emobank-PAD

October 10, 2024

1 Step 1: Import Libraries

```
[122]: import pandas as pd
       import numpy as np
       import seaborn as sns
       import matplotlib.pyplot as plt
       from sklearn.model_selection import train_test_split
       from sklearn.decomposition import LatentDirichletAllocation
       from sklearn.feature_extraction.text import TfidfVectorizer
       from sklearn.linear_model import Ridge
       from sklearn.metrics import mean_squared_error
       from sklearn.ensemble import RandomForestRegressor
       from sklearn.neural_network import MLPRegressor
       import requests
       import io
       from sklearn.feature_extraction.text import CountVectorizer
```

2 Step 2: Load Dataset

```
[123]: def get_dataset(url = "https://raw.githubusercontent.com/JULIELab/EmoBank/
         ↪master/corpus/emobank.csv"):
           response = requests.get(url)
           if response.status_code == 200:
               return pd.read_csv(io.StringIO(response.text))
           else:
               raise Exception(f"Failed to download the dataset. Status code:␣
         ↪{response.status_code}")
```

```
[124]: data = get_dataset()
```

```
[125]: data.head()
```

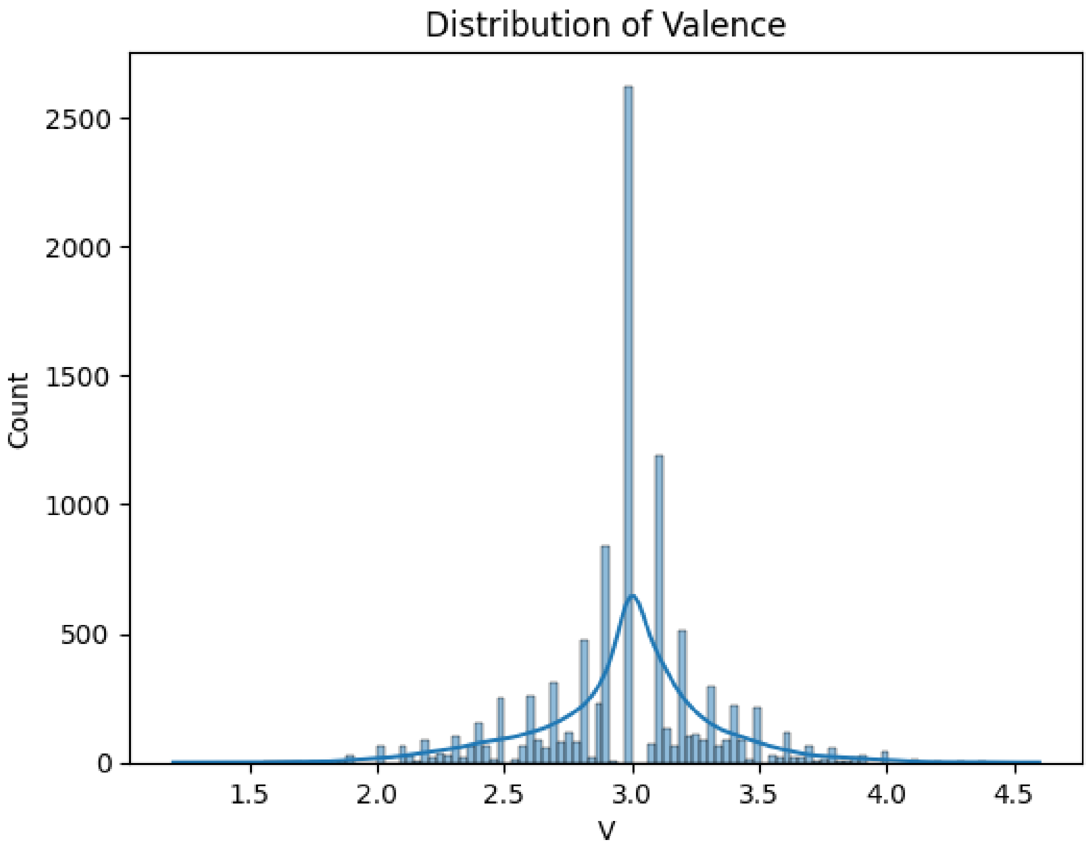
```
[125]:                     id  split     V     A     D  \
       0  110CYL068_1036_1079  train  3.00  3.00  3.20
       1  110CYL068_1079_1110   test  2.80  3.10  2.80
       2  110CYL068_1127_1130  train  3.00  3.00  3.00
```

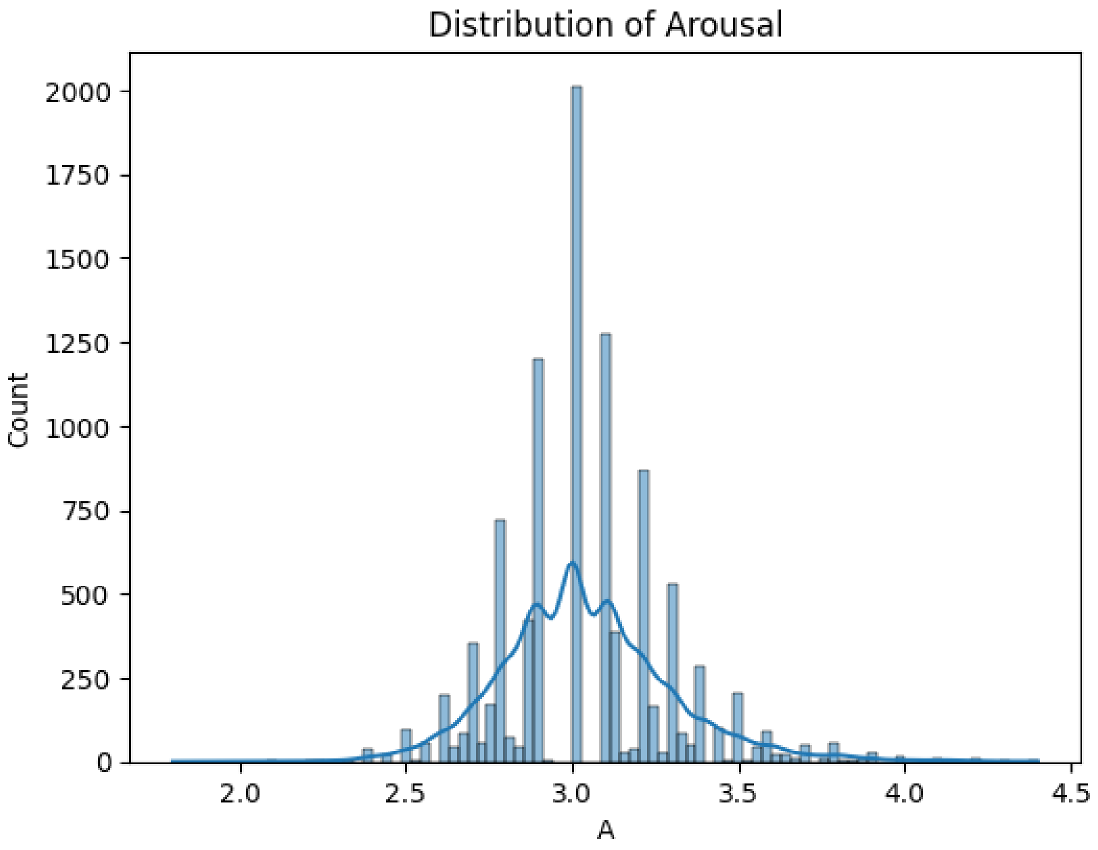
```
3  110CYL068_1137_1188  train  3.44  3.00  3.22
4  110CYL068_1189_1328  train  3.55  3.27  3.46

                                              text
0
Remember what she said in my last letter? "
1
If I wasn't working here.
2
.."
3
Goodwill helps people get off of public assistance.
4  Sherry learned through our Future Works class that she could rise out of the
mire of the welfare system and support her family.
```

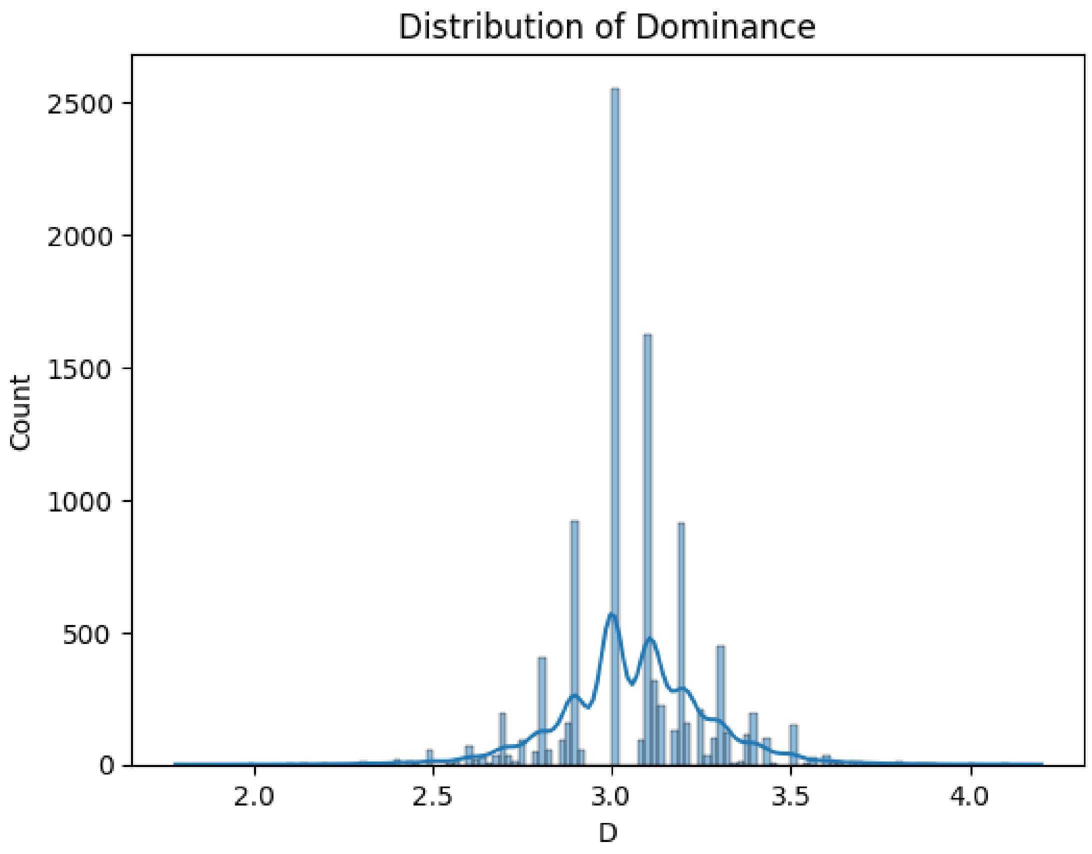
3 Step 3: Exploratory Data Analysis

Visualize the distribution of Valence, Arousal, and Dominance

[126]:
```python
sns.histplot(data['V'], kde=True).set_title('Distribution of Valence')
plt.show()

sns.histplot(data['A'], kde=True).set_title('Distribution of Arousal')
plt.show()

sns.histplot(data['D'], kde=True).set_title('Distribution of Dominance')
plt.show()
```


Distribution of Valence
Count
V
0
500
1000
1500
2000
2500
1.5
2.0
2.5
3.0
3.5
4.0
4.5


Distribution of Arousal
Count
A
0
250
500
750
1000
1250
1500
1750
2000
2.0
2.5
3.0
3.5
4.0
4.5

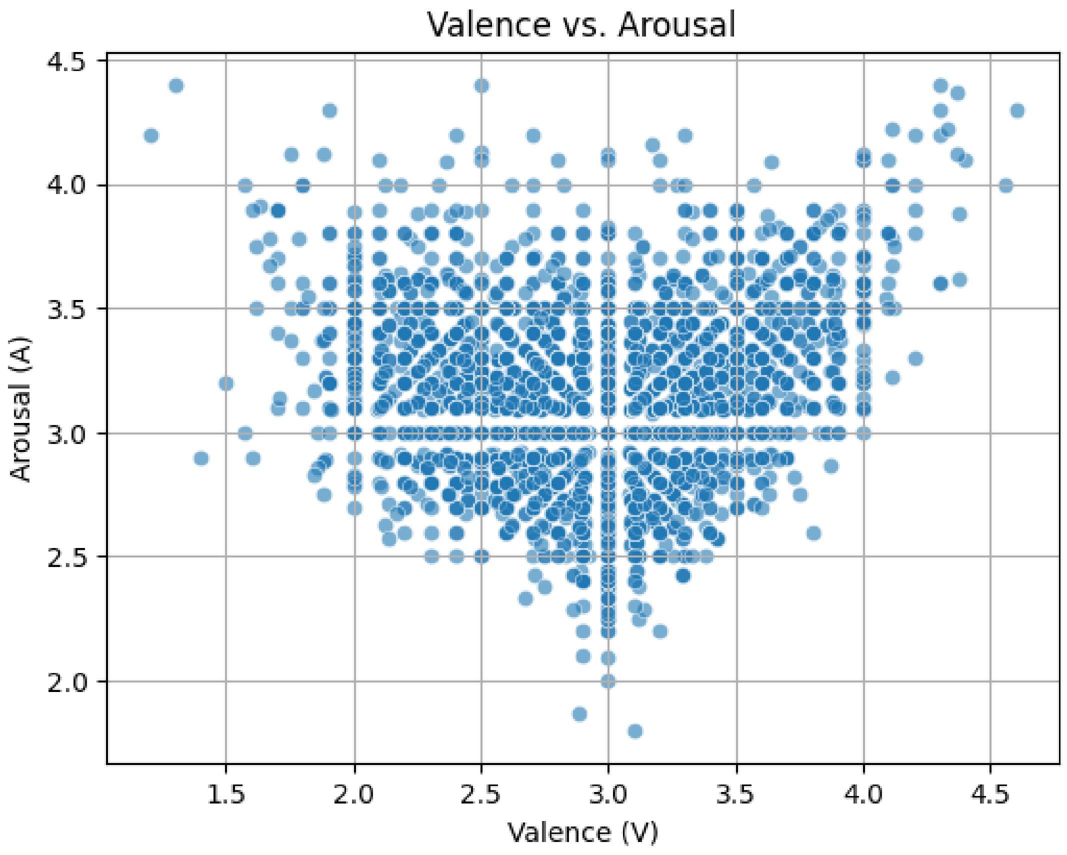


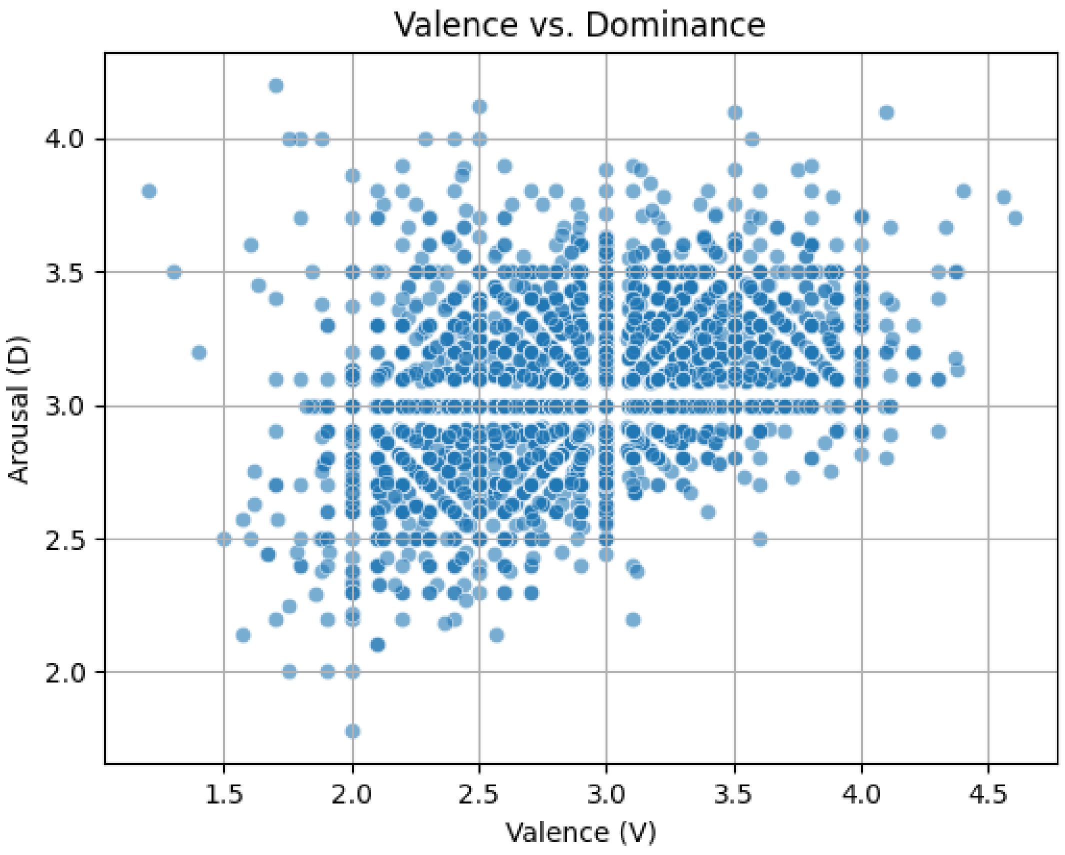
4 Step 4: Preprocessing

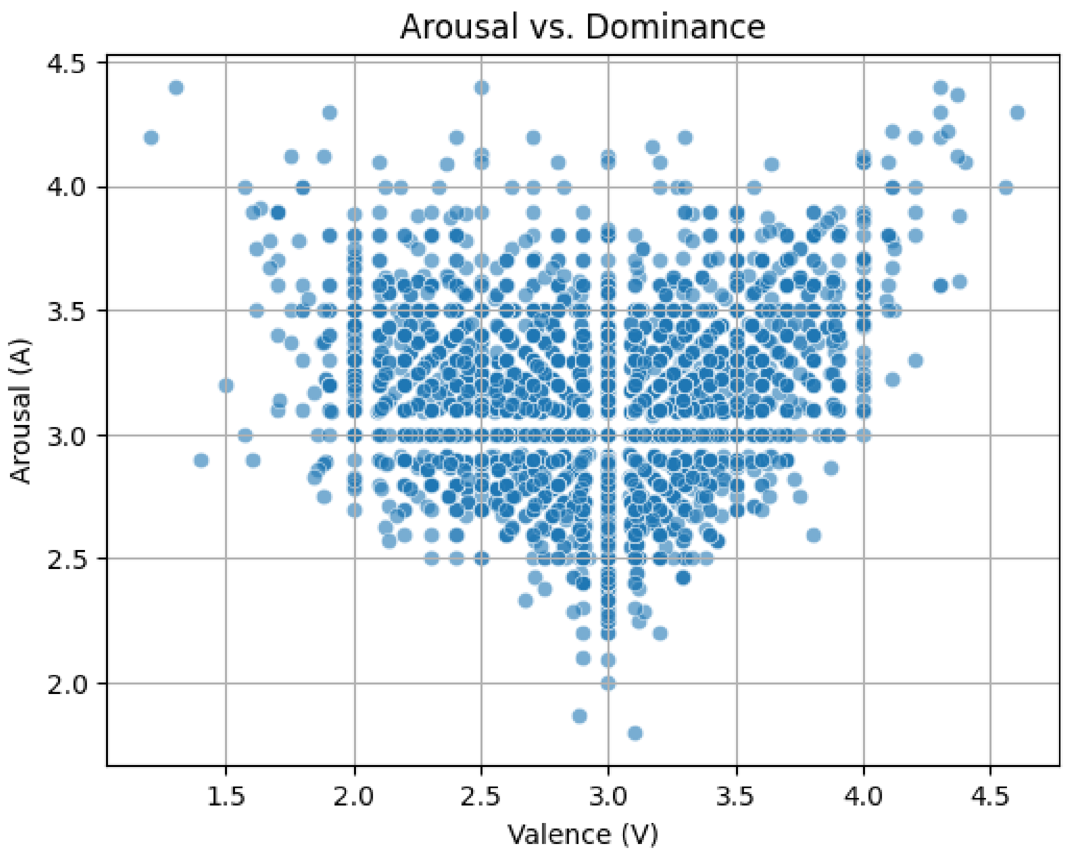
Split into training and testing sets

[127]:
```
# Drop rows with NaN in V or A
data = data.dropna(subset=['V', 'A', 'D'])

# Scatter plot for Valence (V) and Arousal (A)
sns.scatterplot(x='V', y='A', data=data, alpha=0.6)

# Customize the plot
plt.xlabel("Valence (V)")
plt.ylabel("Arousal (A)")
plt.title("Valence vs. Arousal")

# Add grid lines
plt.grid(True)

# Show the plot
```

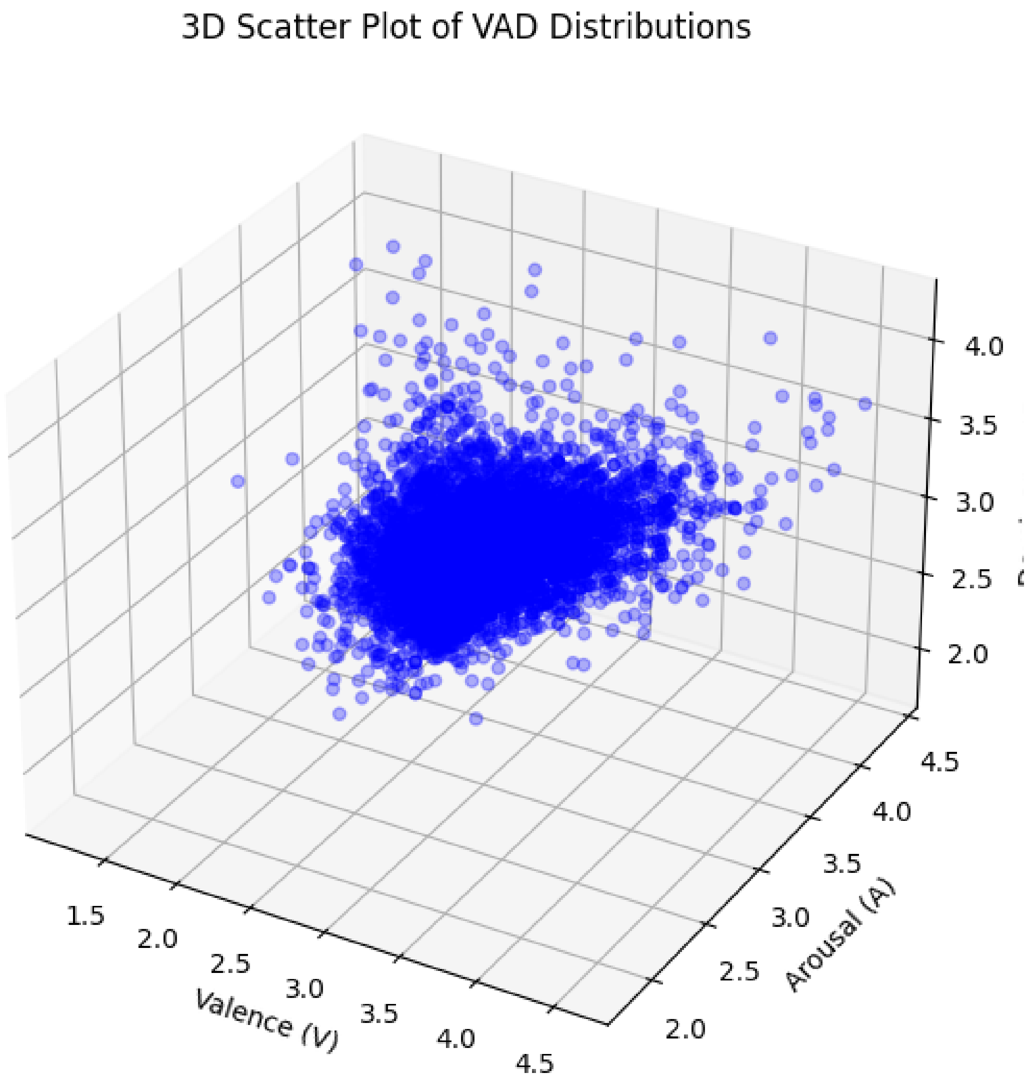
```python
plt.show()

# Scatter plot for Valence (V) and Dominance (D)
sns.scatterplot(x='V', y='D', data=data, alpha=0.6)

# Customize the plot
plt.xlabel("Valence (V)")
plt.ylabel("Arousal (D)")
plt.title("Valence vs. Dominance")

# Add grid lines
plt.grid(True)

# Show the plot
plt.show()

# Scatter plot for Arousal (A) and Dominance (D)
sns.scatterplot(x='V', y='A', data=data, alpha=0.6)

# Customize the plot
plt.xlabel("Valence (V)")
plt.ylabel("Arousal (A)")
plt.title("Arousal vs. Dominance")

# Add grid lines
plt.grid(True)

# Show the plot
plt.show()
```


Valence vs. Arousal
Arousal (A)
Valence (V)
4.5
4.0
3.5
3.0
2.5
2.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5


Valence vs. Dominance
Arousal (D)
Valence (V)
4.0
3.5
3.0
2.5
2.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5



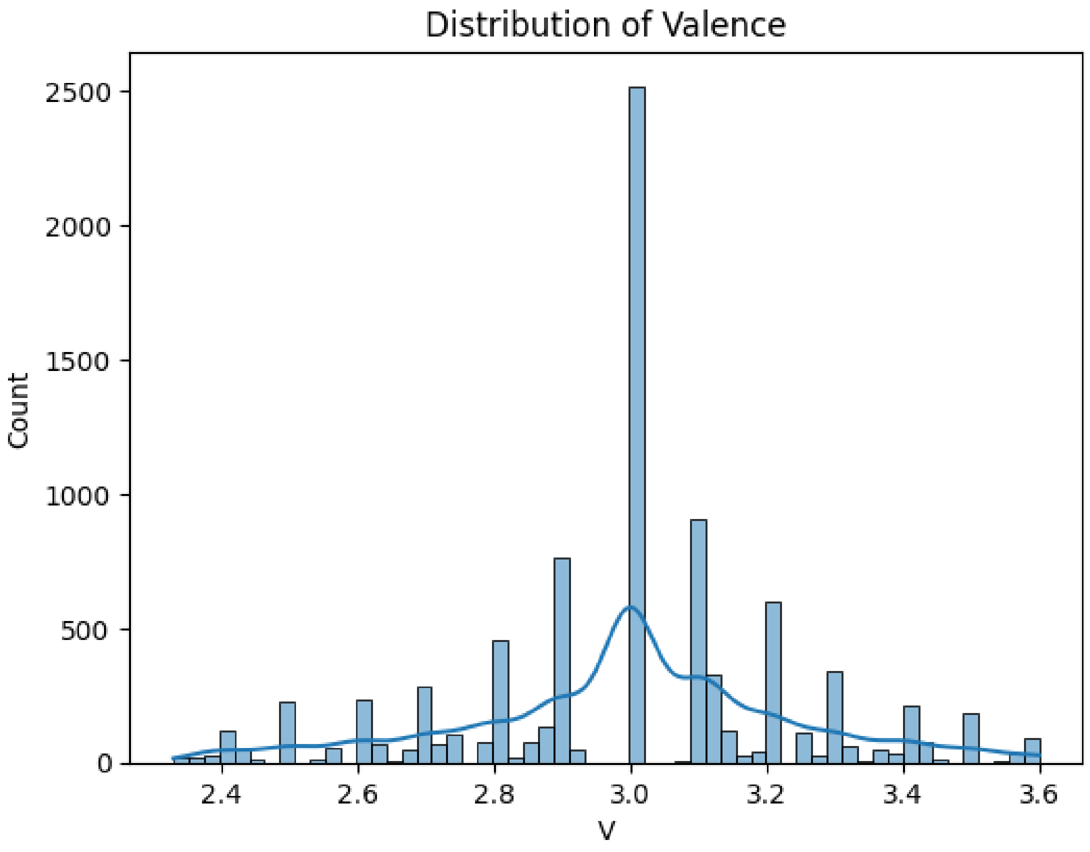
```
[128]: # Create a figure for 3D plot
       fig = plt.figure(figsize=(10, 7))
       ax = fig.add_subplot(111, projection='3d')

       ax.scatter(data['V'], data['A'], data['D'], color='blue', label='Actual VAD',␣
        ↪alpha=0.3)

       ax.set_xlabel('Valence (V)')
       ax.set_ylabel('Arousal (A)')
       ax.set_zlabel('Dominance (D)')
       ax.set_title('3D Scatter Plot of VAD Distributions')

       # Show the plot
       plt.show()
```

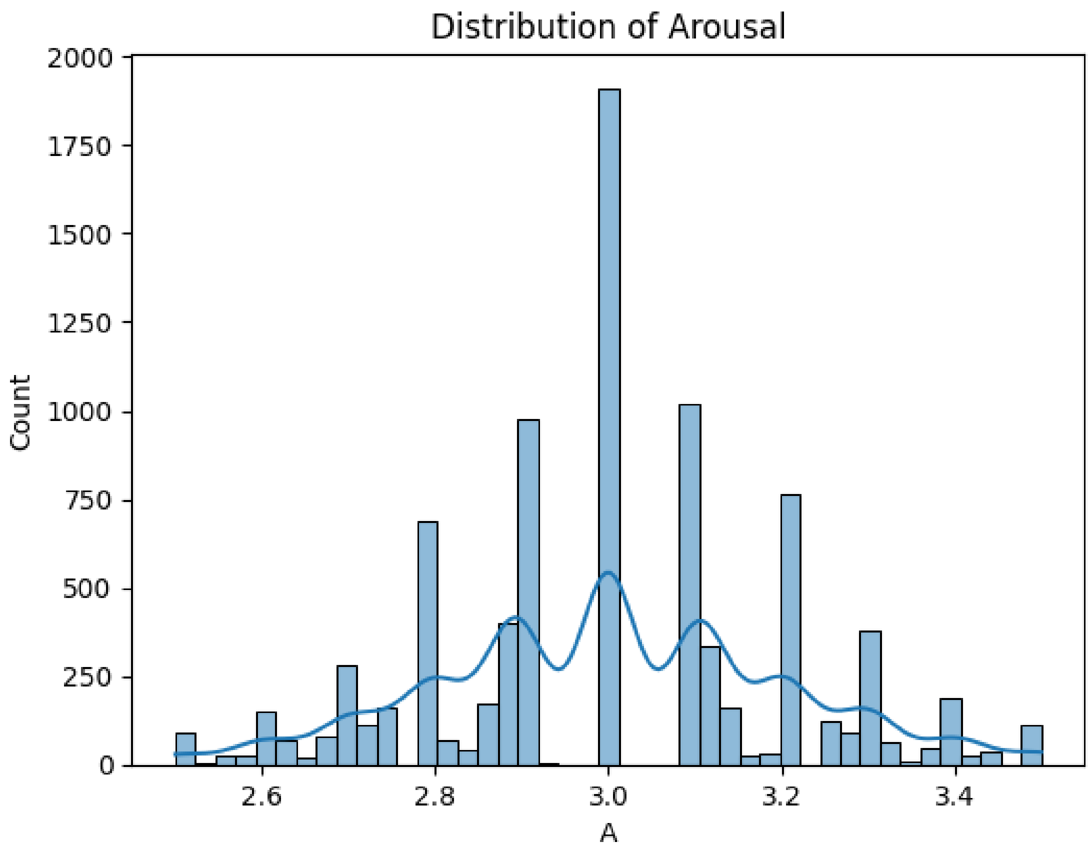


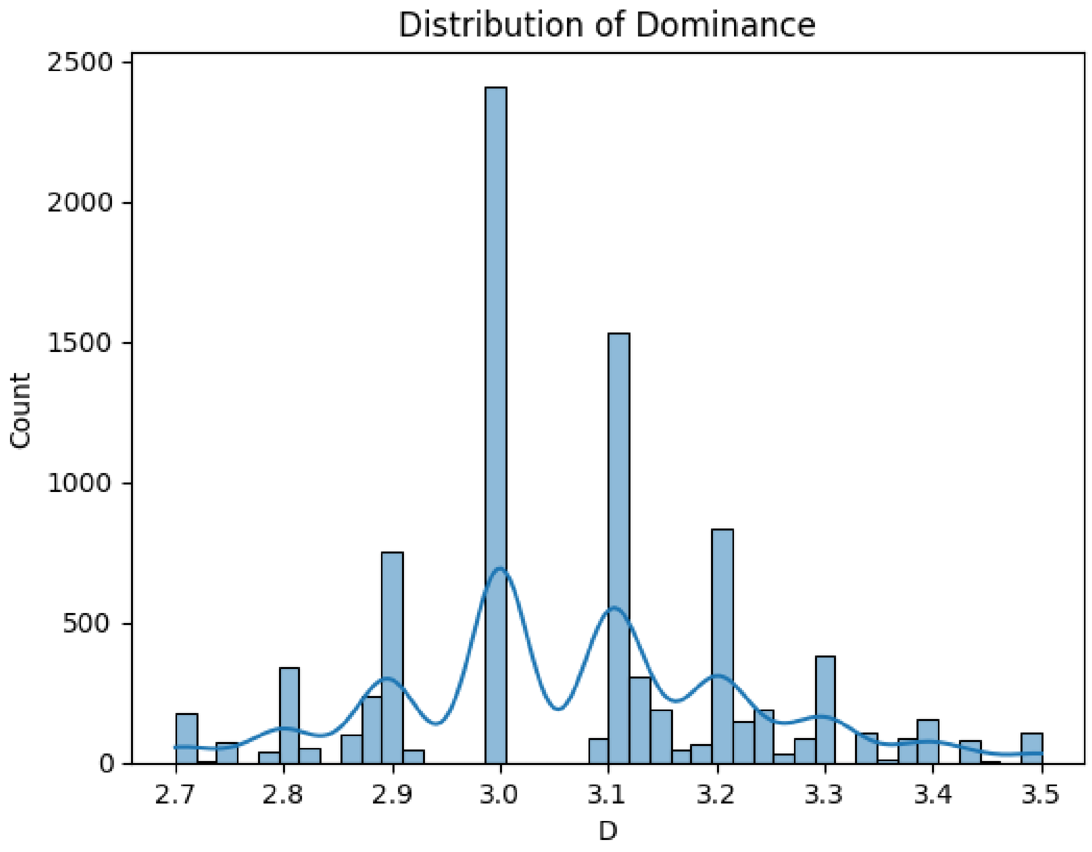
[129]:
```
# Define a function to remove outliers using IQR
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Apply the function to Valence (V), Arousal (A), and Dominance (D)
data = remove_outliers(data, 'V')
data = remove_outliers(data, 'A')
data = remove_outliers(data, 'D')
```

```python
# Replace NaN with a placeholder (e.g., an empty string)
data['text'] = data['text'].fillna('')

# Continue with train-test split and TF-IDF processing
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)

# Using CountVectorizer for LDA
count_vectorizer = CountVectorizer(max_features=5000, stop_words='english')
data_counts = count_vectorizer.fit_transform(data['text'])
X_train_counts  = count_vectorizer.transform(train_data['text'])
X_test_counts = count_vectorizer.transform(test_data['text'])

# Applying LDA
lda = LatentDirichletAllocation(n_components=20, random_state=42)
lda.fit_transform(data_counts)
X_train = lda.transform(X_train_counts)
X_test = lda.transform(X_test_counts)

# Choose targets for Valence, Arousal, and Dominance
v_train, v_test = train_data['V'], test_data['V']
a_train, a_test = train_data['A'], test_data['A']
d_train, d_test = train_data['D'], test_data['D']
```

[130]:
```python
sns.histplot(data['V'], kde=True).set_title('Distribution of Valence')
plt.show()

sns.histplot(data['A'], kde=True).set_title('Distribution of Arousal')
plt.show()

sns.histplot(data['D'], kde=True).set_title('Distribution of Dominance')
plt.show()
```


Distribution of Valence
Count
2500
2000
1500
1000
500
0
2.4
2.6
2.8
3.0
3.2
3.4
3.6
V


Distribution of Arousal
Count
A
0
250
500
750
1000
1250
1500
1750
2000
2.6
2.8
3.0
3.2
3.4



5 Step 5: Build Models

6 5.1 Ridge

[131]:
```
model_v = Ridge(alpha=1.0)
model_v.fit(X_train, v_train)

model_a = Ridge(alpha=1.0)
model_a.fit(X_train, a_train)

model_d = Ridge(alpha=1.0)
model_d.fit(X_train, d_train)

ridge_model = (model_v, model_a, model_d)
```

6.1 5.2 Random Forest

[132]:
```
model_v = RandomForestRegressor(n_estimators=100, random_state=42)
model_v.fit(X_train, v_train)

model_a = RandomForestRegressor(n_estimators=100, random_state=42)
model_a.fit(X_train, a_train)

model_d = RandomForestRegressor(n_estimators=100, random_state=42)
model_d.fit(X_train, d_train)

rf_model = (model_v, model_a, model_d)
```

6.2 5.3 Neural Network Regression

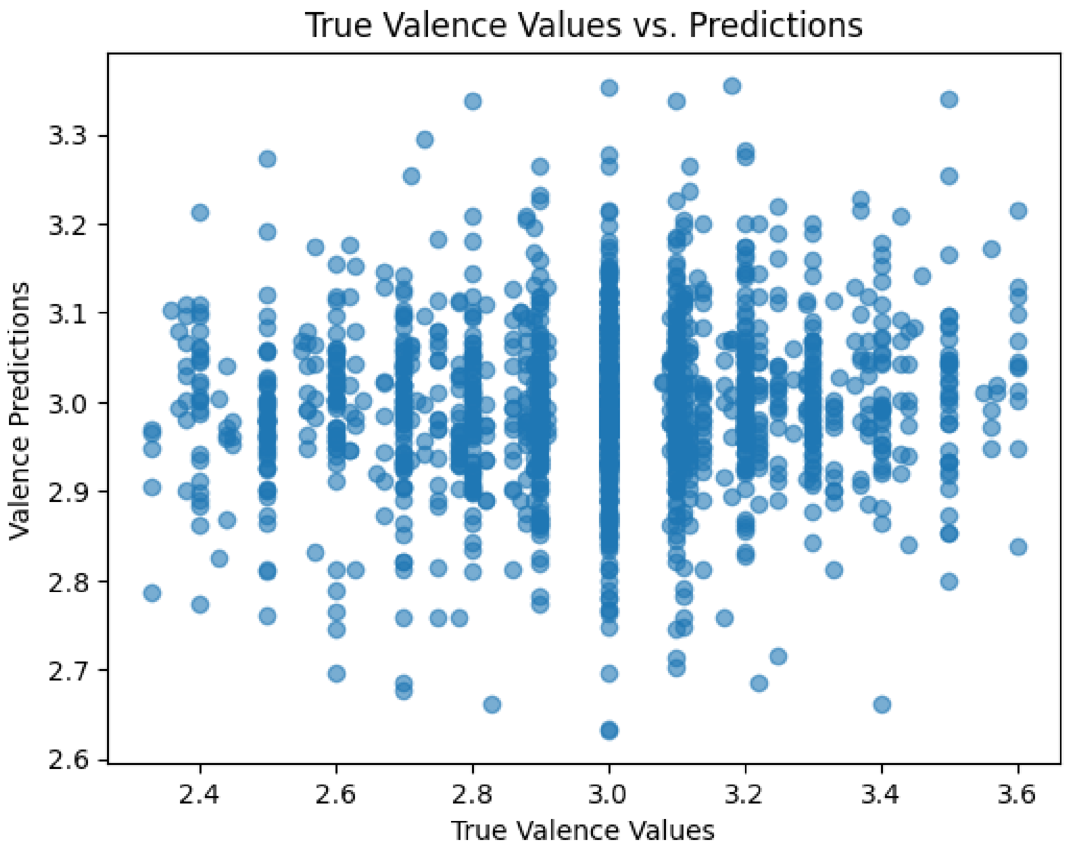
[133]:
```
model_v = MLPRegressor(hidden_layer_sizes=(50, 30), activation='relu',␣
  ↪solver='adam', random_state=42, max_iter=500)
model_v.fit(X_train, v_train)

model_a = MLPRegressor(hidden_layer_sizes=(50, 30), activation='relu',␣
  ↪solver='adam', random_state=42, max_iter=500)
model_a.fit(X_train, a_train)

model_d = MLPRegressor(hidden_layer_sizes=(50, 30), activation='relu',␣
  ↪solver='adam', random_state=42, max_iter=500)
model_d.fit(X_train, d_train)

mlp_model = (model_v, model_a, model_d)
```

[134]:
```
# models = [ridge_model, rf_model, mlp_model]
models = [rf_model]
```

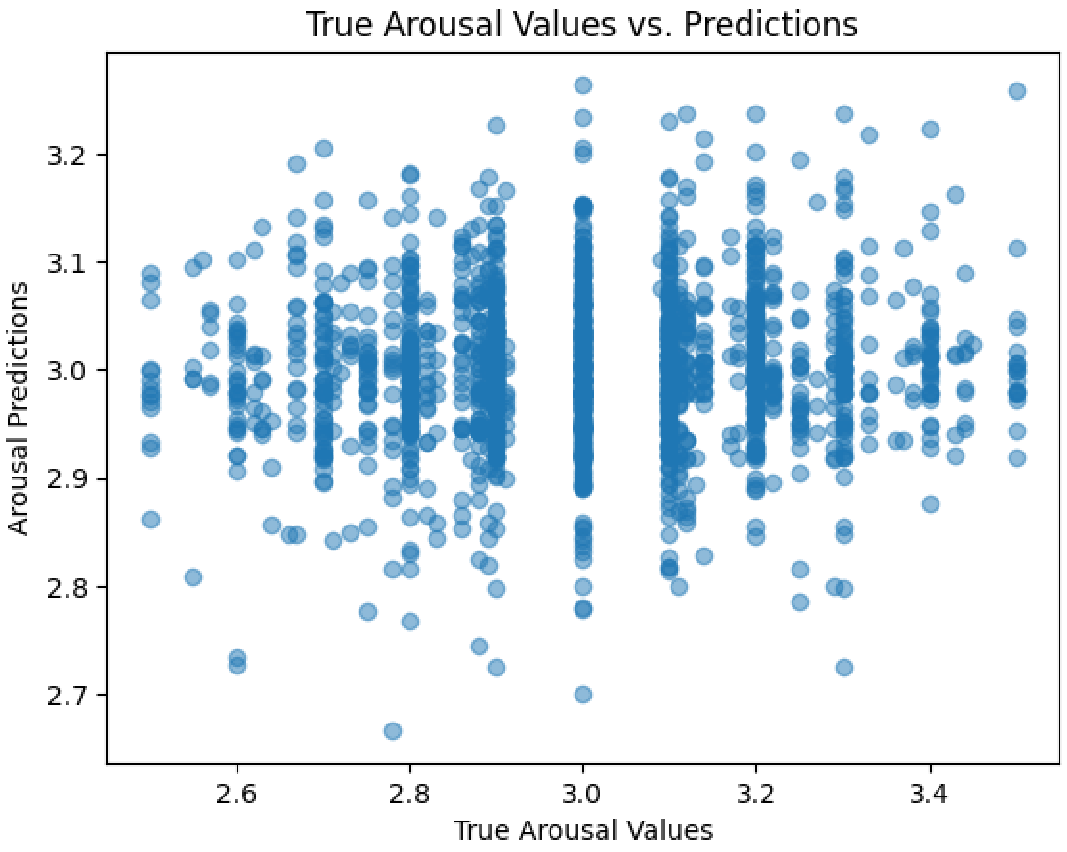
7 Step 6: Evaluate the Models

[135]:
```
from mpl_toolkits.mplot3d import Axes3D

for model in models:
    print(f"For model:", model)
    (model_v, model_a, model_d) = model

    v_pred = model_v.predict(X_test)

    # Calculate Mean Squared Error (MSE)
    mse = mean_squared_error(v_test, v_pred)
    print("Mean Squared Error:", mse)
    # Visualize predictions vs. true values
    plt.scatter(v_test, v_pred, alpha=0.6)
    plt.xlabel("True Valence Values")
    plt.ylabel("Valence Predictions")
    plt.title("True Valence Values vs. Predictions")
    plt.show()

    a_pred = model_a.predict(X_test)

    # Calculate Mean Squared Error (MSE)
    mse = mean_squared_error(a_test, a_pred)
    print("Mean Squared Error:", mse)

    # Visualize predictions vs. true values
    plt.scatter(a_test, a_pred, alpha=0.5)
    plt.xlabel("True Arousal Values")
    plt.ylabel("Arousal Predictions")
    plt.title("True Arousal Values vs. Predictions")
    plt.show()

    d_pred = model_d.predict(X_test)

    # Calculate Mean Squared Error (MSE)
    mse = mean_squared_error(d_test, d_pred)
    print("Mean Squared Error:", mse)

    # Visualize predictions vs. true values
    plt.scatter(d_test, d_pred, alpha=0.6)
    plt.xlabel("True Dominance Values")
    plt.ylabel("Dominance Predictions")
    plt.title("True Dominance Values vs. Predictions")
    plt.show()

    # Create a figure for 3D plot
    fig = plt.figure(figsize=(10, 7))
    ax = fig.add_subplot(111, projection='3d')

    # Scatter true VAD values (from the test set)
    ax.scatter(v_test, a_test, d_test, color='blue', label='Actual VAD',␣
↪alpha=0.3)

    # Scatter predicted VAD values
    ax.scatter(v_pred, a_pred, d_pred, color='red', label='Predicted VAD',␣
↪alpha=0.4)

    # Set labels
    ax.set_xlabel('Valence (V)')
    ax.set_ylabel('Arousal (A)')
    ax.set_zlabel('Dominance (D)')
    ax.set_title('3D Scatter Plot of Predicted vs. Actual VAD')

    # Add a legend
    ax.legend()

    # Show the plot
    plt.show()
```

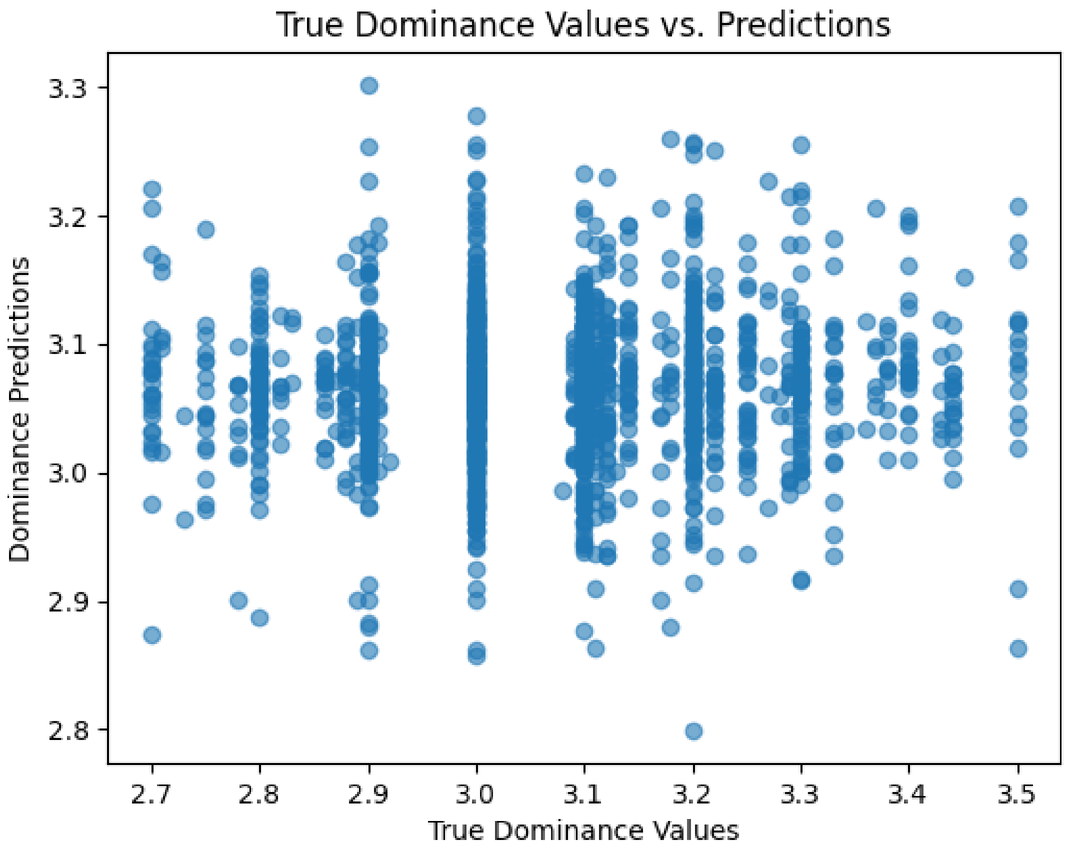
```
For model: (RandomForestRegressor(random_state=42),
RandomForestRegressor(random_state=42), RandomForestRegressor(random_state=42))
Mean Squared Error: 0.06393539720984456
```

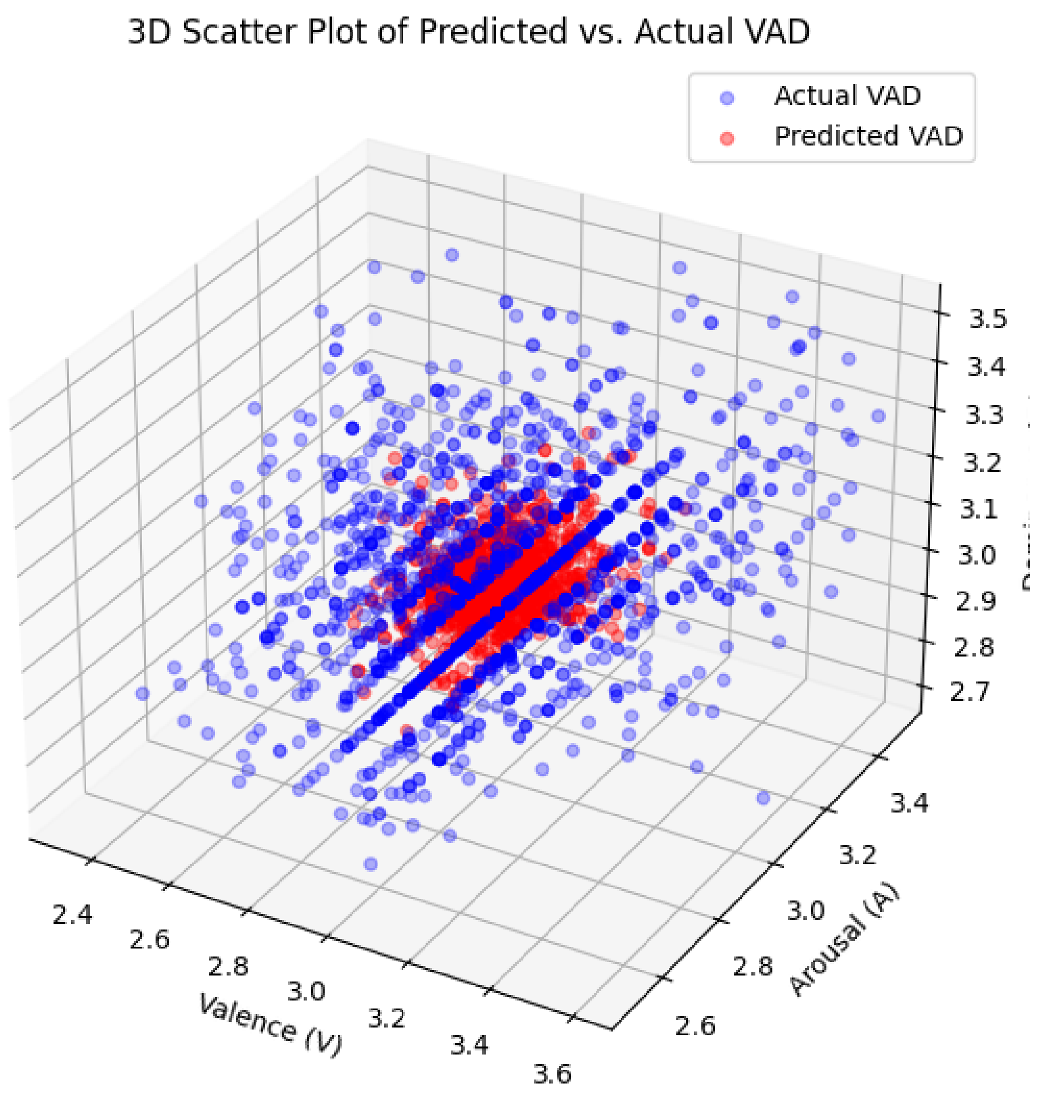


```
Mean Squared Error: 0.04362417508990723
```



```
Mean Squared Error: 0.027560723612094693
```


True Dominance Values vs. Predictions
Dominance Predictions
True Dominance Values
3.3
3.2
3.1
3.0
2.9
2.8
2.7
2.8
2.9
3.0
3.1
3.2
3.3
3.4
3.5



8 Step7: Making Predictions

[136]:
```
def predict(message, model):
    (model_v, model_a, model_d) = model
    v = model_v.predict(message)
    a = model_a.predict(message)
    d = model_d.predict(message)
    return v, a, d
```

[137]:
```
# Get a few sample texts from the test data for prediction
sampled_data = test_data.sample(25)  # Select 5 samples from the test data
# sampled_data = test_data
```

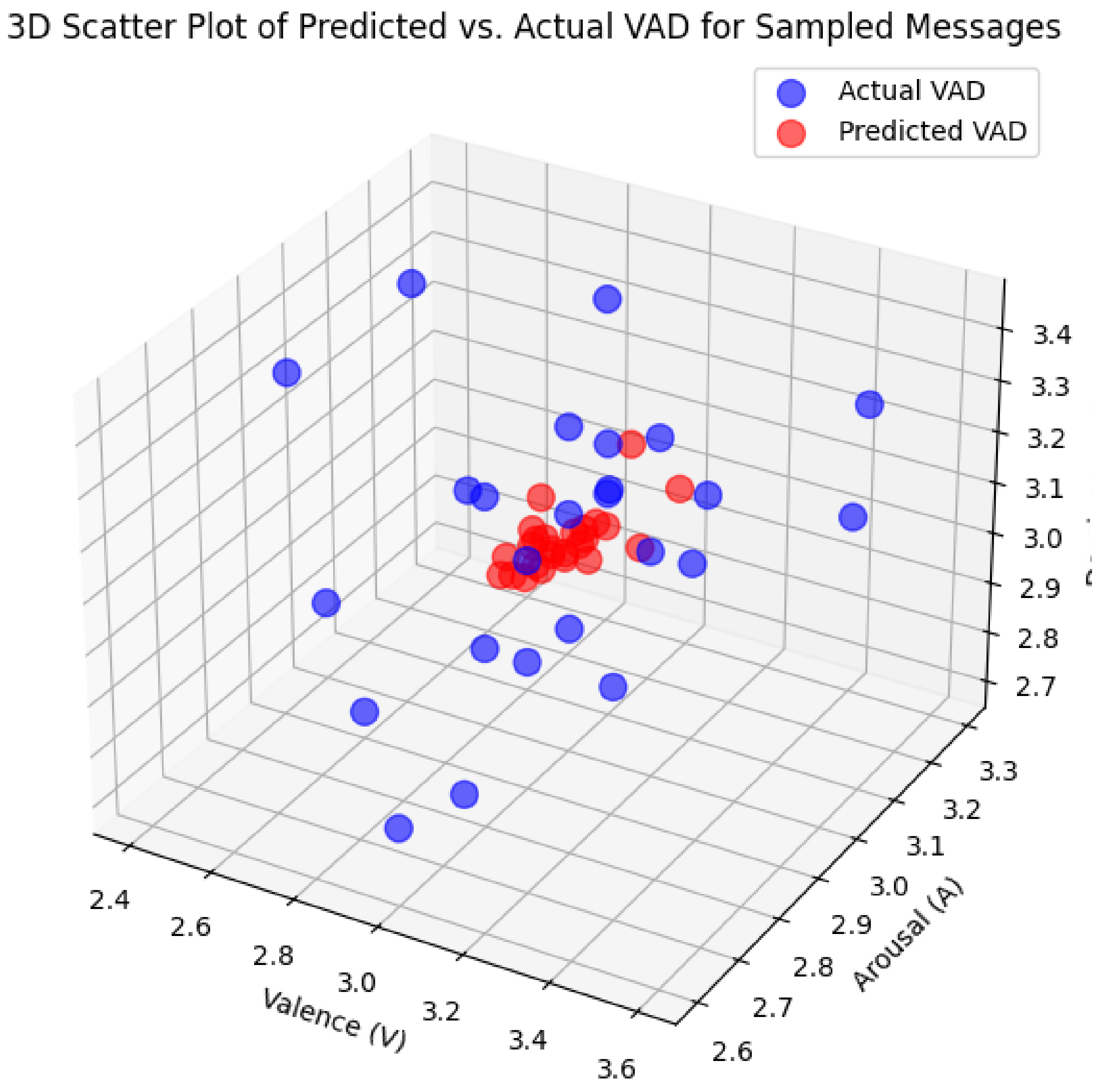
```
messages = sampled_data['text'].values
messages_counts = count_vectorizer.transform(messages)
input_message = lda.transform(messages_counts)

for model in models:
    print(f"For model:", model)
    (model_v, model_a, model_d) = model

    # Get predictions for the new messages
    v_pred, a_pred, d_pred = predict(input_message, model)

    # Display results for each message
    for i, msg in enumerate(messages):
        print(f"Message: {msg}")
        print(f"Predicted Valence: {v_pred[i]:.2f}, Arousal: {a_pred[i]:.2f},␣
↪Dominance: {d_pred[i]:.2f}")
        print(f"Actual Valence: {sampled_data['V'].values[i]:.2f}, Arousal:␣
↪{sampled_data['A'].values[i]:.2f}, Dominance: {sampled_data['D'].values[i]:.
↪2f}\n")

    # Create a 3D plot
    fig = plt.figure(figsize=(10, 7))
    ax = fig.add_subplot(111, projection='3d')

    # Actual VAD values for the 5 sampled messages
    v_actual = sampled_data['V'].values
    a_actual = sampled_data['A'].values
    d_actual = sampled_data['D'].values

    # Scatter actual values
    ax.scatter(v_actual, a_actual, d_actual, color='blue', label='Actual VAD',␣
↪alpha=0.6, s=100)

    # Scatter predicted values
    ax.scatter(v_pred, a_pred, d_pred, color='red', label='Predicted VAD',␣
↪alpha=0.6, s=100)

    # Set labels and title
    ax.set_xlabel('Valence (V)')
    ax.set_ylabel('Arousal (A)')
    ax.set_zlabel('Dominance (D)')
    ax.set_title('3D Scatter Plot of Predicted vs. Actual VAD for Sampled␣
↪Messages')

    # Add a legend
    ax.legend()
```

```python
    # Show the plot
    plt.show()
```

```
For model: (RandomForestRegressor(random_state=42),
RandomForestRegressor(random_state=42), RandomForestRegressor(random_state=42))
Message: Today Big Sisters serves over 1,000 girls in central Indiana.
Predicted Valence: 2.97, Arousal: 2.95, Dominance: 3.05
Actual Valence: 3.20, Arousal: 3.10, Dominance: 3.00

Message: Ninety seconds from the bomb drop, Tibbets put the plane on auto-pilot.
Predicted Valence: 2.97, Arousal: 2.95, Dominance: 3.10
Actual Valence: 2.80, Arousal: 3.30, Dominance: 3.30

Message: "an effective headcount-control program."
Predicted Valence: 2.99, Arousal: 3.02, Dominance: 3.06
Actual Valence: 3.11, Arousal: 2.89, Dominance: 3.00

Message: Culture buffs are well catered to, and there is always a varied program
of events, ranging from world-class concerts to local amateur dramatic
productions.
Predicted Valence: 2.92, Arousal: 3.12, Dominance: 2.99
Actual Valence: 3.17, Arousal: 3.17, Dominance: 3.08

Message: The monastery is strictly vegetarian, and visitors are warned not to
bring any meat with them.
Predicted Valence: 3.01, Arousal: 3.03, Dominance: 3.05
Actual Valence: 2.90, Arousal: 3.20, Dominance: 3.10

Message: We are seeking to build partnerships between businesses and our not-
for-profit agency.
Predicted Valence: 3.13, Arousal: 3.14, Dominance: 3.10
Actual Valence: 3.60, Arousal: 3.10, Dominance: 3.40

Message: This website and everything on it is in the Public Domain, so you may
use it freely, even without attribution.
Predicted Valence: 3.00, Arousal: 2.94, Dominance: 3.12
Actual Valence: 3.00, Arousal: 3.10, Dominance: 3.10

Message: Karnes, 87, is a retired history professor who lives in suburban Austin
with his wife, Virginia, who also worked for the 509th as a civilian bookkeeper
for six months.
Predicted Valence: 2.98, Arousal: 2.93, Dominance: 3.10
Actual Valence: 3.00, Arousal: 3.00, Dominance: 3.12

Message: Detroit is a ghost town, and it's inhabited by ghosts: the only people
left here are those who can't or won't leave.
Predicted Valence: 2.88, Arousal: 2.97, Dominance: 3.03
```

```
Actual Valence: 2.40, Arousal: 3.00, Dominance: 2.80

Message: "a great deal of money,"
Predicted Valence: 2.95, Arousal: 3.01, Dominance: 3.03
Actual Valence: 3.10, Arousal: 3.10, Dominance: 3.00

Message: "You don't see a meaning because there is no meaning,"
Predicted Valence: 2.99, Arousal: 3.04, Dominance: 3.03
Actual Valence: 2.56, Arousal: 2.78, Dominance: 3.44

Message: Dear Friend:
Predicted Valence: 3.04, Arousal: 2.95, Dominance: 3.09
Actual Valence: 3.30, Arousal: 2.80, Dominance: 3.00

Message: Brad 14. Oct, 2010 at 7:02 pm
Predicted Valence: 3.08, Arousal: 2.97, Dominance: 3.07
Actual Valence: 3.00, Arousal: 2.80, Dominance: 3.00

Message: "The only no-kill shelter that’s there when you need us" Little Shelter
is crowded with animals waiting for homes.
Predicted Valence: 2.89, Arousal: 2.97, Dominance: 3.00
Actual Valence: 3.00, Arousal: 3.00, Dominance: 3.29

Message: "Tasha --"
Predicted Valence: 2.98, Arousal: 3.01, Dominance: 3.02
Actual Valence: 3.00, Arousal: 2.90, Dominance: 3.10

Message: At long last, Nathan reached the end and set down his reading on the
table between them.
Predicted Valence: 3.11, Arousal: 3.06, Dominance: 3.04
Actual Valence: 3.00, Arousal: 2.90, Dominance: 2.90

Message: Pan-Mayanism has succeeded in part due to its focus on issues of
ethnicity; it largely avoids specifically political overtones, instead
concentrating on the development of the reemerging Mayan culture (England 2003:
734).
Predicted Valence: 2.92, Arousal: 3.02, Dominance: 3.12
Actual Valence: 3.00, Arousal: 2.60, Dominance: 2.80

Message: AND this time apart will make you stronger and make you realize
marriage is work and you need to put some in or the marriage dies and you are
left with an empty hole in your heart.
Predicted Valence: 3.02, Arousal: 3.05, Dominance: 3.08
Actual Valence: 2.50, Arousal: 3.25, Dominance: 2.88

Message: His first order appointed Karnes adjutant for the 509th.
Predicted Valence: 2.97, Arousal: 2.97, Dominance: 3.03
Actual Valence: 2.90, Arousal: 2.90, Dominance: 3.20
```

```
Message: I travel way too much.
Predicted Valence: 2.96, Arousal: 2.93, Dominance: 3.03
Actual Valence: 2.50, Arousal: 3.12, Dominance: 3.38

Message: Teen hacks Venezuelan government Web sites
Predicted Valence: 3.03, Arousal: 3.06, Dominance: 3.06
Actual Valence: 2.60, Arousal: 2.90, Dominance: 2.70

Message: But this claim is merely feel-good propaganda, intended to keep the
neo-cons in power.
Predicted Valence: 2.93, Arousal: 3.03, Dominance: 3.00
Actual Valence: 2.90, Arousal: 3.20, Dominance: 3.00

Message: On the other hand, we need color in order to distinguish black and
white.
Predicted Valence: 2.92, Arousal: 2.99, Dominance: 3.08
Actual Valence: 3.11, Arousal: 3.11, Dominance: 3.22

Message: "Ah," Malaquez said.
Predicted Valence: 2.93, Arousal: 2.91, Dominance: 3.05
Actual Valence: 3.00, Arousal: 2.75, Dominance: 2.75

Message: Everyone I've talked to who has previously attended says wonderful
things about it; this semester, one of my friends told me she was going: I
should join her!
Predicted Valence: 2.92, Arousal: 3.24, Dominance: 3.08
Actual Valence: 3.40, Arousal: 3.30, Dominance: 3.00
```



9 Step8: Infering Emotions

9.1 (Merhabian VAD)

[138]:
```
def normalize_vad(v, a, d):
    """Normalize VAD values from EmoBank's 1-5 scale to -1 to 1 scale"""
    return (v - 3) / 2, (a - 3) / 2, (d - 3) / 2

emotion_coords = {
    'joy': (0.76, 0.48, 0.35),
    'anger': (-0.51, 0.59, 0.25),
    'fear': (-0.64, 0.60, -0.43),
    'sadness': (-0.63, -0.27, -0.33),
    'surprise': (0.40, 0.67, -0.13),
    'disgust': (-0.60, 0.35, 0.11),
    'contentment': (0.82, -0.18, 0.21),
    'boredom': (-0.65, -0.62, -0.33),
    'acceptance': (0.46, -0.09, -0.19)
}

def infer_emotion_mehrabian(v, a, d):
    v_norm, a_norm, d_norm = normalize_vad(v, a, d)
    vad = np.array([v_norm, a_norm, d_norm])

    distances = {emotion: np.linalg.norm(vad - np.array(coords))
                 for emotion, coords in emotion_coords.items()}

    return min(distances, key=distances.get)

# Function to get the PAD values for a given emotion
def get_pad_values(emotion):
    if emotion in emotion_coords:
        return emotion_coords[emotion]
    else:
        return None

# Function to find the closest emotions given PAD values
def find_closest_emotions(v, a, d, n=3):
    v_norm, a_norm, d_norm = normalize_vad(v, a, d)
    vad = np.array([v_norm, a_norm, d_norm])

    distances = {emotion: np.linalg.norm(vad - np.array(coords))
                 for emotion, coords in emotion_coords.items()}

    sorted_emotions = sorted(distances.items(), key=lambda x: x[1])
    return [emotion for emotion, _ in sorted_emotions[:n]]
```

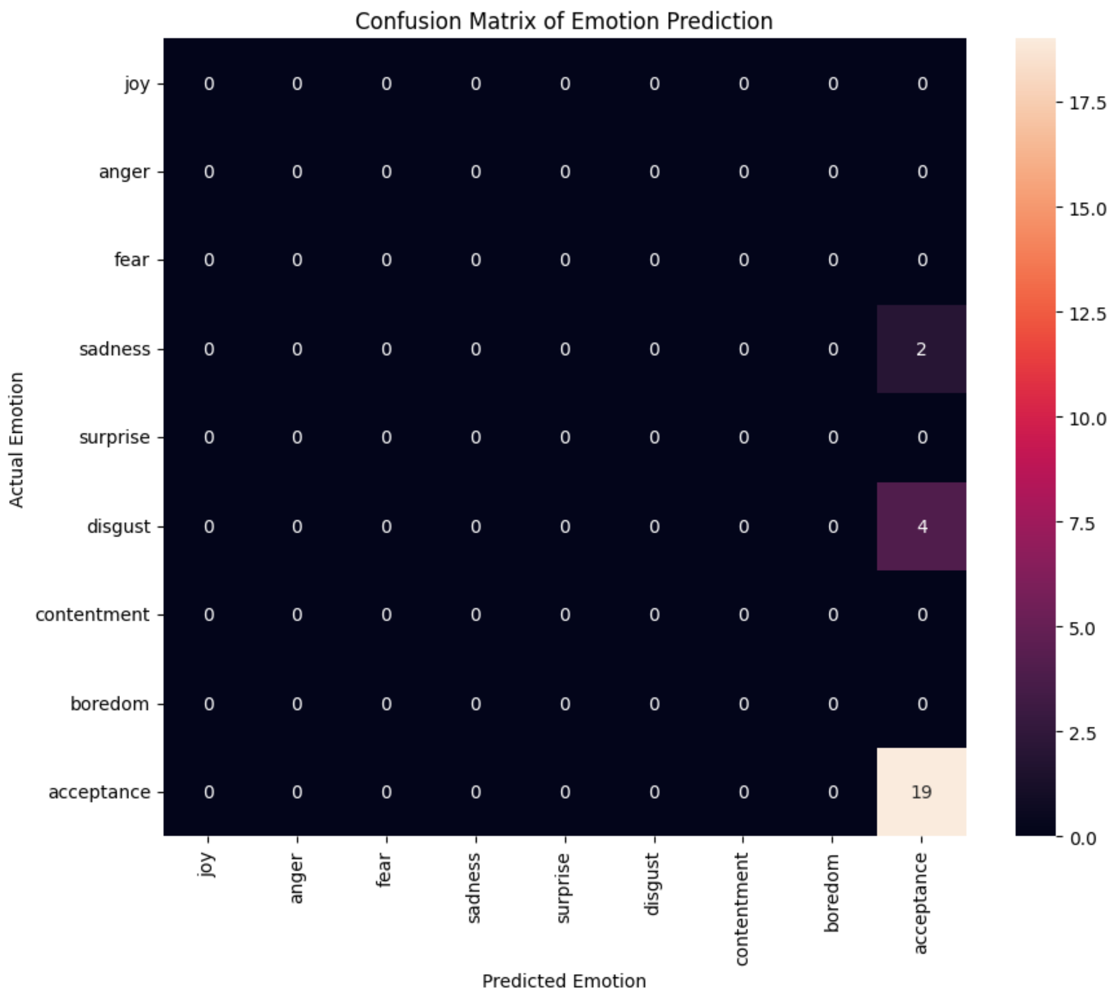
[139]:
```
v_pred, a_pred, d_pred = predict(input_message, rf_model)

# Create a DataFrame to store the results
results = pd.DataFrame({
    'Message': sampled_data['text'].values,
    'Actual_V': sampled_data['V'].values,
    'Actual_A': sampled_data['A'].values,
    'Actual_D': sampled_data['D'].values,
    'Predicted_V': v_pred,
    'Predicted_A': a_pred,
    'Predicted_D': d_pred
})
```

```
# Infer emotions and find closest emotions for actual and predicted VAD values
results['Actual_Emotion'] = results.apply(lambda row:␣
↪infer_emotion_mehrabian(row['Actual_V'], row['Actual_A'], row['Actual_D']),␣
↪axis=1)
results['Predicted_Emotion'] = results.apply(lambda row:␣
↪infer_emotion_mehrabian(row['Predicted_V'], row['Predicted_A'],␣
↪row['Predicted_D']), axis=1)
results['Actual_Top3'] = results.apply(lambda row:␣
↪find_closest_emotions(row['Actual_V'], row['Actual_A'], row['Actual_D']),␣
↪axis=1)
results['Predicted_Top3'] = results.apply(lambda row:␣
↪find_closest_emotions(row['Predicted_V'], row['Predicted_A'],␣
↪row['Predicted_D']), axis=1)

# Display the results
pd.set_option('display.max_colwidth', None)
print(results[['Message', 'Actual_Top3', 'Predicted_Top3']])

# Calculate accuracy
accuracy = (results['Actual_Emotion'] == results['Predicted_Emotion']).mean()
print(f"\nAccuracy of emotion prediction: {accuracy:.2%}")

# Calculate top-3 accuracy
top3_accuracy = results.apply(lambda row: row['Actual_Emotion'] in␣
↪row['Predicted_Top3'], axis=1).mean()
print(f"Top-3 accuracy of emotion prediction: {top3_accuracy:.2%}")

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(results['Actual_Emotion'], results['Predicted_Emotion'],␣
↪labels=list(emotion_coords.keys()))
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=emotion_coords.keys(),␣
↪yticklabels=emotion_coords.keys())
plt.title('Confusion Matrix of Emotion Prediction')
plt.xlabel('Predicted Emotion')
plt.ylabel('Actual Emotion')
plt.show()
```

```
                                                            Message  \
0
Today Big Sisters serves over 1,000 girls in central Indiana.
1
Ninety seconds from the bomb drop, Tibbets put the plane on auto-pilot.
2
"an effective headcount-control program."
```

3 Culture
buffs are well catered to, and there is always a varied program of events,
ranging from world-class concerts to local amateur dramatic productions.
4
The monastery is strictly vegetarian, and visitors are warned not to bring any
meat with them.
5
We are seeking to build partnerships between businesses and our not-for-profit
agency.
6
This website and everything on it is in the Public Domain, so you may use it
freely, even without attribution.
7 Karnes, 87, is a
retired history professor who lives in suburban Austin with his wife, Virginia,
who also worked for the 509th as a civilian bookkeeper for six months.
8
Detroit is a ghost town, and it's inhabited by ghosts: the only people left here
are those who can't or won't leave.
9
"a great deal of money,"
10
"You don't see a meaning because there is no meaning,"
11
Dear Friend:
12
Brad 14. Oct, 2010 at 7:02 pm
13
"The only no-kill shelter that's there when you need us" Little Shelter is
crowded with animals waiting for homes.
14
"Tasha --"
15
At long last, Nathan reached the end and set down his reading on the table
between them.
16 Pan-Mayanism has succeeded in part due to its focus on issues of ethnicity;
it largely avoids specifically political overtones, instead concentrating on the
development of the reemerging Mayan culture (England 2003: 734).
17 AND this time apart will make you
stronger and make you realize marriage is work and you need to put some in or
the marriage dies and you are left with an empty hole in your heart.
18
His first order appointed Karnes adjutant for the 509th.
19
I travel way too much.
20
Teen hacks Venezuelan government Web sites
21
But this claim is merely feel-good propaganda, intended to keep the neo-cons in

```
power.
22
On the other hand, we need color in order to distinguish black and white.
23
"Ah," Malaquez said.
24                                                                  Everyone I've
talked to who has previously attended says wonderful things about it; this
semester, one of my friends told me she was going: I should join her!

                            Actual_Top3                      Predicted_Top3
0       [acceptance, surprise, disgust]   [acceptance, disgust, sadness]
1          [disgust, anger, acceptance]   [acceptance, disgust, sadness]
2        [acceptance, disgust, sadness]   [acceptance, disgust, sadness]
3       [acceptance, surprise, disgust]   [acceptance, disgust, sadness]
4          [acceptance, disgust, anger]  [acceptance, disgust, surprise]
5        [acceptance, contentment, joy]  [acceptance, surprise, disgust]
6       [acceptance, disgust, surprise]   [acceptance, disgust, sadness]
7        [acceptance, disgust, sadness]   [acceptance, disgust, sadness]
8             [sadness, disgust, anger]   [acceptance, disgust, sadness]
9       [acceptance, surprise, disgust]   [acceptance, disgust, sadness]
10            [disgust, sadness, anger]   [acceptance, disgust, sadness]
11  [acceptance, contentment, surprise]   [acceptance, disgust, sadness]
12       [acceptance, sadness, disgust]  [acceptance, disgust, surprise]
13         [acceptance, disgust, anger]   [acceptance, disgust, sadness]
14       [acceptance, disgust, sadness]   [acceptance, disgust, sadness]
15       [acceptance, sadness, disgust]  [acceptance, disgust, surprise]
16       [acceptance, sadness, boredom]   [acceptance, disgust, sadness]
17            [disgust, sadness, anger]  [acceptance, disgust, surprise]
18       [acceptance, disgust, sadness]   [acceptance, disgust, sadness]
19            [disgust, anger, sadness]   [acceptance, disgust, sadness]
20       [sadness, disgust, acceptance]  [acceptance, disgust, surprise]
21         [acceptance, disgust, anger]   [acceptance, disgust, sadness]
22      [acceptance, disgust, surprise]   [acceptance, disgust, sadness]
23       [acceptance, sadness, disgust]   [acceptance, disgust, sadness]
24  [acceptance, surprise, contentment]     [acceptance, disgust, anger]

Accuracy of emotion prediction: 76.00%
Top-3 accuracy of emotion prediction: 96.00%
```


Confusion Matrix of Emotion Prediction
Actual Emotion
Predicted Emotion
joy
anger
fear
sadness
surprise
disgust
contentment
boredom
acceptance
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 4
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 19
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0