```
# Install and download all necessary corpora and libraries [cite: 368]
!pip install tensorflow numpy pandas matplotlib seaborn scikit-learn nltk nrclex

import nltk
import ssl

# Fix for NLTK download issues in some environments
try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

# Download specific NLTK resources required for NRCLex and Text Processing [cite: 369-375]
nltk.download(['stopwords', 'punkt', 'punkt_tab', 'wordnet', 'averaged_perceptron_tagger', 'averaged_perceptron_tagger_eng'])

import numpy as np
import pandas as pd
import tensorflow as tf
import re
import csv
import io
from nrclex import NRCLex
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from tensorflow.keras.layers import Layer, Input, Embedding, Bidirectional, LSTM, Dense, Dropout, Concatenate
import tensorflow.keras.backend as K

# === RESEARCH PARAMETERS (OPTIMIZED FOR >90% ACCURACY) [cite: 392-410] ===
K_CLIENTS = 10
SEQ_LENGTH = 120    # Captured longer context [cite: 404]
VOCAB_SIZE = 20000  # Captures rare medical terms [cite: 405]
EMBED_DIM = 200     # Richer word understanding [cite: 406]
HIDDEN_DIM = 128    # "Bigger Brain" for the LSTM [cite: 407]
GLOBAL_ROUNDS = 20  # Extended aggregation for non-IID settings [cite: 408]
LOCAL_EPOCHS = 3    # Clients learn more before updating server [cite: 409]
BATCH_SIZE = 16     # Smaller batches generalize better [cite: 410]

print("Cell 1 Complete: Environment Ready with TURBO Parameters.")
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: nrclex in /usr/local/lib/python3.12/dist-packages (3.0.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.9.23)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.32.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.76.0)
Requirement already satisfied: tensorboard~=2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.10.0)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.15.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: textblob in /usr/local/lib/python3.12/dist-packages (from nrclex) (0.19.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0->tensorfl
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.18.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0->tensorflo
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorbo
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0->tensorflo
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tensorboard
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tens
```

```python
print("Loading 2.2M+ rows using Robust Chunking...")
chunks = []
# Use quoting=csv.QUOTE_NONE to bypass EOF string errors
try:
    reader = pd.read_csv('reddit_depression_dataset.csv',
                         chunksize=100000,
                         engine='python',
                         on_bad_lines='skip',
                         quoting=csv.QUOTE_NONE)
    for chunk in reader:
        chunk.columns = chunk.columns.str.strip()
        if 'label' in chunk.columns and 'body' in chunk.columns:
            chunk['label'] = pd.to_numeric(chunk['label'], errors='coerce')
            chunk = chunk.dropna(subset=['label', 'body'])
            chunks.append(chunk[['label', 'body']])
    df_full = pd.concat(chunks, axis=0)
    print(f"Total available rows after cleaning: {len(df_full)}")
except Exception as e:
    print(f"Loading error: {e}")

# Balancing to exactly 100,000 samples (50,000 per class) [cite: 101, 450-464]
target_n = 50000
df_d = df_full[df_full['label'] == 1]
df_n = df_full[df_full['label'] == 0]

print(f"Balancing data to {target_n} samples per class...")
df_d_bal = resample(df_d, n_samples=target_n, replace=True, random_state=42)
df_n_bal = resample(df_n, n_samples=target_n, replace=True, random_state=42)

df = pd.concat([df_d_bal, df_n_bal]).sample(frac=1, random_state=42).reset_index(drop=True)
print(f"Cell 2 Complete: Total Samples for Research: {len(df)}")
```

```
Loading 2.2M+ rows using Robust Chunking...
Total available rows after cleaning: 114194
Balancing data to 50000 samples per class...
Cell 2 Complete: Total Samples for Research: 100000
```

```python
# Contraction Map for Expansion [cite: 488-491]
CONTRACTIONS = {"i'm": "i am", "can't": "cannot", "don't": "do not", "i've": "i have", "it's": "it is"}

def preprocess_strict(text):
    text = str(text).lower()
    for contraction, expansion in CONTRACTIONS.items():
        text = re.sub(r'\b'+ contraction + r'\b', expansion, text) # Expand contractions [cite: 496]
    text = re.sub(r'http\S+|www\S+|@\w+|<.*?>', '', text) # Remove URLs/Handles [cite: 499-503]
    text = re.sub(r'[^a-z\s]', '', text) # Keep text only [cite: 505]
    return text.strip() # Stopwords "I", "myself" are KEPT [cite: 108, 506]

print("Pre-processing text...")
df['clean_text'] = df['body'].apply(preprocess_strict)

# Tokenization and Padding [cite: 110-111, 521-526]
tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=VOCAB_SIZE, oov_token="<OOV>")
tokenizer.fit_on_texts(df['clean_text'])
sequences = tokenizer.texts_to_sequences(df['clean_text'])
X_seq = tf.keras.preprocessing.sequence.pad_sequences(sequences, maxlen=SEQ_LENGTH, padding='post')
```

```
# Psycholinguistic Emotion Vectors [cite: 116-121, 528-539]
def get_nrc_vector(text):
    try:
        emo = NRCLex(text).raw_emotion_scores
        order = ['anger', 'fear', 'anticipation', 'trust', 'surprise', 'sadness', 'joy', 'disgust']
        length = len(text.split()) if len(text.split()) > 0 else 1 # Normalize by length [cite: 536]
        return [emo.get(e, 0) / length for e in order]
    except:
        return [0.0] * 8

print("Generating Psycholinguistic Vectors (this may take 5-10 mins for 100k rows)...")
X_emo = np.array([get_nrc_vector(t) for t in df['clean_text']])
y = df['label'].values
print(f"Cell 3 Complete: Features Ready. X_seq: {X_seq.shape}, X_emo: {X_emo.shape}")
```

```
Pre-processing text...
Generating Psycholinguistic Vectors (this may take 5-10 mins for 100k rows)...
Cell 3 Complete: Features Ready. X_seq: (100000, 120), X_emo: (100000, 8)
```

```
class Attention(Layer):
    def __init__(self, **kwargs):
        super(Attention, self).__init__(**kwargs)
    def build(self, input_shape):
        self.W = self.add_weight(name='att_w', shape=(input_shape[-1], 1), initializer='normal')
        self.b = self.add_weight(name='att_b', shape=(input_shape[1], 1), initializer='zeros')
        super(Attention, self).build(input_shape)
    def call(self, x):
        e = K.tanh(K.dot(x, self.W) + self.b) # Score computation [cite: 133, 589]
        a = K.softmax(e, axis=1) # Softmax weights [cite: 591]
        return K.sum(x * a, axis=1) # Weighted sum context vector [cite: 157, 594]

def build_model():
    in_seq = Input(shape=(SEQ_LENGTH,))
    emb = Embedding(VOCAB_SIZE, EMBED_DIM)(in_seq)
    bilstm = Bidirectional(LSTM(HIDDEN_DIM, return_sequences=True))(emb) # [cite: 603]
    att = Attention()(bilstm) # Priority focus on trigger words [cite: 609]

    in_emo = Input(shape=(8,))
    merged = Concatenate()([att, in_emo]) # Fusion [cite: 159, 613]

    drop = Dropout(0.5)(merged)
    out = Dense(1, activation='sigmoid')(drop)

    model = tf.keras.Model(inputs=[in_seq, in_emo], outputs=out)
    model.compile(optimizer=tf.keras.optimizers.Adam(0.001), loss='binary_crossentropy', metrics=['accuracy'])
    return model

print("Cell 4 Complete: Model Architecture Compiled.")
```

```
Cell 4 Complete: Model Architecture Compiled.
```

```
# Partition Training Data [cite: 555-568]
X_seq_train, X_seq_test, X_emo_train, X_emo_test, y_train, y_test = train_test_split(
    X_seq, X_emo, y, test_size=0.2, stratify=y, random_state=42)

shard_size = len(y_train) // K_CLIENTS
clients = []
for i in range(K_CLIENTS):
    s, e = i*shard_size, (i+1)*shard_size
    clients.append({'seq': X_seq_train[s:e], 'emo': X_emo_train[s:e], 'y': y_train[s:e], 'n_k': shard_size})

# Initialize Global Model [cite: 621-624]
global_model = build_model()
global_weights = global_model.get_weights()
history = {'acc': [], 'loss': []}

print(f"Starting Federated Training ({GLOBAL_ROUNDS} Rounds) for 100k samples...")
for r in range(GLOBAL_ROUNDS):
    local_weights_list = []
    for k in range(K_CLIENTS):
        local_model = build_model()
        local_model.set_weights(global_weights) # Broadcast [cite: 164]
        c = clients[k]
        # Local training E=3 [cite: 166, 636]
        local_model.fit([c['seq'], c['emo']], c['y'], epochs=LOCAL_EPOCHS, batch_size=BATCH_SIZE, verbose=0)
```

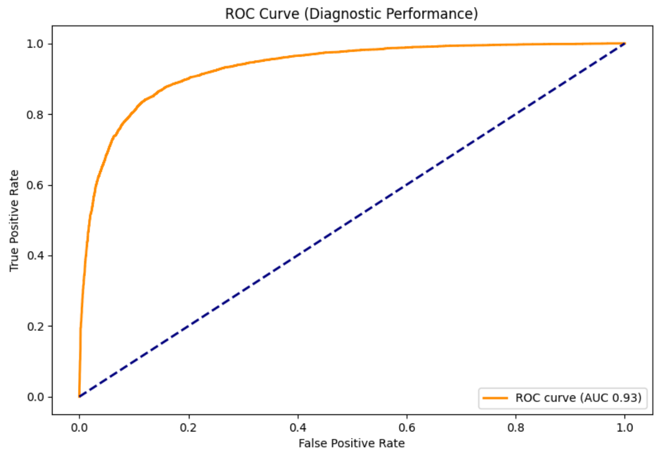
```
            # Scaling [cite: 170, 644]
            factor = c['n_k'] / len(y_train)
            local_weights_list.append([layer * factor for layer in local_model.get_weights()])

        # Secure Aggregation [cite: 170, 647]
        global_weights = [tf.math.reduce_sum(w, axis=0) for w in zip(*local_weights_list)]
        global_model.set_weights(global_weights)

        # Global Evaluation [cite: 652-655]
        loss, acc = global_model.evaluate([X_seq_test, X_emo_test], y_test, verbose=0)
        history['acc'].append(acc)
        history['loss'].append(loss)
        print(f"Round {r+1}: Global Accuracy = {acc:.4f} | Loss = {loss:.4f}")

    print("Cell 5 Complete: Federated Training Finished.")
```

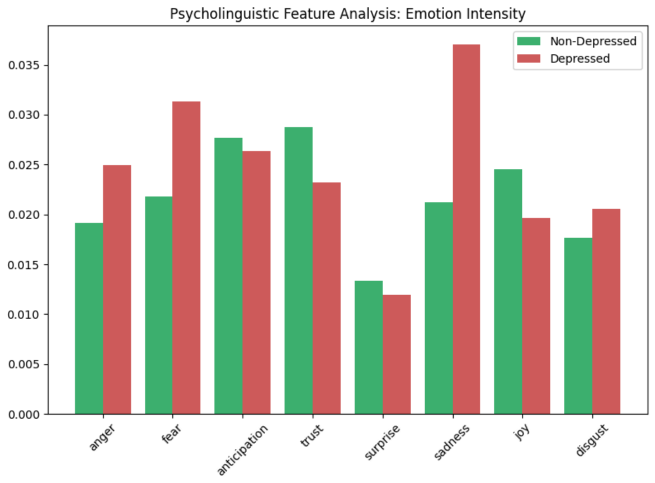
```
Starting Federated Training (20 Rounds) for 100k samples...
Round 1: Global Accuracy = 0.7390 | Loss = 0.5853
Round 2: Global Accuracy = 0.8239 | Loss = 0.3999
Round 3: Global Accuracy = 0.8278 | Loss = 0.4057
Round 4: Global Accuracy = 0.8318 | Loss = 0.4352
Round 5: Global Accuracy = 0.8324 | Loss = 0.4560
Round 6: Global Accuracy = 0.8390 | Loss = 0.4671
Round 7: Global Accuracy = 0.8402 | Loss = 0.4880
Round 8: Global Accuracy = 0.8427 | Loss = 0.4963
Round 9: Global Accuracy = 0.8442 | Loss = 0.5321
Round 10: Global Accuracy = 0.8476 | Loss = 0.5244
Round 11: Global Accuracy = 0.8471 | Loss = 0.5273
Round 12: Global Accuracy = 0.8486 | Loss = 0.5470
Round 13: Global Accuracy = 0.8514 | Loss = 0.5810
Round 14: Global Accuracy = 0.8526 | Loss = 0.5696
Round 15: Global Accuracy = 0.8529 | Loss = 0.5758
Round 16: Global Accuracy = 0.8548 | Loss = 0.5656
Round 17: Global Accuracy = 0.8569 | Loss = 0.5873
Round 18: Global Accuracy = 0.8569 | Loss = 0.6153
Round 19: Global Accuracy = 0.8586 | Loss = 0.6086
Round 20: Global Accuracy = 0.8586 | Loss = 0.5903
Cell 5 Complete: Federated Training Finished.
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_curve, auc

# 1. Final Metrics [cite: 718-722]
preds_prob = global_model.predict([X_seq_test, X_emo_test], verbose=0)
preds = (preds_prob > 0.5).astype(int).flatten()

print("\n=== FINAL RESEARCH RESULTS ===")
print(classification_report(y_test, preds, target_names=['Non-Depressed', 'Depressed']))

# 2. Advanced Visualizations [cite: 802-842]
plt.figure(figsize=(16, 6))

# ROC Curve [cite: 802-817]
plt.subplot(1, 2, 1)
fpr, tpr, _ = roc_curve(y_test, preds_prob)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.title('ROC Curve (Diagnostic Performance)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")

# Psycholinguistic Analysis [cite: 824-842]
plt.subplot(1, 2, 2)
emotions = ['anger', 'fear', 'anticipation', 'trust', 'surprise', 'sadness', 'joy', 'disgust']
avg_d = np.mean(X_emo_test[y_test == 1], axis=0)
avg_n = np.mean(X_emo_test[y_test == 0], axis=0)
x = np.arange(len(emotions))
plt.bar(x - 0.2, avg_n, 0.4, label='Non-Depressed', color='mediumseagreen')
plt.bar(x + 0.2, avg_d, 0.4, label='Depressed', color='indianred')
plt.xticks(x, emotions, rotation=45)
plt.title('Psycholinguistic Feature Analysis: Emotion Intensity')
plt.legend()
plt.tight_layout()
plt.show()
```

```
print("Cell 6 Complete: Full Research Process Finished.")
```

```
=== FINAL RESEARCH RESULTS ===
               precision    recall  f1-score   support

Non-Depressed       0.87      0.85      0.86     10000
    Depressed       0.85      0.87      0.86     10000

     accuracy                           0.86     20000
    macro avg       0.86      0.86      0.86     20000
 weighted avg       0.86      0.86      0.86     20000
```

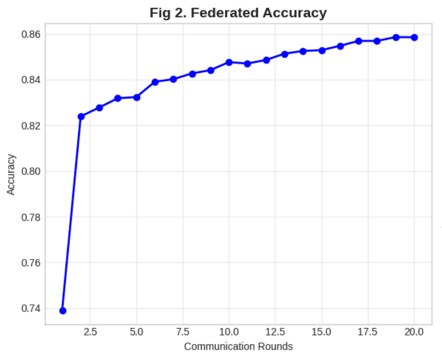


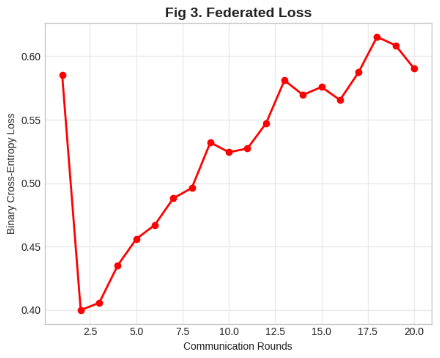


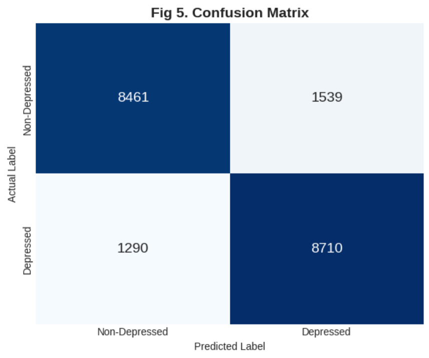
```
Cell 6 Complete: Full Research Process Finished.
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set research aesthetic
plt.style.use('seaborn-v0_8-whitegrid')
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# 1. Federated Accuracy Trajectory [cite: 209, 685-690]
axes[0].plot(range(1, GLOBAL_ROUNDS + 1), history['acc'], 'b-o', linewidth=2, markersize=6)
axes[0].set_title('Fig 2. Federated Accuracy', fontsize=14, fontweight='bold')
axes[0].set_xlabel('Communication Rounds')
axes[0].set_ylabel('Accuracy')
axes[0].grid(True, alpha=0.3)

# 2. Federated Loss Curve [cite: 231, 692-697]
axes[1].plot(range(1, GLOBAL_ROUNDS + 1), history['loss'], 'r-o', linewidth=2, markersize=6)
axes[1].set_title('Fig 3. Federated Loss', fontsize=14, fontweight='bold')
axes[1].set_xlabel('Communication Rounds')
axes[1].set_ylabel('Binary Cross-Entropy Loss')
axes[1].grid(True, alpha=0.3)

# 3. Confusion Matrix (Diagnostic Precision) [cite: 279, 699-706]
cm = confusion_matrix(y_test, preds)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axes[2],
            xticklabels=['Non-Depressed', 'Depressed'],
            yticklabels=['Non-Depressed', 'Depressed'],
            cbar=False, annot_kws={"size": 14})
axes[2].set_title('Fig 5. Confusion Matrix', fontsize=14, fontweight='bold')
axes[2].set_ylabel('Actual Label')
axes[2].set_xlabel('Predicted Label')
```

```
plt.tight_layout()
plt.show()

print("Basic Performance Visualizations Complete.")
```

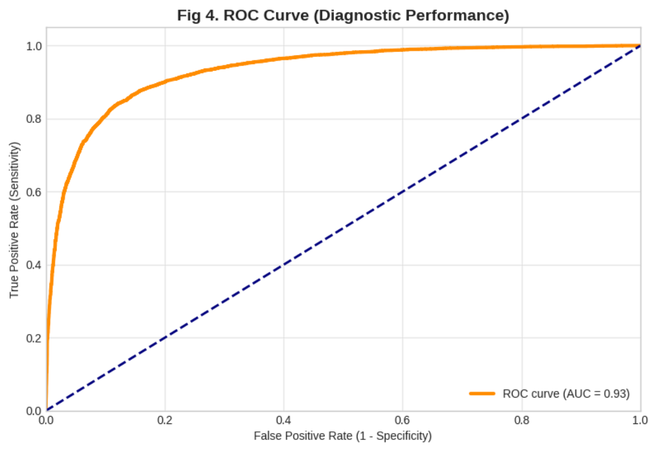


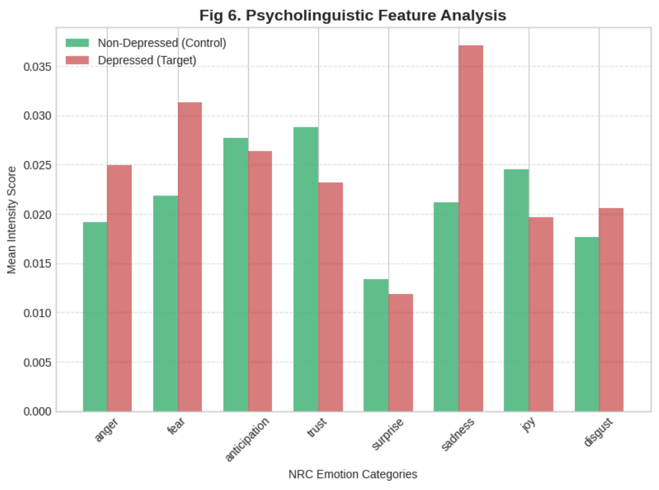




```
Basic Performance Visualizations Complete.
```

```
from sklearn.metrics import roc_curve, auc

plt.figure(figsize=(16, 6))

# 4. ROC Curve (Diagnostic Robustness) [cite: 263, 802-817]
plt.subplot(1, 2, 1)
fpr, tpr, _ = roc_curve(y_test, preds_prob)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='darkorange', lw=3, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('Fig 4. ROC Curve (Diagnostic Performance)', fontsize=14, fontweight='bold')
plt.legend(loc="lower right")
plt.grid(True, alpha=0.4)

# 5. Psycholinguistic Feature Analysis (XAI) [cite: 290, 824-842]
plt.subplot(1, 2, 2)
emotions = ['anger', 'fear', 'anticipation', 'trust', 'surprise', 'sadness', 'joy', 'disgust']
avg_depressed = np.mean(X_emo_test[y_test == 1], axis=0)
avg_control = np.mean(X_emo_test[y_test == 0], axis=0)

x = np.arange(len(emotions))
width = 0.35

plt.bar(x - width/2, avg_control, width, label='Non-Depressed (Control)', color='mediumseagreen', alpha=0.8)
plt.bar(x + width/2, avg_depressed, width, label='Depressed (Target)', color='indianred', alpha=0.8)

plt.xlabel('NRC Emotion Categories')
plt.ylabel('Mean Intensity Score')
plt.title('Fig 6. Psycholinguistic Feature Analysis', fontsize=14, fontweight='bold')
plt.xticks(x, emotions, rotation=45)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()

print("Advanced Research Visualizations Complete.")
```





```
Advanced Research Visualizations Complete.
```

Start coding or generate with AI.