speech', 'offensive', 'n

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
=== Model Evaluation ===

Accuracy: 0.8892

Classification Report:
              precision    recall  f1-score   support

 hate_speech       0.51      0.32      0.40       285
   offensive       0.92      0.95      0.93      3837
     neutral       0.82      0.82      0.82       832

    accuracy                           0.89      4954
   macro avg       0.75      0.70      0.72      4954
weighted avg       0.88      0.89      0.88      4954


Confusion Matrix:
[[  92  168   25]
 [  75 3633  129]
 [  12  140  680]]
```

```
import os

# Create trained_models directory in Colab
os.makedirs('trained_models', exist_ok=True)

# Save both models
with open('trained_models/hate_speech_classifier.pkl', 'wb') as f:
    pickle.dump(clf, f)

with open('trained_models/vectorizer.pkl', 'wb') as f:
    pickle.dump(vectorizer, f)

# Check file sizes
classifier_size = os.path.getsize('trained_models/hate_speech_classifie
vectorizer_size = os.path.getsize('trained_models/vectorizer.pkl') / (1

print("\n✅ Models saved successfully!")
print(f"📦 hate_speech_classifier.pkl size: {classifier_size:.2f} MB")
print(f"📦 vectorizer.pkl size: {vectorizer_size:.2f} MB")
```

```
✅ Models saved successfully!
📦 hate_speech_classifier.pkl size: 14.90 MB
📦 vectorizer.pkl size: 0.18 MB
```

```
def test_prediction(text, vectorizer, model):
    """Test model on custom text"""
    clean = preprocess_text(text)
    vec = vectorizer.transform([clean])
    pred = model.predict(vec)[0]
    proba = model.predict_proba(vec)[0]

    labels = ['hate_speech', 'offensive', 'neutral']
    return labels[pred], proba[pred]

# Test examples
test_texts = [
    "I love this beautiful day!",
    "You are so stupid and worthless",
    "Go back to your country!",
    "This is offensive language",
    "Have a great day everyone!"
]

print("\n=== Test Predictions ===")
for text in test_texts:
    label, confidence = test_prediction(text, vectorizer, clf)
    print(f"\n📝 Text: {text}")
    print(f"🎯 Prediction: {label} (confidence: {confidence:.2f})")
```

```
=== Test Predictions ===

📝 Text: I love this beautiful day!
🎯 Prediction: neutral (confidence: 0.88)

📝 Text: You are so stupid and worthless
🎯 Prediction: hate_speech (confidence: 0.63)

📝 Text: Go back to your country!
🎯 Prediction: offensive (confidence: 0.62)

📝 Text: This is offensive language
🎯 Prediction: neutral (confidence: 0.95)

📝 Text: Have a great day everyone!
🎯 Prediction: neutral (confidence: 0.96)
```

```
from google.colab import files

print("\n📩 Downloading trained models to your computer...")
print("⏳ Please wait, this may take a minute...")

# Download both files
files.download('trained_models/hate_speech_classifier.pkl')
files.download('trained_models/vectorizer.pkl')

print("\n✅ Downloads complete!")
```

```
📩 Downloading trained models to your computer...
⏳ Please wait, this may take a minute...

✅ Downloads complete!
```

Start coding or generate with AI.