```
import os
import joblib
import numpy as np
import pandas as pd
from scipy.sparse import hstack
from sentence_transformers import SentenceTransformer
import warnings
import traceback

warnings.filterwarnings("ignore")
print("Libraries imported successfully.")
```

```
c:\Users\yagni\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please
update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

WARNING:tensorflow:From c:\Users\yagni\AppData\Local\Programs\Python\
Python311\Lib\site-packages\tf_keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

Libraries imported successfully.
```

```
MODEL_DIR = "../src/models"

LABELS = [
    'addiction',
    'adhd',
    'anxiety',
    'autism',
    'bipolar',
    'bpd',
    'depression',
    'ocd',
    'psychosis',
    'ptsd',
    'suicide'
]

LABEL2ID = {label: i for i, label in enumerate(LABELS)}
ID2LABEL = {i: label for i, label in enumerate(LABELS)}

print(f"Models will be loaded from: {MODEL_DIR}")
print(f"Configured with {len(LABELS)} labels: {LABELS}")
```

```
Models will be loaded from: ../src/models
Configured with 11 labels: ['addiction', 'adhd', 'anxiety', 'autism',
'bipolar', 'bpd', 'depression', 'ocd', 'psychosis', 'ptsd', 'suicide']
```

```
print("Loading all models and vectorizers...")

# Advanced models
svc_model = joblib.load(os.path.join(MODEL_DIR, "svc_model.pkl"))
tfidf_word = joblib.load(os.path.join(MODEL_DIR, "tfidf_word.pkl"))
tfidf_char = joblib.load(os.path.join(MODEL_DIR, "tfidf_char.pkl"))
lr_sbert = joblib.load(os.path.join(MODEL_DIR, "lr_sbert.pkl"))
print("-> Advanced models (SVC, SBERT-LR) loaded.")

# Baseline models
baseline_lr = joblib.load(os.path.join(MODEL_DIR,
"baseline_logistic_regression.pkl"))
baseline_rf = joblib.load(os.path.join(MODEL_DIR,
"baseline_random_forest.pkl"))
baseline_vectorizer = joblib.load(os.path.join(MODEL_DIR,
"tfidf_vectorizer.pkl"))
print("-> Baseline models (LR, RF) loaded.")

# Sentence Transformer (SBERT) model
sbert_model = SentenceTransformer("sentence-transformers/all-MiniLM-
L6-v2")
print("-> SBERT model loaded.")

print("\n All artifacts loaded successfully!")
```

```
Loading all models and vectorizers...
-> Advanced models (SVC, SBERT-LR) loaded.
-> Baseline models (LR, RF) loaded.
-> SBERT model loaded.

 All artifacts loaded successfully!
```

```
def predict_top3(texts, weights=None):
    if weights is None:
        # Define the weights for combining model predictions
        weights = {'svc': 0.4, 'sbert': 0.3, 'lr': 0.15, 'rf': 0.15}

    results = []
    try:
        # --- Feature Generation ---
        # 1. Advanced TF-IDF for SVC
        X_word = tfidf_word.transform(texts)
        X_char = tfidf_char.transform(texts)
        X_advanced = hstack([X_word, X_char])

        # 2. SBERT embeddings for SBERT-LR
        embeddings = sbert_model.encode(texts, batch_size=32,
convert_to_numpy=True)

        # 3. Baseline TF-IDF for baseline models
```

```
        X_baseline = baseline_vectorizer.transform(texts)

        # --- Probability Prediction ---
        svc_probs = svc_model.predict_proba(X_advanced)
        sbert_probs = lr_sbert.predict_proba(embeddings)
        lr_probs = baseline_lr.predict_proba(X_baseline)
        rf_probs = baseline_rf.predict_proba(X_baseline)

        # --- Weighted Ensemble ---
        ensemble_probs = (weights['svc'] * svc_probs +
                          weights['sbert'] * sbert_probs +
                          weights['lr'] * lr_probs +
                          weights['rf'] * rf_probs)

        # --- Process Results ---
        for i, text in enumerate(texts):
            final_probs = ensemble_probs[i]
            top_indices = np.argsort(final_probs)[::-1][:3]

            top3 = [(ID2LABEL[idx], float(final_probs[idx])) for idx
in top_indices]

            results.append({
                'text': text[:100] + "..." if len(text) > 100 else
text,
                'predictions': top3
            })

    except Exception as e:
        print(f"❌ An error occurred during prediction: {e}")
        traceback.print_exc()
        # Return an error message for all texts if one fails
        results = [{'text': text, 'predictions': [('ERROR', 0.0)]} for
text in texts]

    return results

print("Prediction function is defined.")
```

```
Prediction function is defined.
```

```
# List of texts to test the model pipeline
test_texts = [
    "I feel very anxious and can't sleep at night. My heart races
constantly.",
    "Lately I feel on top of the world, very energetic and happy,
barely need sleep.",
    "I find it hard to focus and get frustrated easily. I can't sit
still for long.",
    "I have these intrusive thoughts and I need to check things over
```

```
and over again.",
    "I feel empty inside and nothing brings me joy anymore. It's hard
to even get out of bed.",
    "My emotions are all over the place, my relationships are so
intense and then they just end badly."
]

print("Running ensemble prediction on sample texts...")
prediction_results = predict_top3(test_texts)
print("✅ Prediction complete.")
```

```
Running ensemble prediction on sample texts...
✅ Prediction complete.
```

```
for i, result in enumerate(prediction_results, 1):
    print(f"--- Text #{i} ---\n'{result['text']}'")
    print("Top 3 Predictions:")
    for rank, (label, confidence) in enumerate(result['predictions'],
1):
        confidence_pct = confidence * 100
        bar = "█" * int(confidence_pct / 5) + "░" * (20 -
int(confidence_pct / 5))
        print(f"  {rank}. {label.upper():<12} |{bar}|
{confidence_pct:5.1f}%")
    print("-" * 60)
```

```
--- Text #1 ---
'I feel very anxious and can't sleep at night. My heart races
constantly.'
Top 3 Predictions:
  1. ANXIETY      |██████████████░░░░░░|  71.2%
  2. BIPOLAR      |█░░░░░░░░░░░░░░░░░░░|   5.0%
  3. PSYCHOSIS    |░░░░░░░░░░░░░░░░░░░░|   3.2%
------------------------------------------------------------
--- Text #2 ---
'Lately I feel on top of the world, very energetic and happy, barely
need sleep.'
Top 3 Predictions:
  1. DEPRESSION   |███░░░░░░░░░░░░░░░░░|  19.2%
  2. SUICIDE      |███░░░░░░░░░░░░░░░░░|  16.6%
  3. BIPOLAR      |███░░░░░░░░░░░░░░░░░|  16.5%
------------------------------------------------------------
--- Text #3 ---
'I find it hard to focus and get frustrated easily. I can't sit still
for long.'
Top 3 Predictions:
  1. ADHD         |██████▒▒▒▒▒▒▒▒▒▒▒▒▒▒|  39.7%
  2. AUTISM       |██▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|  18.4%
  3. ANXIETY      |█▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|   8.2%
------------------------------------------------------------
```

```
--- Text #4 ---
'I have these intrusive thoughts and I need to check things over and
over again.'
Top 3 Predictions:
  1. OCD          |████████████▒▒▒▒▒▒▒▒|  78.6%
  2. PSYCHOSIS    |█▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|   7.6%
  3. ANXIETY      |▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|   2.0%
------------------------------------------------------------
--- Text #5 ---
'I feel empty inside and nothing brings me joy anymore. It's hard to
even get out of bed.'
Top 3 Predictions:
  1. DEPRESSION   |███████▒▒▒▒▒▒▒▒▒▒▒▒|  43.4%
  2. PSYCHOSIS    |██▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|  10.4%
  3. AUTISM       |██▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|   8.9%
------------------------------------------------------------
--- Text #6 ---
'My emotions are all over the place, my relationships are so intense
and then they just end badly.'
Top 3 Predictions:
  1. BPD          |██████████▒▒▒▒▒▒▒▒▒|  63.3%
  2. SUICIDE      |▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|   7.2%
  3. AUTISM       |▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒|   5.7%
------------------------------------------------------------
```

```python
def analyze_individual_models(text):
    print(f"\n Individual Model Analysis for:\n'{text}'")
    print("=" * 60)

    # --- Feature Prep ---
    X_advanced = hstack([tfidf_word.transform([text]),
tfidf_char.transform([text])])
    embedding = sbert_model.encode([text], convert_to_numpy=True)
    X_baseline = baseline_vectorizer.transform([text])

    # --- Predictions ---
    models_to_test = {
        "SVC (Advanced TF-IDF)": (svc_model, X_advanced),
        "SBERT-LR": (lr_sbert, embedding),
        "Baseline LR (TF-IDF)": (baseline_lr, X_baseline),
        "Baseline RF (TF-IDF)": (baseline_rf, X_baseline)
    }

    for model_name, (model, features) in models_to_test.items():
        probs = model.predict_proba(features)[0]
        prediction_idx = np.argmax(probs)
        prediction_label = ID2LABEL[prediction_idx]
        confidence = probs[prediction_idx]

        print(f"-> {model_name}:")
```

```
        print(f"   Prediction: {prediction_label.upper()}
({confidence:.2%})")

print("Individual model analysis function is defined.")
```

```
Individual model analysis function is defined.
```

```
# Choose any text you want to analyze in detail
sample_text_for_analysis = "I can't stop checking if I locked the door
and washing my hands repeatedly"

analyze_individual_models(sample_text_for_analysis)
```

```
 Individual Model Analysis for:
'I can't stop checking if I locked the door and washing my hands
repeatedly'
============================================================
-> SVC (Advanced TF-IDF):
   Prediction: OCD (77.75%)
-> SBERT-LR:
   Prediction: OCD (85.64%)
-> Baseline LR (TF-IDF):
   Prediction: OCD (53.09%)
-> Baseline RF (TF-IDF):
   Prediction: AUTISM (10.63%)
```