In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dataset = pd.read_excel('Final Autism Dataset.xlsx')
```

In [3]:
```
dataset.head(5)
```

Out[3]:

	age	gender	level_ASD	speak_verbally	follow_instruction	maintain_interaction	s
0	6-9	Male	Level 1	Sometimes	Yes	Yes	
1	6-9	Male	Level 1	Sometimes	Yes	Yes	
2	Below 6	Male	Level 1	No	Yes	Yes	
3	Below 6	Male	Level 1	Sometimes	Yes	No	
4	Below 6	Female	Level 2	No	Yes	Yes	

In [4]:
```
dataset.shape
```

Out[4]:
```
(225, 18)
```

In [5]:
```
dataset.isnull().sum()
```

Out[5]:
```
age                         0
gender                      0
level_ASD                   0
speak_verbally              0
follow_instruction          0
maintain_interaction        0
socialize_other_children    0
eye_contact                 0
role_playing                0
facial_expression           0
understand_others_feeling   0
look_at_pointed_toys        0
respond_when_called         0
keep_attention              0
interest_in_gadget          0
behaviour                   0
parents_objective_1         0
plan_therapy_1              0
dtype: int64
```

In [6]:
```
dataset.describe()
```

Out[6]:

	age	gender	level_ASD	speak_verbally	follow_instruction	maintain_interacti
count	225	225	225	225	225	2
unique	3	2	3	3	3	
top	Below 6	Male	Level 1	No	Yes	Y
freq	138	179	150	112	144	1

In [7]:

```
dataset.nunique()
```

Out[7]:

```
age                         3
gender                      2
level_ASD                   3
speak_verbally              3
follow_instruction          3
maintain_interaction        3
socialize_other_children    3
eye_contact                 3
role_playing                3
facial_expression           3
understand_others_feeling   3
look_at_pointed_toys        3
respond_when_called         3
keep_attention              4
interest_in_gadget          3
behaviour                   6
parents_objective_1        11
plan_therapy_1              9
dtype: int64
```

In [8]:

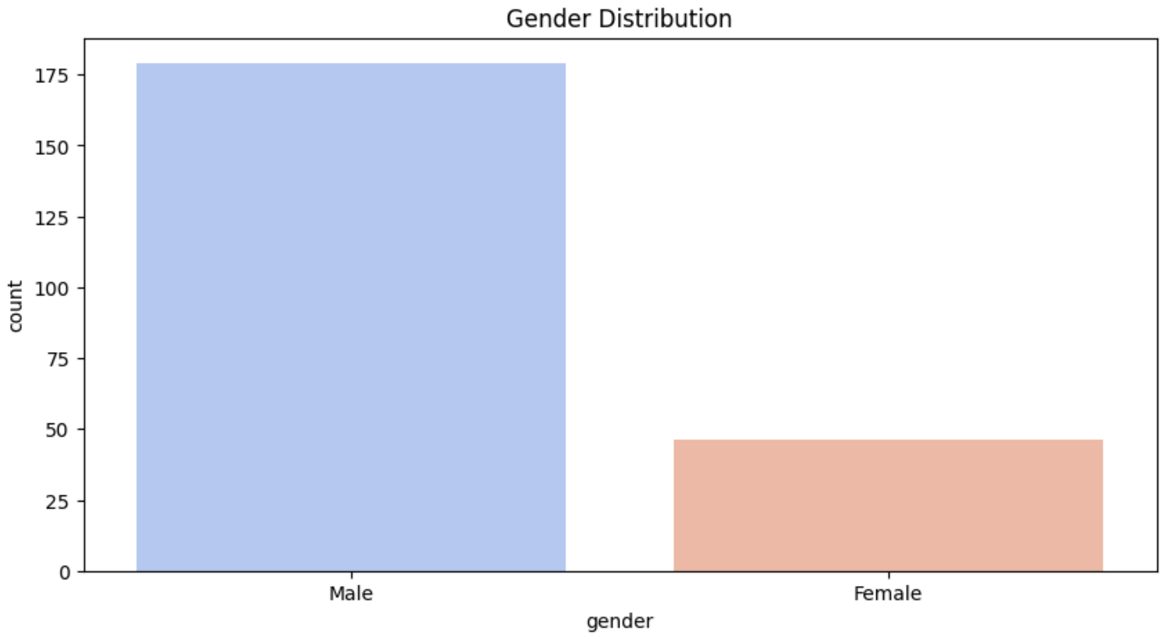
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
sns.countplot(x='gender', data=dataset, palette='coolwarm')
plt.title('Gender Distribution')
plt.show()
```

```
C:\Users\geekp\AppData\Local\Temp\ipykernel_10232\707396516.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='gender', data=dataset, palette='coolwarm')
```

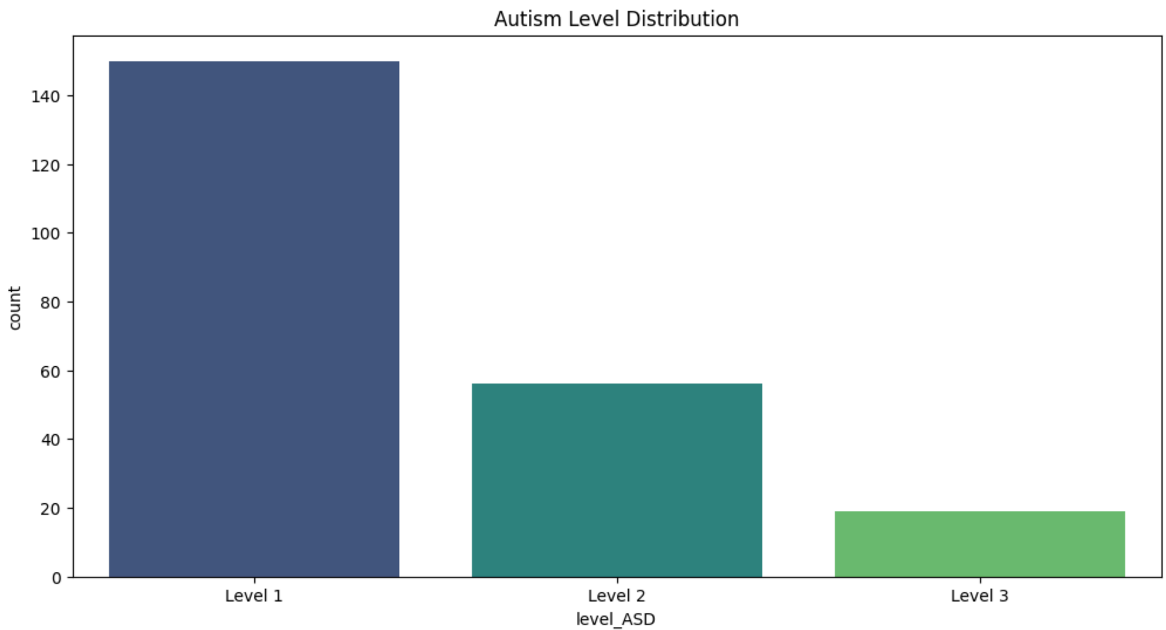


In [9]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='level_ASD', data=dataset, palette='viridis')
plt.title('Autism Level Distribution')
plt.show()
```

```
C:\Users\geekp\AppData\Local\Temp\ipykernel_10232\527715418.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='level_ASD', data=dataset, palette='viridis')
```

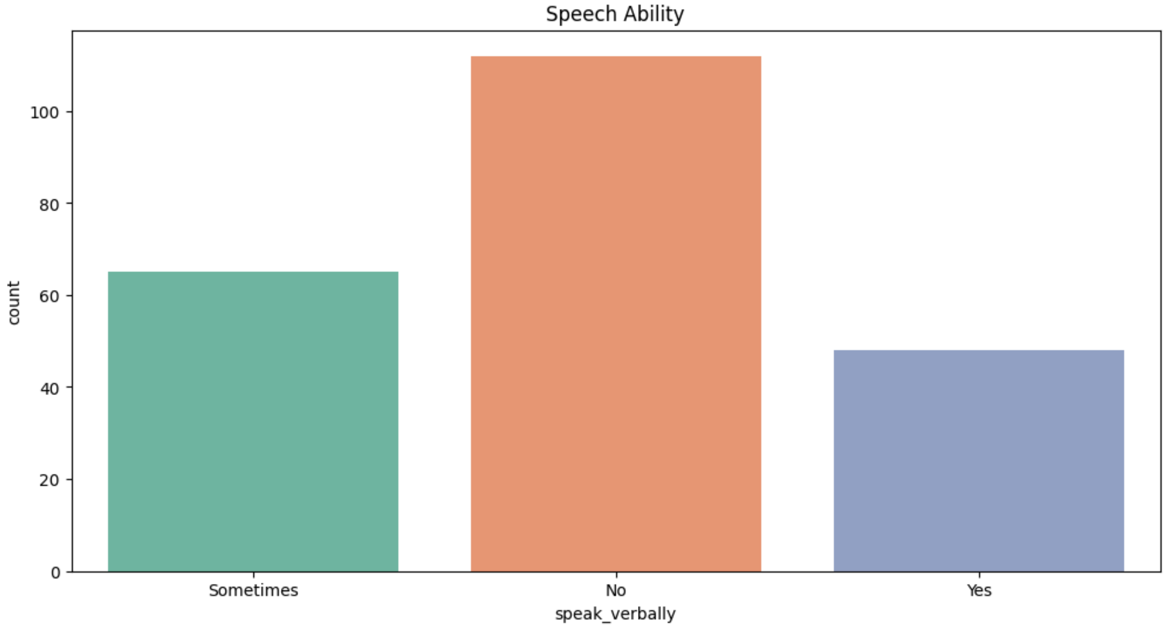


In [10]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='speak_verbally', data=dataset, palette='Set2')
plt.title('Speech Ability')
plt.show()
```

```
C:\Users\geekp\AppData\Local\Temp\ipykernel_10232\2843719687.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='speak_verbally', data=dataset, palette='Set2')
```

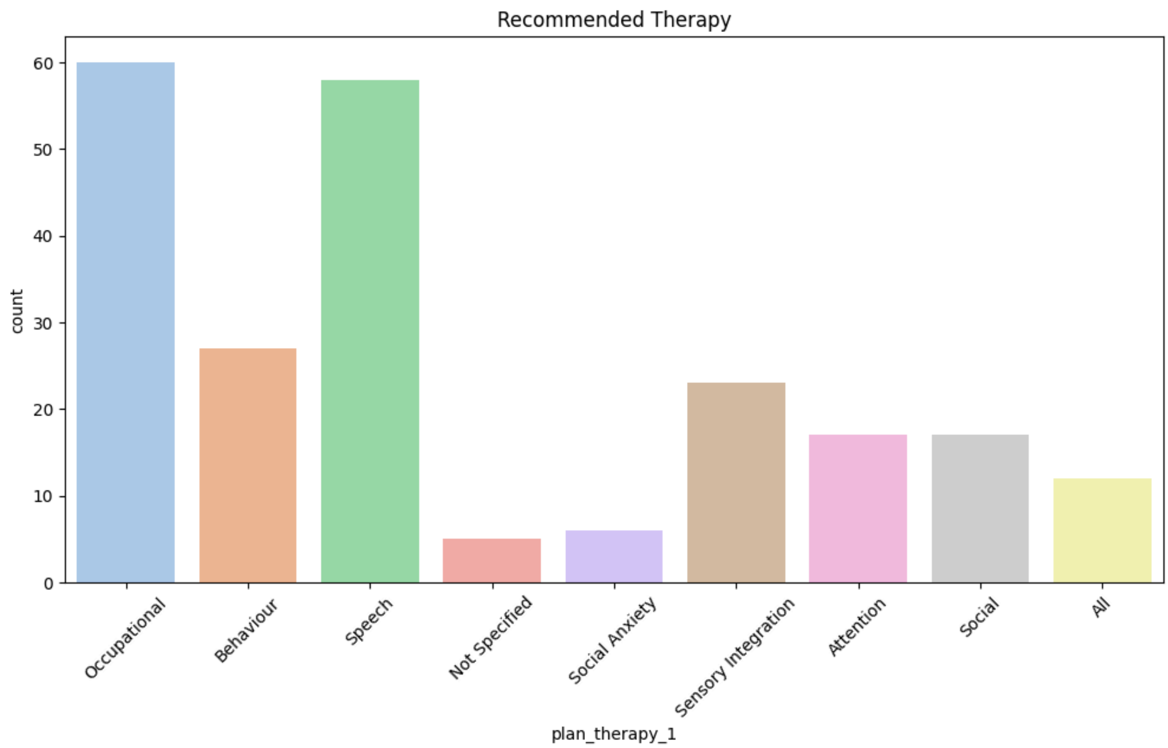


In [11]:
```python
plt.figure(figsize=(12, 6))
sns.countplot(x='plan_therapy_1', data=dataset, palette='pastel')
plt.title('Recommended Therapy')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\geekp\AppData\Local\Temp\ipykernel_10232\3246047537.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='plan_therapy_1', data=dataset, palette='pastel')
```



In [12]:
```
therapy_autism_ct = pd.crosstab(dataset['level_ASD'], dataset['plan_therapy_1'])
print("\nAutism Level vs Therapy Plan:\n", therapy_autism_ct)
```

```
Autism Level vs Therapy Plan:
 plan_therapy_1  All  Attention  Behaviour  Not Specified  Occupational  \
level_ASD
Level 1           3         10         15              4            42
Level 2           5          4          9              1            15
Level 3           4          3          3              0             3

plan_therapy_1  Sensory Integration  Social  Social Anxiety  Speech
level_ASD
Level 1                          17      12               1      46
Level 2                           6       2               3      11
Level 3                           0       3               2       1
```

In [13]:
```
from scipy.stats import chi2_contingency, spearmanr

target_col = "plan_therapy_1"

df_encoded = dataset.apply(lambda x: pd.factorize(x)[0] if x.dtype == 'object' e

def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x, y)
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))
    r_corr = r - ((r-1)**2)/(n-1)
    k_corr = k - ((k-1)**2)/(n-1)
    return np.sqrt(phi2corr / min((k_corr-1), (r_corr-1)))

correlation_results = {}

for col in df_encoded.columns:
    if col != target_col:
```

```python
        if df_encoded[col].nunique() > 10:
            correlation, _ = spearmanr(df_encoded[col], df_encoded[target_col])
        else:
            correlation = cramers_v(df_encoded[col], df_encoded[target_col])

        correlation_results[col] = correlation

correlation_df = pd.DataFrame(list(correlation_results.items()), columns=["Featu
correlation_df = correlation_df.sort_values(by="Correlation", ascending=False)

print("\nCorrelation of All Features with Target Column:\n")
print(correlation_df)

plt.figure(figsize=(15, 8))
sns.barplot(x=correlation_df["Feature"], y=correlation_df["Correlation"], palett
plt.xticks(rotation=45)
plt.ylabel("Correlation")
plt.title(f"Feature Correlation with {target_col}")
plt.show()
```

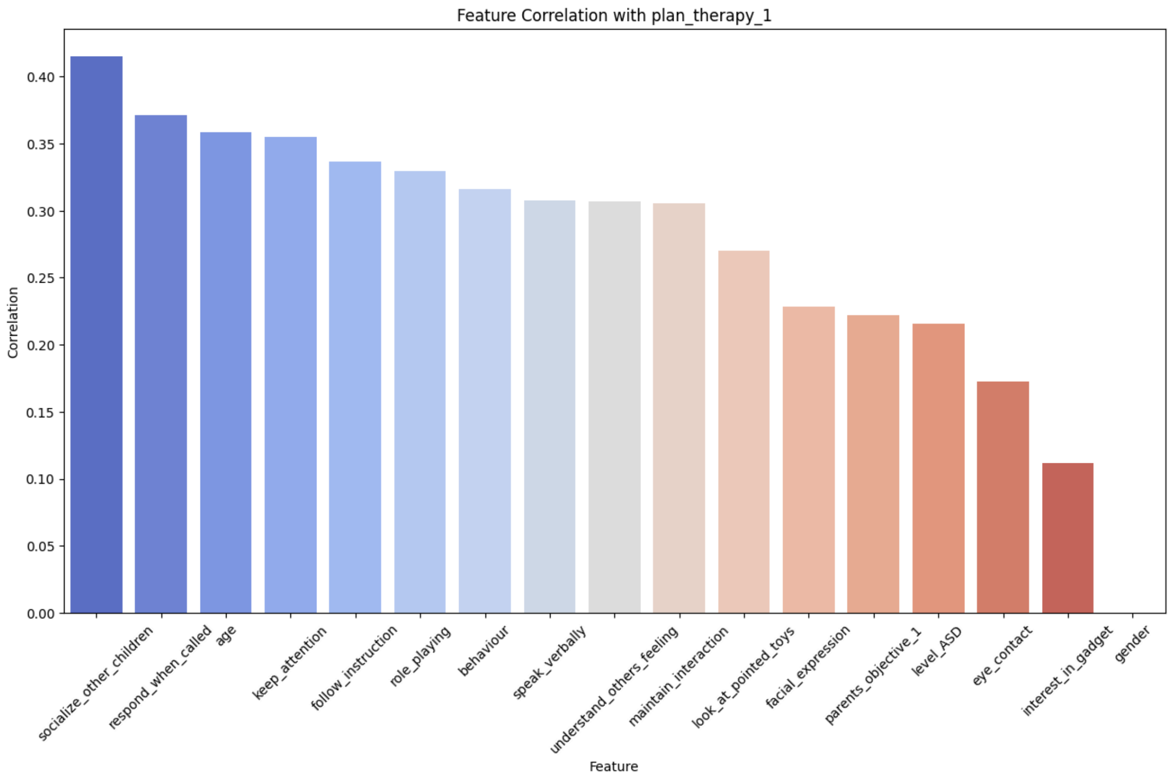
```
Correlation of All Features with Target Column:

                     Feature  Correlation
6    socialize_other_children     0.414951
12         respond_when_called     0.371012
0                          age     0.358697
13              keep_attention     0.354792
4           follow_instruction     0.336502
8                 role_playing     0.329356
15                   behaviour     0.315808
3                speak_verbally     0.307299
10   understand_others_feeling     0.306569
5         maintain_interaction     0.305806
11         look_at_pointed_toys     0.270019
9            facial_expression     0.228510
16         parents_objective_1     0.221934
2                    level_ASD     0.215310
7                  eye_contact     0.172170
14           interest_in_gadget     0.112000
1                       gender     0.000000
```

```
C:\Users\geekp\AppData\Local\Temp\ipykernel_10232\62164871.py:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=correlation_df["Feature"], y=correlation_df["Correlation"], palet
te="coolwarm")
```



In [14]:

```
from sklearn.model_selection import train_test_split

important_features = [
    "socialize_other_children",
    "respond_when_called",
    "age",
    "keep_attention",
    "follow_instruction",
    "role_playing",
    "behaviour",
    "speak_verbally",
    "understand_others_feeling",
    "maintain_interaction",
    "look_at_pointed_toys",
    "facial_expression",
    "parents_objective_1",
    "level_ASD",
    "eye_contact"
]
```

In [15]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225 entries, 0 to 224
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   age                       225 non-null    object
 1   gender                    225 non-null    object
 2   level_ASD                 225 non-null    object
 3   speak_verbally            225 non-null    object
 4   follow_instruction        225 non-null    object
 5   maintain_interaction      225 non-null    object
 6   socialize_other_children  225 non-null    object
 7   eye_contact               225 non-null    object
 8   role_playing              225 non-null    object
 9   facial_expression         225 non-null    object
 10  understand_others_feeling 225 non-null    object
 11  look_at_pointed_toys      225 non-null    object
 12  respond_when_called       225 non-null    object
 13  keep_attention            225 non-null    object
 14  interest_in_gadget        225 non-null    object
 15  behaviour                 225 non-null    object
 16  parents_objective_1       225 non-null    object
 17  plan_therapy_1            225 non-null    object
dtypes: object(18)
memory usage: 31.8+ KB
```

In [16]:
```
dataset["age"] = pd.to_numeric(dataset["age"], errors="coerce")  # Convert to nu
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
for col in dataset.columns:
    dataset[col] = le.fit_transform(dataset[col])  # Convert categorical to numb
```

In [18]:
```
dataset = pd.get_dummies(dataset, columns=["gender", "level_ASD"], drop_first=Tr
```

In [19]:
```
dataset.head()
```

Out[19]:

	age	speak_verbally	follow_instruction	maintain_interaction	socialize_other_children
0	0	1	2	2	2
1	0	1	2	2	2
2	0	0	2	2	0
3	0	1	2	0	2
4	0	0	2	2	2

In [20]:
```
dataset.isnull().sum()
```

Out[20]:
```
age                          0
speak_verbally               0
follow_instruction           0
maintain_interaction         0
socialize_other_children     0
eye_contact                  0
role_playing                 0
facial_expression            0
understand_others_feeling    0
look_at_pointed_toys         0
respond_when_called          0
keep_attention               0
interest_in_gadget           0
behaviour                    0
parents_objective_1          0
plan_therapy_1               0
gender_1                     0
level_ASD_1                  0
level_ASD_2                  0
dtype: int64
```

In [21]:
```
important_features = [
    "socialize_other_children",
    "respond_when_called",
    "age",
    "keep_attention",
    "follow_instruction",
    "role_playing",    "behaviour",
    "speak_verbally",
    "understand_others_feeling",
    "maintain_interaction",
    "look_at_pointed_toys",
    "facial_expression",
    "parents_objective_1",
    "level_ASD_1",
    "level_ASD_2",
    "eye_contact"
]
```

In [22]:
```
X = dataset.drop(['plan_therapy_1'],axis=1)
y = dataset["plan_therapy_1"]
```

In [23]:
```
X
```

Out[23]:

	age	speak_verbally	follow_instruction	maintain_interaction	socialize_other_childre
0	0	1	2	2	
1	0	1	2	2	
2	0	0	2	2	
3	0	1	2	0	
4	0	0	2	2	
...	...	...	...	...	
220	0	2	1	0	
221	0	2	1	0	
222	0	0	0	0	
223	0	2	0	0	
224	0	0	1	0	

225 rows × 18 columns

In [24]:

```
y
```

Out[24]:

```
0      4
1      2
2      8
3      3
4      7
      ..
220    0
221    0
222    4
223    4
224    2
Name: plan_therapy_1, Length: 225, dtype: int64
```

In [25]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [26]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.utils.class_weight import compute_class_weight
from imblearn.over_sampling import SMOTE

# 1. Prepare your dataset
X = X.copy()
for col in X.columns:
    if X[col].dtype == 'bool':
```

```python
        X[col] = X[col].astype(int)

X_np = X.to_numpy().astype(float)
y_np = y.to_numpy()  # assuming y is a pandas Series

# 2. Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_np)

# 3. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_np, test_size=0.

# 4. SMOTE for oversampling
smote = SMOTE(random_state=42, k_neighbors=2)
X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)

# 5. Compute class weights (optional since SMOTE balances it, but still useful)
classes = np.unique(y_train_sm)
weights = compute_class_weight(class_weight='balanced', classes=classes, y=y_tra
class_weight_dict = dict(zip(classes, weights))

# 6. Define models with class_weight
rf = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='bal
svm = SVC(probability=True, kernel='linear', random_state=42, class_weight='bala
xgb = XGBClassifier(n_estimators=100, use_label_encoder=False, eval_metric='mlog

# 7. Voting classifier
voting_model = VotingClassifier(
    estimators=[('rf', rf), ('xgb', xgb), ('svm', svm)],
    voting='soft'
)

# 8. Train
voting_model.fit(X_train_sm, y_train_sm)

# 9. Predict
y_pred_train = voting_model.predict(X_train_sm)
y_pred_test = voting_model.predict(X_test)

# 10. Evaluate
print("Training Accuracy:", accuracy_score(y_train_sm, y_pred_train))
print("Testing Accuracy:", accuracy_score(y_test, y_pred_test))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_test))
print("\nClassification Report:\n", classification_report(y_test, y_pred_test))
```

```
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\xgboost
\training.py:183: UserWarning: [16:09:42] WARNING: C:\actions-runner\_work\xgboos
t\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
Training Accuracy: 0.9953703703703703
Testing Accuracy: 0.6222222222222222

Confusion Matrix:
 [[ 1  0  0  0  0  0  1  0  0]
 [ 0  2  1  0  0  0  0  0  0]
 [ 0  0  6  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0]
 [ 0  1  3  0  4  1  3  0  0]
 [ 0  0  0  0  0  5  0  0  0]
 [ 0  0  0  1  2  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0]
 [ 0  0  0  0  2  0  0  0 10]]

Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.67      0.67      0.67         3
           2       0.60      1.00      0.75         6
           3       0.00      0.00      0.00         1
           4       0.40      0.33      0.36        12
           5       0.83      1.00      0.91         5
           6       0.00      0.00      0.00         3
           7       0.00      0.00      0.00         1
           8       1.00      0.83      0.91        12

    accuracy                           0.62        45
   macro avg       0.50      0.48      0.47        45
weighted avg       0.63      0.62      0.61        45
```

```
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [27]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep

rf = RandomForestClassifier(n_estimators=100, random_state=42)
xgb = XGBClassifier(n_estimators=100, use_label_encoder=False, eval_metric='mlog
svm = SVC(probability=True, kernel='linear', random_state=42)

meta_classifier = LogisticRegression()
```

```python
stacked_model = StackingClassifier(
    estimators=[('rf', rf), ('xgb', xgb), ('svm', svm)],
    final_estimator=meta_classifier
)

stacked_model.fit(X_train, y_train)

train_pred = stacked_model.predict(X_train)
test_pred = stacked_model.predict(X_test)

train_acc = accuracy_score(y_train, train_pred)
test_acc = accuracy_score(y_test, test_pred)
conf_matrix = confusion_matrix(y_test, test_pred)
class_report = classification_report(y_test, test_pred)

print(f"Training Accuracy: {train_acc:.4f}")
print(f"Testing Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\xgboost
\training.py:183: UserWarning: [16:09:43] WARNING: C:\actions-runner\_work\xgboos
t\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\model_selection\_split.py:805: UserWarning: The least populated class in y has o
nly 4 members, which is less than n_splits=5.
  warnings.warn(
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\model_selection\_split.py:805: UserWarning: The least populated class in y has o
nly 4 members, which is less than n_splits=5.
  warnings.warn(
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\xgboost
\training.py:183: UserWarning: [16:09:45] WARNING: C:\actions-runner\_work\xgboos
t\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\xgboost
\training.py:183: UserWarning: [16:09:46] WARNING: C:\actions-runner\_work\xgboos
t\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\xgboost
\training.py:183: UserWarning: [16:09:47] WARNING: C:\actions-runner\_work\xgboos
t\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\model_selection\_split.py:805: UserWarning: The least populated class in y has o
nly 4 members, which is less than n_splits=5.
  warnings.warn(
```

```
Training Accuracy: 0.8722
Testing Accuracy: 0.6667

Confusion Matrix:
[[1 0 0 0 1 0 0 0 0]
 [0 2 1 0 0 0 0 0 0]
 [0 0 6 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]
 [0 2 2 0 7 1 0 0 0]
 [0 0 0 0 0 5 0 0 0]
 [0 0 0 0 3 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]
 [0 0 0 0 3 0 0 0 9]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.50      0.67      0.57         3
           2       0.67      1.00      0.80         6
           3       0.00      0.00      0.00         1
           4       0.44      0.58      0.50        12
           5       0.83      1.00      0.91         5
           6       0.00      0.00      0.00         3
           7       0.00      0.00      0.00         1
           8       1.00      0.75      0.86        12

    accuracy                           0.67        45
   macro avg       0.49      0.50      0.48        45
weighted avg       0.64      0.67      0.64        45
```

```
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\geekp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [28]:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense, Input, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = X.copy()
for col in X.columns:
    if X[col].dtype == 'bool':
        X[col] = X[col].astype(int)
```

```python
X_np = X.to_numpy()  # or X.values


X_np = X_np.astype(float)  # ensure float for StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_np)

X_reshaped = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))


X_train, X_test, y_train, y_test = train_test_split(X_reshaped, y, test_size=0.2

y_train = to_categorical(y_train, num_classes=9)
y_test = to_categorical(y_test, num_classes=9)

input_layer = Input(shape=(X_reshaped.shape[1], X_reshaped.shape[2]))

lstm_out = LSTM(128, return_sequences=False)(input_layer)
lstm_out = Dropout(0.3)(lstm_out)

dense_out = Dense(64, activation="relu")(lstm_out)
dense_out = Dropout(0.3)(dense_out)

output_layer = Dense(9, activation="softmax")(dense_out)

optimizer = Adam(learning_rate=0.0005)
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['ac

model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")
```

```
Epoch 1/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 5s 160ms/step - accuracy: 0.1665 - loss: 2.1665 - val_ac
curacy: 0.1778 - val_loss: 2.1782
Epoch 2/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.2452 - loss: 2.1473 - val_acc
uracy: 0.2667 - val_loss: 2.1627
Epoch 3/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.2919 - loss: 2.1278 - val_acc
uracy: 0.2667 - val_loss: 2.1471
Epoch 4/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.3340 - loss: 2.1242 - val_acc
uracy: 0.4000 - val_loss: 2.1306
Epoch 5/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.3348 - loss: 2.1098 - val_acc
uracy: 0.4444 - val_loss: 2.1129
Epoch 6/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.3851 - loss: 2.0850 - val_acc
uracy: 0.4667 - val_loss: 2.0945
Epoch 7/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.4121 - loss: 2.0457 - val_acc
uracy: 0.4444 - val_loss: 2.0752
Epoch 8/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.4074 - loss: 2.0341 - val_acc
uracy: 0.4444 - val_loss: 2.0552
Epoch 9/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.4546 - loss: 1.9984 - val_acc
uracy: 0.4444 - val_loss: 2.0338
Epoch 10/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4660 - loss: 1.9804 - val_acc
uracy: 0.4444 - val_loss: 2.0118
Epoch 11/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.4692 - loss: 1.9591 - val_acc
uracy: 0.4000 - val_loss: 1.9884
Epoch 12/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.4142 - loss: 1.9310 - val_acc
uracy: 0.4000 - val_loss: 1.9642
Epoch 13/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.3976 - loss: 1.9166 - val_acc
uracy: 0.4222 - val_loss: 1.9387
Epoch 14/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.4235 - loss: 1.8674 - val_acc
uracy: 0.4222 - val_loss: 1.9124
Epoch 15/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.4785 - loss: 1.8019 - val_acc
uracy: 0.4222 - val_loss: 1.8852
Epoch 16/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4497 - loss: 1.8040 - val_acc
uracy: 0.4222 - val_loss: 1.8583
Epoch 17/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.4327 - loss: 1.8078 - val_acc
uracy: 0.4222 - val_loss: 1.8324
Epoch 18/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.4493 - loss: 1.7372 - val_acc
uracy: 0.4222 - val_loss: 1.8068
Epoch 19/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5000 - loss: 1.6430 - val_acc
uracy: 0.4222 - val_loss: 1.7843
Epoch 20/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.4210 - loss: 1.6762 - val_acc
uracy: 0.4222 - val_loss: 1.7638
```

```
Epoch 21/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.4688 - loss: 1.6225 - val_acc
uracy: 0.4222 - val_loss: 1.7428
Epoch 22/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4776 - loss: 1.5860 - val_acc
uracy: 0.4222 - val_loss: 1.7230
Epoch 23/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.4393 - loss: 1.6076 - val_acc
uracy: 0.4222 - val_loss: 1.7055
Epoch 24/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.4741 - loss: 1.5680 - val_acc
uracy: 0.4444 - val_loss: 1.6874
Epoch 25/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.4206 - loss: 1.5990 - val_acc
uracy: 0.4444 - val_loss: 1.6715
Epoch 26/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.4502 - loss: 1.5651 - val_acc
uracy: 0.4444 - val_loss: 1.6544
Epoch 27/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.4905 - loss: 1.5101 - val_acc
uracy: 0.4667 - val_loss: 1.6396
Epoch 28/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 0.4703 - loss: 1.5182 - val_acc
uracy: 0.4889 - val_loss: 1.6253
Epoch 29/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.4851 - loss: 1.4722 - val_acc
uracy: 0.4889 - val_loss: 1.6122
Epoch 30/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4818 - loss: 1.4591 - val_acc
uracy: 0.4889 - val_loss: 1.6008
Epoch 31/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.4761 - loss: 1.4716 - val_acc
uracy: 0.4889 - val_loss: 1.5890
Epoch 32/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5005 - loss: 1.4841 - val_acc
uracy: 0.4889 - val_loss: 1.5762
Epoch 33/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.4822 - loss: 1.4110 - val_acc
uracy: 0.5111 - val_loss: 1.5647
Epoch 34/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5219 - loss: 1.4152 - val_acc
uracy: 0.5333 - val_loss: 1.5530
Epoch 35/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5074 - loss: 1.4067 - val_acc
uracy: 0.5333 - val_loss: 1.5397
Epoch 36/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5497 - loss: 1.3231 - val_acc
uracy: 0.5333 - val_loss: 1.5281
Epoch 37/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5661 - loss: 1.3563 - val_acc
uracy: 0.5333 - val_loss: 1.5177
Epoch 38/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.5080 - loss: 1.3685 - val_acc
uracy: 0.5111 - val_loss: 1.5084
Epoch 39/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.5627 - loss: 1.3089 - val_acc
uracy: 0.5111 - val_loss: 1.4987
Epoch 40/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.4857 - loss: 1.3846 - val_acc
uracy: 0.5111 - val_loss: 1.4882
```

```
Epoch 41/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.5180 - loss: 1.3212 - val_acc
uracy: 0.5111 - val_loss: 1.4802
Epoch 42/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.5887 - loss: 1.2422 - val_acc
uracy: 0.5111 - val_loss: 1.4689
Epoch 43/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5566 - loss: 1.2706 - val_acc
uracy: 0.5111 - val_loss: 1.4609
Epoch 44/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.5569 - loss: 1.2744 - val_acc
uracy: 0.5111 - val_loss: 1.4496
Epoch 45/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5409 - loss: 1.2667 - val_acc
uracy: 0.5111 - val_loss: 1.4389
Epoch 46/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.5270 - loss: 1.2576 - val_acc
uracy: 0.5111 - val_loss: 1.4316
Epoch 47/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6007 - loss: 1.1940 - val_acc
uracy: 0.5111 - val_loss: 1.4247
Epoch 48/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5555 - loss: 1.2347 - val_acc
uracy: 0.5111 - val_loss: 1.4189
Epoch 49/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5575 - loss: 1.1408 - val_acc
uracy: 0.5111 - val_loss: 1.4126
Epoch 50/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6108 - loss: 1.1757 - val_acc
uracy: 0.5111 - val_loss: 1.4054
Epoch 51/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5570 - loss: 1.2133 - val_acc
uracy: 0.5111 - val_loss: 1.3975
Epoch 52/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.5909 - loss: 1.1707 - val_acc
uracy: 0.4889 - val_loss: 1.3911
Epoch 53/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5625 - loss: 1.1775 - val_acc
uracy: 0.4889 - val_loss: 1.3853
Epoch 54/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.5721 - loss: 1.0937 - val_acc
uracy: 0.4889 - val_loss: 1.3834
Epoch 55/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.5837 - loss: 1.1372 - val_acc
uracy: 0.4889 - val_loss: 1.3795
Epoch 56/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.6073 - loss: 1.1265 - val_acc
uracy: 0.4889 - val_loss: 1.3753
Epoch 57/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5824 - loss: 1.0972 - val_acc
uracy: 0.4889 - val_loss: 1.3701
Epoch 58/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6035 - loss: 1.1541 - val_acc
uracy: 0.4889 - val_loss: 1.3656
Epoch 59/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.5223 - loss: 1.1469 - val_acc
uracy: 0.4667 - val_loss: 1.3642
Epoch 60/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5291 - loss: 1.1869 - val_acc
uracy: 0.4889 - val_loss: 1.3620
```

```
Epoch 61/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5868 - loss: 1.1653 - val_acc
uracy: 0.4667 - val_loss: 1.3614
Epoch 62/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5362 - loss: 1.1827 - val_acc
uracy: 0.4667 - val_loss: 1.3632
Epoch 63/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.6027 - loss: 1.0293 - val_acc
uracy: 0.4667 - val_loss: 1.3611
Epoch 64/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6309 - loss: 1.0695 - val_acc
uracy: 0.4667 - val_loss: 1.3602
Epoch 65/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6413 - loss: 0.9947 - val_acc
uracy: 0.4667 - val_loss: 1.3572
Epoch 66/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.6098 - loss: 1.0787 - val_acc
uracy: 0.4889 - val_loss: 1.3573
Epoch 67/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.5921 - loss: 1.0806 - val_acc
uracy: 0.4889 - val_loss: 1.3571
Epoch 68/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.6438 - loss: 1.0455 - val_acc
uracy: 0.4889 - val_loss: 1.3526
Epoch 69/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6429 - loss: 0.9999 - val_acc
uracy: 0.4889 - val_loss: 1.3520
Epoch 70/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.6114 - loss: 1.1020 - val_acc
uracy: 0.4889 - val_loss: 1.3511
Epoch 71/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6033 - loss: 1.0930 - val_acc
uracy: 0.5111 - val_loss: 1.3516
Epoch 72/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6604 - loss: 1.0363 - val_acc
uracy: 0.5111 - val_loss: 1.3513
Epoch 73/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6659 - loss: 1.0363 - val_acc
uracy: 0.4889 - val_loss: 1.3505
Epoch 74/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.6611 - loss: 0.9272 - val_acc
uracy: 0.5111 - val_loss: 1.3513
Epoch 75/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.6370 - loss: 1.0102 - val_acc
uracy: 0.5111 - val_loss: 1.3518
Epoch 76/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.6400 - loss: 0.9784 - val_acc
uracy: 0.5111 - val_loss: 1.3539
Epoch 77/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.6483 - loss: 0.9845 - val_acc
uracy: 0.5111 - val_loss: 1.3550
Epoch 78/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.6343 - loss: 1.0165 - val_acc
uracy: 0.5111 - val_loss: 1.3575
Epoch 79/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6728 - loss: 0.9308 - val_acc
uracy: 0.5111 - val_loss: 1.3610
Epoch 80/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.6022 - loss: 1.0136 - val_acc
uracy: 0.5111 - val_loss: 1.3607
```

```
Epoch 81/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6894 - loss: 0.9411 - val_acc
uracy: 0.5111 - val_loss: 1.3626
Epoch 82/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6021 - loss: 1.0180 - val_acc
uracy: 0.5111 - val_loss: 1.3628
Epoch 83/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.6197 - loss: 1.0465 - val_acc
uracy: 0.5111 - val_loss: 1.3616
Epoch 84/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.6343 - loss: 0.9470 - val_acc
uracy: 0.5111 - val_loss: 1.3607
Epoch 85/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6405 - loss: 0.9838 - val_acc
uracy: 0.4889 - val_loss: 1.3608
Epoch 86/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6840 - loss: 0.9229 - val_acc
uracy: 0.4889 - val_loss: 1.3628
Epoch 87/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6739 - loss: 0.9053 - val_acc
uracy: 0.5111 - val_loss: 1.3663
Epoch 88/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6789 - loss: 0.8680 - val_acc
uracy: 0.5111 - val_loss: 1.3717
Epoch 89/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6650 - loss: 0.9358 - val_acc
uracy: 0.5333 - val_loss: 1.3754
Epoch 90/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.7207 - loss: 0.8462 - val_acc
uracy: 0.5111 - val_loss: 1.3769
Epoch 91/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.6019 - loss: 0.9665 - val_acc
uracy: 0.5111 - val_loss: 1.3754
Epoch 92/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6371 - loss: 0.9892 - val_acc
uracy: 0.5111 - val_loss: 1.3736
Epoch 93/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.6667 - loss: 0.9067 - val_acc
uracy: 0.5111 - val_loss: 1.3731
Epoch 94/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.7059 - loss: 0.9127 - val_acc
uracy: 0.5111 - val_loss: 1.3721
Epoch 95/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.6988 - loss: 0.8202 - val_acc
uracy: 0.5111 - val_loss: 1.3702
Epoch 96/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.7002 - loss: 0.8491 - val_acc
uracy: 0.5111 - val_loss: 1.3675
Epoch 97/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.7170 - loss: 0.8627 - val_acc
uracy: 0.5333 - val_loss: 1.3690
Epoch 98/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6411 - loss: 0.8835 - val_acc
uracy: 0.4889 - val_loss: 1.3712
Epoch 99/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.7210 - loss: 0.8215 - val_acc
uracy: 0.4889 - val_loss: 1.3750
Epoch 100/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.6784 - loss: 0.8527 - val_acc
uracy: 0.4889 - val_loss: 1.3813
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.4822 - loss: 1.3617
Test Accuracy: 0.4889
```

In [29]:

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense, Input, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils.class_weight import compute_class_weight

# 1. Preprocess
X = X.copy()
for col in X.columns:
    if X[col].dtype == 'bool':
        X[col] = X[col].astype(int)

X_np = X.to_numpy().astype(float)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_np)

# 2. Reshape for LSTM
X_reshaped = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))

# 3. Train-test split
X_train, X_test, y_train_raw, y_test_raw = train_test_split(X_reshaped, y, test_

# 4. Compute class weights
import numpy as np
from collections import Counter
from sklearn.utils.class_weight import compute_class_weight

classes = np.unique(y_train_raw)
class_weights = compute_class_weight(class_weight='balanced', classes=classes, y
class_weights_dict = dict(zip(classes, class_weights))
print("Class weights:", class_weights_dict)

# 5. One-hot encode targets
y_train = to_categorical(y_train_raw, num_classes=9)
y_test = to_categorical(y_test_raw, num_classes=9)

# 6. Build LSTM Model
input_layer = Input(shape=(X_reshaped.shape[1], X_reshaped.shape[2]))
lstm_out = LSTM(128, return_sequences=False)(input_layer)
lstm_out = Dropout(0.3)(lstm_out)
dense_out = Dense(64, activation="relu")(lstm_out)
dense_out = Dropout(0.3)(dense_out)
output_layer = Dense(9, activation="softmax")(dense_out)

optimizer = Adam(learning_rate=0.0005)
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['ac

# 7. Train with class weights
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,

# 8. Evaluate
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")
```

```
Class weights: {np.int64(0): np.float64(2.0), np.int64(1): np.float64(1.538461538
4615385), np.int64(2): np.float64(0.9090909090909091), np.int64(3): np.float64(5.
0), np.int64(4): np.float64(0.4166666666666667), np.int64(5): np.float64(1.111111
1111111112), np.int64(6): np.float64(1.6666666666666667), np.int64(7): np.float64
(5.0), np.int64(8): np.float64(0.40816326530612246)}
Epoch 1/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 4s 130ms/step - accuracy: 0.1176 - loss: 2.1835 - val_ac
curacy: 0.1333 - val_loss: 2.1790
Epoch 2/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.1531 - loss: 2.3647 - val_acc
uracy: 0.1556 - val_loss: 2.1745
Epoch 3/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.1256 - loss: 2.2508 - val_acc
uracy: 0.1778 - val_loss: 2.1691
Epoch 4/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.1934 - loss: 2.2626 - val_acc
uracy: 0.2667 - val_loss: 2.1627
Epoch 5/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.2415 - loss: 2.1992 - val_acc
uracy: 0.3556 - val_loss: 2.1559
Epoch 6/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.2692 - loss: 2.0376 - val_acc
uracy: 0.3333 - val_loss: 2.1487
Epoch 7/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.2522 - loss: 1.9527 - val_acc
uracy: 0.3333 - val_loss: 2.1409
Epoch 8/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.2774 - loss: 2.2241 - val_acc
uracy: 0.3111 - val_loss: 2.1340
Epoch 9/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.2665 - loss: 2.1505 - val_acc
uracy: 0.3111 - val_loss: 2.1266
Epoch 10/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.3687 - loss: 2.1735 - val_acc
uracy: 0.3111 - val_loss: 2.1189
Epoch 11/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.3762 - loss: 1.9138 - val_acc
uracy: 0.2889 - val_loss: 2.1099
Epoch 12/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 0.3889 - loss: 2.1242 - val_acc
uracy: 0.2889 - val_loss: 2.1013
Epoch 13/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.3766 - loss: 2.0894 - val_acc
uracy: 0.2889 - val_loss: 2.0919
Epoch 14/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.3763 - loss: 1.9998 - val_acc
uracy: 0.2889 - val_loss: 2.0821
Epoch 15/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.3119 - loss: 2.0138 - val_acc
uracy: 0.2889 - val_loss: 2.0708
Epoch 16/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.3357 - loss: 2.1197 - val_acc
uracy: 0.2889 - val_loss: 2.0590
Epoch 17/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.3248 - loss: 1.9259 - val_acc
uracy: 0.2889 - val_loss: 2.0468
Epoch 18/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4165 - loss: 2.0468 - val_acc
uracy: 0.3333 - val_loss: 2.0347
Epoch 19/100
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.3974 - loss: 1.9360 - val_acc
uracy: 0.3333 - val_loss: 2.0217
Epoch 20/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.3820 - loss: 2.0145 - val_acc
uracy: 0.3556 - val_loss: 2.0087
Epoch 21/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.3482 - loss: 2.0491 - val_acc
uracy: 0.3556 - val_loss: 1.9954
Epoch 22/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.4229 - loss: 1.8774 - val_acc
uracy: 0.3556 - val_loss: 1.9816
Epoch 23/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.3837 - loss: 1.8897 - val_acc
uracy: 0.3556 - val_loss: 1.9670
Epoch 24/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.4133 - loss: 1.8532 - val_acc
uracy: 0.3556 - val_loss: 1.9525
Epoch 25/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.3734 - loss: 1.8010 - val_acc
uracy: 0.3778 - val_loss: 1.9376
Epoch 26/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.3997 - loss: 1.8569 - val_acc
uracy: 0.4000 - val_loss: 1.9229
Epoch 27/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.3637 - loss: 1.7326 - val_acc
uracy: 0.4000 - val_loss: 1.9064
Epoch 28/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.3882 - loss: 1.6654 - val_acc
uracy: 0.4000 - val_loss: 1.8900
Epoch 29/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.4143 - loss: 1.8036 - val_acc
uracy: 0.4000 - val_loss: 1.8743
Epoch 30/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.4430 - loss: 1.6368 - val_acc
uracy: 0.4000 - val_loss: 1.8569
Epoch 31/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.4310 - loss: 1.7309 - val_acc
uracy: 0.4000 - val_loss: 1.8400
Epoch 32/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4343 - loss: 1.6131 - val_acc
uracy: 0.4000 - val_loss: 1.8237
Epoch 33/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.4459 - loss: 1.5352 - val_acc
uracy: 0.4000 - val_loss: 1.8071
Epoch 34/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.4533 - loss: 1.7056 - val_acc
uracy: 0.4222 - val_loss: 1.7902
Epoch 35/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.3441 - loss: 1.6020 - val_acc
uracy: 0.4444 - val_loss: 1.7731
Epoch 36/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4351 - loss: 1.6863 - val_acc
uracy: 0.4444 - val_loss: 1.7561
Epoch 37/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.4111 - loss: 1.4889 - val_acc
uracy: 0.4444 - val_loss: 1.7390
Epoch 38/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4727 - loss: 1.6135 - val_acc
uracy: 0.4444 - val_loss: 1.7219
Epoch 39/100
```

```
6/6 0s 32ms/step - accuracy: 0.4526 - loss: 1.5498 - val_acc
uracy: 0.4444 - val_loss: 1.7065
Epoch 40/100
6/6 0s 29ms/step - accuracy: 0.4441 - loss: 1.4182 - val_acc
uracy: 0.4444 - val_loss: 1.6904
Epoch 41/100
6/6 0s 31ms/step - accuracy: 0.4558 - loss: 1.5153 - val_acc
uracy: 0.4444 - val_loss: 1.6742
Epoch 42/100
6/6 0s 28ms/step - accuracy: 0.4901 - loss: 1.3421 - val_acc
uracy: 0.4444 - val_loss: 1.6587
Epoch 43/100
6/6 0s 35ms/step - accuracy: 0.4961 - loss: 1.3213 - val_acc
uracy: 0.4444 - val_loss: 1.6444
Epoch 44/100
6/6 0s 36ms/step - accuracy: 0.4924 - loss: 1.3113 - val_acc
uracy: 0.4444 - val_loss: 1.6301
Epoch 45/100
6/6 0s 38ms/step - accuracy: 0.4971 - loss: 1.4936 - val_acc
uracy: 0.4444 - val_loss: 1.6164
Epoch 46/100
6/6 0s 27ms/step - accuracy: 0.4563 - loss: 1.3514 - val_acc
uracy: 0.4444 - val_loss: 1.6016
Epoch 47/100
6/6 0s 40ms/step - accuracy: 0.4778 - loss: 1.3166 - val_acc
uracy: 0.4444 - val_loss: 1.5872
Epoch 48/100
6/6 0s 30ms/step - accuracy: 0.4860 - loss: 1.3407 - val_acc
uracy: 0.4444 - val_loss: 1.5746
Epoch 49/100
6/6 0s 31ms/step - accuracy: 0.4908 - loss: 1.3128 - val_acc
uracy: 0.4444 - val_loss: 1.5626
Epoch 50/100
6/6 0s 27ms/step - accuracy: 0.4887 - loss: 1.1717 - val_acc
uracy: 0.4444 - val_loss: 1.5497
Epoch 51/100
6/6 0s 27ms/step - accuracy: 0.5035 - loss: 1.3144 - val_acc
uracy: 0.4444 - val_loss: 1.5380
Epoch 52/100
6/6 0s 27ms/step - accuracy: 0.4781 - loss: 1.3436 - val_acc
uracy: 0.4667 - val_loss: 1.5262
Epoch 53/100
6/6 0s 28ms/step - accuracy: 0.4867 - loss: 1.3151 - val_acc
uracy: 0.4667 - val_loss: 1.5154
Epoch 54/100
6/6 0s 25ms/step - accuracy: 0.4566 - loss: 1.2994 - val_acc
uracy: 0.4667 - val_loss: 1.5078
Epoch 55/100
6/6 0s 28ms/step - accuracy: 0.5507 - loss: 1.1238 - val_acc
uracy: 0.4667 - val_loss: 1.5000
Epoch 56/100
6/6 0s 29ms/step - accuracy: 0.4847 - loss: 1.2771 - val_acc
uracy: 0.4667 - val_loss: 1.4932
Epoch 57/100
6/6 0s 28ms/step - accuracy: 0.4944 - loss: 1.2042 - val_acc
uracy: 0.4667 - val_loss: 1.4880
Epoch 58/100
6/6 0s 28ms/step - accuracy: 0.4838 - loss: 1.1988 - val_acc
uracy: 0.4667 - val_loss: 1.4817
Epoch 59/100
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.5081 - loss: 1.0949 - val_acc
uracy: 0.4667 - val_loss: 1.4749
Epoch 60/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.5176 - loss: 1.1783 - val_acc
uracy: 0.4667 - val_loss: 1.4698
Epoch 61/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.5009 - loss: 1.1287 - val_acc
uracy: 0.4667 - val_loss: 1.4636
Epoch 62/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.3975 - loss: 1.2156 - val_acc
uracy: 0.4889 - val_loss: 1.4567
Epoch 63/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.4749 - loss: 1.1822 - val_acc
uracy: 0.4667 - val_loss: 1.4509
Epoch 64/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5432 - loss: 1.1569 - val_acc
uracy: 0.4444 - val_loss: 1.4441
Epoch 65/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5053 - loss: 0.9867 - val_acc
uracy: 0.4444 - val_loss: 1.4361
Epoch 66/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4774 - loss: 1.0701 - val_acc
uracy: 0.4222 - val_loss: 1.4321
Epoch 67/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5064 - loss: 1.1249 - val_acc
uracy: 0.4222 - val_loss: 1.4292
Epoch 68/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.5858 - loss: 0.9651 - val_acc
uracy: 0.4222 - val_loss: 1.4273
Epoch 69/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5055 - loss: 1.1373 - val_acc
uracy: 0.4222 - val_loss: 1.4250
Epoch 70/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4967 - loss: 1.0068 - val_acc
uracy: 0.4000 - val_loss: 1.4213
Epoch 71/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.5354 - loss: 1.1226 - val_acc
uracy: 0.3778 - val_loss: 1.4211
Epoch 72/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5725 - loss: 1.0603 - val_acc
uracy: 0.3778 - val_loss: 1.4188
Epoch 73/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5652 - loss: 1.0735 - val_acc
uracy: 0.3778 - val_loss: 1.4149
Epoch 74/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.5905 - loss: 0.9803 - val_acc
uracy: 0.4000 - val_loss: 1.4101
Epoch 75/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4839 - loss: 1.0241 - val_acc
uracy: 0.4000 - val_loss: 1.4060
Epoch 76/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5254 - loss: 0.9864 - val_acc
uracy: 0.4000 - val_loss: 1.4026
Epoch 77/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5779 - loss: 0.9518 - val_acc
uracy: 0.4000 - val_loss: 1.4010
Epoch 78/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5070 - loss: 1.0323 - val_acc
uracy: 0.4000 - val_loss: 1.4005
Epoch 79/100
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.5355 - loss: 0.9526 - val_acc
uracy: 0.4000 - val_loss: 1.3987
Epoch 80/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.5352 - loss: 1.0556 - val_acc
uracy: 0.4000 - val_loss: 1.3978
Epoch 81/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.5838 - loss: 1.0070 - val_acc
uracy: 0.3778 - val_loss: 1.3970
Epoch 82/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.5847 - loss: 0.9633 - val_acc
uracy: 0.3778 - val_loss: 1.3960
Epoch 83/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6095 - loss: 0.9388 - val_acc
uracy: 0.3778 - val_loss: 1.3948
Epoch 84/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.5532 - loss: 0.8760 - val_acc
uracy: 0.3778 - val_loss: 1.3935
Epoch 85/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.5369 - loss: 1.0044 - val_acc
uracy: 0.3556 - val_loss: 1.3941
Epoch 86/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.5310 - loss: 0.9396 - val_acc
uracy: 0.3778 - val_loss: 1.3944
Epoch 87/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5556 - loss: 0.9208 - val_acc
uracy: 0.3778 - val_loss: 1.3962
Epoch 88/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5370 - loss: 0.8535 - val_acc
uracy: 0.3556 - val_loss: 1.3971
Epoch 89/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5256 - loss: 0.9078 - val_acc
uracy: 0.3778 - val_loss: 1.3957
Epoch 90/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.5581 - loss: 0.9026 - val_acc
uracy: 0.4000 - val_loss: 1.3962
Epoch 91/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 0.5578 - loss: 0.9105 - val_acc
uracy: 0.3778 - val_loss: 1.3947
Epoch 92/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.5455 - loss: 0.9464 - val_acc
uracy: 0.4000 - val_loss: 1.3937
Epoch 93/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.5182 - loss: 0.9297 - val_acc
uracy: 0.4000 - val_loss: 1.3930
Epoch 94/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.5668 - loss: 0.9084 - val_acc
uracy: 0.3778 - val_loss: 1.3915
Epoch 95/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6619 - loss: 0.7642 - val_acc
uracy: 0.3778 - val_loss: 1.3900
Epoch 96/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.6424 - loss: 0.8199 - val_acc
uracy: 0.4000 - val_loss: 1.3910
Epoch 97/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.5905 - loss: 0.8691 - val_acc
uracy: 0.4000 - val_loss: 1.3910
Epoch 98/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.5398 - loss: 0.8672 - val_acc
uracy: 0.4000 - val_loss: 1.3905
Epoch 99/100
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.6105 - loss: 0.8292 - val_acc
uracy: 0.4000 - val_loss: 1.3904
Epoch 100/100
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.5703 - loss: 0.8099 - val_acc
uracy: 0.4000 - val_loss: 1.3887
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.4229 - loss: 1.3795
Test Accuracy: 0.4000
```

In [ ]: