Name: Gullas Rainer L.

Section: BSCPE32S3

Date Perfomred: 04/15/2024

Date Submitted: 04/19/2024

Instructor: Engr. Roman Richard

```
!pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
```

```
from ucimlrepo import fetch_ucirepo

wine = fetch_ucirepo(id=109)

X = wine.data.features
y = wine.data.targets

print(wine.metadata)
print(wine.variables)
```

```
{'uci_id': 109, 'name': 'Wine', 'repository_url': 'https://archive.ics.uci.edu/dataset/109/wine', 'data_url': 'https://archive.ics.uci.e
                          name     role         type demographic  \
0                        class   Target  Categorical        None
1                      Alcohol  Feature   Continuous        None
2                    Malicacid  Feature   Continuous        None
3                          Ash  Feature   Continuous        None
4              Alcalinity_of_ash  Feature   Continuous        None
5                    Magnesium  Feature      Integer        None
6                Total_phenols  Feature   Continuous        None
7                   Flavanoids  Feature   Continuous        None
8           Nonflavanoid_phenols  Feature   Continuous        None
9              Proanthocyanins  Feature   Continuous        None
10              Color_intensity  Feature   Continuous        None
11                          Hue  Feature   Continuous        None
12  0D280_0D315_of_diluted_wines  Feature   Continuous        None
13                      Proline  Feature      Integer        None

   description units missing_values
0         None  None             no
1         None  None             no
2         None  None             no
3         None  None             no
4         None  None             no
5         None  None             no
6         None  None             no
7         None  None             no
8         None  None             no
9         None  None             no
10        None  None             no
11        None  None             no
12        None  None             no
13        None  None             no
```

```
X
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanoids	N
0	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	

178 rows × 13 columns

Next steps: View recommended plots

```
1 y
```

	class
0	1
1	1
2	1
3	1
4	1
...	...
173	3
174	3
175	3
176	3
177	3

178 rows × 1 columns

Next steps: View recommended plots

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 from sklearn.metrics import roc_curve, auc
6 from sklearn.model_selection import train_test_split
7 from sklearn.preprocessing import StandardScaler
8 from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_auc_score, roc_curve, accuracy_score
9 from sklearn.ensemble import RandomForestClassifier
10
11 import seaborn as sns
12 %matplotlib inline
13
14 from keras.models  import Sequential
15 from keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
16 from keras.optimizers import Adam, SGD, RMSprop
17 from sklearn.model_selection import cross_val_score, KFold, train_test_split
18 from sklearn.ensemble import RandomForestClassifier
19 from sklearn.metrics import accuracy_score, classification_report
20 from sklearn.datasets import make_classification
```

```
null_values = X.isnull().sum()

print(null_values)
```

```
Alcohol                         0
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavanoids                      0
Nonflavanoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
0D280_0D315_of_diluted_wines    0
Proline                         0
dtype: int64
```

```
null_values = X.isnull().sum()

print(null_values)
```

```
Alcohol                         0
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavanoids                      0
Nonflavanoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
0D280_0D315_of_diluted_wines    0
Proline                         0
dtype: int64
```

```
X_1 = X.iloc[:, 0].values
y_1 = y["class"].values
```

Saving the Model as HDF5 format

```
import joblib
import h5py
import tempfile
import shutil
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)

from keras.layers import GaussianNoise
input_shape = (X_train_norm.shape[1],)
model = Sequential([
    Dense(10, input_shape=input_shape, activation='relu'),
    GaussianNoise(0.5),
    Dense(3, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)
```

```
<keras.src.callbacks.History at 0x7d37055beb90>
```

```
scores = model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy: %.2f%%" % (scores[1]*100))
```

```
Test Accuracy: 19.44%
```

```
model.save('classification_model.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
```

Saving the model as JSON and YMAL format

```
import json
import yaml
```

```
from tensorflow.keras.models import load_model

model_json = model.to_json()
with open('classification_model.json', 'w') as json_file:
    json_file.write(model_json)

from keras.models import model_from_json
with open("classification_model.json", "r") as json_file:
    loaded_model_json = json_file.read()
loaded_model = model_from_json(loaded_model_json)

loaded_model = load_model("/content/classification_model.h5")

loaded_model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
scores_loaded = loaded_model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy (Loaded Model): %.2f%%" % (scores_loaded[1] * 100))

```

```
Test Accuracy (Loaded Model): 19.44%
```

```
json_file_path = '/content/classification_model.json'
with open(json_file_path, 'r') as json_file:
    loaded_model_json = json_file.read()

json_file.close()

loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights("/content/classification_model.h5")

loaded_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
scores_loaded = loaded_model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy (Loaded Model): %.2f%%" % (scores_loaded[1] * 100))
```

```
WARNING:tensorflow:5 out of the last 321 calls to <function Model.make_test_function.<locals>.test_function at 0x7d3706de4160> triggered
Test Accuracy (Loaded Model): 19.44%
```

Checkpoint Neural Network Model Improvements

```
from keras.callbacks import ModelCheckpoint
filepath = "weights-improvement-{epoch:02d}-{val_accuracy:.2f}.keras"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only = True)
callbacks_list = [checkpoint]

history = model.fit(X, y, validation_split = 0.33, epochs = 50, batch_size = 10, callbacks = callbacks_list, verbose = 0)
```

```
Epoch 31: val_accuracy did not improve from 0.00000

Epoch 32: val_accuracy did not improve from 0.00000

Epoch 33: val_accuracy did not improve from 0.00000

Epoch 34: val_accuracy did not improve from 0.00000

Epoch 35: val_accuracy did not improve from 0.00000

Epoch 36: val_accuracy did not improve from 0.00000

Epoch 37: val_accuracy did not improve from 0.00000

Epoch 38: val_accuracy did not improve from 0.00000

Epoch 39: val_accuracy did not improve from 0.00000

Epoch 40: val_accuracy did not improve from 0.00000

Epoch 41: val_accuracy did not improve from 0.00000

Epoch 42: val_accuracy did not improve from 0.00000

Epoch 43: val_accuracy did not improve from 0.00000

Epoch 44: val_accuracy did not improve from 0.00000

Epoch 45: val_accuracy did not improve from 0.00000

Epoch 46: val_accuracy did not improve from 0.00000

Epoch 47: val_accuracy did not improve from 0.00000

Epoch 48: val_accuracy did not improve from 0.00000

Epoch 49: val_accuracy did not improve from 0.00000

Epoch 50: val_accuracy did not improve from 0.00000
```

Checkpoint Best Neural Network Model only

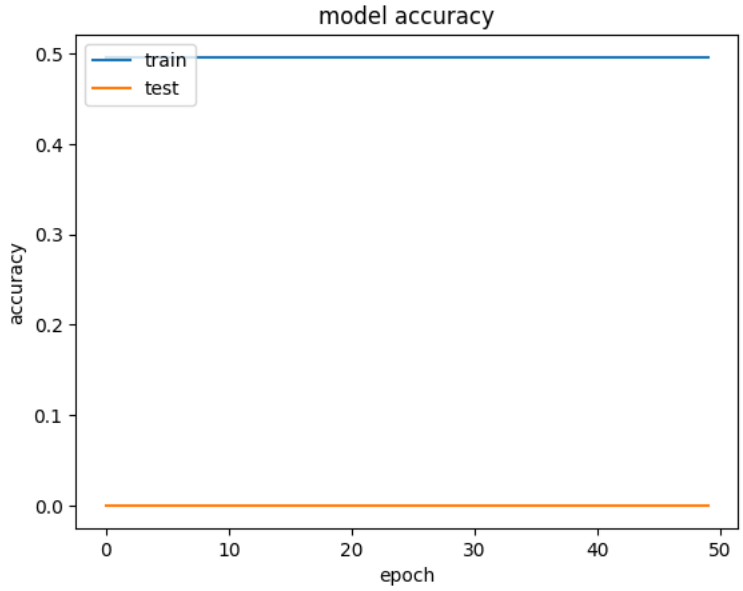
```
print("Created model and loaded weights from file")
scores = model.evaluate(X, y, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

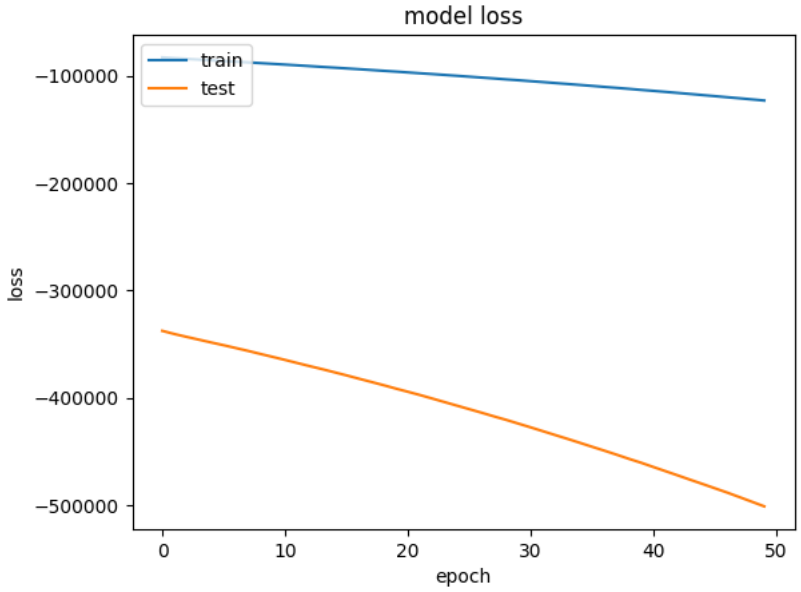
```
Created model and loaded weights from file
accuracy: 33.15%
```

Visualize Model Training History in Keras

```
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```





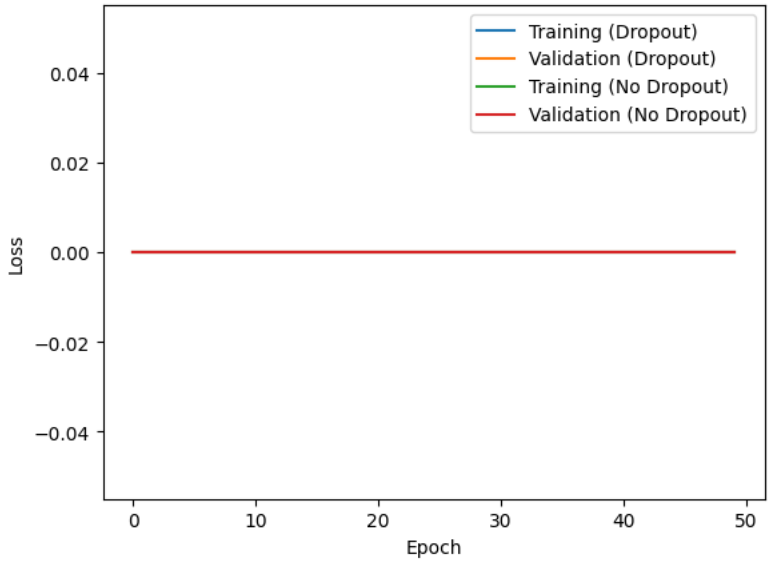
Show the application of Dropout Regularization

```
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split

model = keras.Sequential([
    Dense(64, input_shape=input_shape, activation='relu'),
    keras.layers.Dropout(0.5),
    Dense(10, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=50, validation_data=(X_val, y_val))
```

```
5/5 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 26/50
5/5 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 27/50
5/5 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 28/50
5/5 [==============================] - 0s 22ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 29/50
5/5 [==============================] - 0s 21ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 30/50
5/5 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 31/50
5/5 [==============================] - 0s 11ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 32/50
5/5 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 33/50
5/5 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 34/50
5/5 [==============================] - 0s 51ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 35/50
5/5 [==============================] - 0s 28ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 36/50
5/5 [==============================] - 0s 17ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 37/50
5/5 [==============================] - 0s 19ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 38/50
5/5 [==============================] - 0s 22ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 39/50
5/5 [==============================] - 0s 36ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 40/50
5/5 [==============================] - 0s 22ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 41/50
5/5 [==============================] - 0s 22ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 42/50
5/5 [==============================] - 0s 24ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 43/50
5/5 [==============================] - 0s 21ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 44/50
5/5 [==============================] - 0s 33ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 45/50
5/5 [==============================] - 0s 39ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 46/50
5/5 [==============================] - 0s 22ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 47/50
5/5 [==============================] - 0s 32ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 48/50
5/5 [==============================] - 0s 33ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 49/50
5/5 [==============================] - 0s 25ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
Epoch 50/50
5/5 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accuracy: 0.3169 - val_loss: 0.0000e+00 - val_accuracy: 0.388
```

```
plt.plot(history.history['loss'], label='Training (Dropout)')
plt.plot(history.history['val_loss'], label='Validation (Dropout)')
plt.plot(history.history['loss'], label='Training (No Dropout)')
plt.plot(history.history['val_loss'], label='Validation (No Dropout)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Show the application of Dropout on the visible layer

Show the application of Dropout on the hidden layer

```
model = keras.Sequential([
    #Visible Layer
    keras.layers.Input(shape=(input_shape)),
    keras.layers.Dropout(0.2),

    #Hidden Layer
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dropout(0.3),  #
    keras.layers.Dense(units=16, activation='relu'),
])
```

Show the application of a time-based learning rate schedule

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
from tensorflow.keras.optimizers import SGD

dropout_rate1 = 0.5
dropout_rate2 = 0.3

encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)
X_train, X_val, y_train, y_val = train_test_split(X, y_encoded, test_size=0.33, random_state=42)

model = Sequential()
model.add(Input(shape=(X.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dropout(dropout_rate1))  # Apply dropout with defined rate
model.add(Dense(32, activation='relu'))
model.add(Dropout(dropout_rate2))  # Apply dropout with defined rate
model.add(Dense(1, activation='sigmoid'))

learning_rate = 0.0001
momentum = 0.8
sgd = SGD(learning_rate=learning_rate, momentum=momentum)
model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
epochs = 150
batch_size = 28
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=epochs, batch_size=batch_size)

```

```
Epoch 148/150
5/5 [==============================] - 0s 15ms/step - loss: 0.2220 - accuracy: 0.3950 - val_loss: -2.9264 - val_accuracy: 0.4068
Epoch 149/150
5/5 [==============================] - 0s 12ms/step - loss: -1.8861 - accuracy: 0.3950 - val_loss: -2.9284 - val_accuracy: 0.4068
Epoch 150/150
5/5 [==============================] - 0s 13ms/step - loss: -2.8177 - accuracy: 0.3950 - val_loss: -3.2164 - val_accuracy: 0.4068
```

Show the application of a drop-based learning rate schedule

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
from tensorflow.keras.optimizers import SGD

# Define the learning rate schedule function (as explained before)
def drop_based_learning_rate(epoch, initial_learning_rate=0.1, drop_rate=0.1, drop_interval=5):
    if epoch % drop_interval == 0 and epoch > 0:
        return initial_learning_rate * (1 - drop_rate)
    else:
        return initial_learning_rate

encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)


X_train, X_val, y_train, y_val = train_test_split(X, y_encoded, test_size=0.33, random_state=42)

# Define model architecture
model = Sequential()
model.add(Input(shape=(X.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))  #=


initial_learning_rate = 0.1
drop_rate = 0.1
drop_interval = 5


optimizer = SGD(learning_rate=drop_based_learning_rate(epoch=0, initial_learning_rate=initial_learning_rate, drop_rate=drop_rate, drop_in

model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])


epochs = 150
batch_size = 28


history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=epochs, batch_size=batch_size)
```

```
5/5 [==============================] - 0s 17ms/step - loss: 0.2003 - accuracy: 0.3950 - val_loss: 0.3032 - val_accuracy: 0.4068
Epoch 117/150
5/5 [==============================] - 0s 12ms/step - loss: 0.2003 - accuracy: 0.3950 - val_loss: 0.3037 - val_accuracy: 0.4068
Epoch 118/150
5/5 [==============================] - 0s 12ms/step - loss: 0.2002 - accuracy: 0.3950 - val_loss: 0.3038 - val_accuracy: 0.4068
Epoch 119/150
5/5 [==============================] - 0s 12ms/step - loss: 0.2010 - accuracy: 0.3950 - val_loss: 0.3030 - val_accuracy: 0.4068
Epoch 120/150
5/5 [==============================] - 0s 16ms/step - loss: 0.2005 - accuracy: 0.3950 - val_loss: 0.3023 - val_accuracy: 0.4068
Epoch 121/150
5/5 [==============================] - 0s 12ms/step - loss: 0.2002 - accuracy: 0.3950 - val_loss: 0.3028 - val_accuracy: 0.4068
Epoch 122/150
```