Time Series Analisys on Geoespatial Data with Python

Author: João Otavio Nascimento Firigato

email: joaootavionf007@gmail.com

LinkedIn: https://www.linkedin.com/in/jo%C3%A3o-otavio-firigato-4876b3aa/

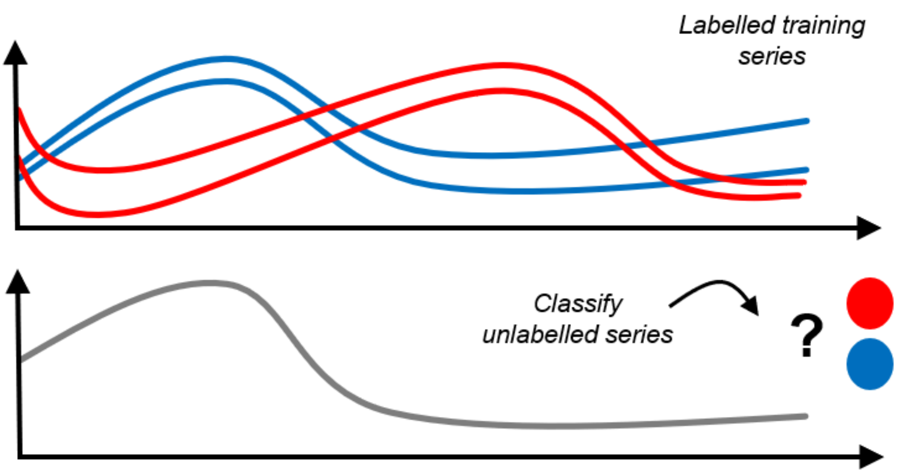
First instructions:

✅ Access the link to join our private WhatsApp community for students: https://chat.whatsapp.com/EPn27ZgR07lF3e1vnj8FiI

❗ It is important to access the Whatsapp Group to get the Colab Notebooks, as the PDF files are protected from text copying.

Chapter 14 - Time Series Classification

The Time Series Classification (TSC) task involves training a model from a collection of time series (real-valued, ordered, data) to predict a target variable. For example, we might want to build a model that can predict whether a patient is sick based on an ECG reading, or predict whether a device will fail based on some sensor reading.



We now give a formal definition of a Time Series Classification problem. Suppose we have a set of objects with the same structure (e.g., real values, vectors or matrices with the same size, etc.) and a fixed set of different classes. We define a dataset as a collection of (object, class) pairs, which means that each object is associated with a certain class. Given a dataset, a Classification problem is to build a model that associates to a new object, with the same structure as the others, the probability of belonging to the possible classes, according to the characteristics of the objects associated with each class.

A univariate time series is an ordered set of real values, while an M-dimensional multivariate time series consists of M different univariate time series with the same length. A Time Series Classification problem is a Classification problem in which the objects of the dataset are univariate or multivariate time series.

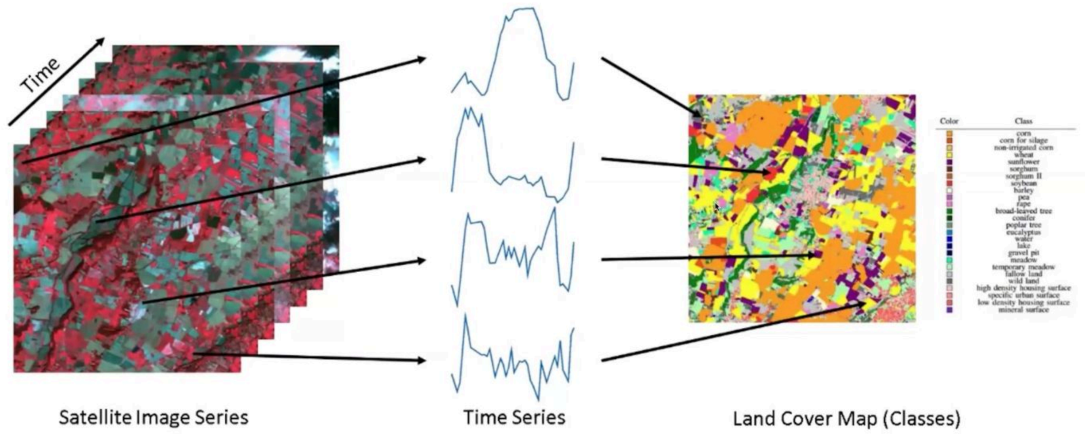


image: F Petitjean, Monash Univ

Let's install and then import the necessary libraries:

In [ ]:
```
!pip install rasterio
```

```
Collecting rasterio
  Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl.metadata (9.1 kB)
Collecting affine (from rasterio)
  Downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: attrs in /usr/local/lib/python3.11/dist-packages
(from rasterio) (25.3.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages
(from rasterio) (2025.4.26)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.11/dist-packa
ges (from rasterio) (8.2.1)
Collecting cligj>=0.5 (from rasterio)
  Downloading cligj-0.7.2-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.11/dist-pack
ages (from rasterio) (2.0.2)
Collecting click-plugins (from rasterio)
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.11/dist-packag
es (from rasterio) (3.2.3)
Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (22.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22.2/22.2 MB 35.9 MB/s eta 0:00:00
Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Downloading affine-2.4.0-py3-none-any.whl (15 kB)
Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Installing collected packages: cligj, click-plugins, affine, rasterio
Successfully installed affine-2.4.0 click-plugins-1.1.1 cligj-0.7.2 rasterio-1.4.
3
```

```
In [ ]: import ee
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import geemap
import geopandas as gpd
import warnings
import os
import rasterio
from prophet import Prophet
warnings.filterwarnings('ignore')
```

We authenticate and initialize the GEE:

```
In [ ]: ee.Authenticate()
ee.Initialize(project='my-project-1527255156007')
```

Let's create a set of points for different crops:

```
In [ ]: Class_Wine = ee.Geometry.MultiPoint([[-119.82298223898039, 36.61282934632966],
         [-119.80341284200773, 36.6125537661724],
         [-119.87791387960539, 36.61448280659136],
         [-119.80341284200773, 36.58099332673207],
         [-119.769080566171, 36.60084069916139],
         [-119.7416147463046, 36.60993570555981],
         [-119.72307531759367, 36.611038057703475],
         [-119.72410528585539, 36.629224592987626],
         [-119.8569711916171, 36.586782670945986],
         [-119.86057608053312, 36.595328052162635],
         [-119.87207739278898, 36.60028945218614],
         [-119.88821356222257, 36.608971134508366],
         [-119.88787023946867, 36.596292793820254],
         [-119.83843176290617, 36.60759315493638],
         [-119.82195227071867, 36.60938452357907],
         [-119.80598776266203, 36.60966011506182],
         [-119.79963629171476, 36.60938452357907],
         [-119.79311315939054, 36.61062467749615],
         [-119.74264471456632, 36.60800655139306],
         [-119.66162054464445, 36.62550496889934],
         [-119.65492575094328, 36.61627401519815],
         [-119.64891760274992, 36.62206071246241],
         [-119.6394762270175, 36.627433685279705],
         [-119.61681692525968, 36.62454059261014],
         [-119.64055796058997, 36.65397596987176],
         [-119.65223093422279, 36.67600742053306],
         [-119.65961237343177, 36.67229005555964],
         [-119.66733713539466, 36.656730246079036],
         [-119.6884514847599, 36.64681439063314],
         [-119.68930979164466, 36.64392202561094],
         [-119.69325800331458, 36.64667666142838],
         [-119.69411631019935, 36.6435088217412],
         [-119.69548960121497, 36.63951440338949],
         [-119.76824834675485, 36.559245235000425],
         [-119.76807668537789, 36.553453826148214],
         [-119.76121023029977, 36.552074855302834],
         [-119.7577770027607, 36.558831577334466],
         [-119.80378225178414, 36.550695859852084],
         [-119.80601384968453, 36.549041032832655],
```

```
          [-119.81305196613961, 36.55579802011514]])
Class_Almonds = ee.Geometry.MultiPoint([[-119.83021810383492, 36.557039035190684
          [-119.83742788166695, 36.556487475395514],
          [-119.83742788166695, 36.552074855302834],
          [-119.83742788166695, 36.55110956107087],
          [-119.85184743733102, 36.56489833447415],
          [-119.85871389240914, 36.57110247966871],
          [-119.86867025227242, 36.570551020275815],
          [-119.87210347981149, 36.572481110923626],
          [-119.87553670735055, 36.549178936437535],
          [-119.86918523640328, 36.5517990581811],
          [-119.81682851643258, 36.56310593308617],
          [-119.82094838947945, 36.5788224977055],
          [-119.7878177437275, 36.579511747497385],
          [-119.77906301350289, 36.57909819836062],
          [-119.77528646320992, 36.577719685240126],
          [-119.75829198689156, 36.57468686974122],
          [-119.7299678596943, 36.56379532315709],
          [-119.8673774617767, 36.638741811461195],
          [-119.88626021324154, 36.63819083542564],
          [-119.89552992759701, 36.63943052596441],
          [-119.89261168418881, 36.64549094809744],
          [-119.86497420249935, 36.65141317266768],
          [-119.85810774742123, 36.653065805219775],
          [-119.8563911336517, 36.666285588686094],
          [-119.84162825523373, 36.64466455497453],
          [-119.82034224449154, 36.658023490057126],
          [-119.83373183189389, 36.66559711433901],
          [-119.84214323936459, 36.62413961520817],
          [-119.73407650471735, 36.62524176428721],
          [-119.89775562764216, 36.5943756336181],
          [-119.89329243184137, 36.591481300549994],
          [-119.87406635762262, 36.58734635071557],
          [-119.8697748231988, 36.581832739667945],
          [-119.86994648457575, 36.58045427537243],
          [-119.8917474794488, 36.631578815651686],
          [-119.85844517231989, 36.62868587858294],
          [-119.86067677022028, 36.62951244311265],
          [-119.85638523579645, 36.628548116966115],
          [-119.84179401875544, 36.622624134492355],
          [-119.83750248433161, 36.62165972216027]])
Class_Cherries = ee.Geometry.MultiPoint([[-119.89157581807184, 36.5564650569373
          [-119.89003086567926, 36.55563770962265],
          [-119.89878559590387, 36.54888070832333],
          [-119.89672565938044, 36.54915651585914],
          [-119.85140705586481, 36.59354869239014],
          [-119.84797382832575, 36.595753849304934],
          [-119.84763050557184, 36.59299739331493],
          [-119.85123539448786, 36.59272174230038],
          [-119.83329678059626, 36.582728728261316],
          [-119.8302068758111, 36.582659806431],
          [-119.78385830403376, 36.570666471155896],
          [-119.78214169026423, 36.571080065451426],
          [-119.77973843098688, 36.57149365753204],
          [-119.77870846272516, 36.57052860589851],
          [-119.76111317158747, 36.569977142407964],
          [-119.75999737263727, 36.571080065451426],
          [-119.7584524202447, 36.56956354220599],
          [-119.72187713136236, 36.59182571503345],
          [-119.72058967103521, 36.59396200944671],
```

```
          [-119.56841394348035, 36.5560169396553],
          [-119.56746980590711, 36.55546537256447],
          [-119.58266183776746, 36.595512691402895],
          [-119.58120271606336, 36.595512691402895],
          [-119.58266183776746, 36.593789912563814],
          [-119.58137437744031, 36.59358317651843],
          [-119.54197809142957, 36.587932176813325],
          [-119.54051896972547, 36.58779434238304],
          [-119.78892756676589, 36.661116232973875],
          [-119.78824092125808, 36.661116232973875],
          [-119.8398251650325, 36.62558053204331],
          [-119.83785105919753, 36.62558053204331],
          [-119.83750773644363, 36.62440950327035],
          [-119.83888102745925, 36.624202849287144],
          [-119.83973933434402, 36.624202849287144],
          [-119.83699275231277, 36.6240650796572],
          [-119.79922724938308, 36.62599383206842],
          [-119.79854060387527, 36.62613159825094],
          [-119.79931308007156, 36.62468504105281],
          [-119.79854060387527, 36.62482280957467],
          [-119.79888392662917, 36.62385842475059]])
Class_Citrus = ee.Geometry.MultiPoint([[-119.69358809714616, 36.6652603648613],
          [-119.68929656272233, 36.66594884222072],
          [-119.68620665793718, 36.66691270017729],
          [-119.68534835105241, 36.66484727548946],
          [-119.67127211814226, 36.6729709594935],
          [-119.66680892234147, 36.67200717741508],
          [-119.66595061545671, 36.67710131738032],
          [-119.67831023459733, 36.67613758703974],
          [-119.68860991721452, 36.675311522853484],
          [-119.68431838279069, 36.6757245560553],
          [-119.69393141990007, 36.67283327707854],
          [-119.69650634055436, 36.67710131738032],
          [-119.69375975852311, 36.68233278571631],
          [-119.69255812888444, 36.680818449891404],
          [-119.69341643576921, 36.691005225635045],
          [-119.69530471091569, 36.69788741834709],
          [-119.66165908103288, 36.696235748280706],
          [-119.65959914450944, 36.694721686213605],
          [-119.657539207986, 36.69719922679858],
          [-119.65444930320085, 36.69788741834709],
          [-119.64964278464616, 36.695547541946574],
          [-119.64998610740007, 36.69403346632616],
          [-119.62252028708757, 36.658650668864],
          [-119.61943038230241, 36.65947691190202],
          [-119.61977370505632, 36.657686707444974],
          [-119.61668380027116, 36.65754899768551],
          [-119.61634047751726, 36.65396845747989],
          [-119.60466750388444, 36.66498497219307],
          [-119.59780104880632, 36.66484727548946],
          [-119.60037596946061, 36.66828961917539],
          [-119.60037596946061, 36.6655357565441],
          [-119.59574111228288, 36.66815192838417],
          [-119.60501082663835, 36.62352714349036],
          [-119.60707076316179, 36.62214942403314],
          [-119.58921797995866, 36.63069088772247],
          [-119.58372481589616, 36.63041537141072],
          [-119.57960494284929, 36.63013985411392],
          [-119.57067855124772, 36.628486729647825],
          [-119.56140883689226, 36.62821120545593],
```

```
          [-119.52707656150163, 36.602307533766435]])
Class_Alfalfa = ee.Geometry.MultiPoint([[-119.49358547717651, 36.70314585101399]
          [-119.49264133960327, 36.702044816902315],
          [-119.4947012761267, 36.70004915237429],
          [-119.49573124438842, 36.70225126199967],
          [-119.49598873645385, 36.69812225470046],
          [-119.49779118091186, 36.69626412905685],
          [-119.4987353184851, 36.694199492334796],
          [-119.49804867297729, 36.69296068368799],
          [-119.49573124438842, 36.69268539016698],
          [-119.49452961474975, 36.697846979661264],
          [-119.49246967822631, 36.69977388423524],
          [-119.49135387927612, 36.69819107330622],
          [-119.48251331836303, 36.69970506704646],
          [-119.48216999560913, 36.70465974716411],
          [-119.48311413318237, 36.69598884736411],
          [-119.48165501147827, 36.69887925597554],
          [-119.50783337146362, 36.69413067015585],
          [-119.50783337146362, 36.69585120614808],
          [-119.50869167834838, 36.693442444977975],
          [-119.58702744324405, 36.64591439129269],
          [-119.5865124591132, 36.64426160526685],
          [-119.585396660163, 36.64563892941801],
          [-119.58548249085148, 36.64371066871017],
          [-119.58659828980167, 36.64343519895408],
          [-119.5894307025214, 36.649839615958825],
          [-119.58737076599796, 36.649908477763766],
          [-119.58599747498234, 36.649908477763766],
          [-119.58857239563663, 36.64970189216424],
          [-119.58471001465519, 36.64481253788284],
          [-119.58479584534366, 36.64343519895408],
          [-119.4993900760629, 36.69315967458978],
          [-119.49814553107998, 36.69205849744163],
          [-119.49621434058926, 36.69216173346922],
          [-119.49494873578573, 36.696488440446814],
          [-119.4955924659493, 36.70154655132576],
          [-119.49400459821248, 36.701856220771276],
          [-119.48207413251424, 36.70130569645003],
          [-119.48138748700643, 36.700411086018306],
          [-119.48945557172323, 36.69321941623659],
          [-119.48906933362508, 36.69235912360564]])
```

We can use geemap to visualize these points:

In [ ]:

```python
import geemap
import ee

Map = geemap.Map(center=[36.6, -119.8], zoom=10)

# Add the USDA Cropland Data Layer
usda_crop = ee.ImageCollection('USDA/NASS/CDL') \
    .filterDate('2021-01-01', '2021-12-31') \
    .mosaic().select('cropland')

vis_params = {
    'min': 1,
    'max': 254,
    'palette': ['006400', 'ffbb22', 'ffff4c', 'f096ff', 'fa0000', 'b4b4b4',
                'f0f0f0', '0064c8', '0096a0', '00cf75', 'fae6a0', '008000',
```

```
                '00cf00', 'a0d000', 'ffff00', 'a0a0a0', '000080', '800080',
                'a00000', 'ffa0a0', 'a0ffa0', '00ffff', '008080', '0000ff',
                '0000bc', '8080ff', 'c8c8c8', '800000']
}
Map.addLayer(usda_crop, vis_params, 'USDA Cropland Data Layer')


# Function to plot points by class
def plot_points_by_class(geometry, class_name):
    style = {'color': 'black', 'fillColor': '', 'radius': 3}
    if class_name == 'Wine':
      style['fillColor'] = 'blue'
    elif class_name == 'Almonds':
      style['fillColor'] = 'red'
    elif class_name == 'Cherries':
      style['fillColor'] = 'green'
    elif class_name == 'Citrus':
      style['fillColor'] = 'yellow'
    elif class_name == 'Alfafa':
      style['fillColor'] = 'purple'

    Map.addLayer(geometry, style, class_name)


# Plot the points for each class
plot_points_by_class(Class_Wine, 'Wine')
plot_points_by_class(Class_Almonds, 'Almonds')
plot_points_by_class(Class_Cherries, 'Cherries')
plot_points_by_class(Class_Citrus, 'Citrus')
plot_points_by_class(Class_Alfalfa, 'Alfafa')

# Display the map
Map
```

Let's join all the points into a single FeatureCollection:

In [ ]:

```
feature_collection = ee.FeatureCollection([
    ee.Feature(Class_Wine, {'Class': 'Wine'}),
    ee.Feature(Class_Almonds, {'Class': 'Almonds'}),
    ee.Feature(Class_Cherries, {'Class': 'Cherries'}),
    ee.Feature(Class_Citrus, {'Class': 'Citrus'}),
    ee.Feature(Class_Alfalfa, {'Class': 'Alfalfa'})
])
```

In [ ]:

```
def multipoint_to_points(feature):
    multipoint = feature.geometry()
    class_name = feature.get('Class')

    # Get coordinates of the MultiPoint
    coords = multipoint.coordinates()

    # Create a list of Features from coordinates
    features = coords.map(lambda coord: ee.Feature(ee.Geometry.Point(coord)).set

    # Return a FeatureCollection of individual points
    return ee.FeatureCollection(features)

point_features = feature_collection.map(multipoint_to_points).flatten()
```

Now let's extract the NDVI time series for each point:

```
In [ ]: def addNDVI(image):
    ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI')
    return image.addBands(ndvi)

def extract_ndvi_timeseries(feature):
    point = feature.geometry()
    class_id = feature.get('Class').getInfo()

    s2_sr_harmonized = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
        .filterBounds(point) \
        .filterDate('2023-01-01', '2023-12-31') \
        .map(addNDVI)

    months = ee.List.sequence(1, 12)
    years = ee.List.sequence(2023, 2023)

    def monthly(collection):
      img_coll = ee.ImageCollection([])
      for y in years.getInfo():
        for m in months.getInfo():
          filtered = collection.filter(ee.Filter.calendarRange(y, y, 'year')).fi
          filtered = filtered.median()
          img_coll = img_coll.merge(filtered.set('year', y).set('month', m).set(
          img_coll = img_coll.select('NDVI')
      return img_coll

    monthly_s2_collection = monthly(s2_sr_harmonized)

    ndvi_timeseries = monthly_s2_collection.select('NDVI') \
        .getRegion(point, 30) \
        .getInfo()

    class_list.append(class_id)
    df = pd.DataFrame(ndvi_timeseries[1:], columns=ndvi_timeseries[0])
    df['date'] = df['time']
    df = df.set_index('date')
    df = df[['NDVI']]
    return df

dfs = []
class_list = []


for feature in point_features.getInfo()['features']:
  dfs.append(extract_ndvi_timeseries(ee.Feature(feature)))
```

And convert to a DataFrame:

```
In [ ]: ndvi_df = pd.concat(dfs, axis=1)
```

```
In [ ]: ndvi_df = ndvi_df.T.reset_index(drop=True)
```

```
In [ ]: ndvi_df['classe'] = class_list
```

In [ ]:

```
ndvi_df
```

Out[ ]:

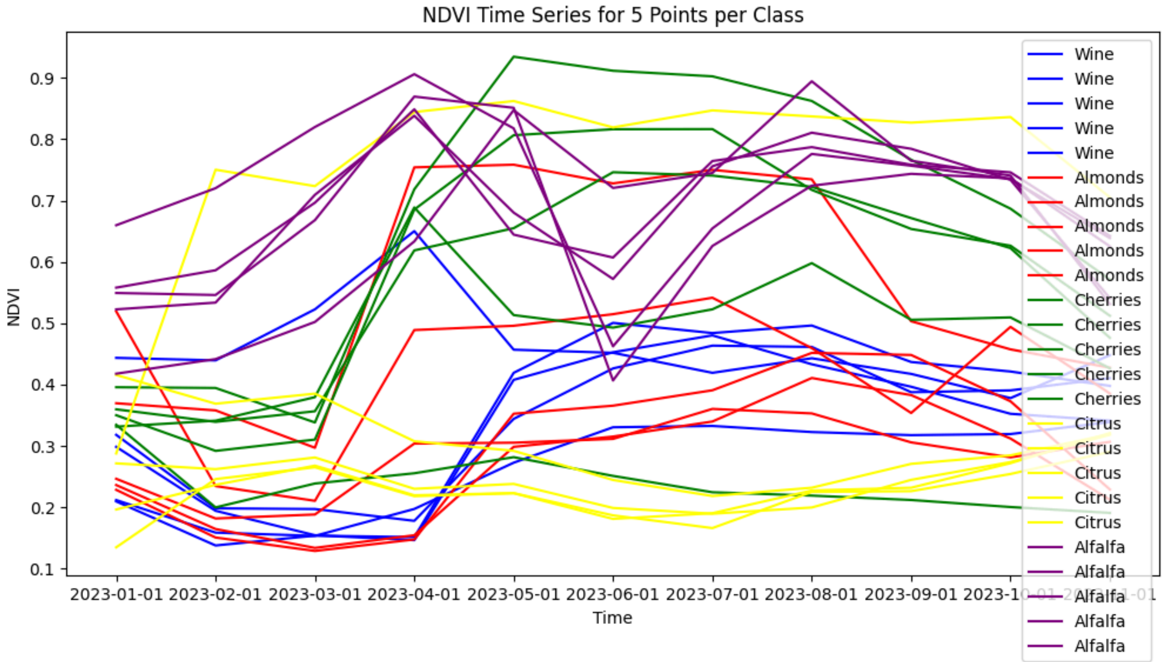
date	2023-01-01	2023-02-01	2023-03-01	2023-04-01	2023-05-01	2023-06-01	2023-07-01	2023-08-01	
0	0.318135	0.198323	0.197208	0.177967	0.344116	0.426382	0.463539	0.461427	0.
1	0.211793	0.158740	0.153425	0.197292	0.272719	0.330481	0.332641	0.322678	0.
2	0.298569	0.194311	0.154483	0.146479	0.407806	0.452722	0.480268	0.433181	0.
3	0.443601	0.439508	0.522739	0.650218	0.456934	0.451865	0.419228	0.442965	0.
4	0.209651	0.137768	0.153815	0.151413	0.419170	0.500637	0.484049	0.496346	0.
...	...	...	...	...	...	...	...	...	
195	0.589531	0.629662	0.764611	0.899719	0.719161	0.555190	0.753784	0.783859	0.
196	0.836447	0.714871	0.624605	0.568544	0.624511	0.570010	0.723826	0.587678	0.
197	0.843162	0.726048	0.683235	0.631695	0.588760	0.547373	0.651241	0.605400	0.
198	0.429343	0.432568	0.439226	0.612217	0.815699	0.635619	0.799945	0.834310	0.
199	0.454886	0.382710	0.468032	0.629751	0.768730	0.546013	0.770378	0.833859	0.

200 rows × 13 columns

Let's visualize some points of each class:

In [ ]:

```
import matplotlib.pyplot as plt

class_colors = {
    'Wine': 'blue',
    'Almonds': 'red',
    'Cherries': 'green',
    'Citrus': 'yellow',
    'Alfalfa': 'purple'
}

# Plot time series for 5 points of each class
plt.figure(figsize=(12, 6))
for class_name in ndvi_df['classe'].unique():
    class_df = ndvi_df[ndvi_df['classe'] == class_name].head(5)
    class_df = class_df.drop(columns='classe')
     # Select first 5 points
    for index, row in class_df.iterrows():
        # The change is on this line. Use row.index[:-1] to select the same colu
        plt.plot(row.index[:-1], row.values[:-1], label=class_name, color=class_

plt.xlabel('Time')
plt.ylabel('NDVI')
plt.title('NDVI Time Series for 5 Points per Class')
plt.legend()
plt.show()
```



Algorithms used

Many different categories of algorithms have been investigated to deal with time series classification. To compare time series, specific metrics and kernels have been proposed. Some algorithms consist of extracting features from time series that can be used as input to a standard machine learning classifier, while other algorithms work on raw time series. Bag-of-words models that rely on discretized time series are popular, with many algorithms being developed. Transforming time series into images can also be studied.

Standard classifiers

We can use SkLearn's default classifiers, such as RandomForest:

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

We will use the dates as features:

In [ ]:
```
X = ndvi_df.drop('classe', axis=1)  # Features
y = ndvi_df['classe']  # Target variable

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

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (160, 12)
X_test shape: (40, 12)
y_train shape: (160,)
y_test shape: (40,)
```

We can then obtain the accuracy on the test data:

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

accuracy_score(y_test, y_pred)
```

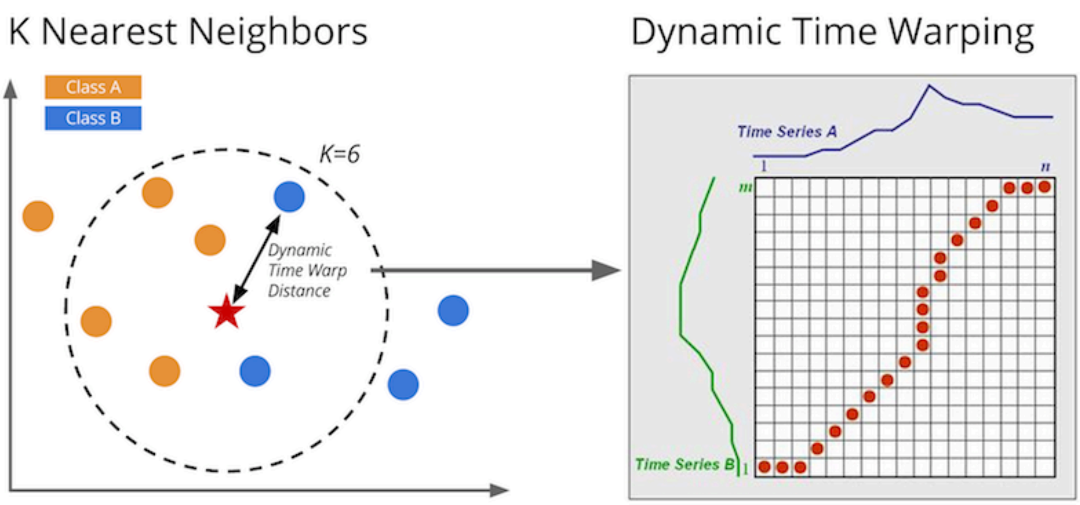
Out[ ]:
```
0.7
```

KNN with DTW

Nearest neighbor methods are one of the most intuitive algorithms for supervised learning: the prediction for a new sample is based on the target value of similar samples. A key element of nearest neighbor algorithms is the metric, i.e. the mathematical function that defines the (dis)similarity between any pair of samples.

Although Euclidean distance is the most common metric, it is not suitable for comparing time series for two main reasons: (i) it is defined only for two vectors of the same length, while time series in a given dataset are usually of different lengths, and (ii) it compares the values of both time series at each time point independently, while the values of the time series are correlated. Considering the minimum of the Euclidean distances between the smaller time series and the same-length subsequences of the larger time series may not be optimal. A concrete example from automatic speech recognition is considering a given sentence, with one being pronounced more slowly than the other. Not only will the corresponding time series have different lengths, but the Euclidean distance between the smaller time series and any same-length subsequence with consecutive time points of the larger time series will not be small, even if the same sentence was uttered and a relevant metric should yield a small value. Another concrete example could consist of sequences of geolocations, with several people walking on the same route, but at different speeds.

Dynamic temporal warping (DTW) is a metric for time series that addresses both limitations of Euclidean distance [9]. The nearest neighbor classifier with the DTW metric is often considered the baseline algorithm for time series classification.



Let's install tslearn to use time series specific classifiers:

In [ ]:
```
!pip install tslearn
```

```
Collecting tslearn
  Downloading tslearn-0.6.3-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from tslearn) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages
(from tslearn) (1.15.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-pac
kages (from tslearn) (1.6.1)
Requirement already satisfied: numba in /usr/local/lib/python3.11/dist-packages
(from tslearn) (0.60.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages
(from tslearn) (1.5.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/pytho
n3.11/dist-packages (from numba->tslearn) (0.43.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/
dist-packages (from scikit-learn->tslearn) (3.6.0)
Downloading tslearn-0.6.3-py3-none-any.whl (374 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 374.4/374.4 kB 7.4 MB/s eta 0:00:00
Installing collected packages: tslearn
Successfully installed tslearn-0.6.3
```

In [ ]:
```
from tslearn.metrics import dtw
from sklearn.neighbors import KNeighborsClassifier
from tslearn.utils import to_time_series_dataset
from tslearn.neighbors import KNeighborsTimeSeriesClassifier
```

First let's use the default KNN:

In [ ]:
```
clf = KNeighborsClassifier(n_neighbors=3, metric=dtw)
```

In [ ]:
```
clf.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

accuracy_score(y_test, y_pred)
```

Out[ ]: 0.675

To use KNN for time series, we need to convert the original dataframe to be compatible with the tslearn library:

In [ ]:
```
X_tslearn = to_time_series_dataset(X)
```

We divide it into training and testing:

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X_tslearn, y, test_size=0.2,
```

We apply KNN:

In [ ]:
```
knn = KNeighborsTimeSeriesClassifier(n_neighbors=3,
                                     metric="dtw")
knn.fit(X_train,y_train)
```

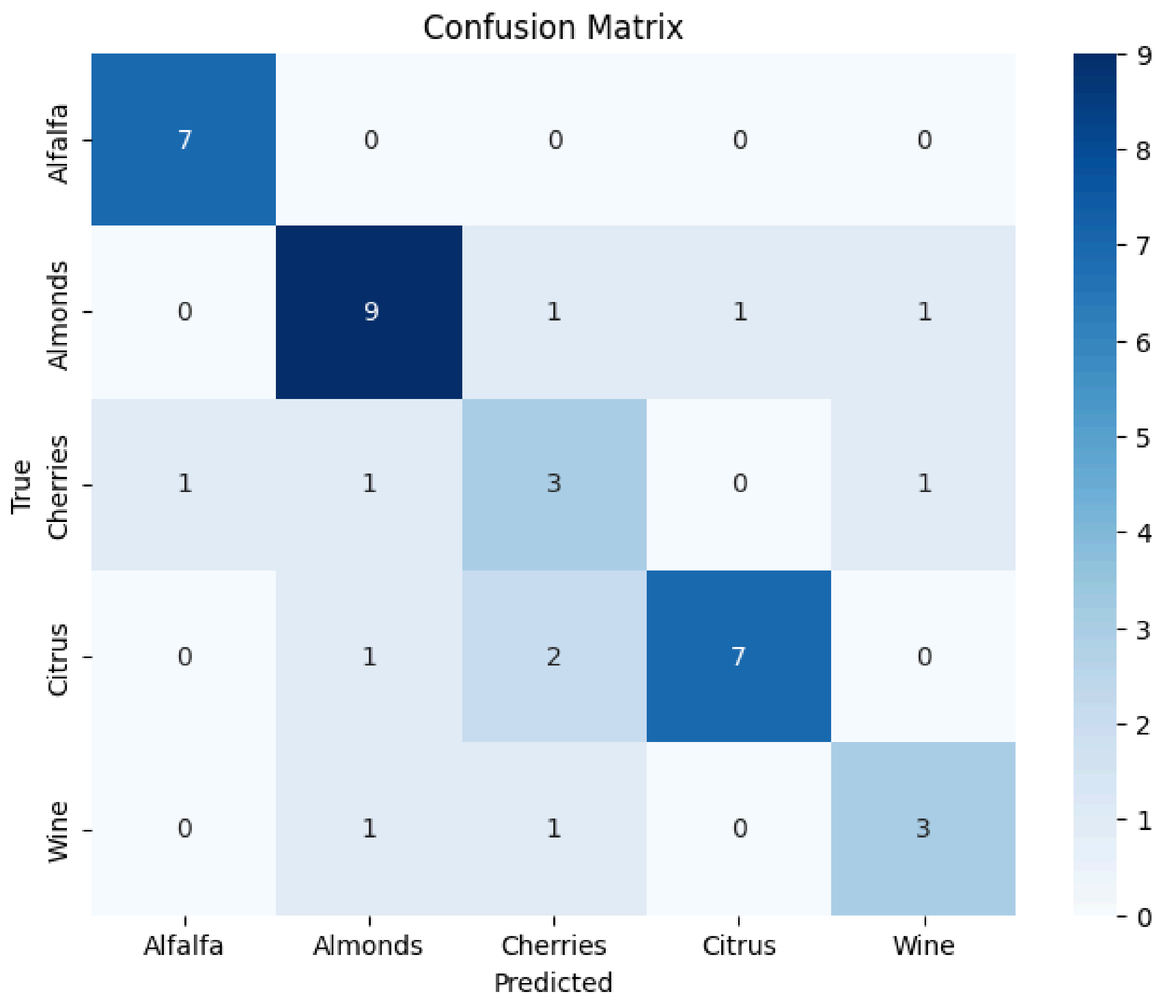
Out[ ]:
```
KNeighborsTimeSeriesClassifier
KNeighborsTimeSeriesClassifier(n_neighbors=3)
```

In [ ]:
```
y_pred = knn.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[ ]:
```
0.725
```

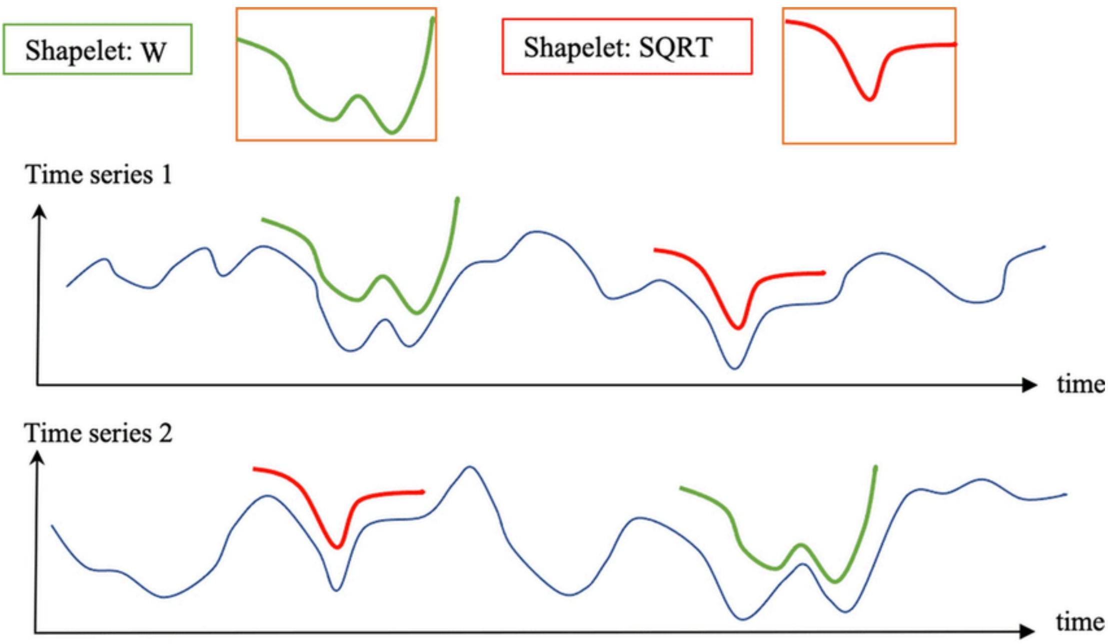
We can generate the Confusion Matrix:

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns


cm = confusion_matrix(y_test, y_pred)


plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```



Shapelets

A shapelet is defined as a subsequence of consecutive observations of a time series. In some use cases, specific shapelets may be characteristic of classes and thus useful for discriminating between them. Several algorithms rely on shapelets, either by extracting the best shapelets from the training dataset or by learning them directly.



Shapelet Extraction

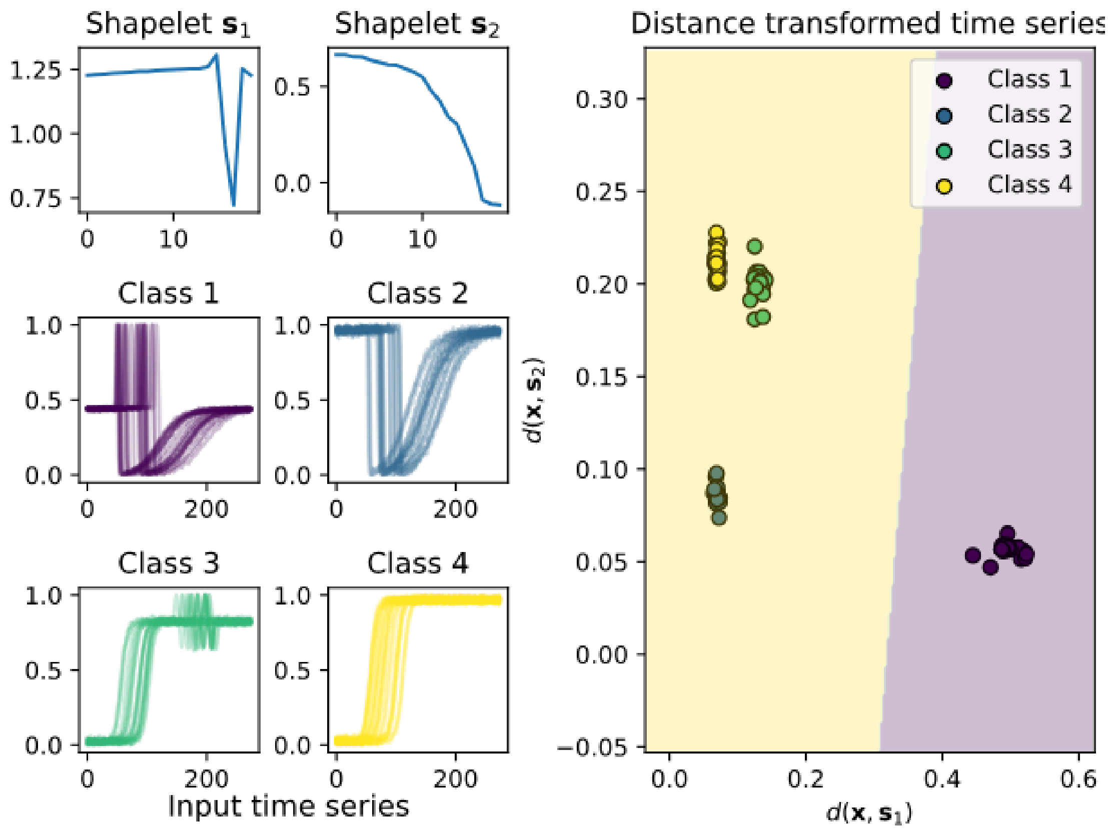
The algorithm extracts all shapelets whose length falls within a range, where range is a hyperparameter of the algorithm, and selects the k best shapelets, where k is another hyperparameter of the algorithm. This process can be thought of as univariate feature extraction, where each feature is the distance between a given shapelet and all time series in the dataset. Shapelets are classified based on the F-statistic from the analysis of variance test that compares between- and within-class variability. To extract features (i.e., shapelets) that are not highly correlated, self-similar shapelets are removed, with any pair of shapelets being considered self-similar if they are from the same time series and have any overlapping indices.

Once the k best shapelets have been identified and corresponding features generated, any standard machine learning classifier can be applied to this new dataset.

Learning shapelets

To address the limitations of the Shapelet Transform algorithm, another algorithm that relies on learning shapelets rather than extracting them was proposed.

The distance between a shapelet and a time series depends on the minimum function, which is not differentiable. Similar to the soft-DTW variant of DTW, the minimum function is derived from a soft minimum function, namely the LogSumExp function, which is differentiable. The logistic regression algorithm is used as the machine learning classifier built on top of the transformation. The image below illustrates learned shapelets and the distances between both shapelets and time series, highlighting that each shapelet is class-specific.



Since the transformation and classification functions are differentiable, the chain rule allows us to compute the gradients of the objection function with respect to shapelets and the logistic regression coefficients, respectively, thus minimizing the objective function that can be attempted to be solved by gradient descent.

Learning shapelets rather than extracting them has several advantages. First, it can lead to shapelets that are not from the dataset but are discriminative of the classes. Second, it does not require traversing the entire dataset and can therefore be faster to train, especially with stochastic variants of gradient descent.

However, learning shapelets also has drawbacks. Since both the shapelets and the logistic regression coefficients need to be learned, the objective function is not convex (one can see the analogy with some clustering algorithms, such as k-means and Gaussian mixture models, where both the parameters and the cluster memberships need to be learned). Therefore, the optimization algorithm may converge to a poor local minimum. This also leads to more hyperparameters, since the optimization process is a key component of the algorithm. Finally, since learning shapelets is embedded in the algorithm, it may not be optimal to try other classifiers besides logistic regression later, whereas the Shapelet Transform algorithm is independent of the machine learning classifier, and therefore the transformation step can be computed only once and many classifiers can be built on top of it to find the best performing classifier.

Let's install Sktime:

In [ ]:
```
!pip install sktime
```

```
Requirement already satisfied: sktime in /usr/local/lib/python3.11/dist-packages
(0.36.1)
Requirement already satisfied: joblib<1.5,>=1.2.0 in /usr/local/lib/python3.11/di
st-packages (from sktime) (1.4.2)
Requirement already satisfied: numpy<2.3,>=1.21 in /usr/local/lib/python3.11/dist
-packages (from sktime) (2.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packag
es (from sktime) (24.2)
Requirement already satisfied: pandas<2.3.0,>=1.1 in /usr/local/lib/python3.11/di
st-packages (from sktime) (2.2.2)
Requirement already satisfied: scikit-base<0.13.0,>=0.6.1 in /usr/local/lib/pytho
n3.11/dist-packages (from sktime) (0.12.2)
Requirement already satisfied: scikit-learn<1.7.0,>=0.24 in /usr/local/lib/python
3.11/dist-packages (from sktime) (1.6.1)
Requirement already satisfied: scipy<2.0.0,>=1.2 in /usr/local/lib/python3.11/dis
t-packages (from sktime) (1.14.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.1
1/dist-packages (from pandas<2.3.0,>=1.1->sktime) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pac
kages (from pandas<2.3.0,>=1.1->sktime) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-p
ackages (from pandas<2.3.0,>=1.1->sktime) (2025.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/
dist-packages (from scikit-learn<1.7.0,>=0.24->sktime) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-package
s (from python-dateutil>=2.8.2->pandas<2.3.0,>=1.1->sktime) (1.17.0)
```

In [ ]:

```
from sktime.classification.shapelet_based import ShapeletTransformClassifier
from tslearn.preprocessing import TimeSeriesScalerMinMax
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.optimizers import Adam
from tslearn.shapelets import LearningShapelets
```

We normalize the data and divide it into training and testing:

In [ ]:

```
X_tslearn = TimeSeriesScalerMinMax().fit_transform(X_tslearn)
```

In [ ]:

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

In [ ]:

```
shapelet_sizes = {12: 4}

# Define the model and fit it using the training data
shp_clf = LearningShapelets(n_shapelets_per_size=shapelet_sizes,
                            weight_regularizer=0.0001,
                            optimizer=Adam(learning_rate=0.01),
                            max_iter=500,
                            verbose=1,
                            scale=False,
                            random_state=42)
shp_clf.fit(X_train, y_train)
```

```
Epoch 1/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - categorical_accuracy: 0.2125 - categorical_
crossentropy: 1.6168 - loss: 1.6172
Epoch 2/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 117ms/step - categorical_accuracy: 0.2125 - categoric
al_crossentropy: 1.6135 - loss: 1.6139
Epoch 3/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - categorical_accuracy: 0.1875 - categoric
al_crossentropy: 1.6104 - loss: 1.6108
Epoch 4/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - categorical_accuracy: 0.1187 - categoric
al_crossentropy: 1.6074 - loss: 1.6078
Epoch 5/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 215ms/step - categorical_accuracy: 0.1562 - categoric
al_crossentropy: 1.6045 - loss: 1.6049
Epoch 6/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - categorical_accuracy: 0.1875 - categoric
al_crossentropy: 1.6018 - loss: 1.6022
Epoch 7/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 260ms/step - categorical_accuracy: 0.1937 - categoric
al_crossentropy: 1.5992 - loss: 1.5996
Epoch 8/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 117ms/step - categorical_accuracy: 0.1937 - categoric
al_crossentropy: 1.5966 - loss: 1.5970
Epoch 9/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5941 - loss: 1.5945
Epoch 10/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 187ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5917 - loss: 1.5920
Epoch 11/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5892 - loss: 1.5896
Epoch 12/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5868 - loss: 1.5872
Epoch 13/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5844 - loss: 1.5848
Epoch 14/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 226ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5820 - loss: 1.5824
Epoch 15/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 198ms/step - categorical_accuracy: 0.2000 - categoric
al_crossentropy: 1.5796 - loss: 1.5801
Epoch 16/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - categorical_accuracy: 0.2000 - categorica
l_crossentropy: 1.5772 - loss: 1.5777
Epoch 17/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step - categorical_accuracy: 0.1937 - categoric
al_crossentropy: 1.5748 - loss: 1.5752
Epoch 18/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 232ms/step - categorical_accuracy: 0.1875 - categoric
al_crossentropy: 1.5723 - loss: 1.5728
Epoch 19/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - categorical_accuracy: 0.1937 - categorica
l_crossentropy: 1.5698 - loss: 1.5703
Epoch 20/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - categorical_accuracy: 0.2000 - categorica
l_crossentropy: 1.5673 - loss: 1.5677
```

```
Epoch 21/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - categorical_accuracy: 0.2000 - categoric
al_crossentropy: 1.5647 - loss: 1.5651
Epoch 22/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step - categorical_accuracy: 0.2000 - categorica
l_crossentropy: 1.5620 - loss: 1.5625
Epoch 23/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - categorical_accuracy: 0.1937 - categoric
al_crossentropy: 1.5593 - loss: 1.5598
Epoch 24/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5566 - loss: 1.5571
Epoch 25/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 184ms/step - categorical_accuracy: 0.2125 - categoric
al_crossentropy: 1.5538 - loss: 1.5542
Epoch 26/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 225ms/step - categorical_accuracy: 0.2125 - categoric
al_crossentropy: 1.5509 - loss: 1.5514
Epoch 27/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step - categorical_accuracy: 0.2125 - categoric
al_crossentropy: 1.5480 - loss: 1.5485
Epoch 28/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 180ms/step - categorical_accuracy: 0.2125 - categoric
al_crossentropy: 1.5450 - loss: 1.5455
Epoch 29/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 262ms/step - categorical_accuracy: 0.2062 - categoric
al_crossentropy: 1.5419 - loss: 1.5424
Epoch 30/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - categorical_accuracy: 0.2125 - categorica
l_crossentropy: 1.5388 - loss: 1.5393
Epoch 31/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - categorical_accuracy: 0.2313 - categorica
l_crossentropy: 1.5356 - loss: 1.5362
Epoch 32/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - categorical_accuracy: 0.2375 - categoric
al_crossentropy: 1.5324 - loss: 1.5330
Epoch 33/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - categorical_accuracy: 0.2688 - categoric
al_crossentropy: 1.5291 - loss: 1.5297
Epoch 34/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 161ms/step - categorical_accuracy: 0.3000 - categoric
al_crossentropy: 1.5258 - loss: 1.5264
Epoch 35/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 122ms/step - categorical_accuracy: 0.3063 - categoric
al_crossentropy: 1.5224 - loss: 1.5230
Epoch 36/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - categorical_accuracy: 0.3250 - categoric
al_crossentropy: 1.5189 - loss: 1.5195
Epoch 37/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - categorical_accuracy: 0.3375 - categorica
l_crossentropy: 1.5154 - loss: 1.5160
Epoch 38/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - categorical_accuracy: 0.3375 - categoric
al_crossentropy: 1.5118 - loss: 1.5124
Epoch 39/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - categorical_accuracy: 0.3375 - categorica
l_crossentropy: 1.5081 - loss: 1.5088
Epoch 40/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - categorical_accuracy: 0.3438 - categorica
l_crossentropy: 1.5044 - loss: 1.5051
```

```
Epoch 41/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - categorical_accuracy: 0.3750 - categoric
al_crossentropy: 1.5006 - loss: 1.5013
Epoch 42/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 260ms/step - categorical_accuracy: 0.4000 - categoric
al_crossentropy: 1.4968 - loss: 1.4975
Epoch 43/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 245ms/step - categorical_accuracy: 0.4125 - categoric
al_crossentropy: 1.4928 - loss: 1.4936
Epoch 44/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 169ms/step - categorical_accuracy: 0.4250 - categoric
al_crossentropy: 1.4888 - loss: 1.4896
Epoch 45/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - categorical_accuracy: 0.4313 - categoric
al_crossentropy: 1.4848 - loss: 1.4855
Epoch 46/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 185ms/step - categorical_accuracy: 0.4313 - categoric
al_crossentropy: 1.4806 - loss: 1.4814
Epoch 47/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step - categorical_accuracy: 0.4375 - categoric
al_crossentropy: 1.4764 - loss: 1.4772
Epoch 48/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 298ms/step - categorical_accuracy: 0.4563 - categoric
al_crossentropy: 1.4722 - loss: 1.4730
Epoch 49/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 310ms/step - categorical_accuracy: 0.4688 - categoric
al_crossentropy: 1.4678 - loss: 1.4687
Epoch 50/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - categorical_accuracy: 0.4750 - categoric
al_crossentropy: 1.4634 - loss: 1.4643
Epoch 51/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 162ms/step - categorical_accuracy: 0.4875 - categoric
al_crossentropy: 1.4590 - loss: 1.4598
Epoch 52/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 313ms/step - categorical_accuracy: 0.4875 - categoric
al_crossentropy: 1.4544 - loss: 1.4553
Epoch 53/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 286ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.4499 - loss: 1.4508
Epoch 54/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 251ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.4452 - loss: 1.4461
Epoch 55/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.4405 - loss: 1.4415
Epoch 56/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.4358 - loss: 1.4367
Epoch 57/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - categorical_accuracy: 0.5125 - categorica
l_crossentropy: 1.4310 - loss: 1.4320
Epoch 58/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.5125 - categoric
al_crossentropy: 1.4262 - loss: 1.4272
Epoch 59/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - categorical_accuracy: 0.5125 - categoric
al_crossentropy: 1.4213 - loss: 1.4223
Epoch 60/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 285ms/step - categorical_accuracy: 0.5000 - categoric
al_crossentropy: 1.4164 - loss: 1.4174
```

```
Epoch 61/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 340ms/step - categorical_accuracy: 0.5063 - categoric
al_crossentropy: 1.4114 - loss: 1.4125
Epoch 62/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 222ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.4064 - loss: 1.4075
Epoch 63/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - categorical_accuracy: 0.5188 - categorica
l_crossentropy: 1.4014 - loss: 1.4025
Epoch 64/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.3963 - loss: 1.3975
Epoch 65/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 317ms/step - categorical_accuracy: 0.5063 - categoric
al_crossentropy: 1.3912 - loss: 1.3924
Epoch 66/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 253ms/step - categorical_accuracy: 0.5000 - categoric
al_crossentropy: 1.3861 - loss: 1.3873
Epoch 67/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - categorical_accuracy: 0.5000 - categoric
al_crossentropy: 1.3810 - loss: 1.3823
Epoch 68/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - categorical_accuracy: 0.4938 - categorica
l_crossentropy: 1.3759 - loss: 1.3772
Epoch 69/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - categorical_accuracy: 0.4938 - categorica
l_crossentropy: 1.3708 - loss: 1.3721
Epoch 70/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - categorical_accuracy: 0.4938 - categorica
l_crossentropy: 1.3656 - loss: 1.3670
Epoch 71/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 117ms/step - categorical_accuracy: 0.4875 - categoric
al_crossentropy: 1.3605 - loss: 1.3619
Epoch 72/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step - categorical_accuracy: 0.4875 - categoric
al_crossentropy: 1.3554 - loss: 1.3568
Epoch 73/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step - categorical_accuracy: 0.4875 - categoric
al_crossentropy: 1.3503 - loss: 1.3517
Epoch 74/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.3452 - loss: 1.3467
Epoch 75/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 256ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.3401 - loss: 1.3416
Epoch 76/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - categorical_accuracy: 0.4938 - categorica
l_crossentropy: 1.3351 - loss: 1.3366
Epoch 77/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 234ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.3301 - loss: 1.3316
Epoch 78/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 181ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.3251 - loss: 1.3266
Epoch 79/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - categorical_accuracy: 0.4938 - categorica
l_crossentropy: 1.3201 - loss: 1.3217
Epoch 80/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.3152 - loss: 1.3168
```

```
Epoch 81/500
1/1 0s 125ms/step - categorical_accuracy: 0.4938 - categoric
al_crossentropy: 1.3103 - loss: 1.3120
Epoch 82/500
1/1 0s 68ms/step - categorical_accuracy: 0.5000 - categorica
l_crossentropy: 1.3054 - loss: 1.3071
Epoch 83/500
1/1 0s 77ms/step - categorical_accuracy: 0.5063 - categorica
l_crossentropy: 1.3006 - loss: 1.3023
Epoch 84/500
1/1 0s 155ms/step - categorical_accuracy: 0.5063 - categoric
al_crossentropy: 1.2958 - loss: 1.2976
Epoch 85/500
1/1 0s 342ms/step - categorical_accuracy: 0.5063 - categoric
al_crossentropy: 1.2911 - loss: 1.2929
Epoch 86/500
1/1 0s 109ms/step - categorical_accuracy: 0.5125 - categoric
al_crossentropy: 1.2864 - loss: 1.2883
Epoch 87/500
1/1 0s 99ms/step - categorical_accuracy: 0.5063 - categorica
l_crossentropy: 1.2818 - loss: 1.2837
Epoch 88/500
1/1 0s 149ms/step - categorical_accuracy: 0.5125 - categoric
al_crossentropy: 1.2772 - loss: 1.2791
Epoch 89/500
1/1 0s 116ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.2727 - loss: 1.2746
Epoch 90/500
1/1 0s 50ms/step - categorical_accuracy: 0.5188 - categorica
l_crossentropy: 1.2682 - loss: 1.2702
Epoch 91/500
1/1 0s 50ms/step - categorical_accuracy: 0.5250 - categorica
l_crossentropy: 1.2638 - loss: 1.2658
Epoch 92/500
1/1 0s 53ms/step - categorical_accuracy: 0.5250 - categorica
l_crossentropy: 1.2594 - loss: 1.2615
Epoch 93/500
1/1 0s 78ms/step - categorical_accuracy: 0.5188 - categorica
l_crossentropy: 1.2551 - loss: 1.2572
Epoch 94/500
1/1 0s 139ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.2508 - loss: 1.2529
Epoch 95/500
1/1 0s 81ms/step - categorical_accuracy: 0.5188 - categorica
l_crossentropy: 1.2466 - loss: 1.2487
Epoch 96/500
1/1 0s 205ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.2424 - loss: 1.2446
Epoch 97/500
1/1 0s 183ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.2383 - loss: 1.2405
Epoch 98/500
1/1 0s 136ms/step - categorical_accuracy: 0.5188 - categoric
al_crossentropy: 1.2342 - loss: 1.2365
Epoch 99/500
1/1 0s 131ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2302 - loss: 1.2325
Epoch 100/500
1/1 0s 176ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2262 - loss: 1.2286
```

```
Epoch 101/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2223 - loss: 1.2247
Epoch 102/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 233ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2185 - loss: 1.2209
Epoch 103/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2147 - loss: 1.2171
Epoch 104/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 323ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2109 - loss: 1.2134
Epoch 105/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2072 - loss: 1.2097
Epoch 106/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 123ms/step - categorical_accuracy: 0.5250 - categoric
al_crossentropy: 1.2035 - loss: 1.2061
Epoch 107/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step - categorical_accuracy: 0.5312 - categoric
al_crossentropy: 1.1999 - loss: 1.2025
Epoch 108/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - categorical_accuracy: 0.5375 - categorica
l_crossentropy: 1.1963 - loss: 1.1989
Epoch 109/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - categorical_accuracy: 0.5375 - categorica
l_crossentropy: 1.1928 - loss: 1.1954
Epoch 110/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - categorical_accuracy: 0.5437 - categorica
l_crossentropy: 1.1893 - loss: 1.1920
Epoch 111/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - categorical_accuracy: 0.5562 - categorica
l_crossentropy: 1.1858 - loss: 1.1886
Epoch 112/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 202ms/step - categorical_accuracy: 0.5562 - categoric
al_crossentropy: 1.1824 - loss: 1.1852
Epoch 113/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 275ms/step - categorical_accuracy: 0.5562 - categoric
al_crossentropy: 1.1791 - loss: 1.1819
Epoch 114/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - categorical_accuracy: 0.5562 - categoric
al_crossentropy: 1.1757 - loss: 1.1786
Epoch 115/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step - categorical_accuracy: 0.5562 - categoric
al_crossentropy: 1.1724 - loss: 1.1753
Epoch 116/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - categorical_accuracy: 0.5562 - categorica
l_crossentropy: 1.1692 - loss: 1.1721
Epoch 117/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step - categorical_accuracy: 0.5562 - categoric
al_crossentropy: 1.1660 - loss: 1.1689
Epoch 118/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - categorical_accuracy: 0.5625 - categorica
l_crossentropy: 1.1628 - loss: 1.1657
Epoch 119/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - categorical_accuracy: 0.5625 - categorica
l_crossentropy: 1.1596 - loss: 1.1626
Epoch 120/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 188ms/step - categorical_accuracy: 0.5625 - categoric
al_crossentropy: 1.1565 - loss: 1.1595
```

JOAO OTAVIO NASCIMENTO FIRIGATO, joaootavionf007@gmail.com

```
Epoch 121/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 203ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1534 - loss: 1.1565
Epoch 122/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1503 - loss: 1.1534
Epoch 123/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - categorical_accuracy: 0.5688 - categorica
l_crossentropy: 1.1473 - loss: 1.1504
Epoch 124/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 188ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1442 - loss: 1.1474
Epoch 125/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 222ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1412 - loss: 1.1444
Epoch 126/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 230ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1382 - loss: 1.1415
Epoch 127/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 315ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1353 - loss: 1.1386
Epoch 128/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 282ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1323 - loss: 1.1357
Epoch 129/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step - categorical_accuracy: 0.5688 - categoric
al_crossentropy: 1.1294 - loss: 1.1328
Epoch 130/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - categorical_accuracy: 0.5750 - categoric
al_crossentropy: 1.1265 - loss: 1.1299
Epoch 131/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 185ms/step - categorical_accuracy: 0.5750 - categoric
al_crossentropy: 1.1236 - loss: 1.1270
Epoch 132/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step - categorical_accuracy: 0.5813 - categoric
al_crossentropy: 1.1207 - loss: 1.1242
Epoch 133/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step - categorical_accuracy: 0.5813 - categoric
al_crossentropy: 1.1178 - loss: 1.1213
Epoch 134/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 319ms/step - categorical_accuracy: 0.5813 - categoric
al_crossentropy: 1.1150 - loss: 1.1185
Epoch 135/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - categorical_accuracy: 0.5875 - categoric
al_crossentropy: 1.1121 - loss: 1.1157
Epoch 136/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.5875 - categoric
al_crossentropy: 1.1093 - loss: 1.1129
Epoch 137/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 270ms/step - categorical_accuracy: 0.5875 - categoric
al_crossentropy: 1.1064 - loss: 1.1101
Epoch 138/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step - categorical_accuracy: 0.5875 - categoric
al_crossentropy: 1.1036 - loss: 1.1073
Epoch 139/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.1007 - loss: 1.1045
Epoch 140/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step - categorical_accuracy: 0.5938 - categoric
al_crossentropy: 1.0979 - loss: 1.1017
```

```
Epoch 141/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.0951 - loss: 1.0989
Epoch 142/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.0922 - loss: 1.0961
Epoch 143/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.0894 - loss: 1.0932
Epoch 144/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.0865 - loss: 1.0904
Epoch 145/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.0837 - loss: 1.0876
Epoch 146/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.5938 - categorica
l_crossentropy: 1.0808 - loss: 1.0848
Epoch 147/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.6000 - categorica
l_crossentropy: 1.0779 - loss: 1.0820
Epoch 148/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6000 - categorica
l_crossentropy: 1.0751 - loss: 1.0792
Epoch 149/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - categorical_accuracy: 0.6062 - categorica
l_crossentropy: 1.0722 - loss: 1.0763
Epoch 150/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.6125 - categorica
l_crossentropy: 1.0693 - loss: 1.0735
Epoch 151/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6125 - categorica
l_crossentropy: 1.0664 - loss: 1.0706
Epoch 152/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6125 - categorica
l_crossentropy: 1.0635 - loss: 1.0678
Epoch 153/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0606 - loss: 1.0649
Epoch 154/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0577 - loss: 1.0620
Epoch 155/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0547 - loss: 1.0591
Epoch 156/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.6250 - categoric
al_crossentropy: 1.0518 - loss: 1.0562
Epoch 157/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0488 - loss: 1.0533
Epoch 158/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0459 - loss: 1.0504
Epoch 159/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0429 - loss: 1.0474
Epoch 160/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0399 - loss: 1.0445
```

```
Epoch 161/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0369 - loss: 1.0415
Epoch 162/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0338 - loss: 1.0385
Epoch 163/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0308 - loss: 1.0355
Epoch 164/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0278 - loss: 1.0325
Epoch 165/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0247 - loss: 1.0295
Epoch 166/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.6250 - categorica
l_crossentropy: 1.0216 - loss: 1.0265
Epoch 167/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 1.0185 - loss: 1.0234
Epoch 168/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 1.0154 - loss: 1.0204
Epoch 169/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 1.0123 - loss: 1.0173
Epoch 170/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 1.0092 - loss: 1.0142
Epoch 171/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 1.0060 - loss: 1.0112
Epoch 172/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 1.0029 - loss: 1.0080
Epoch 173/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - categorical_accuracy: 0.6313 - categorica
l_crossentropy: 0.9997 - loss: 1.0049
Epoch 174/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6438 - categorica
l_crossentropy: 0.9965 - loss: 1.0018
Epoch 175/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - categorical_accuracy: 0.6562 - categorica
l_crossentropy: 0.9933 - loss: 0.9987
Epoch 176/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6562 - categorica
l_crossentropy: 0.9901 - loss: 0.9955
Epoch 177/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6562 - categorica
l_crossentropy: 0.9869 - loss: 0.9923
Epoch 178/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6562 - categorica
l_crossentropy: 0.9837 - loss: 0.9892
Epoch 179/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.6562 - categorica
l_crossentropy: 0.9805 - loss: 0.9860
Epoch 180/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - categorical_accuracy: 0.6687 - categorica
l_crossentropy: 0.9772 - loss: 0.9828
```

```
Epoch 181/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - categorical_accuracy: 0.6687 - categoric
al_crossentropy: 0.9740 - loss: 0.9796
Epoch 182/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6750 - categorica
l_crossentropy: 0.9707 - loss: 0.9764
Epoch 183/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.6812 - categorica
l_crossentropy: 0.9674 - loss: 0.9731
Epoch 184/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.6875 - categorica
l_crossentropy: 0.9641 - loss: 0.9699
Epoch 185/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.6938 - categorica
l_crossentropy: 0.9608 - loss: 0.9667
Epoch 186/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.6938 - categorica
l_crossentropy: 0.9575 - loss: 0.9634
Epoch 187/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.6938 - categoric
al_crossentropy: 0.9542 - loss: 0.9602
Epoch 188/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7000 - categorica
l_crossentropy: 0.9509 - loss: 0.9569
Epoch 189/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9476 - loss: 0.9537
Epoch 190/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - categorical_accuracy: 0.7063 - categoric
al_crossentropy: 0.9443 - loss: 0.9504
Epoch 191/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9409 - loss: 0.9471
Epoch 192/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9376 - loss: 0.9438
Epoch 193/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9342 - loss: 0.9405
Epoch 194/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9309 - loss: 0.9373
Epoch 195/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - categorical_accuracy: 0.7063 - categoric
al_crossentropy: 0.9275 - loss: 0.9340
Epoch 196/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9242 - loss: 0.9307
Epoch 197/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9208 - loss: 0.9274
Epoch 198/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7063 - categorica
l_crossentropy: 0.9175 - loss: 0.9241
Epoch 199/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.9141 - loss: 0.9208
Epoch 200/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.9108 - loss: 0.9175
```

```
Epoch 201/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.9074 - loss: 0.9142
Epoch 202/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.9040 - loss: 0.9109
Epoch 203/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.9007 - loss: 0.9076
Epoch 204/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.8973 - loss: 0.9043
Epoch 205/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.8940 - loss: 0.9010
Epoch 206/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7125 - categorica
l_crossentropy: 0.8906 - loss: 0.8977
Epoch 207/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7188 - categorica
l_crossentropy: 0.8872 - loss: 0.8944
Epoch 208/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - categorical_accuracy: 0.7188 - categorica
l_crossentropy: 0.8839 - loss: 0.8911
Epoch 209/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7188 - categorica
l_crossentropy: 0.8806 - loss: 0.8878
Epoch 210/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7250 - categorica
l_crossentropy: 0.8772 - loss: 0.8845
Epoch 211/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7250 - categorica
l_crossentropy: 0.8739 - loss: 0.8813
Epoch 212/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7250 - categoric
al_crossentropy: 0.8705 - loss: 0.8780
Epoch 213/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7250 - categorica
l_crossentropy: 0.8672 - loss: 0.8747
Epoch 214/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7250 - categorica
l_crossentropy: 0.8639 - loss: 0.8715
Epoch 215/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7250 - categoric
al_crossentropy: 0.8606 - loss: 0.8682
Epoch 216/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7188 - categorica
l_crossentropy: 0.8573 - loss: 0.8650
Epoch 217/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - categorical_accuracy: 0.7188 - categoric
al_crossentropy: 0.8540 - loss: 0.8618
Epoch 218/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7250 - categorica
l_crossentropy: 0.8507 - loss: 0.8586
Epoch 219/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7250 - categorica
l_crossentropy: 0.8474 - loss: 0.8553
Epoch 220/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8442 - loss: 0.8521
```

```
Epoch 221/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7312 - categoric
al_crossentropy: 0.8409 - loss: 0.8490
Epoch 222/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8377 - loss: 0.8458
Epoch 223/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8344 - loss: 0.8426
Epoch 224/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8312 - loss: 0.8395
Epoch 225/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8280 - loss: 0.8363
Epoch 226/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8248 - loss: 0.8332
Epoch 227/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7375 - categorica
l_crossentropy: 0.8216 - loss: 0.8301
Epoch 228/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7312 - categorica
l_crossentropy: 0.8185 - loss: 0.8270
Epoch 229/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7375 - categorica
l_crossentropy: 0.8153 - loss: 0.8239
Epoch 230/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7437 - categorica
l_crossentropy: 0.8122 - loss: 0.8208
Epoch 231/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7375 - categorica
l_crossentropy: 0.8091 - loss: 0.8178
Epoch 232/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7375 - categorica
l_crossentropy: 0.8060 - loss: 0.8147
Epoch 233/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7375 - categorica
l_crossentropy: 0.8029 - loss: 0.8117
Epoch 234/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7437 - categoric
al_crossentropy: 0.7998 - loss: 0.8087
Epoch 235/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step - categorical_accuracy: 0.7437 - categoric
al_crossentropy: 0.7968 - loss: 0.8058
Epoch 236/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7437 - categorica
l_crossentropy: 0.7938 - loss: 0.8028
Epoch 237/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7908 - loss: 0.7999
Epoch 238/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7878 - loss: 0.7970
Epoch 239/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7849 - loss: 0.7941
Epoch 240/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7625 - categoric
al_crossentropy: 0.7819 - loss: 0.7912
```

```
Epoch 241/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7790 - loss: 0.7884
Epoch 242/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7761 - loss: 0.7856
Epoch 243/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7733 - loss: 0.7828
Epoch 244/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7704 - loss: 0.7800
Epoch 245/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7676 - loss: 0.7772
Epoch 246/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7648 - loss: 0.7745
Epoch 247/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7620 - loss: 0.7718
Epoch 248/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7593 - loss: 0.7691
Epoch 249/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.7566 - loss: 0.7664
Epoch 250/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7625 - categoric
al_crossentropy: 0.7539 - loss: 0.7639
Epoch 251/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7513 - loss: 0.7613
Epoch 252/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7488 - loss: 0.7589
Epoch 253/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7462 - loss: 0.7564
Epoch 254/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7437 - loss: 0.7539
Epoch 255/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7411 - loss: 0.7514
Epoch 256/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7387 - loss: 0.7490
Epoch 257/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7364 - loss: 0.7467
Epoch 258/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7340 - loss: 0.7444
Epoch 259/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7316 - loss: 0.7421
Epoch 260/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7293 - loss: 0.7399
```

```
Epoch 261/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7270 - loss: 0.7377
Epoch 262/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7248 - loss: 0.7355
Epoch 263/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7500 - categorica
l_crossentropy: 0.7225 - loss: 0.7332
Epoch 264/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7203 - loss: 0.7311
Epoch 265/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.7182 - loss: 0.7290
Epoch 266/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7160 - loss: 0.7269
Epoch 267/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.7139 - loss: 0.7248
Epoch 268/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7117 - loss: 0.7227
Epoch 269/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7096 - loss: 0.7207
Epoch 270/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7076 - loss: 0.7187
Epoch 271/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7056 - loss: 0.7167
Epoch 272/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.7035 - loss: 0.7148
Epoch 273/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.7015 - loss: 0.7128
Epoch 274/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6995 - loss: 0.7108
Epoch 275/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6975 - loss: 0.7089
Epoch 276/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6956 - loss: 0.7070
Epoch 277/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6937 - loss: 0.7052
Epoch 278/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6918 - loss: 0.7033
Epoch 279/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6899 - loss: 0.7015
Epoch 280/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6880 - loss: 0.6997
```

```
Epoch 281/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6861 - loss: 0.6979
Epoch 282/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step - categorical_accuracy: 0.7563 - categoric
al_crossentropy: 0.6843 - loss: 0.6961
Epoch 283/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - categorical_accuracy: 0.7625 - categoric
al_crossentropy: 0.6825 - loss: 0.6943
Epoch 284/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - categorical_accuracy: 0.7563 - categorica
l_crossentropy: 0.6807 - loss: 0.6926
Epoch 285/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - categorical_accuracy: 0.7750 - categoric
al_crossentropy: 0.6789 - loss: 0.6909
Epoch 286/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.6773 - loss: 0.6893
Epoch 287/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.6758 - loss: 0.6879
Epoch 288/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7688 - categorica
l_crossentropy: 0.6747 - loss: 0.6868
Epoch 289/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7750 - categorica
l_crossentropy: 0.6734 - loss: 0.6856
Epoch 290/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.6715 - loss: 0.6838
Epoch 291/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.6692 - loss: 0.6815
Epoch 292/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step - categorical_accuracy: 0.7812 - categoric
al_crossentropy: 0.6680 - loss: 0.6804
Epoch 293/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.6672 - loss: 0.6795
Epoch 294/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.6651 - loss: 0.6775
Epoch 295/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6636 - loss: 0.6760
Epoch 296/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7625 - categorica
l_crossentropy: 0.6627 - loss: 0.6752
Epoch 297/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.6611 - loss: 0.6736
Epoch 298/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6596 - loss: 0.6722
Epoch 299/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7688 - categorica
l_crossentropy: 0.6585 - loss: 0.6712
Epoch 300/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.6571 - loss: 0.6698
```

```
Epoch 301/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6559 - loss: 0.6686
Epoch 302/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.6546 - loss: 0.6674
Epoch 303/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6533 - loss: 0.6661
Epoch 304/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6522 - loss: 0.6650
Epoch 305/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - categorical_accuracy: 0.7750 - categorica
l_crossentropy: 0.6509 - loss: 0.6638
Epoch 306/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.6496 - loss: 0.6625
Epoch 307/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6486 - loss: 0.6615
Epoch 308/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.6474 - loss: 0.6604
Epoch 309/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.6461 - loss: 0.6591
Epoch 310/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6450 - loss: 0.6581
Epoch 311/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6439 - loss: 0.6570
Epoch 312/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6427 - loss: 0.6558
Epoch 313/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6416 - loss: 0.6548
Epoch 314/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6405 - loss: 0.6537
Epoch 315/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6394 - loss: 0.6526
Epoch 316/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6383 - loss: 0.6516
Epoch 317/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6372 - loss: 0.6505
Epoch 318/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.6361 - loss: 0.6495
Epoch 319/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.6351 - loss: 0.6485
Epoch 320/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6340 - loss: 0.6474
```

```
Epoch 321/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6329 - loss: 0.6464
Epoch 322/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6319 - loss: 0.6454
Epoch 323/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6308 - loss: 0.6444
Epoch 324/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6298 - loss: 0.6434
Epoch 325/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6288 - loss: 0.6424
Epoch 326/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6278 - loss: 0.6414
Epoch 327/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6267 - loss: 0.6405
Epoch 328/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6258 - loss: 0.6395
Epoch 329/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6248 - loss: 0.6386
Epoch 330/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6238 - loss: 0.6376
Epoch 331/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6228 - loss: 0.6367
Epoch 332/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6218 - loss: 0.6357
Epoch 333/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6208 - loss: 0.6348
Epoch 334/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6199 - loss: 0.6339
Epoch 335/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6189 - loss: 0.6330
Epoch 336/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6180 - loss: 0.6321
Epoch 337/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6171 - loss: 0.6311
Epoch 338/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6161 - loss: 0.6302
Epoch 339/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6152 - loss: 0.6294
Epoch 340/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6143 - loss: 0.6285
```

```
Epoch 341/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6134 - loss: 0.6276
Epoch 342/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6125 - loss: 0.6267
Epoch 343/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6116 - loss: 0.6259
Epoch 344/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6107 - loss: 0.6250
Epoch 345/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6098 - loss: 0.6242
Epoch 346/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6089 - loss: 0.6233
Epoch 347/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6080 - loss: 0.6225
Epoch 348/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6072 - loss: 0.6216
Epoch 349/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6063 - loss: 0.6208
Epoch 350/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6055 - loss: 0.6200
Epoch 351/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6047 - loss: 0.6193
Epoch 352/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6039 - loss: 0.6185
Epoch 353/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6032 - loss: 0.6179
Epoch 354/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.6025 - loss: 0.6173
Epoch 355/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6020 - loss: 0.6167
Epoch 356/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.6013 - loss: 0.6161
Epoch 357/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.6003 - loss: 0.6151
Epoch 358/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5992 - loss: 0.6140
Epoch 359/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5982 - loss: 0.6131
Epoch 360/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5976 - loss: 0.6125
```

```
Epoch 361/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5971 - loss: 0.6121
Epoch 362/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5965 - loss: 0.6114
Epoch 363/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5956 - loss: 0.6106
Epoch 364/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5947 - loss: 0.6097
Epoch 365/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5940 - loss: 0.6091
Epoch 366/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5934 - loss: 0.6086
Epoch 367/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5928 - loss: 0.6080
Epoch 368/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5921 - loss: 0.6072
Epoch 369/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5912 - loss: 0.6064
Epoch 370/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5906 - loss: 0.6058
Epoch 371/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5900 - loss: 0.6053
Epoch 372/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5894 - loss: 0.6047
Epoch 373/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5887 - loss: 0.6040
Epoch 374/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5880 - loss: 0.6033
Epoch 375/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5873 - loss: 0.6027
Epoch 376/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5867 - loss: 0.6021
Epoch 377/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5861 - loss: 0.6016
Epoch 378/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5855 - loss: 0.6010
Epoch 379/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5848 - loss: 0.6003
Epoch 380/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5842 - loss: 0.5997
```

```
Epoch 381/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5835 - loss: 0.5991
Epoch 382/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5830 - loss: 0.5985
Epoch 383/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5824 - loss: 0.5980
Epoch 384/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5818 - loss: 0.5974
Epoch 385/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5812 - loss: 0.5969
Epoch 386/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5806 - loss: 0.5963
Epoch 387/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5799 - loss: 0.5957
Epoch 388/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5793 - loss: 0.5951
Epoch 389/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5788 - loss: 0.5946
Epoch 390/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5782 - loss: 0.5940
Epoch 391/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5776 - loss: 0.5935
Epoch 392/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5771 - loss: 0.5929
Epoch 393/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5765 - loss: 0.5924
Epoch 394/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5759 - loss: 0.5918
Epoch 395/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5753 - loss: 0.5913
Epoch 396/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5748 - loss: 0.5908
Epoch 397/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5742 - loss: 0.5902
Epoch 398/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5736 - loss: 0.5897
Epoch 399/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5731 - loss: 0.5892
Epoch 400/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5725 - loss: 0.5886
```

```
Epoch 401/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5720 - loss: 0.5881
Epoch 402/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5714 - loss: 0.5876
Epoch 403/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5709 - loss: 0.5871
Epoch 404/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5704 - loss: 0.5866
Epoch 405/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5698 - loss: 0.5861
Epoch 406/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5693 - loss: 0.5856
Epoch 407/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5688 - loss: 0.5851
Epoch 408/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5683 - loss: 0.5847
Epoch 409/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5679 - loss: 0.5843
Epoch 410/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5676 - loss: 0.5840
Epoch 411/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5674 - loss: 0.5838
Epoch 412/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5673 - loss: 0.5838
Epoch 413/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - categorical_accuracy: 0.8000 - categorica
l_crossentropy: 0.5673 - loss: 0.5838
Epoch 414/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5667 - loss: 0.5832
Epoch 415/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 148ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5655 - loss: 0.5820
Epoch 416/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5644 - loss: 0.5809
Epoch 417/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5641 - loss: 0.5807
Epoch 418/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5642 - loss: 0.5808
Epoch 419/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5638 - loss: 0.5804
Epoch 420/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5629 - loss: 0.5795
```

```
Epoch 421/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5622 - loss: 0.5789
Epoch 422/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 152ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5620 - loss: 0.5787
Epoch 423/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5619 - loss: 0.5786
Epoch 424/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5613 - loss: 0.5780
Epoch 425/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5606 - loss: 0.5773
Epoch 426/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5602 - loss: 0.5770
Epoch 427/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5601 - loss: 0.5769
Epoch 428/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5596 - loss: 0.5764
Epoch 429/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5590 - loss: 0.5758
Epoch 430/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5586 - loss: 0.5754
Epoch 431/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5583 - loss: 0.5752
Epoch 432/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5580 - loss: 0.5749
Epoch 433/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5574 - loss: 0.5744
Epoch 434/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5570 - loss: 0.5740
Epoch 435/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5567 - loss: 0.5737
Epoch 436/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7937 - categorica
l_crossentropy: 0.5564 - loss: 0.5734
Epoch 437/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5559 - loss: 0.5730
Epoch 438/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5555 - loss: 0.5725
Epoch 439/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5551 - loss: 0.5722
Epoch 440/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5548 - loss: 0.5719
```

```
Epoch 441/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5544 - loss: 0.5716
Epoch 442/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5540 - loss: 0.5712
Epoch 443/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5536 - loss: 0.5708
Epoch 444/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5533 - loss: 0.5705
Epoch 445/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5529 - loss: 0.5702
Epoch 446/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5526 - loss: 0.5698
Epoch 447/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5522 - loss: 0.5694
Epoch 448/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5518 - loss: 0.5691
Epoch 449/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5515 - loss: 0.5688
Epoch 450/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5511 - loss: 0.5684
Epoch 451/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5508 - loss: 0.5681
Epoch 452/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5504 - loss: 0.5677
Epoch 453/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5500 - loss: 0.5674
Epoch 454/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5497 - loss: 0.5671
Epoch 455/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5493 - loss: 0.5668
Epoch 456/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5490 - loss: 0.5664
Epoch 457/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5486 - loss: 0.5661
Epoch 458/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5483 - loss: 0.5658
Epoch 459/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5479 - loss: 0.5655
Epoch 460/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5476 - loss: 0.5651
```

```
Epoch 461/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5473 - loss: 0.5648
Epoch 462/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5469 - loss: 0.5645
Epoch 463/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5466 - loss: 0.5642
Epoch 464/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5462 - loss: 0.5639
Epoch 465/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5459 - loss: 0.5635
Epoch 466/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5456 - loss: 0.5632
Epoch 467/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5452 - loss: 0.5629
Epoch 468/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5449 - loss: 0.5626
Epoch 469/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5446 - loss: 0.5623
Epoch 470/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5442 - loss: 0.5620
Epoch 471/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5439 - loss: 0.5617
Epoch 472/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5436 - loss: 0.5614
Epoch 473/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5433 - loss: 0.5611
Epoch 474/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5429 - loss: 0.5608
Epoch 475/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5426 - loss: 0.5605
Epoch 476/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5423 - loss: 0.5602
Epoch 477/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5420 - loss: 0.5599
Epoch 478/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5417 - loss: 0.5596
Epoch 479/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5413 - loss: 0.5593
Epoch 480/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5410 - loss: 0.5590
```

```
Epoch 481/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5407 - loss: 0.5587
Epoch 482/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5404 - loss: 0.5584
Epoch 483/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5401 - loss: 0.5581
Epoch 484/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5398 - loss: 0.5578
Epoch 485/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5395 - loss: 0.5575
Epoch 486/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5391 - loss: 0.5572
Epoch 487/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5388 - loss: 0.5570
Epoch 488/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5385 - loss: 0.5567
Epoch 489/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5382 - loss: 0.5564
Epoch 490/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7875 - categorica
l_crossentropy: 0.5380 - loss: 0.5561
Epoch 491/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.5377 - loss: 0.5559
Epoch 492/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5374 - loss: 0.5557
Epoch 493/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - categorical_accuracy: 0.7812 - categorica
l_crossentropy: 0.5373 - loss: 0.5555
Epoch 494/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5371 - loss: 0.5554
Epoch 495/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5371 - loss: 0.5554
Epoch 496/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - categorical_accuracy: 0.8000 - categorica
l_crossentropy: 0.5372 - loss: 0.5555
Epoch 497/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5373 - loss: 0.5556
Epoch 498/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - categorical_accuracy: 0.8000 - categorica
l_crossentropy: 0.5370 - loss: 0.5554
Epoch 499/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - categorical_accuracy: 0.7875 - categoric
al_crossentropy: 0.5363 - loss: 0.5547
Epoch 500/500
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - categorical_accuracy: 0.7937 - categoric
al_crossentropy: 0.5353 - loss: 0.5537
```

```
Out[ ]:
▾ LearningShapelets
LearningShapelets(max_iter=500, n_shapelets_per_size={12: 4},
                  optimizer=<keras.src.optimizers.adam.Adam object at 0
x7c6fa00d1dd0>,
                  random_state=42, verbose=1, weight_regularizer=0.000
1)
```

```
In [ ]: y_pred = shp_clf.predict(X_test)
```

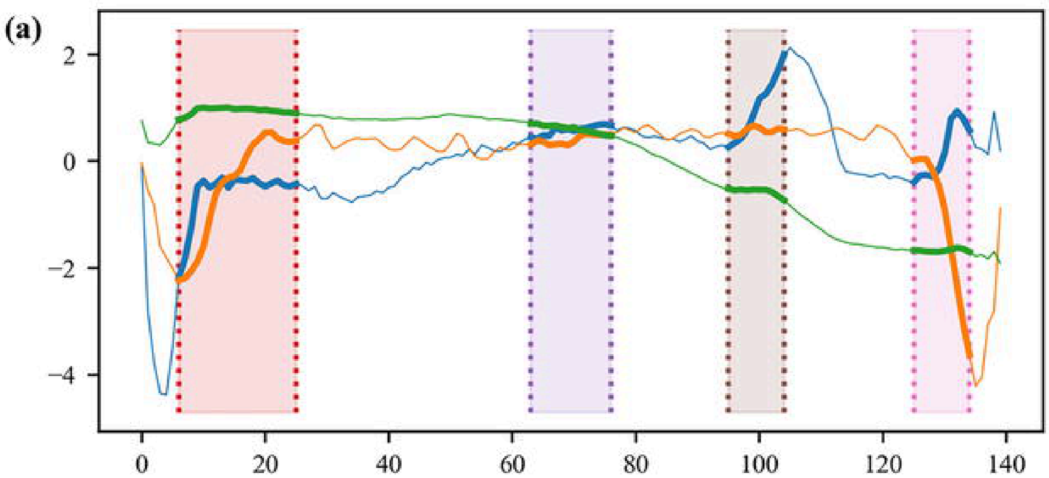
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
```

```
In [ ]: accuracy_score(y_test, y_pred)
```

```
Out[ ]: 0.675
```

Tree-Based Algorithms

One of the first algorithms proposed based on the random forest algorithm is called time series forest and is relatively simple. The algorithm considers information from subsequences of the time series. Given a minimum length for the subsequences, which is a hyperparameter, random intervals are generated, with the start indices, end indices, and lengths of all intervals being all randomly generated. For a given time series and a given interval, the corresponding subsequence is the ordered set of time series values belonging to the interval. From each subsequence, three features are extracted: the mean, the standard deviation, and the slope.



(b)

	Interval 1			Interval 2			Interval 3			Interval 4		
	Mean	SD	Slope	Mean	SD	Slope	Mean	SD	Slope	Mean	SD	Slope
Time series 1	-0.61	0.504	0.052	0.58	0.086	0.02	1.004	0.572	0.197	0.189	0.504	0.156
Time series 2	-0.467	0.977	0.158	0.403	0.084	0.017	0.568	0.061	0.011	-1.204	1.305	-0.431
Time series 3	0.944	0.061	0.002	0.604	0.075	-0.018	-0.57	0.065	-0.017	-1.671	0.024	0.003

The total number of extracted features is therefore three times the number of intervals considered. A random forest classifier is then trained on these extracted features.

Predictions for new time series are obtained in the same way: Given the intervals already generated, the three features are extracted from each subsequence, then the fine-tuned random forest classifier outputs its prediction.

In [ ]:
```
from sktime.classification.interval_based import TimeSeriesForestClassifier
from sktime.datatypes import convert_to
from tslearn.utils import to_sktime_dataset
```

Let's prepare our data to apply this algorithm:

In [ ]:
```
X_tslearn
```

Out[ ]:
```
array([[[0.31813478],
        [0.19832304],
        [0.19720842],
        ...,
        [0.39059409],
        [0.41053399],
        [0.13294899]],

       [[0.21179298],
        [0.15874021],
        [0.15342478],
        ...,
        [0.31940517],
        [0.33896241],
        [0.13153593]],

       [[0.29856932],
        [0.19431072],
        [0.15448278],
        ...,
        [0.35244396],
        [0.3413209 ],
        [0.13553786]],

       ...,

       [[0.84316188],
        [0.72604793],
        [0.68323469],
        ...,
        [0.81082076],
        [0.82439131],
        [0.20189609]],

       [[0.42934331],
        [0.43256783],
        [0.439226  ],
        ...,
        [0.70383036],
        [0.45613879],
        [0.16385904]],

       [[0.4548862 ],
        [0.38271028],
        [0.46803191],
        ...,
        [0.69260389],
        [0.43452859],
        [0.16668539]]])
```

Let's convert the data to the format that sktime needs:

In [ ]:
```
X_sktime = convert_to(X_tslearn, to_type="pd-multiindex")
```

In [ ]:
```
X_sktime = to_sktime_dataset(X_tslearn)
```

In [ ]:
```
X_sktime
```

Out[ ]:

	dim_0
0	0 0.318135 1 0.198323 2 0.197208 3...
1	0 0.211793 1 0.158740 2 0.153425 3...
2	0 0.298569 1 0.194311 2 0.154483 3...
3	0 0.443601 1 0.439508 2 0.522739 3...
4	0 0.209651 1 0.137768 2 0.153815 3...
...	...
195	0 0.589531 1 0.629662 2 0.764611 3...
196	0 0.836447 1 0.714871 2 0.624605 3...
197	0 0.843162 1 0.726048 2 0.683235 3...
198	0 0.429343 1 0.432568 2 0.439226 3...
199	0 0.454886 1 0.382710 2 0.468032 3...

200 rows × 1 columns

Then we can split the data into training and testing. Then we apply the algorithm:

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X_sktime, y, test_size=0.2,
```

In [ ]:
```
clf = TimeSeriesForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)
```

Out[ ]:
```
▾ TimeSeriesForestClassifier
TimeSeriesForestClassifier(n_estimators=100)
```

In [ ]:
```
y_pred = clf.predict(X_test)
```

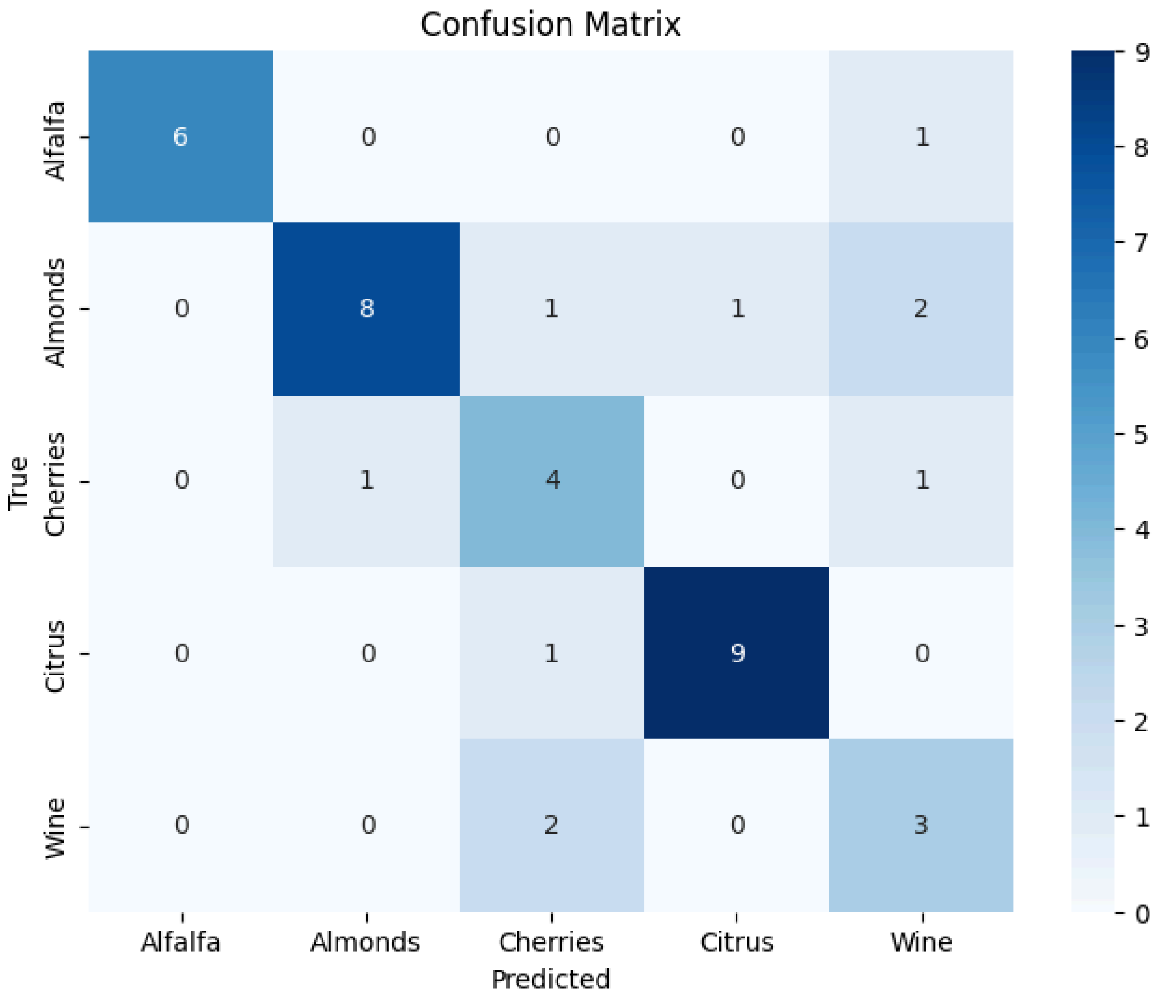
Let's check the results:

In [ ]:
```
accuracy_score(y_test, y_pred)
```

Out[ ]: 0.75

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns


cm = confusion_matrix(y_test, y_pred)


plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel("Predicted")
plt.ylabel("True")
```

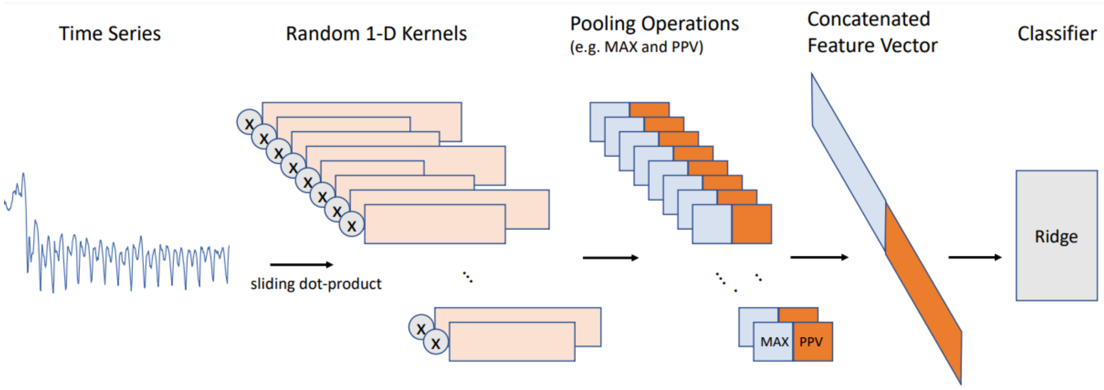
```
plt.title("Confusion Matrix")
plt.show()
```



True \ Predicted	Alfalfa	Almonds	Cherries	Citrus	Wine
Alfalfa	6	0	0	0	1
Almonds	0	8	1	1	2
Cherries	0	1	4	0	1
Citrus	0	0	1	9	0
Wine	0	0	2	0	3

Random Convolutional Kernel Transform (ROCKET)

This algorithm extracts features from time series using a large number of random convolutional kernels, meaning that all parameters of all kernels (length, weights, bias, dilation, and padding) are randomly generated from fixed distributions. Instead of extracting a single feature for each kernel, such as the maximum or the mean, as is typically done in convolutional neural networks, two features are extracted: the maximum and the proportion of positive values.



The classifier built on top of the transformation is responsible for selecting the most relevant features to perform the classification. A ridge regression classifier was originally proposed for several reasons.

- First, it is highly efficient when the number of classes is large, because the multi-class classification task is treated as a multi-output regression task, with the predicted class corresponding to the output with the highest value; therefore, the projection matrix needs to be computed only once.

- Second, optimizing the λ parameter (controlling the amount of regularization) using leave-one-out cross-validation is also highly efficient. Logistic regression was used for datasets in which the number of training time series was much larger than the number of extracted time series due to the better scalability of logistic regression solved with stochastic gradient descent to large numbers of training samples.

The ROCKET algorithm combined with a linear classifier has a much lower computational complexity than the best performing time series classification algorithms, although it performs comparably. Its reported performance is actually higher on average than that of convolutional neural networks on commonly compared datasets. Given its high predictive performance and low computational time, ROCKET is one of the most prominent transformation algorithms for time series classification.

Several recent extensions have been proposed. MiniROCKET reduces the randomness of kernel parameters by using a fixed value or sampling from smaller distributions. Furthermore, it extracts only the proportion of positive values for each kernel. These modifications also allow for more optimization and lead to much lower computational complexity while maintaining similar performance. MultiROCKET extends MiniROCKET by extracting possibly multiple features, leading to slightly higher computational time but better accuracy. In particular, the authors found that the proportion of positive values and the longest period of consecutive positive values are the most effective features to extract from convolutional time series outputs.

Let's apply the Rocket classifier to our data:

In [ ]:
```
from sktime.classification.kernel_based import RocketClassifier
```

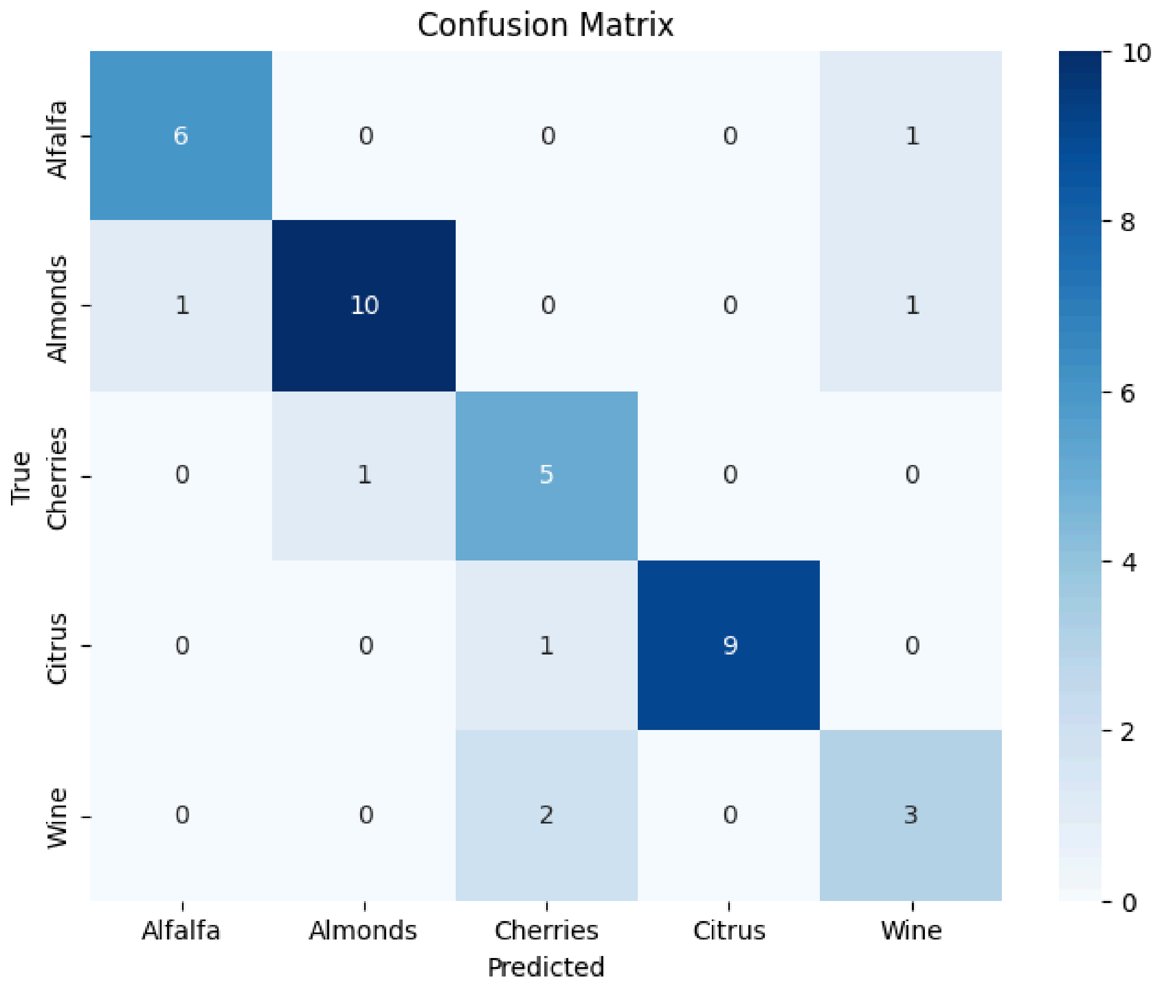
In [ ]:
```
rocket = RocketClassifier(num_kernels=2000)
rocket.fit(X_train, y_train)
```

Out[ ]:
```
▾ RocketClassifier
RocketClassifier(num_kernels=2000)
```

In [ ]:
```
y_pred = rocket.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[ ]:
```
0.825
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns


cm = confusion_matrix(y_test, y_pred)


plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

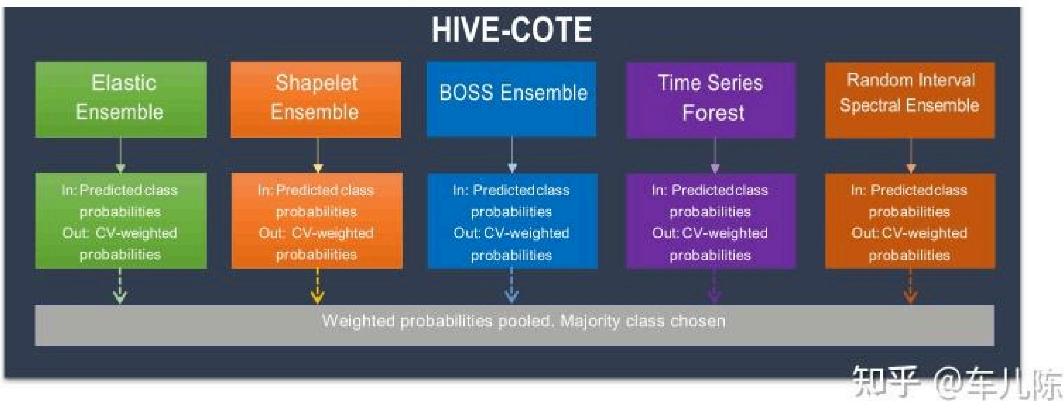


Ensemble Models

Averaging the predictions of multiple independently trained models into a single prediction is a common approach to building a better final model by decreasing the variance of the predictions. In traditional ensemble methods, all base classifiers belong to a certain type of algorithm. For example, in a random forest, all base classifiers are decision trees. However, using a single type of algorithm limits the advantages and disadvantages of the final model to those of the base classifier. On the other hand, using multiple types of algorithms allows learning a more diverse representation of the data. In particular, for time series classification, ensemble models that combine different types of algorithms (bag-of-words approaches, shapelet-based algorithms, convolutions, etc.) have been developed. They are usually state-of-the-art in terms of predictive performance, at the cost of high computational complexity.

The Collective of Transformation-Based Ensembles (COTE) algorithm was the first proposed ensemble classifier. The most effective ensemble strategy was found to combine all classifiers into a flat hierarchy, and the corresponding model is often referred to as Flat-COTE. Flat-COTE combines 35 classifiers on four data representations: 11 classifiers based on integer series similarity measures, 8 shapelet-transform based classifiers, 8 based on autocorrelation features, and 8 based on power spectrum.

The Hierarchical Vote Collective of Transformation-Based Ensembles (HIVE-COTE) algorithm is an extension of COTE with significant modifications, including a new type of spectral classifier called Random Interval Spectral Ensemble, two more classifiers (BOSS and Time Series Forest), and a hierarchical voting procedure, defined as a weighted average of the probabilities returned by each classifier, with the weights being proportional to the classification accuracy estimated through cross-validation.



Finally, we applied HIVECOTEV2 to our data:

In [ ]:
```
from sktime.classification.hybrid import HIVECOTEV2
```

In [ ]:
```
hc2 = HIVECOTEV2(time_limit_in_minutes=0.2)
hc2.fit(X_train, y_train)
```

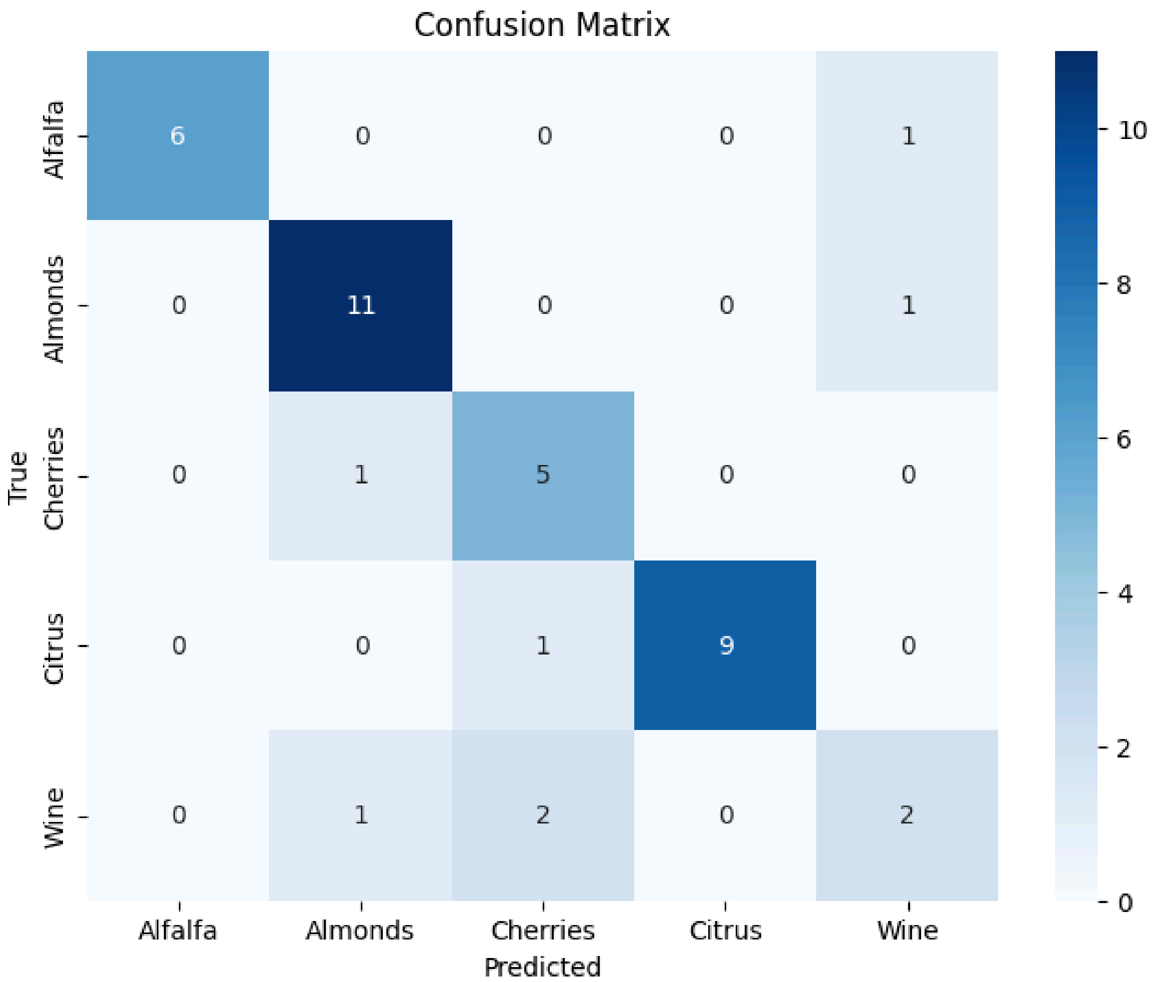
Out[ ]:
```
HIVECOTEV2
HIVECOTEV2(time_limit_in_minutes=0.2)
```

In [ ]:
```
y_pred = hc2.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[ ]: 0.825

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

```
cm = confusion_matrix(y_test, y_pred)


plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```



Thank you! See you in the next Chapter!

References:

https://www.intechopen.com/chapters/1185930

https://www.sktime.net/en/v0.20.0/examples/02_classification.html

https://medium.com/@quantclubiitkgp/time-series-classification-using-dynamic-time-warping-k-nearest-neighbour-e683896e0861

https://medium.com/version-1/an-introduction-to-shapelets-the-shapes-in-time-series-c55b94205614

https://tslearn.readthedocs.io/en/latest/user_guide/shapelets.html

https://github.com/ashishpatel26/Shapelet-time-Series-Classification/blob/main/Time_Series_Classification_Shaplet_Learning.ipynb

https://www.kaggle.com/code/somertonman/time-series-classification-using-deep-learning