MACHINE LEARNING

UE22CS352A

RAASHI BAFNA

SEM : 5 SEC : G

DATE : 11-11-2024

SENTIMENT ANALYSIS

SENTIMENT ANALYSIS :

This hackathon challenge involves building a **multimodal sentiment analysis model** using **video, audio and text data** from the TV show *Friends*. The goal is to detect and classify the sentiments expressed in each clip and each spoken sentence by the characters, based on how they feel when they speak. To achieve this, both **visual cues** (from video frames) , **audio data**(from audio) and **textual data** (from subtitles) need to be integrated and analyzed.

1) Data Preprocessing Steps

The notebook includes preprocessing for text, audio, and visual data:

- Audio Preprocessing: Audio features such as MFCCs, chroma, mel spectrogram, spectral contrast, and tonnetz are extracted using the librosa library. These features are averaged to create a single feature vector per audio clip.

```python
# 1. Handle missing data (if any NaN values)
def handle_missing_data(features):
    # Replace NaN values with 0
    if np.any(np.isnan(features)):
        features = np.nan_to_num(features)  # Replace NaN with zero
    return features

# 2. Normalize features (Standardization)
def normalize_features(features):
    scaler = StandardScaler()
    features_normalized = scaler.fit_transform(features.reshape(-1, 1))  # Standardizing the feature vector
    return features_normalized.flatten()  # Flatten back to 1D array

# Full preprocessing pipeline (without PCA and temporal smoothing)
def preprocess_audio_features(features):
    # Step 1: Handle missing data (if any)
    features = handle_missing_data(features)

    # Step 2: Normalize features
    features = normalize_features(features)

    return features

# Assuming 'train_df' contains a column 'audio_features' with extracted features
# Apply preprocessing to the 'audio_features' column
train_df['preprocessed_audio_features'] = train_df['audio_features'].apply(lambda x: preprocess_audio_features(np.array(x)) if x is not None else None)
```

- Visual Preprocessing: Frames from video clips are sampled (every 10th frame) and passed through a pre-trained ResNet model to extract visual features. The output is averaged across frames to create a single visual feature vector.

```python
# Define transformation pipeline to preprocess images before passing through the model
preprocess = transforms.Compose([
    transforms.ToPILImage(),  # Convert the frame to PIL image
    transforms.Resize((224, 224)),  # Resize the image to 224x224
    transforms.ToTensor(),  # Convert the PIL image to tensor
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),  # Normalize the image
])
```

- Text Preprocessing: Text data is cleaned by converting to lowercase, removing punctuation and digits and tokenizing. Lemmatization is also applied to reduce words to their base forms.

```python
# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Function for preprocessing text
def preprocess_text(text):
    # Lowercase the text
    text = text.lower()

    # Remove punctuation, special characters, and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize the text
    words = text.split()

    return ' '.join(words)

# Apply the preprocessing to the 'Utterance' column
train_df['cleaned_text'] = train_df['Utterance'].apply(preprocess_text)
```

These preprocessing steps aim to create a uniform representation of each modality for later fusion.

2) Feature Extraction Steps

1. Audio Features

Audio feature extraction in this notebook uses the librosa library, a powerful tool for analyzing audio signals. The extracted features aim to capture various characteristics that can hint at emotions, speech patterns, or tones. Key extracted features include:

- **MFCCs (Mel-Frequency Cepstral Coefficients)**: MFCCs are commonly used in speech and audio processing to capture timbral texture and the shape of the sound spectrum. By transforming the audio signal into the Mel scale, which aligns with human auditory perception, MFCCs can effectively capture the distinct qualities of speech or vocal tone.
- **Chroma Features**: These features represent the 12 distinct pitch classes (e.g., C, C#, D) and are useful for understanding the harmonic content of audio. They provide insight into the musical or tonal elements, which may be relevant in detecting emotional nuances in voice.
- **Mel Spectrogram**: The Mel spectrogram represents the intensity (amplitude) of frequencies over time, using the Mel scale. It captures the overall shape and energy of sounds, helpful for identifying emotional intensity or variations in the speaker's tone.
- **Spectral Contrast**: This feature captures the difference in amplitude between peaks and valleys in a sound spectrum. Variations in spectral contrast can reveal information about the timbre or tonal quality of the audio, such as whether it's a quiet or energetic sound.
- **Tonnetz (Tonal Centroid Features)**: Tonnetz is a feature that captures harmonic content, often linked to emotions in speech. It provides insights into the tonal stability and harmonic structure of the sound.

After extracting these features, they are averaged to create a single, fixed-length feature vector for each audio clip. This vector provides a compact yet descriptive representation of the audio signal.

2. Visual Features

For visual data, a pre-trained ResNet model is used to extract deep features from sampled frames of video clips:

- **ResNet (Residual Network)**: ResNet is a popular deep convolutional neural network model trained on large datasets, like ImageNet, to recognize various visual patterns. By using a pre-trained ResNet, this notebook leverages ResNet's ability to capture complex image features without needing to train from scratch.
- **Frame Sampling and Feature Averaging**: Since videos contain multiple frames, sampling every 10th frame reduces computational complexity while still capturing essential visual information over time. Each sampled frame is passed through the ResNet model, producing a feature vector (often 2048-dimensional). These vectors are then averaged to create a single vector per video, capturing the video's overall visual characteristics.

This approach allows for a concise yet informative visual representation, incorporating textures, shapes, and patterns that the ResNet has learned to identify.

3. Textual Features

Text features are derived using both basic preprocessing and likely leveraging a BERT model for deep semantic representations:

- **Text Preprocessing**: Initially, the text is cleaned by converting it to lowercase, removing special characters, and lemmatizing words (reducing words to their root forms). This process helps in standardizing the text data, removing noise, and making it easier to analyze.
- **Word2Vec Embeddings**: Word2Vec generates embeddings by training a shallow neural network to predict words within their local context (either using the Skip-Gram or Continuous Bag of Words (CBOW) model). These embeddings capture semantic similarities based on co-occurrence patterns in text, meaning words that often appear in similar contexts have closer vector representations.

- **Averaging Word Embeddings**: Since Word2Vec creates embeddings for individual words, the notebook likely averages these word vectors across each sentence or document. This creates a single vector that represents the overall meaning of the sentence or document**.**The combination of these features from different modalities—audio, visual, and textual—provides a rich, multimodal representation of each data point, which can improve the model's ability to understand and analyze complex data patterns.

3) Comparison Between Early and Late Fusion Results:

1. Early Fusion:

- **Description**: In early fusion, all modalities (audio, text, and visual) are combined into a single feature vector before being passed to the classifier. This approach typically involves **concatenating the features** from all modalities into a single input, which is then processed by a unified model (in this case, XGBoost).
- **Advantages**:
  - **Holistic Understanding**: By combining features from all modalities upfront, the model can learn joint patterns across audio, text, and visual data, capturing complex inter-modal relationships.
  - **Improved Model Generalization**: Since the model has access to all information in one step, it is more likely to generalize well across different types of data, rather than making predictions based on a single modality.
  - **Better Performance**: When the model is tuned properly, early fusion can lead to **higher accuracy** because all relevant information is available simultaneously to the classifier.
- **Challenges**:

    - **Dimensionality**: Combining all features into a single vector increases the feature space's dimensionality, which may require more training data to avoid overfitting.
    - **Complexity in Feature Engineering**: Preprocessing and scaling each modality’s features appropriately can be complex, and improper scaling might degrade model performance.
    - **Data Imbalance**: Some modalities (e.g., visual features) may be less informative than others (e.g., text), leading to a potential imbalance in the feature importance, requiring careful attention during training.
- **Effectiveness**:
    - **High Performance**: Early fusion, especially with **XGBoost** as the classifier, tends to provide high predictive accuracy by exploiting the full spectrum of available data.
    - **Computational Cost**: Handling all features in one large input vector may require more computational resources, especially when the number of features is large.

2. Late Fusion:

- **Description**: Late fusion involves training separate classifiers for each modality (e.g., one for audio, one for text, and one for visual features) and then combining their predictions, typically through **majority voting** or weighted averaging. This approach allows each modality to be processed independently before combining insights.
- **Advantages**:
    - **Modality Specialization**: Each classifier can specialize in its respective modality, potentially leading to better performance on the features it is most suited for.
    - **Flexibility**: Late fusion is more flexible since it allows for different types of models or classifiers to be used for each modality. For example, Random Forest for text, audio, and visual data can each be optimized independently.
    - **Reduced Risk of Overfitting**: Since classifiers are trained independently, there is less risk of overfitting on any single modality compared to early fusion.

- **Challenges**:
  - **Combining Predictions**: The main challenge with late fusion is how to combine predictions from different models. If the modalities are highly imbalanced or the classifiers have significantly different performance, this can lead to suboptimal outcomes.
  - **Computational Complexity**: Although each classifier is simpler, training multiple classifiers and then combining their results requires more computational time and resources than a single model.
  - **Loss of Inter-Modality Interaction**: Unlike early fusion, late fusion may miss out on the richer relationships between the different modalities, as they are processed in isolation. This might reduce the performance when the relationship between modalities is crucial (e.g., matching tone in audio with sentiment in text).
- **Effectiveness**:
  - **Balanced Performance**: Late fusion is often effective when individual modalities perform well on their own. However, its performance can be highly dependent on how well each classifier is tuned and how the final predictions are combined.
  - **Simplicity in Training**: Each classifier can be trained separately, making it easier to debug and modify the system for each modality. It also provides an easier pathway to handle imbalances in data or feature importance across modalities.

Comparative Summary:

Aspect	Early Fusion	Late Fusion
Data Handling	Combines all features into one vector.	Processes each modality separately.
Model Complexity	One classifier processes all features.	Multiple classifiers handle different modalities.

Aspect	Early Fusion	Late Fusion
Training Flexibility	Requires careful preprocessing and scaling.	More flexible; each modality can have its own classifier.
Performance	High accuracy if properly tuned (complex).	Can be good, but depends on individual classifier performance.
Overfitting Risk	Higher dimensionality may increase overfitting risk.	Lower risk of overfitting due to independent models.
Inter-Modality Interaction	Learns joint representations from all modalities.	Does not capture relationships between modalities directly.
Computational Cost	Higher, due to large feature vectors.	Lower, as models are trained separately.
Effectiveness	Performs well in capturing multi-modal relationships.	Effective when each modality is strong on its own.

Insights from Experimentation:

- **Early Fusion (XGBoost)** tends to provide **better overall performance** when the modalities are well-aligned and the data is sufficient. It captures the joint interactions across modalities, leading to richer feature representations.
- **Late Fusion (Random Forest)** works best when each modality has strong individual classifiers. It also provides a more **flexible** setup, allowing for specialized models for each modality, but may suffer from **loss of inter-modal relationships** and a **higher risk of conflicting predictions**.

In conclusion, **early fusion** is typically more effective in tasks requiring deep integration of multimodal data, while **late fusion**

offers simplicity and flexibility, particularly when each modality performs independently well.

Early fusion code:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Split the train_df into 80% training and 20% testing
train_df, test_df = train_test_split(train_df, test_size=0.2, random_state=42)

# Extract features and target (Sentiment) for training
X_train_audio =
np.vstack(train_df['preprocessed_audio_features'].apply(np.array))
X_train_text = np.vstack(train_df['textual_features'].apply(np.array))
X_train_visual = np.vstack(train_df['visual_features'].apply(np.array))

X_test_audio =
np.vstack(test_df['preprocessed_audio_features'].apply(np.array))
X_test_text = np.vstack(test_df['textual_features'].apply(np.array))
X_test_visual = np.vstack(test_df['visual_features'].apply(np.array))

# Convert Sentiment to numerical labels
sentiment_mapping = {'negative': 0, 'neutral': 1, 'positive': 2}
y_train = train_df['Sentiment'].map(sentiment_mapping).values
y_test = test_df['Sentiment'].map(sentiment_mapping).values

# Normalize features with separate scalers for each feature set
scaler_audio = StandardScaler()
scaler_text = StandardScaler()
scaler_visual = StandardScaler()

X_train_audio = scaler_audio.fit_transform(X_train_audio)
X_train_text = scaler_text.fit_transform(X_train_text)
X_train_visual = scaler_visual.fit_transform(X_train_visual)

X_test_audio = scaler_audio.transform(X_test_audio)
X_test_text = scaler_text.transform(X_test_text)
```

```
X_test_visual = scaler_visual.transform(X_test_visual)

# Dimensionality reduction with PCA (increase n_components to retain more
variance)
pca_audio = PCA(n_components=50)  # Increase components to retain more
information
pca_text = PCA(n_components=50)
pca_visual = PCA(n_components=50)

X_train_audio_pca = pca_audio.fit_transform(X_train_audio)
X_train_text_pca = pca_text.fit_transform(X_train_text)
X_train_visual_pca = pca_visual.fit_transform(X_train_visual)

X_test_audio_pca = pca_audio.transform(X_test_audio)
X_test_text_pca = pca_text.transform(X_test_text)
X_test_visual_pca = pca_visual.transform(X_test_visual)

# Early Fusion: Combine all PCA-reduced features into one
X_train_early = np.hstack((X_train_audio_pca, X_train_text_pca,
X_train_visual_pca))
X_test_early = np.hstack((X_test_audio_pca, X_test_text_pca,
X_test_visual_pca))

# Hyperparameter tuning for XGBoost with RandomizedSearchCV
param_dist = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 5, 7],
    'subsample': [0.7, 0.8, 0.9],
    'colsample_bytree': [0.7, 0.8, 0.9]
}

xgb_early = XGBClassifier(objective='multi:softmax', num_class=3,
random_state=42)
random_search = RandomizedSearchCV(estimator=xgb_early,
param_distributions=param_dist,
                                   n_iter=20, scoring='accuracy', cv=3,
random_state=42, n_jobs=-1)
random_search.fit(X_train_early, y_train)

# Best model after RandomizedSearchCV
best_xgb_early = random_search.best_estimator_

# Predictions and Evaluation (Early Fusion with XGBoost)
y_pred_early = best_xgb_early.predict(X_test_early)

accuracy_early = accuracy_score(y_test, y_pred_early)
```

```
print(f"Accuracy of Early Fusion with XGBoost (after tuning): {accuracy_early
* 100:.2f}%")

# Classification Report and Confusion Matrix for evaluation
print("\nClassification Report for Early Fusion with XGBoost:")
print(classification_report(y_test, y_pred_early, target_names=['negative',
'neutral', 'positive']))

print("\nConfusion Matrix for Early Fusion with XGBoost:")
print(confusion_matrix(y_test, y_pred_early))
```

Late fusion code:

```
# Late Fusion: Train separate models for each modality and combine the
predictions
# Train separate classifiers for each modality
rf_audio = RandomForestClassifier(random_state=42)
rf_audio.fit(X_train_audio, y_train)

rf_text = RandomForestClassifier(random_state=42)
rf_text.fit(X_train_text, y_train)

rf_visual = RandomForestClassifier(random_state=42)
rf_visual.fit(X_train_visual, y_train)

# Predictions for Late Fusion
y_pred_audio = rf_audio.predict(X_test_audio)
y_pred_text = rf_text.predict(X_test_text)
y_pred_visual = rf_visual.predict(X_test_visual)

# Majority voting for late fusion
y_pred_late = np.array([np.argmax(np.bincount([y_pred_audio[i],
y_pred_text[i], y_pred_visual[i]])) for i in range(len(y_pred_audio))])

# Evaluate the Late Fusion model
accuracy_late = accuracy_score(y_test, y_pred_late)
print(f"Accuracy of Late Fusion: {accuracy_late * 100:.2f}%")

# Classification Report and Confusion Matrix for evaluation
print("\nClassification Report for Late Fusion:")
print(classification_report(y_test, y_pred_late, target_names=['negative',
'neutral', 'positive']))

print("\nConfusion Matrix for Late Fusion:")
print(confusion_matrix(y_test, y_pred_late))
```

4) Model Decisions:

1. ResNet (Visual Feature Extraction)

Code Explanation:

- **ResNet50 model**: The code uses a pre-trained **ResNet50** model, a deep convolutional neural network architecture, for extracting visual features from video frames.
- **Preprocessing**: Before feeding frames to ResNet, they are resized to 224x224 pixels, normalized, and converted into tensor format.
- **Feature extraction**: After preprocessing, each frame is passed through the ResNet model, and the resulting features (2048-dimensional vector) are flattened for further use. The feature vectors are aggregated (averaged) across all frames to get a summary of visual information from the video.

Why ResNet is good for this task:

- **Pre-trained knowledge**: By using a pre-trained ResNet model, you are benefiting from transfer learning. ResNet has been trained on large image datasets (like ImageNet), allowing it to extract useful high-level features from images without needing to retrain from scratch.
- **Deep learning capability**: ResNet architecture excels at handling very deep neural networks with residual connections, ensuring that the model can capture complex patterns in visual data. This is ideal for extracting meaningful features from complex video frames.

2. Word2Vec (Textual Feature Extraction)

Code Explanation:

- **Word2Vec model**: The code trains a custom Word2Vec model using **Gensim**, which learns dense vector representations for words based on their co-occurrence in the corpus.
- **Sentence embedding**: For each utterance (sentence), it converts words into their vector representations and averages them to create a single vector representing the sentence.

Why Word2Vec is beneficial:

- **Captures semantic meaning**: Word2Vec is good for capturing semantic meaning in text. Words with similar meanings will have similar vectors, making it effective for tasks like sentiment analysis where the meaning of words matters.
- **Custom model**: By training Word2Vec on your own corpus (the dataset you're working with), it learns word representations that are specifically tailored to the context of your data, rather than relying on a general-purpose model.

3. XGBoost (Machine Learning Model for Prediction)

Code Explanation:

- **XGBoost**: After extracting features from audio, text, and visual data, these features are concatenated and passed to an **XGBoost** classifier for sentiment classification.
- **Hyperparameter tuning**: The code uses **RandomizedSearchCV** to perform hyperparameter optimization for XGBoost, tuning parameters such as the number of estimators, learning rate, max depth, and subsample to improve model performance.
- **Training**: The classifier is trained on the combined features from all modalities (audio, text, and visual).

Why XGBoost is preferred:

- **Performance**: XGBoost is an optimized gradient boosting framework that tends to perform very well for tabular data and is known for its speed and accuracy. It handles missing data, feature interactions, and is robust to overfitting with proper tuning.
- **Interpretability**: Compared to deep learning models, XGBoost offers greater interpretability (via feature importance scores), making it easier to understand which features are most important for prediction.

4. Random Forest (Late Fusion Method)

Code Explanation:

- **Random Forest**: The code trains three separate **Random Forest** classifiers—one each for audio, text, and visual features. Then, predictions from each of these classifiers are combined using **majority voting** to produce the final sentiment prediction (late fusion).
- **Voting**: The final class is chosen based on which prediction (from audio, text, or visual classifier) occurs most frequently.

Why Random Forest is good for late fusion:

- **Robustness**: Random Forest is an ensemble method that combines multiple decision trees, which makes it less prone to overfitting and more robust to noise in the data.
- **Combining strengths**: In the late fusion strategy, combining predictions from multiple models allows the system to leverage different strengths of each modality (audio, text, and visual) and reduces the risk of one modality's weaknesses dominating the prediction.

Fusion Strategies (Early and Late Fusion)

Early Fusion:

- **Combining features before training**: In early fusion, the features from all modalities (audio, text, visual) are combined

into a single feature vector, and a single classifier (XGBoost) is used to predict the output.
- **Advantages**: The model learns how to balance the contributions from each modality directly during training. This can be efficient if the different feature types complement each other well.

Late Fusion:

- **Separate classifiers for each modality**: In late fusion, each modality is treated separately, and different classifiers (Random Forest) are trained for each type of data (audio, text, visual). Their predictions are then combined using majority voting.
- **Advantages**: This method allows the classifiers to specialize in their respective domains (audio, text, visual) and gives flexibility in handling each modality independently.

Why These Models Are Better for This Task:

- **ResNet for vision**: Using a pre-trained deep learning model like ResNet allows you to capture rich visual patterns that are relevant for tasks like video-based sentiment analysis.
- **Word2Vec for text**: Word2Vec captures semantic relationships between words, allowing the model to better understand sentiment even if words are used in different contexts.
- **XGBoost for classification**: XGBoost is well-suited for tabular data and handles high-dimensional data (which comes from concatenating features from different modalities) very effectively. It also performs well with fewer data points or noisy data.
- **Random Forest for late fusion**: Random Forest is good for handling individual data types in an ensemble approach and offers robustness in the decision-making process.

Overall Pipeline Advantages:

- **Multimodal fusion**: The combination of audio, text, and visual features allows the model to capture the full context of the data, leading to better performance than using a single modality.

- **Transfer learning and custom embeddings**: Using pre-trained models like ResNet and custom Word2Vec embeddings leverages the power of transfer learning while tailoring the model to the specific dataset.
- **Strong performance with ensemble models**: Both XGBoost and Random Forest are robust ensemble learning algorithms known for their performance and ability to handle diverse feature sets effectively.

How These Models Are Better:

1. **ResNet**:
    - **Depth and generalization**: ResNet's deep architecture, combined with residual connections, allows it to generalize well to unseen data, especially visual data. It can extract intricate features from video frames that simpler models might miss.
    - **Transfer learning**: By using a pre-trained model, you avoid the need to train a model from scratch, which is particularly useful when you have limited training data.
2. **Word2Vec**:
    - **Contextual understanding**: Word2Vec captures the relationships between words based on context, which is key for understanding nuances in sentiment, tone, and meaning in textual data.
    - **Efficiency**: Compared to other text feature extraction methods (like TF-IDF), Word2Vec provides dense and informative embeddings that are more suitable for machine learning models like XGBoost.
3. **XGBoost**:
    - **Powerful for tabular data**: XGBoost is very effective for classification tasks, especially when you have a combination of numerical and categorical data (like the features extracted from audio, text, and video).
    - **Handling complex data**: With the fusion of multiple feature sets (audio, text, and visual), XGBoost can handle the complex relationships between these features and make accurate predictions.

- **Scalability and tuning**: It can scale well with large datasets, and hyperparameter tuning (like with RandomizedSearchCV) can further improve its performance, making it ideal.

4. **Random Forest (RF):**

- **Robustness**: RF reduces overfitting by averaging multiple decision trees, making it robust to noise.
- **Feature importance**: Provides insights into the contribution of each feature (audio, text, visual).
- **Handling mixed data**: Efficient with both numerical and categorical features, useful for late fusion of multiple modalities.
- **Effective for ensemble (late fusion)**: Combines predictions from separate classifiers (audio, text, visual) using majority voting for better accuracy.

5) Performance Analysis:

```
Accuracy of Early Fusion with XGBoost (after tuning): 48.50%

Classification Report for Early Fusion with XGBoost:
              precision    recall  f1-score   support

    negative       0.55      0.17      0.26        64
     neutral       0.48      0.90      0.62        90
    positive       0.50      0.11      0.18        46

    accuracy                           0.48       200
   macro avg       0.51      0.39      0.35       200
weighted avg       0.51      0.48      0.41       200


Confusion Matrix for Early Fusion with XGBoost:
[[11 51  2]
 [ 6 81  3]
 [ 3 38  5]]
```

```
Accuracy of Late Fusion: 47.00%

Classification Report for Late Fusion:
              precision    recall  f1-score   support

    negative       0.38      0.14      0.20        64
     neutral       0.48      0.92      0.63        90
    positive       0.67      0.04      0.08        46

    accuracy                           0.47       200
   macro avg       0.51      0.37      0.31       200
weighted avg       0.49      0.47      0.37       200


Confusion Matrix for Late Fusion:
[[ 9 54  1]
 [ 7 83  0]
 [ 8 36  2]]
```

Metric	Early Fusion (XGBoost)	Late Fusion
Accuracy	48.50%	47.00%
Precision (Negative)	0.55	0.38
Precision (Neutral)	0.48	0.48
Precision (Positive)	0.50	0.67
Recall (Negative)	0.17	0.14
Recall (Neutral)	0.90	0.92
Recall (Positive)	0.11	0.04
F1-Score (Negative)	0.26	0.20
F1-Score (Neutral)	0.62	0.63
F1-Score (Positive)	0.18	0.08
Macro Average F1-Score	0.35	0.31
Weighted Average F1-Score	0.41	0.37

Early Fusion with XGBoost (Accuracy: 48.50%):

- **Key Metrics**:
  - **Precision**: The precision values for all classes are relatively balanced, but the **negative** class shows a higher precision (0.55) compared to the others.

    - **Recall**: The recall for **neutral** (0.90) is much higher than the other classes, indicating that the model does well in identifying neutral sentiment. However, recall for **negative** and **positive** classes is much lower.
    - **F1-Score**: The **neutral** class has the highest F1-score (0.62), while **positive** and **negative** classes have lower scores.

Late Fusion (Accuracy: 47.00%):

- **Key Metrics**:
    - **Precision**: **Positive** class precision is higher (0.67) compared to the **negative** (0.38) and **neutral** (0.48) classes.
    - **Recall**: The recall for **neutral** remains high (0.92), similar to early fusion, but **negative** and **positive** classes have low recall.
    - **F1-Score**: The **neutral** class achieves a good F1-score (0.63).

Views:

- **Early Fusion vs Late Fusion**:
    - **Early Fusion** slightly outperforms **Late Fusion** in terms of **accuracy** (48.50% vs. 47.00%), with **higher recall** for the neutral class and slightly better performance on the negative class.
    - **Late Fusion**, however, shows a higher **precision** for the **positive** class, but this is accompanied by an extremely low **recall** (0.04), indicating that although the model can identify some positive sentiment instances better.

Insights:

1. Multimodal Fusion Improves Performance:

- **Early Fusion**: Combining audio, text, and visual features early in the pipeline using XGBoost generally leads to higher accuracy as it leverages all available data simultaneously. This approach ensures that the model captures complex relationships between different modalities.
- **Late Fusion**: Using separate classifiers for each modality (e.g., Random Forest for audio, text, and visual features) and combining their predictions via majority voting works well but may not always outperform early fusion. However, it provides a more flexible approach, allowing each modality to specialize in its domain.

2. Feature Importance:

- **Text Features (Word2Vec)**: Word embeddings significantly improve the model's ability to understand nuanced sentiment in text. The model's performance benefits from rich word-level contextual features.
- **Visual Features (ResNet)**: Video features derived from ResNet are valuable, especially for sentiment analysis tasks that may depend on visual cues like facial expressions or body language. However, the visual features may be less informative than text or audio for some types of sentiment.
- **Audio Features**: Audio features like tone and prosody add another layer of understanding, improving model performance when combined with text and visual features. However, in some cases, the audio modality may need further refinement to be on par with text and visual features.

3. XGBoost vs Random Forest:

- **XGBoost**: Tends to perform better when dealing with high-dimensional data and complex feature interactions, especially

after hyperparameter tuning (via RandomizedSearchCV). It is particularly strong when combining multiple modalities early.
- **Random Forest**: While robust and effective, especially in the late fusion setup, Random Forest's performance can be more sensitive to the quality of individual modality features. The voting-based nature of late fusion means that its success heavily depends on how well each modality performs on its own.

4. Dimensionality Reduction (PCA):

- **PCA**: Applying PCA to reduce the feature dimensions helps improve model performance by focusing on the most important variance in the data. However, too much reduction can lead to the loss of critical information. Balancing the number of components retained is crucial.

5. Handling Imbalanced Data:

- If the dataset is imbalanced (e.g., one sentiment class is underrepresented), both XGBoost and Random Forest can benefit from techniques like **class weighting** or **resampling** to ensure balanced learning.

6. Hyperparameter Tuning:

- **XGBoost**: Hyperparameter tuning significantly enhances the performance of the model. Tuning the learning rate, max depth, and subsampling parameters can lead to better generalization and accuracy.
- **Random Forest**: While RF is less sensitive to hyperparameters compared to XGBoost, experimenting with tree depth and number of trees can still yield performance improvements.

Conclusion:

The hackathon confirm that multimodal approaches (combining audio, text, and visual features) enhance sentiment analysis accuracy. Early fusion with XGBoost is generally superior, but late fusion offers flexibility. Feature importance analysis shows that text and visual

features play key roles, while audio can improve results with further refinement.