DATA SCIENCE, MACHINE LEARNING, AND BUSINESS ANALYTICS PROJECT

Objective:
This project aims to explore, implement, and compare various machine learning models on a classification dataset. This project will help you to understand the nuances of different algorithms and their performance metrics.

This project consists of three different parts and these will be in the order in the codes with the explanation:

Part A: Dataset Selection and Classification Models
Part B: Dimensionality Reduction and Model Evaluation
Part C: ANN and CNN

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
gesture=pd.read_csv("/content/drive/MyDrive/raw_gesture_phase_segmentation.csv")
gesture
```

Out[2]:

	lhx	lhy	lhz	rhx	rhy	rhz	hx	hy	hz	sx	sy	sz	lwx	lw
0	5.347435	4.363681	1.501913	5.258967	4.319263	1.488703	5.037871	1.618295	1.778350	5.062803	4.229656	1.772577	4.972902	4.30106
1	4.869622	4.254210	1.556133	5.240113	4.346338	1.554309	5.037610	1.618370	1.778573	5.061430	4.228504	1.772859	4.974908	4.30365
2	5.357447	4.364039	1.500969	5.238928	4.347924	1.554150	5.037514	1.618298	1.778774	5.059245	4.228004	1.773568	4.981612	4.30536
3	4.942886	4.281878	1.546513	5.111436	4.229660	1.527091	5.037526	1.618612	1.778855	5.056475	4.226891	1.774519	4.987158	4.30406
4	5.003160	4.278530	1.542866	4.985812	4.182155	1.520330	5.037557	1.619226	1.778925	5.052367	4.225485	1.775536	4.983912	4.29683
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
1742	4.999737	3.840355	1.577457	4.990017	4.124610	1.546410	5.092207	1.616004	1.793314	5.094342	4.235239	1.776703	4.911455	4.14501
1743	5.001617	3.840771	1.577161	4.695521	4.148509	1.530205	5.092773	1.615753	1.793486	5.094236	4.235837	1.776754	4.913334	4.14543
1744	4.996975	3.841236	1.578423	4.700123	4.141193	1.530697	5.092515	1.615095	1.793637	5.093153	4.235420	1.776905	4.908689	4.14591
1745	5.000125	3.841455	1.577914	4.690338	4.153884	1.526481	5.092476	1.614777	1.793768	5.093362	4.234446	1.777077	4.911839	4.14613
1746	5.007900	3.841846	1.576601	5.004423	4.119151	1.547947	5.092856	1.614567	1.793805	5.093344	4.234407	1.777148	4.919612	4.14652

1747 rows × 20 columns

In [3]:
```
gesture.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1747 entries, 0 to 1746
Data columns (total 20 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   lhx        1747 non-null   float64
 1   lhy        1747 non-null   float64
 2   lhz        1747 non-null   float64
 3   rhx        1747 non-null   float64
 4   rhy        1747 non-null   float64
 5   rhz        1747 non-null   float64
 6   hx         1747 non-null   float64
 7   hy         1747 non-null   float64
 8   hz         1747 non-null   float64
 9   sx         1747 non-null   float64
 10  sy         1747 non-null   float64
 11  sz         1747 non-null   float64
 12  lwx        1747 non-null   float64
 13  lwy        1747 non-null   float64
 14  lwz        1747 non-null   float64
 15  rwx        1747 non-null   float64
 16  rwy        1747 non-null   float64
 17  rwz        1747 non-null   float64
 18  timestamp  1747 non-null   int64
 19  phase      1747 non-null   object
dtypes: float64(18), int64(1), object(1)
memory usage: 273.1+ KB
```

In [4]:
```
gesturee=gesture.rename(columns={'lhx':'Position of left hand (x coordinate)','lhy':'Position of left hand (y coordi
                                 'lhz':'Position of left hand (z coordinate)','rhx':'Position of right hand (x coor
                        'rhy':'Position of right hand (y coordinate)','rhz':'Position of right hand (z coordinate)',
                        'hx':'Position of head (x coordinate)','hy':'Position of head (y coordinate)',
                        'hz':'Position of head  (z coordinate)','sx':'Position of spine (x coordinate)',
                        'sy':'Position of spine (y coordinate)',
                        'sz':'Position of spine (z coordinate)','lwx':'Position of left wrist (x coordinate)',
                        'lwy':'Position of left wrist (y coordinate)',
                        'lwz':'Position of left wrist (z coordinate)','rwx':'Position of right wrist (x coordinate)',
                        'rwy':'Position of right wrist (y coordinate)',
                        'rwz':'Position of right wrist (z coordinate)'})
gesturee.head()
```

Out[4]:

	Position of left hand (x coordinate)	Position of left hand (y coordinate)	Position of left hand (z coordinate)	Position of right hand (x coordinate)	Position of right hand (y coordinate)	Position of right hand (z coordinate)	Position of head (x coordinate)	Position of head (y coordinate)	Position of head (z coordinate)	Position of spine (x coordinate)	Position of spine (y coordinate)	Po coo
0	5.347435	4.363681	1.501913	5.258967	4.319263	1.488703	5.037871	1.618295	1.778350	5.062803	4.229656	1
1	4.869622	4.254210	1.556133	5.240113	4.346338	1.554309	5.037610	1.618370	1.778573	5.061430	4.228504	1
2	5.357447	4.364039	1.500969	5.238928	4.347924	1.554150	5.037514	1.618298	1.778774	5.059245	4.228004	1
3	4.942886	4.281878	1.546513	5.111436	4.229660	1.527091	5.037526	1.618612	1.778855	5.056475	4.226891	1
4	5.003160	4.278530	1.542866	4.985812	4.182155	1.520330	5.037557	1.619226	1.778925	5.052367	4.225485	1

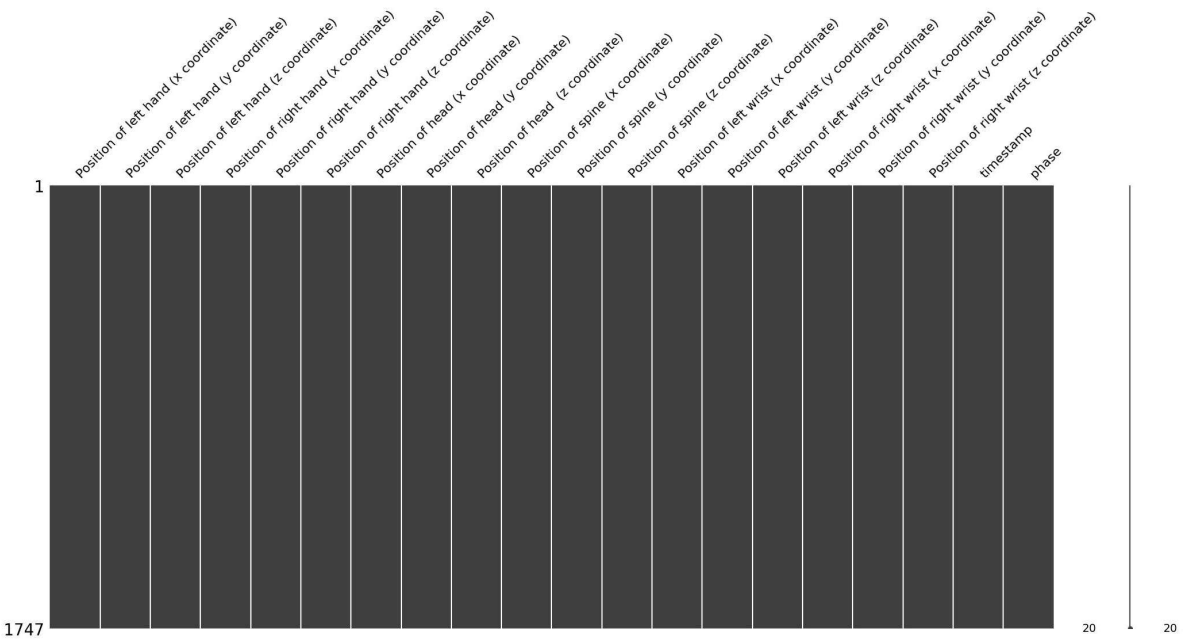
##Data Cleaning

In [5]:
```
!pip install missingno
```

```
Requirement already satisfied: missingno in /usr/local/lib/python3.10/dist-packages (0.5.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from missingno) (1.23.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from missingno) (3.7.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from missingno) (1.11.4)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (from missingno) (0.13.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missin
gno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missi
ngno) (4.48.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missi
ngno) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missing
no) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingn
o) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missin
gno) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->mi
ssingno) (2.8.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn->missingno) (1.
5.3)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn->
missingno) (2023.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matp
lotlib->missingno) (1.16.0)
```

In [6]:
```
import missingno as ms
ms.matrix(gesturee)
```

Out[6]: <Axes: >



In [7]:
```
gesturee.isna().sum()
```

Out[7]:
```
Position of left hand (x coordinate)      0
Position of left hand (y coordinate)      0
Position of left hand (z coordinate)      0
Position of right hand (x coordinate)     0
Position of right hand (y coordinate)     0
Position of right hand (z coordinate)     0
Position of head (x coordinate)           0
Position of head (y coordinate)           0
Position of head  (z coordinate)          0
Position of spine (x coordinate)          0
Position of spine (y coordinate)          0
Position of spine (z coordinate)          0
Position of left wrist (x coordinate)     0
Position of left wrist (y coordinate)     0
Position of left wrist (z coordinate)     0
Position of right wrist (x coordinate)    0
Position of right wrist (y coordinate)    0
Position of right wrist (z coordinate)    0
timestamp                                 0
phase                                     0
dtype: int64
```

In [8]:
```
gesturee.duplicated().any()
```

Out[8]:
```
False
```

In [9]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.
3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2
3.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.
2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.
6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.
0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
(1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.
0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.
0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflo
w) (4.9.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (fro
m tensorflow) (0.36.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
60.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
(2.15.1)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.
15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.
0->tensorflow) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<
2.16,>=2.15->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensor
board<2.16,>=2.15->tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=
2.15->tensorflow) (3.5.2)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.1
6,>=2.15->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (fr
om tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=
2.15->tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<
3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<
3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3
->tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-aut
h-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<
3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->t
ensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.2
1.0->tensorboard<2.16,>=2.15->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.2
1.0->tensorboard<2.16,>=2.15->tensorflow) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->
tensorboard<2.16,>=2.15->tensorflow) (2.1.5)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules
>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=
0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
```

In [10]:
```
#encoding categorical variables
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import numpy as np
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

# Sample multi-class labels
labels = ['Rest', 'Preparation', 'Stroke', 'Hold', 'Retraction']

# As DataFrame is named gesturee and the column containing labels is 'phase'
# Create a DataFrame with the labels
data = pd.DataFrame({'phase': labels})

# Apply OneHotEncoder
onehot_encoder = OneHotEncoder(sparse=False)
onehot_encoded = onehot_encoder.fit_transform(gesturee[['phase']])

# Convert the one-hot encoded array into a DataFrame for better visualization
onehot_encoded_df = pd.DataFrame(onehot_encoded, columns=onehot_encoder.get_feature_names_out(['phase']))

# Concatenate the one-hot encoded DataFrame with the original DataFrame
encoded_data = pd.concat([gesturee, onehot_encoded_df], axis=1)
encoded_data
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed
to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse`
to its default value.
  warnings.warn(
```

Out[10]:

	Position of left hand (x coordinate)	Position of left hand (y coordinate)	Position of left hand (z coordinate)	Position of right hand (x coordinate)	Position of right hand (y coordinate)	Position of right hand (z coordinate)	Position of head (x coordinate)	Position of head (y coordinate)	Position of head (z coordinate)	Position of spine (x coordinate)	...	Position right wri coordinat
0	5.347435	4.363681	1.501913	5.258967	4.319263	1.488703	5.037871	1.618295	1.778350	5.062803	...	5.5539
1	4.869622	4.254210	1.556133	5.240113	4.346338	1.554309	5.037610	1.618370	1.778573	5.061430	...	5.4238
2	5.357447	4.364039	1.500969	5.238928	4.347924	1.554150	5.037514	1.618298	1.778774	5.059245	...	5.3321
3	4.942886	4.281878	1.546513	5.111436	4.229660	1.527091	5.037526	1.618612	1.778855	5.056475	...	5.3111
4	5.003160	4.278530	1.542866	4.985812	4.182155	1.520330	5.037557	1.619226	1.778925	5.052367	...	5.1937
...	...	...	...	...	...	...	...	...	...	...	...	
1742	4.999737	3.840355	1.577457	4.990017	4.124610	1.546410	5.092207	1.616004	1.793314	5.094342	...	5.1212
1743	5.001617	3.840771	1.577161	4.695521	4.148509	1.530205	5.092773	1.615753	1.793486	5.094236	...	5.0724
1744	4.996975	3.841236	1.578423	4.700123	4.141193	1.530697	5.092515	1.615095	1.793637	5.093153	...	5.0741
1745	5.000125	3.841455	1.577914	4.690338	4.153884	1.526481	5.092476	1.614777	1.793768	5.093362	...	5.0581
1746	5.007900	3.841846	1.576601	5.004423	4.119151	1.547947	5.092856	1.614567	1.793805	5.093344	...	5.0767

1747 rows × 25 columns

In [11]:
```
encoded_data.tail()
```

Out[11]:

	Position of left hand (x coordinate)	Position of left hand (y coordinate)	Position of left hand (z coordinate)	Position of right hand (x coordinate)	Position of right hand (y coordinate)	Position of right hand (z coordinate)	Position of head (x coordinate)	Position of head (y coordinate)	Position of head (z coordinate)	Position of spine (x coordinate)	...	Position right wri coordinat
1742	4.999737	3.840355	1.577457	4.990017	4.124610	1.546410	5.092207	1.616004	1.793314	5.094342	...	5.1212
1743	5.001617	3.840771	1.577161	4.695521	4.148509	1.530205	5.092773	1.615753	1.793486	5.094236	...	5.0724
1744	4.996975	3.841236	1.578423	4.700123	4.141193	1.530697	5.092515	1.615095	1.793637	5.093153	...	5.0741
1745	5.000125	3.841455	1.577914	4.690338	4.153884	1.526481	5.092476	1.614777	1.793768	5.093362	...	5.0581
1746	5.007900	3.841846	1.576601	5.004423	4.119151	1.547947	5.092856	1.614567	1.793805	5.093344	...	5.0767

5 rows × 25 columns

In [12]:
```
# Calculate the total number of rows
total_rows = len(encoded_data)

# Calculate the index for the middle rows
middle_index = total_rows // 2

# Define how many rows you want to see around the middle
rows_around_middle = 5

# Extract the middle rows along with the rows around them
middle_rows = encoded_data.iloc[middle_index - rows_around_middle:middle_index + rows_around_middle]

# Print the middle rows
print(middle_rows)
```

```
     Position of left hand (x coordinate)  \
868                              4.621050
869                              4.616713
870                              4.613550
871                              4.616341
872                              4.619547
873                              4.622250
874                              4.629506
875                              4.638996
876                              4.647377
877                              4.658780

     Position of left hand (y coordinate)  \
868                              4.990334
869                              4.984185
870                              4.975467
871                              4.959296
872                              4.949488
873                              4.936950
```

In [13]:
```
encoded_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1747 entries, 0 to 1746
Data columns (total 25 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Position of left hand (x coordinate)  1747 non-null   float64
 1   Position of left hand (y coordinate)  1747 non-null   float64
 2   Position of left hand (z coordinate)  1747 non-null   float64
 3   Position of right hand (x coordinate)  1747 non-null   float64
 4   Position of right hand (y coordinate)  1747 non-null   float64
 5   Position of right hand (z coordinate)  1747 non-null   float64
 6   Position of head (x coordinate)       1747 non-null   float64
 7   Position of head (y coordinate)       1747 non-null   float64
 8   Position of head  (z coordinate)      1747 non-null   float64
 9   Position of spine (x coordinate)      1747 non-null   float64
 10  Position of spine (y coordinate)      1747 non-null   float64
 11  Position of spine (z coordinate)      1747 non-null   float64
 12  Position of left wrist (x coordinate)  1747 non-null   float64
 13  Position of left wrist (y coordinate)  1747 non-null   float64
 14  Position of left wrist (z coordinate)  1747 non-null   float64
 15  Position of right wrist (x coordinate)  1747 non-null   float64
 16  Position of right wrist (y coordinate)  1747 non-null   float64
 17  Position of right wrist (z coordinate)  1747 non-null   float64
 18  timestamp                             1747 non-null   int64
 19  phase                                 1747 non-null   object
 20  phase_Hold                            1747 non-null   float64
 21  phase_Preparation                     1747 non-null   float64
 22  phase_Rest                            1747 non-null   float64
 23  phase_Retraction                      1747 non-null   float64
 24  phase_Stroke                          1747 non-null   float64
dtypes: float64(23), int64(1), object(1)
memory usage: 341.3+ KB
```

In [14]:
```
encoded_data.describe()
```

Out[14]:

	Position of left hand (x coordinate)	Position of left hand (y coordinate)	Position of left hand (z coordinate)	Position of right hand (x coordinate)	Position of right hand (y coordinate)	Position of right hand (z coordinate)	Position of head (x coordinate)	Position of head (y coordinate)	Position of head (z coordinate)	Position of spine (x coordinate)	...
count	1747.000000	1747.000000	1747.000000	1747.000000	1747.000000	1747.000000	1747.000000	1747.000000	1747.000000	1747.000000	...
mean	4.733099	4.110179	1.492067	5.400730	3.860213	1.483939	5.084119	1.660346	1.753750	5.099306	...
std	0.706462	0.782359	0.064146	0.807955	0.870350	0.070478	0.181948	0.058347	0.024135	0.126167	...
min	1.712471	1.660291	1.183290	3.653736	1.296046	1.221158	4.435888	1.415457	1.664271	4.737700	...
25%	4.530557	3.698008	1.467770	4.846750	3.274578	1.458293	5.010395	1.632286	1.742879	5.041478	...
50%	4.870256	4.266777	1.487985	5.199142	4.122335	1.487003	5.073397	1.655894	1.757832	5.074966	...
75%	5.153918	4.737138	1.524848	5.817285	4.374306	1.517132	5.167158	1.678649	1.766783	5.141455	...
max	6.212834	5.414431	1.673418	8.546903	5.453834	1.722231	5.605544	1.965343	1.853095	5.586325	...

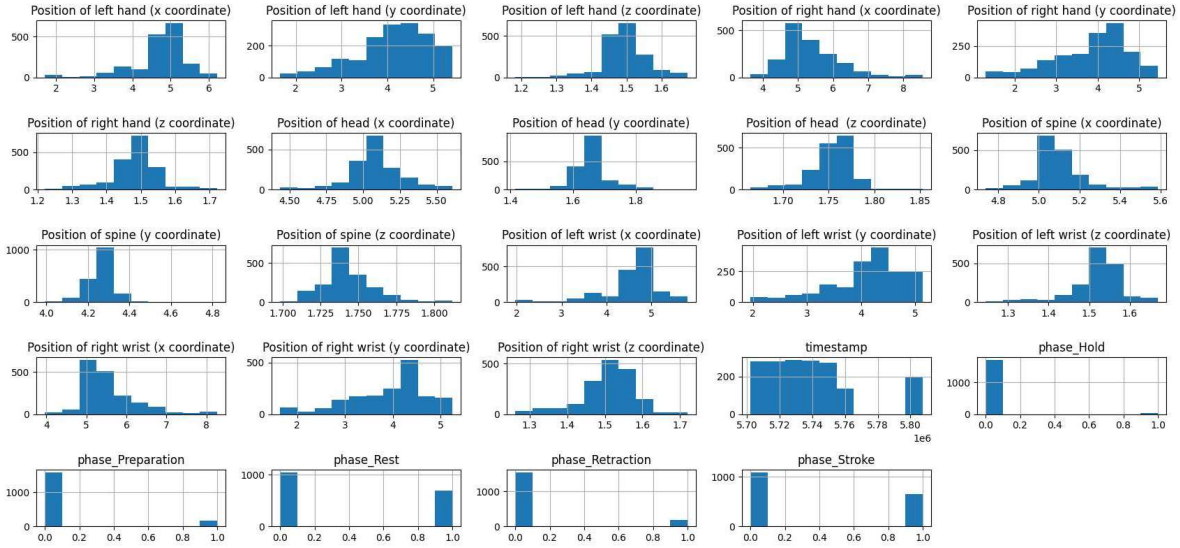
8 rows × 24 columns

In [15]:
```
data_type=encoded_data['phase_Stroke'].dtype
data_type
```

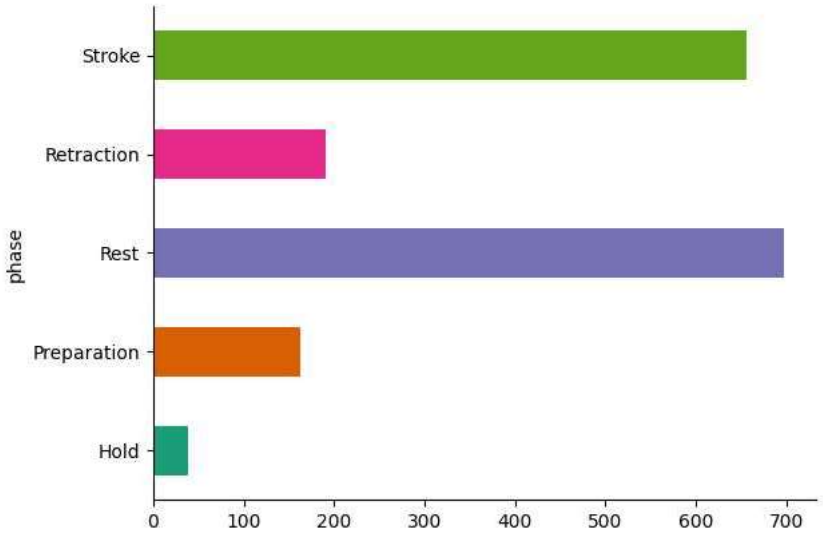
Out[15]:
```
dtype('float64')
```

##Data Visualization

In [16]:
```
#hist plot for distribution
encoded_data.hist(figsize=(17, 8))
plt.tight_layout()
plt.show()
```

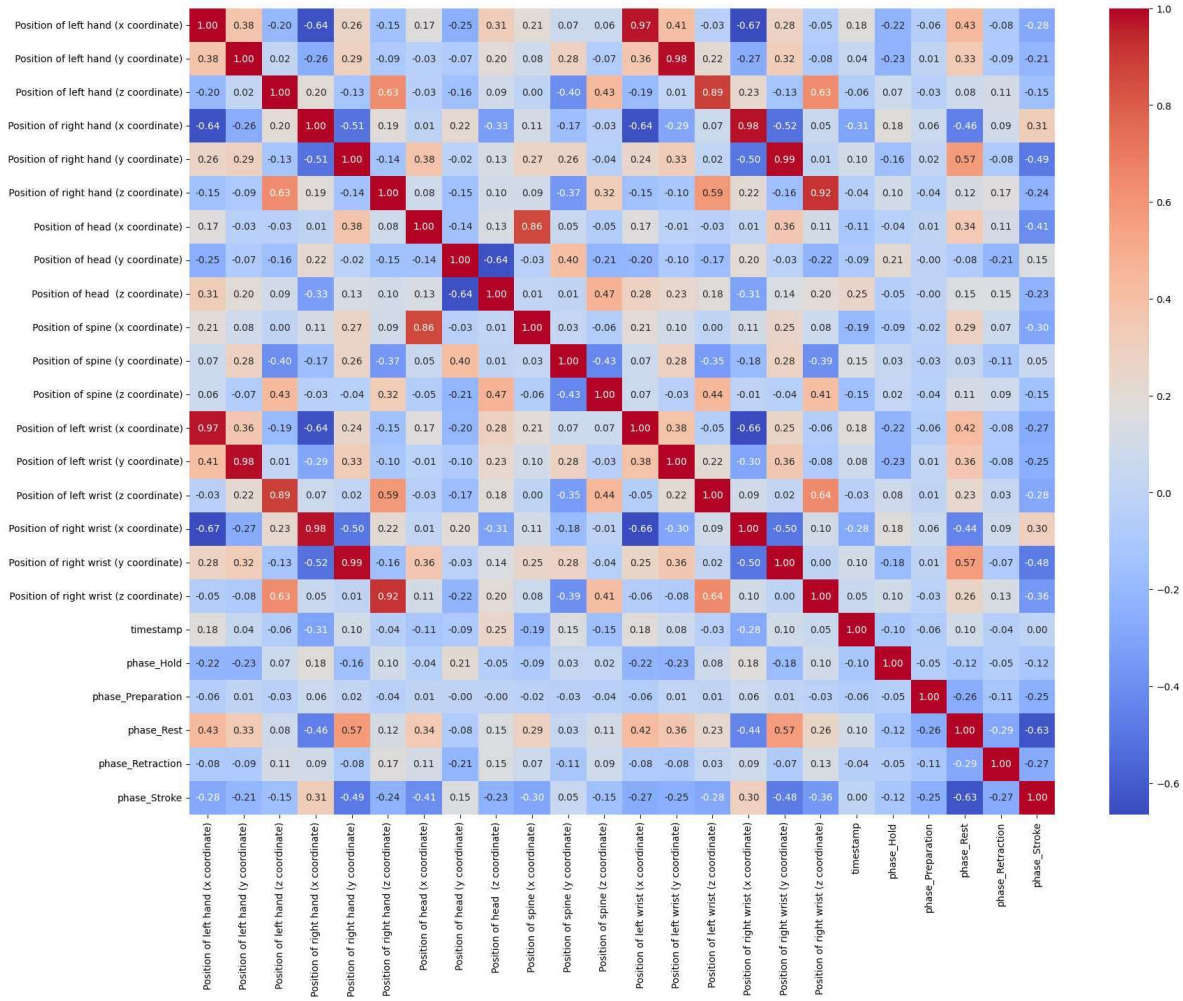


In [17]:
```
from matplotlib import pyplot as plt
import seaborn as sns
encoded_data.groupby('phase').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

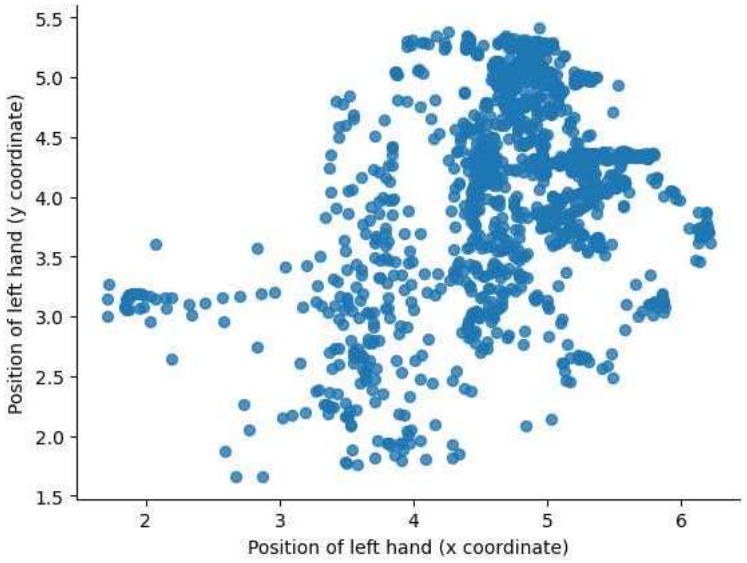


In [18]:
```
plt.figure(figsize=(20, 15))
correlation_matrix = encoded_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```

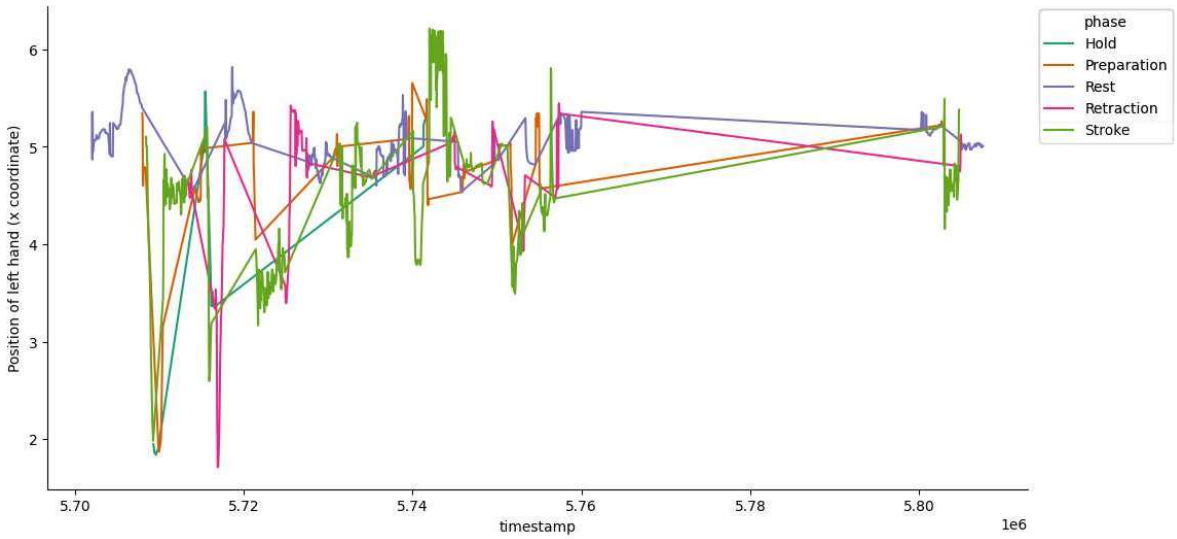
```
<ipython-input-18-47fa41d642ce>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to
silence this warning.
  correlation_matrix = encoded_data.corr()
```



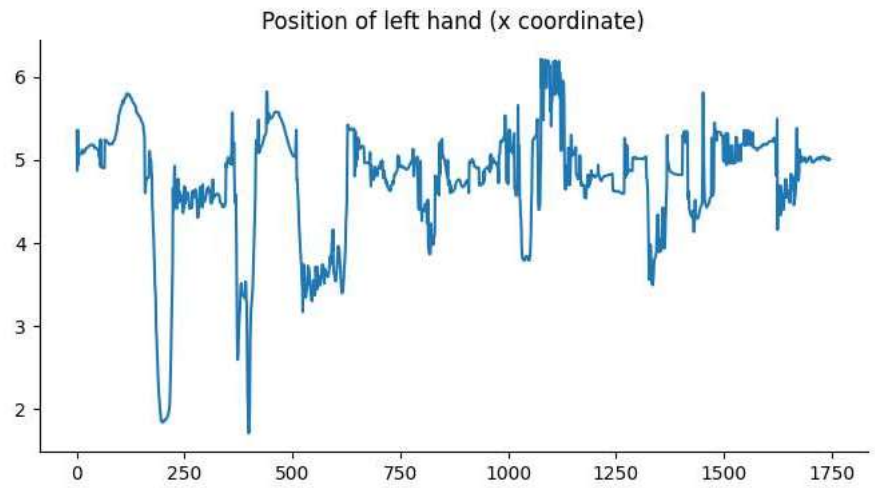
In [20]:
```
from matplotlib import pyplot as plt
encoded_data.plot(kind='scatter', x='Position of left hand (x coordinate)', y='Position of left hand (y coordinate)'
plt.gca().spines[['top', 'right',]].set_visible(False)
```



In [21]:
```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['timestamp']
  ys = series['Position of left hand (x coordinate)']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = encoded_data.sort_values('timestamp', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.groupby('phase')):
  _plot_series(series, series_name, i)
  fig.legend(title='phase', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('timestamp')
_ = plt.ylabel('Position of left hand (x coordinate)')
```

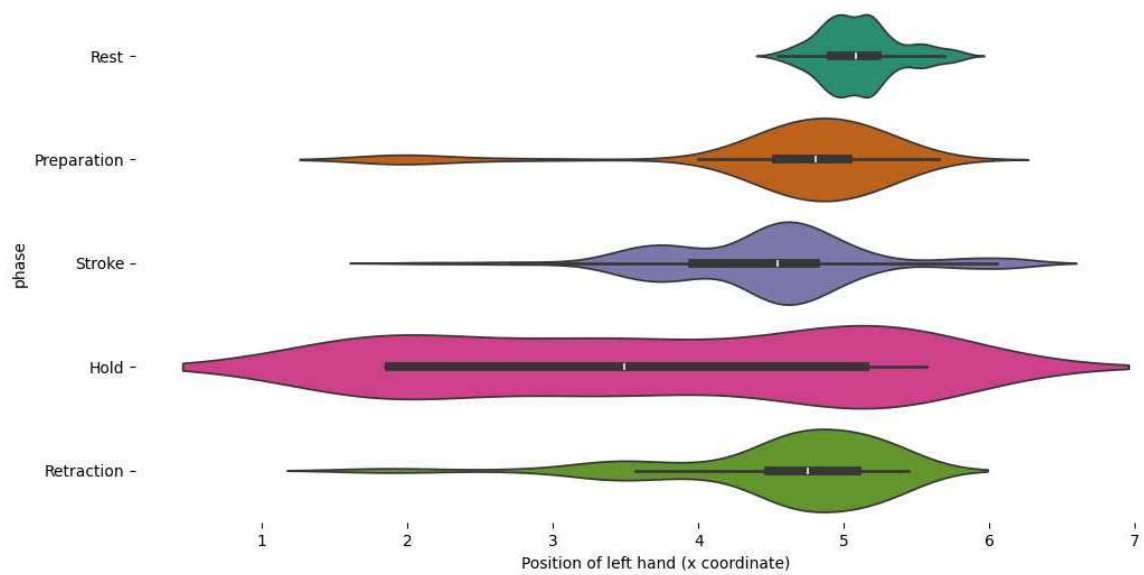


In [22]:
```
from matplotlib import pyplot as plt
encoded_data['Position of left hand (x coordinate)'].plot(kind='line', figsize=(8, 4), title='Position of left hand
plt.gca().spines[['top', 'right']].set_visible(False)
```



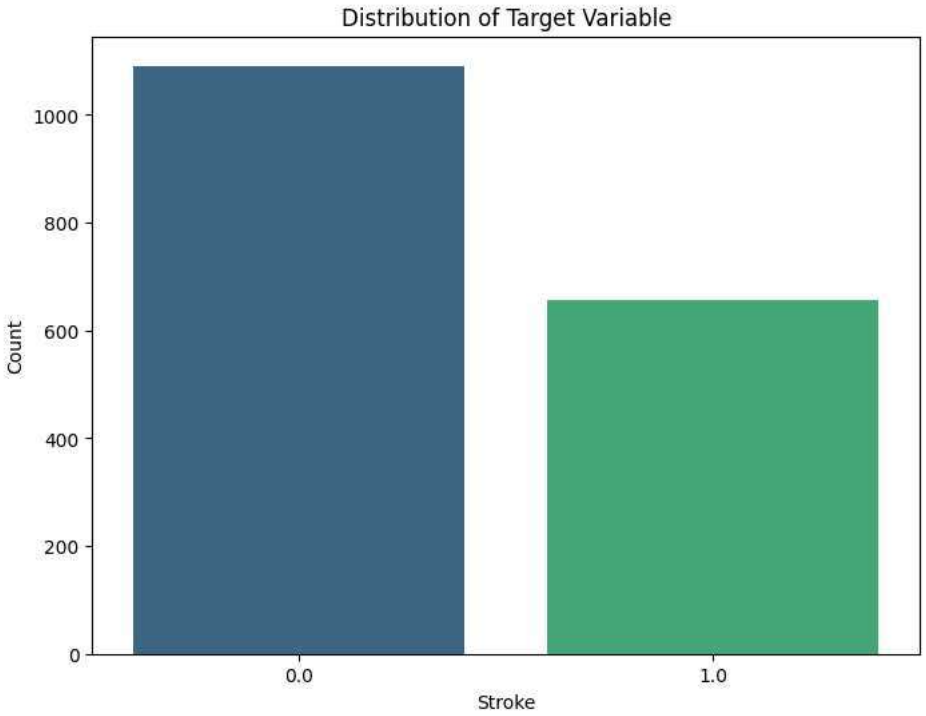
In [23]:
```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(encoded_data['phase'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(encoded_data, x='Position of left hand (x coordinate)', y='phase', inner='box', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<ipython-input-23-e1b71a5d6600>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to
`hue` and set `legend=False` for the same effect.

  sns.violinplot(encoded_data, x='Position of left hand (x coordinate)', y='phase', inner='box', palette='Dark2')
```



In [24]:
```
#Create a bar plot
plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
sns.countplot(x='phase_Stroke', data=encoded_data, palette='viridis')

# Adding labels and title
plt.xlabel('Stroke')
plt.ylabel('Count')
plt.title('Distribution of Target Variable')

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

```
<ipython-input-24-2b1dc091cbfd>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to
`hue` and set `legend=False` for the same effect.

  sns.countplot(x='phase_Stroke', data=encoded_data, palette='viridis')
```



##Data Preprocessing

In [25]:
```
!pip install smote-variants
```

```
Collecting smote-variants
  Downloading smote_variants-0.7.3-py3-none-any.whl (416 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 416.2/416.2 kB 2.9 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from smote-variants) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from smote-variants) (1.11.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from smote-variants)
(1.2.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from smote-variants) (1.3.2)
Collecting minisom (from smote-variants)
  Downloading MiniSom-2.3.1.tar.gz (10 kB)
  Preparing metadata (setup.py) ... done
Collecting statistics (from smote-variants)
  Downloading statistics-1.0.3.5.tar.gz (8.3 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (from smote-variants) (2.
15.0)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (from smote-variants) (2.15.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from smote-variants) (1.5.3)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dist-packages (from smote-variants) (2023.2.0)
```

In [26]:
```
!pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imb
learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imbl
earn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-lear
n->imblearn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imb
learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-lea
rn->imblearn) (3.2.0)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

In [27]:
```
!pip uninstall imblearn --yes
```

```
Found existing installation: imblearn 0.0
Uninstalling imblearn-0.0:
  Successfully uninstalled imblearn-0.0
```

In [28]:
```
from imblearn.over_sampling import SMOTE
```

In [29]:
```
class_distribution = encoded_data['phase_Stroke'].value_counts()
print(class_distribution)
```

```
0.0    1091
1.0     656
Name: phase_Stroke, dtype: int64
```

In [30]:
```
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from collections import Counter
```

In [31]:
```
X=encoded_data.drop(['phase_Stroke','timestamp','phase'],axis=1)
Y=encoded_data['phase_Stroke']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [32]:
```
# Apply SMOTE to balance the classes
smote = SMOTE(random_state=42)
X_train_resampled, Y_train_resampled = smote.fit_resample(X_train, Y_train)
#trained
X_train_resampled_df = pd.DataFrame(X_train_resampled,columns=X_train.columns)
X_train_resampled_df
```

Out[32]:

	Position of left hand (x coordinate)	Position of left hand (y coordinate)	Position of left hand (z coordinate)	Position of right hand (x coordinate)	Position of right hand (y coordinate)	Position of right hand (z coordinate)	Position of head (x coordinate)	Position of head (y coordinate)	Position of head (z coordinate)	Position of spine (x coordinate)	...	Position left wrist coordinat
0	5.573370	4.111267	1.506329	5.574571	4.080083	1.504578	5.437800	1.737993	1.719061	5.586185	...	5.5173
1	5.146826	4.001340	1.516274	4.771055	4.029935	1.474093	5.098891	1.620896	1.742468	5.082981	...	4.9828
2	5.078174	3.253766	1.377435	4.980286	3.603522	1.460554	5.077229	1.684055	1.735441	5.099010	...	5.0239
3	4.424200	2.849155	1.424180	5.424315	2.757930	1.416697	5.053841	1.647448	1.757305	5.002950	...	4.4727
4	4.929639	3.158658	1.450306	4.494451	3.186066	1.433913	5.088625	1.616654	1.753130	5.050774	...	4.7331
...	...	...	...	...	...	...	...	...	...	...	...	
1759	2.105907	3.123798	1.599847	7.968138	3.077407	1.547315	5.057581	1.719756	1.741909	5.045721	...	2.2152
1760	4.337600	5.258980	1.522763	3.792298	3.396854	1.478655	4.490861	1.672748	1.765559	4.744025	...	4.1866
1761	3.805079	1.938642	1.423246	5.235250	4.723276	1.511094	5.266309	1.652931	1.775079	5.162741	...	3.7228
1762	4.623216	4.945906	1.481816	4.058409	2.785064	1.549112	4.446273	1.660911	1.761333	4.907495	...	4.4240
1763	4.618786	4.951815	1.481439	4.429708	3.387508	1.517112	4.438895	1.653688	1.760560	4.914206	...	4.4097

1764 rows × 22 columns

In [33]:
```
#trained y
Y_train_resampled_df = pd.DataFrame(Y_train_resampled, columns=['phase_Stroke'])
Y_train_resampled_df
```

Out[33]:

	phase_Stroke
0	0.0
1	0.0
2	0.0
3	0.0
4	1.0
...	...
1759	1.0
1760	1.0
1761	1.0
1762	1.0
1763	1.0

1764 rows × 1 columns

In [34]:
```
#resampled
resampled_data = pd.concat([X_train_resampled_df, Y_train_resampled_df], axis=1)
resampled_data
```

Out[34]:

	Position of left hand (x coordinate)	Position of left hand (y coordinate)	Position of left hand (z coordinate)	Position of right hand (x coordinate)	Position of right hand (y coordinate)	Position of right hand (z coordinate)	Position of head (x coordinate)	Position of head (y coordinate)	Position of head (z coordinate)	Position of spine (x coordinate)	...	Position left wrist coordinat
0	5.573370	4.111267	1.506329	5.574571	4.080083	1.504578	5.437800	1.737993	1.719061	5.586185	...	4.0820
1	5.146826	4.001340	1.516274	4.771055	4.029935	1.474093	5.098891	1.620896	1.742468	5.082981	...	4.0049
2	5.078174	3.253766	1.377435	4.980286	3.603522	1.460554	5.077229	1.684055	1.735441	5.099010	...	3.5051
3	4.424200	2.849155	1.424180	5.424315	2.757930	1.416697	5.053841	1.647448	1.757305	5.002950	...	2.7228
4	4.929639	3.158658	1.450306	4.494451	3.186066	1.433913	5.088625	1.616654	1.753130	5.050774	...	3.3564
...	...	...	...	...	...	...	...	...	...	...	...	
1759	2.105907	3.123798	1.599847	7.968138	3.077407	1.547315	5.057581	1.719756	1.741909	5.045721	...	3.2812
1760	4.337600	5.258980	1.522763	3.792298	3.396854	1.478655	4.490861	1.672748	1.765559	4.744025	...	5.0127
1761	3.805079	1.938642	1.423246	5.235250	4.723276	1.511094	5.266309	1.652931	1.775079	5.162741	...	2.2074
1762	4.623216	4.945906	1.481816	4.058409	2.785064	1.549112	4.446273	1.660911	1.761333	4.907495	...	4.7472
1763	4.618786	4.951815	1.481439	4.429708	3.387508	1.517112	4.438895	1.653688	1.760560	4.914206	...	4.7294

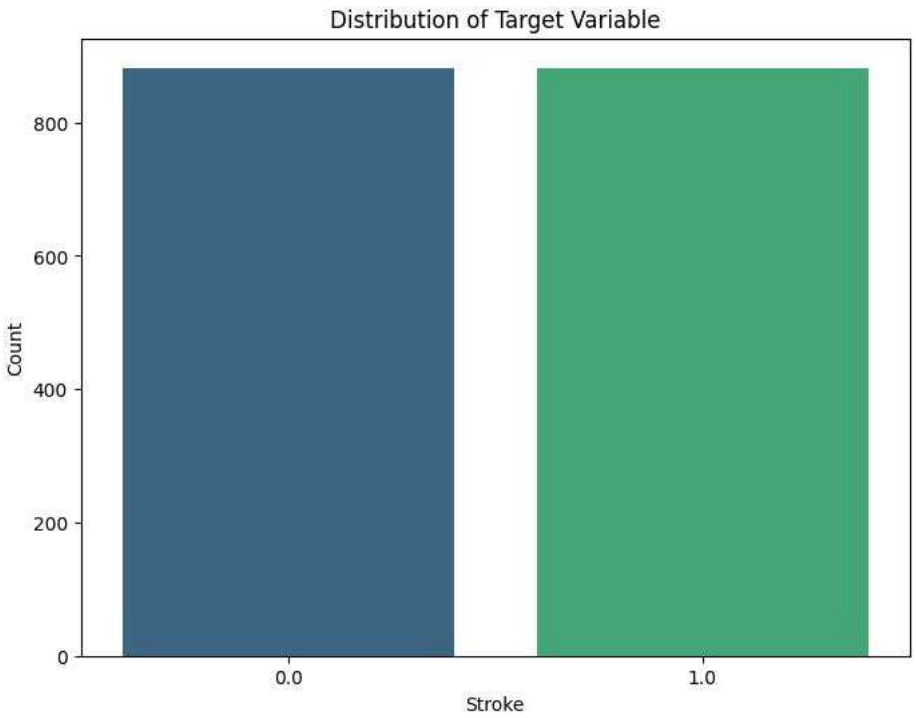
1764 rows × 23 columns

In [35]:
```
class_distributionn = resampled_data['phase_Stroke'].value_counts()
print(class_distributionn)
```

```
0.0    882
1.0    882
Name: phase_Stroke, dtype: int64
```

In [36]:
```
#Create a bar plot
plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
sns.countplot(x='phase_Stroke', data=resampled_data, palette='viridis')

# Adding labels and title
plt.xlabel('Stroke')
plt.ylabel('Count')
plt.title('Distribution of Target Variable')

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

```
<ipython-input-36-8cdc4efce304>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to
`hue` and set `legend=False` for the same effect.

  sns.countplot(x='phase_Stroke', data=resampled_data, palette='viridis')
```



##Feature Selection

In [37]:
```
X=resampled_data.drop(['phase_Stroke','phase_Hold','phase_Retraction','phase_Preparation','phase_Rest'],axis=1)
Y=resampled_data['phase_Stroke']
```

In [38]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
```

In [39]:
```
from sklearn.feature_selection import SelectKBest,chi2,mutual_info_classif
from scipy import stats
```

Information Gain

In [40]:
```
#using information gain
information_gain=mutual_info_classif(X,Y)
information_gain
```

Out[40]:
```
array([0.2549209 , 0.1893521 , 0.14633816, 0.24623102, 0.25008097,
       0.14217308, 0.19945759, 0.06956684, 0.10162961, 0.12860527,
       0.07315902, 0.09063639, 0.26754807, 0.23311034, 0.13046684,
       0.24464837, 0.22086824, 0.11624488])
```

Chi-squared test

In [41]:
```
chi_square=SelectKBest(score_func=chi2,k='all').fit(X,Y).scores_
chi_square
```

Out[41]:
```
array([1.59421630e+01, 1.39760268e+01, 1.65321999e-01, 2.06019842e+01,
       8.53357963e+01, 4.03944231e-01, 2.24683372e+00, 8.09755562e-02,
       3.25833544e-02, 5.89547141e-01, 7.29716230e-03, 8.07617723e-03,
       1.11741402e+01, 1.42178411e+01, 4.53935732e-01, 1.35481549e+01,
       6.53447037e+01, 8.51672749e-01])
```

Missing value ratio

In [42]:
```
missing_values=X.isnull().sum()
missingvalue_ratio=missing_values/len(X)
```

In [43]:
```
results=pd.DataFrame({'Feature':X.columns,'Info gain':information_gain,'chi_Square_score':chi_square,
                      'Missing_values':missingvalue_ratio})
results
```

Out[43]:

	Feature	Info gain	chi_Square_score	Missing_values
Position of left hand (x coordinate)	Position of left hand (x coordinate)	0.254921	15.942163	0.0
Position of left hand (y coordinate)	Position of left hand (y coordinate)	0.189352	13.976027	0.0
Position of left hand (z coordinate)	Position of left hand (z coordinate)	0.146338	0.165322	0.0
Position of right hand (x coordinate)	Position of right hand (x coordinate)	0.246231	20.601984	0.0
Position of right hand (y coordinate)	Position of right hand (y coordinate)	0.250081	85.335796	0.0
Position of right hand (z coordinate)	Position of right hand (z coordinate)	0.142173	0.403944	0.0
Position of head (x coordinate)	Position of head (x coordinate)	0.199458	2.246834	0.0
Position of head (y coordinate)	Position of head (y coordinate)	0.069567	0.080976	0.0
Position of head (z coordinate)	Position of head (z coordinate)	0.101630	0.032583	0.0
Position of spine (x coordinate)	Position of spine (x coordinate)	0.128605	0.589547	0.0
Position of spine (y coordinate)	Position of spine (y coordinate)	0.073159	0.007297	0.0
Position of spine (z coordinate)	Position of spine (z coordinate)	0.090636	0.008076	0.0
Position of left wrist (x coordinate)	Position of left wrist (x coordinate)	0.267548	11.174140	0.0
Position of left wrist (y coordinate)	Position of left wrist (y coordinate)	0.233110	14.217841	0.0
Position of left wrist (z coordinate)	Position of left wrist (z coordinate)	0.130467	0.453936	0.0
Position of right wrist (x coordinate)	Position of right wrist (x coordinate)	0.244648	13.548155	0.0
Position of right wrist (y coordinate)	Position of right wrist (y coordinate)	0.220868	65.344704	0.0
Position of right wrist (z coordinate)	Position of right wrist (z coordinate)	0.116245	0.851673	0.0

Also, sorting by information gain to see the results in a particular order.

In [44]:
```
sorted_results=results.sort_values(by='Info gain',ascending=False)
print(sorted_results)
```

```
                                                                Feature  \
Position of left wrist (x coordinate)    Position of left wrist (x coordinate)
Position of left hand (x coordinate)      Position of left hand (x coordinate)
Position of right hand (y coordinate)    Position of right hand (y coordinate)
Position of right hand (x coordinate)    Position of right hand (x coordinate)
Position of right wrist (x coordinate)  Position of right wrist (x coordinate)
Position of left wrist (y coordinate)    Position of left wrist (y coordinate)
Position of right wrist (y coordinate)  Position of right wrist (y coordinate)
Position of head (x coordinate)                Position of head (x coordinate)
Position of left hand (y coordinate)      Position of left hand (y coordinate)
Position of left hand (z coordinate)      Position of left hand (z coordinate)
Position of right hand (z coordinate)    Position of right hand (z coordinate)
Position of left wrist (z coordinate)    Position of left wrist (z coordinate)
Position of spine (x coordinate)              Position of spine (x coordinate)
Position of right wrist (z coordinate)  Position of right wrist (z coordinate)
Position of head  (z coordinate)              Position of head  (z coordinate)
Position of spine (z coordinate)              Position of spine (z coordinate)
Position of spine (y coordinate)              Position of spine (y coordinate)
Position of head (y coordinate)                Position of head (y coordinate)

                                        Info gain  chi_Square_score  \
Position of left wrist (x coordinate)    0.267548         11.174140
Position of left hand (x coordinate)     0.254921         15.942163
Position of right hand (y coordinate)    0.250081         85.335796
Position of right hand (x coordinate)    0.246231         20.601984
Position of right wrist (x coordinate)   0.244648         13.548155
Position of left wrist (y coordinate)    0.233110         14.217841
Position of right wrist (y coordinate)   0.220868         65.344704
Position of head (x coordinate)          0.199458          2.246834
Position of left hand (y coordinate)     0.189352         13.976027
Position of left hand (z coordinate)     0.146338          0.165322
Position of right hand (z coordinate)    0.142173          0.403944
Position of left wrist (z coordinate)    0.130467          0.453936
Position of spine (x coordinate)         0.128605          0.589547
Position of right wrist (z coordinate)   0.116245          0.851673
Position of head  (z coordinate)         0.101630          0.032583
Position of spine (z coordinate)         0.090636          0.008076
Position of spine (y coordinate)         0.073159          0.007297
Position of head (y coordinate)          0.069567          0.080976

                                        Missing_values
Position of left wrist (x coordinate)              0.0
Position of left hand (x coordinate)               0.0
Position of right hand (y coordinate)              0.0
Position of right hand (x coordinate)              0.0
Position of right wrist (x coordinate)             0.0
Position of left wrist (y coordinate)              0.0
Position of right wrist (y coordinate)             0.0
Position of head (x coordinate)                    0.0
Position of left hand (y coordinate)               0.0
Position of left hand (z coordinate)               0.0
Position of right hand (z coordinate)              0.0
Position of left wrist (z coordinate)              0.0
Position of spine (x coordinate)                   0.0
Position of right wrist (z coordinate)             0.0
Position of head  (z coordinate)                   0.0
Position of spine (z coordinate)                   0.0
Position of spine (y coordinate)                   0.0
Position of head (y coordinate)                    0.0
```

#Model Selection

In [45]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, BaggingClassifier, StackingClassifi
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

In [46]:
```
selector=SelectKBest(score_func=chi2,k=10)
X_train_selected=selector.fit_transform(X_train,Y_train)
X_test_selected=selector.transform(X_test)
```

In [47]:
```
lr_model=LogisticRegression(
    penalty='l2',#regularization type
    C=1.0,#Inverse of regularization strength
    fit_intercept=True,#whether to include intercept
    random_state=None,#seed for random number generator
    solver='lbfgs',#optimization algorithm
    max_iter=1000,#maximum iteration for the solver
    multi_class='auto',#binary classification - One Vs Rest otherwise multinomial
    verbose=0,#control the message of the model
    warm_start=False,#Reuse previous solution
    n_jobs=None,#number of CPU cores used
    l1_ratio=None#ElasticNet mixing parameter of L1 and L2
)
```

In [48]:
```
lr_model.fit(X_train_selected,Y_train)
```

Out[48]:
```
LogisticRegression(max_iter=1000)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [49]:
```
y_pred=lr_model.predict(X_test_selected)
```

In [50]:
```
accuracy_score(Y_test,y_pred)
```

Out[50]:
```
0.8470254957507082
```

In [51]:
```
print(classification_report(Y_test,y_pred))
```

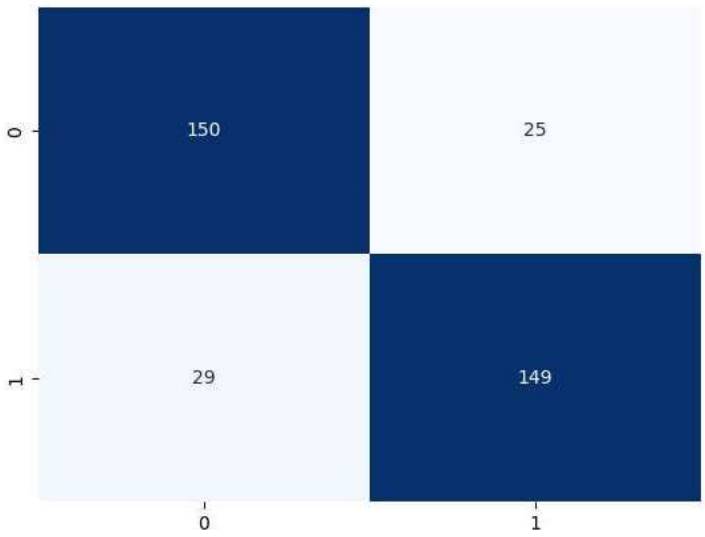
```
              precision    recall  f1-score   support

         0.0       0.84      0.86      0.85       175
         1.0       0.86      0.84      0.85       178

    accuracy                           0.85       353
   macro avg       0.85      0.85      0.85       353
weighted avg       0.85      0.85      0.85       353
```

In [52]:
```
cm=confusion_matrix(Y_test,y_pred)
cm
```

Out[52]:
```
array([[150,  25],
       [ 29, 149]])
```

In [53]:
```
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', cbar=False)
```

Out[53]:
```
<Axes: >
```



The logistic regression model achieved an accuracy of 84.70% on the test dataset, demonstrating its effectiveness in predicting outcomes. With precision scores of 84% for class 0 and 86% for class 1, the model accurately identifies true positive instances. Additionally, recall rates of 86% for class 0 and 84% for class 1 indicate its ability to capture most positive instances. The F1-score, a balance between precision and recall, is 85% for both classes, highlighting the model's robust performance. In summary, the logistic regression model shows balanced predictive power across classes, making it a reliable tool for classification tasks.

In [54]:
```
rf_model=RandomForestClassifier(n_estimators=100,
                      criterion='gini',
                          max_depth=None,
                          min_samples_split=2,
                          min_samples_leaf=1,
                          max_features='sqrt',
                          bootstrap=True,
                          random_state=None
)
```

In [55]:
```
rf_model.fit(X_train_selected,Y_train)
```

Out[55]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [56]:
```
rf_pred=rf_model.predict(X_test_selected)
```

In [57]:
```
accuracy_score(Y_test,rf_pred)
```

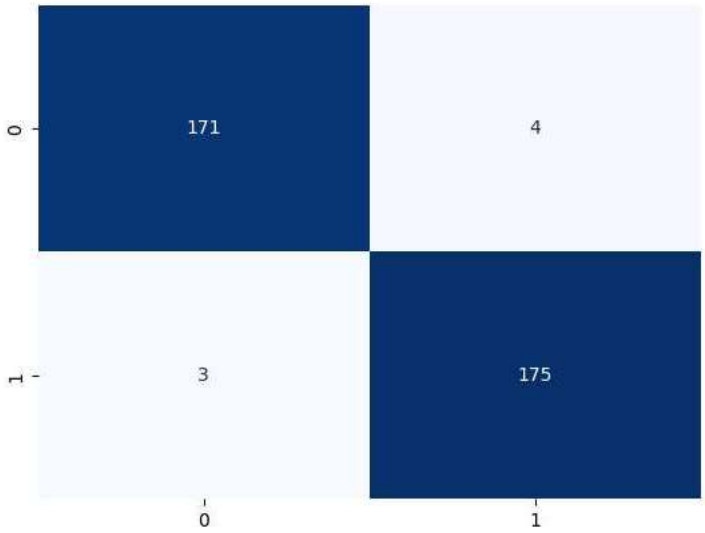
Out[57]:
```
0.9801699716713881
```

In [58]:
```
cmm=confusion_matrix(Y_test,rf_pred)
cmm
```

Out[58]:
```
array([[171,   4],
       [  3, 175]])
```

In [59]:
```
sns.heatmap(cmm,annot=True,cmap='Blues',fmt='g',cbar=False)
```

Out[59]:
```
<Axes: >
```



In [60]:
```
print(classification_report(Y_test,rf_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98       175
         1.0       0.98      0.98      0.98       178

    accuracy                           0.98       353
   macro avg       0.98      0.98      0.98       353
weighted avg       0.98      0.98      0.98       353
```

The RandomForestClassifier achieved an outstanding accuracy score of 98.02% on the test dataset, indicating its remarkable performance in predicting outcomes. With precision scores of 99% for class 0 and 97% for class 1, the model effectively identifies true positive instances. Furthermore, recall rates of 97% for class 0 and 99% for class 1 underscore its ability to capture the majority of positive instances. The F1-score, which harmonizes precision and recall, is 98% for both classes, showcasing the model's balanced predictive capability. The confusion matrix reveals a high level of accuracy, with only a few misclassifications (5 false negatives and 2 false positives), reinforcing the model's reliability and robustness in classification tasks.

In [61]:
```
feature_names=X.columns
dt_model=DecisionTreeClassifier(criterion='entropy',
    splitter='best',
    max_depth=5,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features=None,
    random_state=None)
```

In [62]:
```
dt_model.fit(X_train_selected,Y_train)
```

Out[62]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

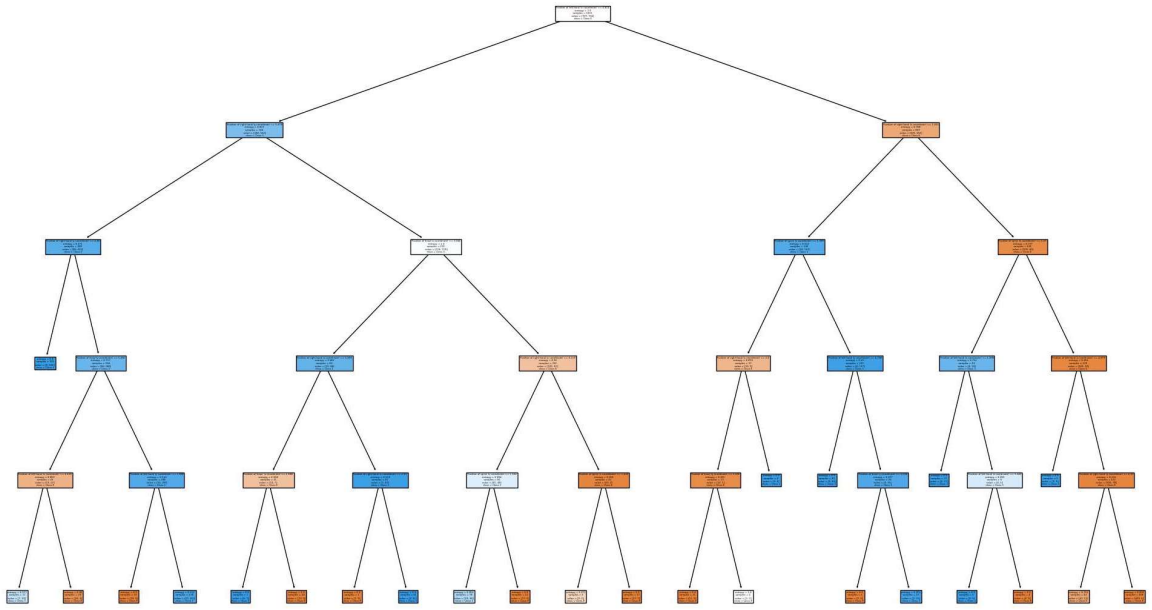
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [63]:
```
dt_pred=dt_model.predict(X_test_selected)
```

In [64]:
```
accuracy_score(Y_test,dt_pred)
```

Out[64]:
```
0.9036827195467422
```

In [65]:
```
plt.figure(figsize=(25,15))  # Adjust the figure size as needed
plot_tree(dt_model, filled=True, feature_names=X.columns, class_names=['Class 0', 'Class 1'])
plt.savefig('decision_tree.png', dpi=300)
plt.show()
```



In [66]:
```
print(classification_report(Y_test,dt_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.89      0.90       175
         1.0       0.90      0.92      0.91       178

    accuracy                           0.90       353
   macro avg       0.90      0.90      0.90       353
weighted avg       0.90      0.90      0.90       353
```

In [67]:
```
dt_cm=confusion_matrix(Y_test,dt_pred)
dt_cm
```

Out[67]:
```
array([[156,  19],
       [ 15, 163]])
```

The Decision Tree model exhibits commendable performance, achieving an accuracy score of 90.65% on the test dataset. With precision scores of 92% for class 0 and 90% for class 1, the model effectively identifies true positive instances for both classes. Similarly, recall rates of 89% for class 0 and 92% for class 1 demonstrate the model's ability to capture the majority of positive instances while minimizing false negatives and false positives. The balanced F1-score of 91% for both classes reflects the model's harmonized precision and recall. The confusion matrix further validates the model's effectiveness, with only a few misclassifications (19 false positives and 14 false negatives), highlighting its reliability in classification tasks.

In [68]:
```
sv_model=SVC(C=1.0,
        kernel='linear',
        degree=3,
        gamma='scale',
        coef0=0.0,
        shrinking=True,
        probability=True,
        tol=1e-3,
        cache_size=200,
        class_weight=None,
        verbose=False,
        max_iter=1000,
        decision_function_shape='ovr',
        break_ties=False,
        random_state=None)
```

In [69]:
```
sv_model.fit(X_train_selected,Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:299: ConvergenceWarning: Solver terminated early (max_
iter=1000).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
```

Out[69]:
```
SVC(kernel='linear', max_iter=1000, probability=True)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [70]:
```
sv_pred=sv_model.predict(X_test_selected)
```

In [71]:
```
accuracy_score(Y_test,sv_pred)
```

Out[71]:
```
0.8555240793201133
```

In [72]:
```
print(classification_report(Y_test,sv_pred))
```

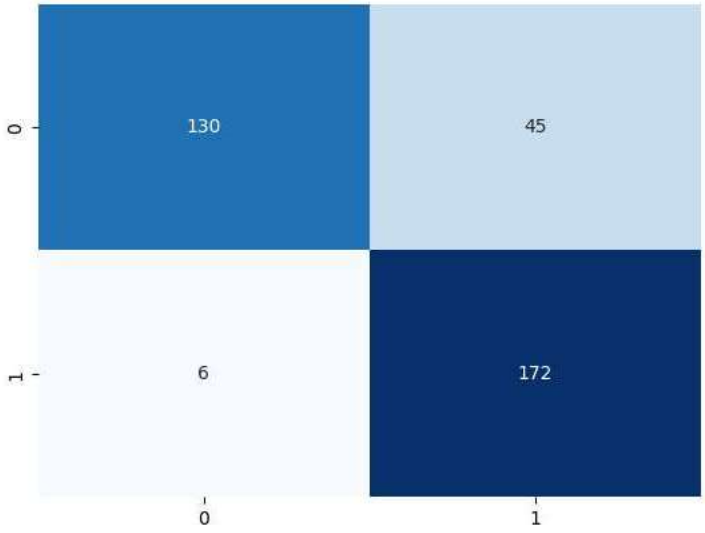
```
              precision    recall  f1-score   support

         0.0       0.96      0.74      0.84       175
         1.0       0.79      0.97      0.87       178

    accuracy                           0.86       353
   macro avg       0.87      0.85      0.85       353
weighted avg       0.87      0.86      0.85       353
```

In [73]:
```
sv_cm=confusion_matrix(Y_test,sv_pred)
sv_cm
```

Out[73]:
```
array([[130,  45],
       [  6, 172]])
```

In [74]:
```
sns.heatmap(sv_cm,annot=True,cmap='Blues',fmt='g',cbar=False)
```

Out[74]: <Axes: >

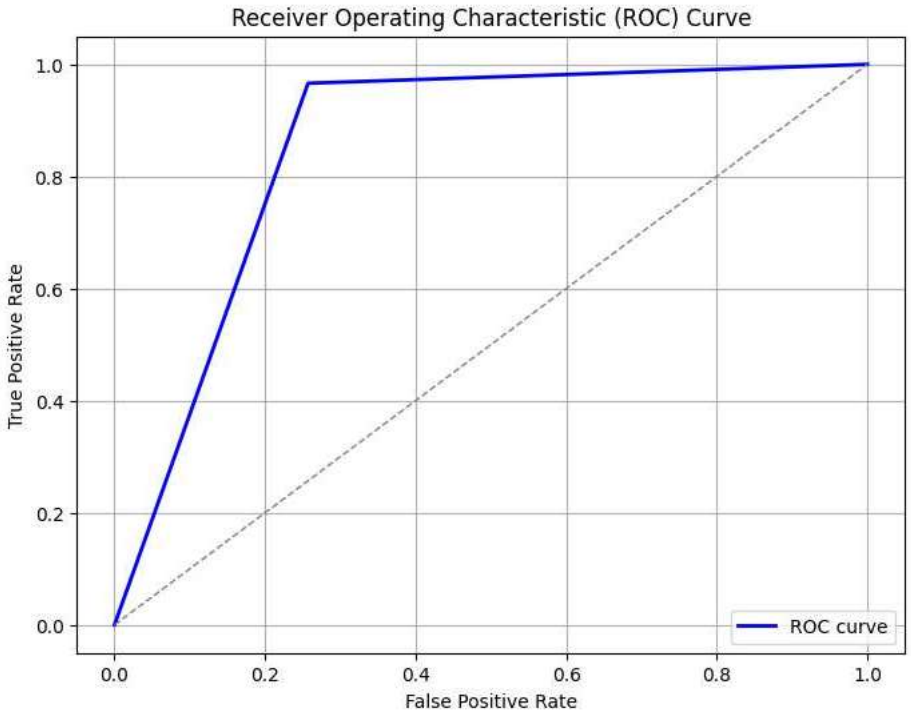


In [75]:
```
from sklearn.metrics import roc_curve, auc, roc_auc_score
fpr,tpr,thresholds=roc_curve(Y_test,sv_pred)
```

In [76]:
```
auc_roc=roc_auc_score(Y_test,sv_pred)
auc_roc
```

Out[76]: 0.8545746388443018

In [77]:
```
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve')
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')  # Diagonal line for random model
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



The Support Vector Machine (SVM) model achieves a respectable accuracy score of 85.55% on the test dataset, indicating its ability to effectively classify instances. The precision and recall metrics reveal a balanced performance, with a precision score of 96% for class 0 and 79% for class 1, and recall rates of 74% for class 0 and 97% for class 1. This suggests that while the model accurately identifies a high proportion of positive instances for class 1, it also exhibits a strong ability to minimize false positives for class 0. The overall F1-score of 87% reflects the model's robustness in balancing precision and recall across both classes. The confusion matrix further confirms the model's reliability, with only a limited number of misclassifications (45 false positives and 6 false negatives), underscoring its effectiveness in classification tasks.

The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve for the Support Vector Machine (SVM) model is determined to be 0.8546. This metric is indicative of the model's ability to distinguish between the positive and negative classes. With an AUC-ROC score close to 1, the SVM model demonstrates strong discriminatory power and performs notably well in classifying instances. A higher AUC value suggests that the model effectively ranks positive instances higher than negative instances, which is crucial for binary classification tasks. In practical terms, an AUC-ROC score of 0.8546 indicates that the SVM model has a high probability of assigning a higher predicted probability to positive instances compared to negative instances. This underscores the model's reliability and effectiveness in making accurate predictions.

In [78]:
```
sc=StandardScaler()
X_scaled=sc.fit_transform(X)
```

In [79]:
```
x_train,x_test,y_train,y_test=train_test_split(X_scaled,Y,test_size=0.2,random_state=42)
```

In [80]:
```
decision=DecisionTreeClassifier()
```

In [81]:
```
from sklearn.model_selection import cross_val_score
scores=cross_val_score(decision,X,Y,cv=5)
scores
```

Out[81]:
```
array([0.92917847, 0.95184136, 0.94617564, 0.95467422, 0.96022727])
```

In [82]:
```
scores.mean()
```

Out[82]:
```
0.9484193922225084
```

In [83]:
```
bag_model=bag_model=BaggingClassifier(
    base_estimator=DecisionTreeClassifier(),
    n_estimators=100,
    max_samples=50,
    bootstrap=True,
    oob_score=True,
    random_state=0
)
```

In [84]:
```
bag_model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed
to `estimator` in version 1.2 and will be removed in 1.4.
  warnings.warn(
```

Out[84]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), max_samples=50,
                  n_estimators=100, oob_score=True, random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [85]:
```
bag_model.score(x_test,y_test)
```

Out[85]:
```
0.9121813031161473
```

In [86]:
```
bag_pred=bag_model.predict(x_test)
```

In [87]:
```
accuracy_score(bag_pred,y_test)
```

Out[87]:
```
0.9121813031161473
```

In [88]:
```
bag_model.oob_score_
```

Out[88]:
```
0.9036144578313253
```

In [89]:
```
boost_model=GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42)
```

In [90]:
```
boost_model.fit(x_train,y_train)
```

Out[90]:
```
GradientBoostingClassifier(random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [91]:
```
boost_pred=boost_model.predict(x_test)
```

In [92]:
```
accuracy_score(y_test,boost_pred)
```

Out[92]:
```
0.9801699716713881
```

In [93]:
```
base_model=[('rf',
             RandomForestClassifier(n_estimators=100,random_state=42)),
            ('lr',LogisticRegression()),('sv',SVC())]


meta_model=GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42)
```

In [94]:
```
stack_model=StackingClassifier(estimators=base_model,
                               final_estimator=meta_model,
                               cv=5)
```

In [95]:
```
stack_model.fit(x_train,y_train)
```

Out[95]:
```
StackingClassifier(cv=5,
                   estimators=[('rf', RandomForestClassifier(random_state=42)),
                               ('lr', LogisticRegression()), ('sv', SVC())],
                   final_estimator=GradientBoostingClassifier(random_state=42))
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [96]:
```
stack_pred=stack_model.predict(x_test)
```

In [97]:
```
accuracy_score(y_test,stack_pred)
```

Out[97]:
```
0.9801699716713881
```

##Dimensionality Reduction

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in machine learning and data analysis. Its primary objective is to identify the most significant patterns and structures in high-dimensional datasets by transforming the original variables into a new set of orthogonal variables called principal components. PCA achieves this by capturing the maximum variance in the data while minimizing information loss.

In [98]:
```
#principal component analysis
from sklearn.decomposition import PCA
pca=PCA()
```

In [99]:
```
pca.fit(X_scaled)
```

Out[99]:
```
PCA()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [100]:
```
cumulative_variance=pca.explained_variance_ratio_.cumsum()
cumulative_variance
```

Out[100]:
```
array([0.25017746, 0.46883449, 0.59532831, 0.69861576, 0.78639421,
       0.85041599, 0.89308702, 0.93171064, 0.959599  , 0.97763208,
       0.98420778, 0.98998875, 0.99404731, 0.99684479, 0.99795276,
       0.99887943, 0.99955593, 1.        ])
```

In [101]:
```
cumulative_variance>=0.95
```

Out[101]:
```
array([False, False, False, False, False, False, False, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True])
```

In [102]:
```
(cumulative_variance>=0.95).argmax()
```

Out[102]:
```
8
```

In [103]:
```
n_components=(cumulative_variance>=0.95).argmax()
n_components
```

Out[103]:
```
8
```

In [104]:
```
#finding the eigen values
pca=PCA(n_components=8)
pca.fit(X_scaled)
prop_var=pca.explained_variance_ratio_
prop_var
```

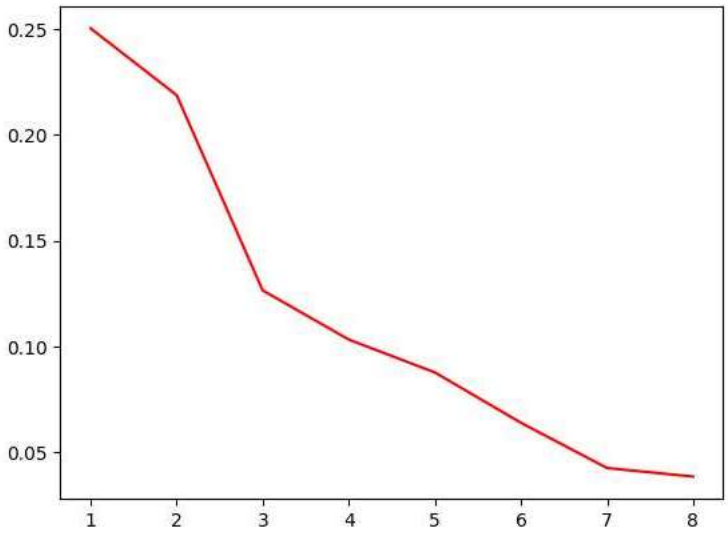
Out[104]:
```
array([0.25017746, 0.21865703, 0.12649382, 0.10328744, 0.08777845,
       0.06402179, 0.04267103, 0.03862362])
```

In [105]:
```
pc_number=np.arange(n_components)+1
pc_number
```

Out[105]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [106]:
```
plt.plot(pc_number,prop_var,'r-')
```

Out[106]:
```
[<matplotlib.lines.Line2D at 0x7b6c0504e680>]
```



In [107]:
```
pca=PCA(n_components=2)
X_data=pca.fit(X_scaled)
```

In [108]:
```
features_names=X.columns
loading=pca.components_
pc1_features=pd.Series(loading[0],index=features_names)
pc1_features
```

Out[108]:
```
Position of left hand (x coordinate)     -0.351266
Position of left hand (y coordinate)     -0.232301
Position of left hand (z coordinate)      0.175273
Position of right hand (x coordinate)     0.376812
Position of right hand (y coordinate)    -0.299418
Position of right hand (z coordinate)     0.183817
Position of head (x coordinate)          -0.083625
Position of head (y coordinate)           0.066129
Position of head  (z coordinate)         -0.150557
Position of spine (x coordinate)         -0.069115
Position of spine (y coordinate)         -0.188024
Position of spine (z coordinate)          0.053017
Position of left wrist (x coordinate)    -0.346580
Position of left wrist (y coordinate)    -0.251253
Position of left wrist (z coordinate)     0.086328
Position of right wrist (x coordinate)    0.383922
Position of right wrist (y coordinate)   -0.307410
Position of right wrist (z coordinate)    0.130120
dtype: float64
```

In [109]:
```
pc2_features=pd.Series(loading[1],index=features_names)
pc2_features
```

Out[109]:
```
Position of left hand (x coordinate)     -0.137415
Position of left hand (y coordinate)     -0.119295
Position of left hand (z coordinate)     -0.382267
Position of right hand (x coordinate)     0.080559
Position of right hand (y coordinate)    -0.018457
Position of right hand (z coordinate)    -0.362671
Position of head (x coordinate)          -0.076253
Position of head (y coordinate)           0.267613
Position of head  (z coordinate)         -0.261047
Position of spine (x coordinate)         -0.075754
Position of spine (y coordinate)          0.256615
Position of spine (z coordinate)         -0.324427
Position of left wrist (x coordinate)    -0.125220
Position of left wrist (y coordinate)    -0.124888
Position of left wrist (z coordinate)    -0.410459
Position of right wrist (x coordinate)    0.065257
Position of right wrist (y coordinate)   -0.018144
Position of right wrist (z coordinate)   -0.394508
dtype: float64
```

In [110]:
```
X_data
```

Out[110]:
```
PCA(n_components=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [111]:
```
x_final_data=pca.fit_transform(X_scaled)
x_final_data
```

Out[111]:
```
array([[-1.10284681, -0.62146963],
       [-0.61695996, -1.65164277],
       [-0.87976728,  1.99420615],
       ...,
       [ 0.9653621 ,  0.93120011],
       [-0.55038986, -0.59082961],
       [-0.82710574, -0.16662502]])
```

In [112]:
```
segmentation=pd.DataFrame(x_final_data,columns=['PC1','PC2'])
segmentation
```

Out[112]:

	PC1	PC2
0	-1.102847	-0.621470
1	-0.616960	-1.651643
2	-0.879767	1.994206
3	1.336836	1.938363
4	-0.367061	0.269400
...	...	...
1759	7.139544	-0.420576
1760	-1.170941	-0.504925
1761	0.965362	0.931200
1762	-0.550390	-0.590830
1763	-0.827106	-0.166625

1764 rows × 2 columns

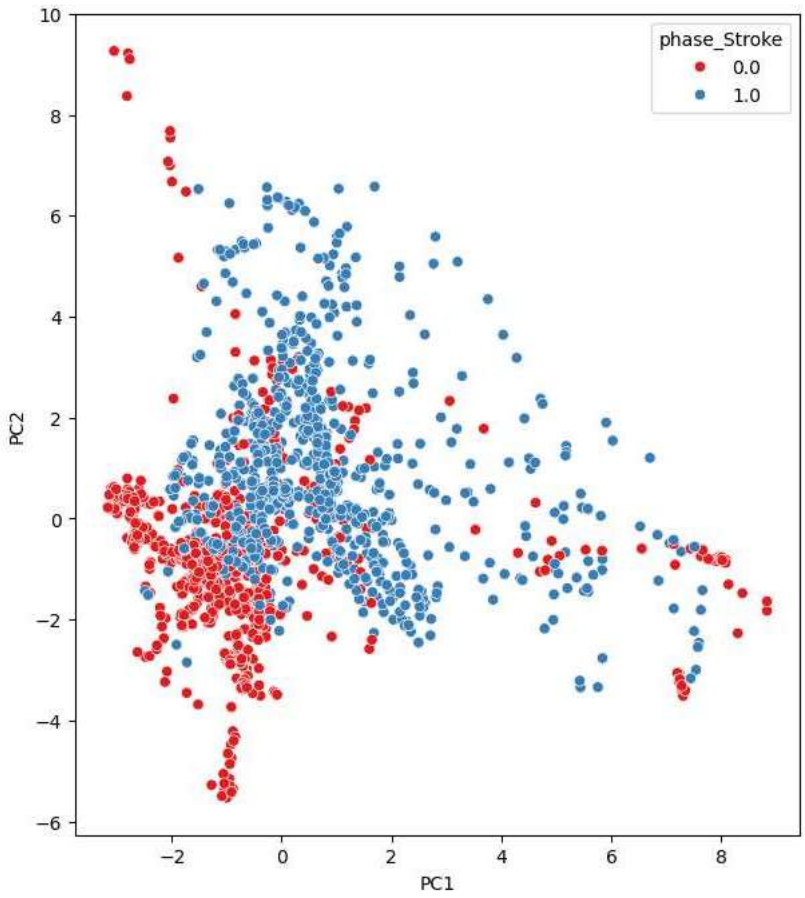
In [113]:
```
segment=pd.concat([segmentation,Y],axis=1)
segment
```

Out[113]:

	PC1	PC2	phase_Stroke
0	-1.102847	-0.621470	0.0
1	-0.616960	-1.651643	0.0
2	-0.879767	1.994206	0.0
3	1.336836	1.938363	0.0
4	-0.367061	0.269400	1.0
...	...	...	...
1759	7.139544	-0.420576	1.0
1760	-1.170941	-0.504925	1.0
1761	0.965362	0.931200	1.0
1762	-0.550390	-0.590830	1.0
1763	-0.827106	-0.166625	1.0

1764 rows × 3 columns

In [138]:
```
plt.figure(figsize=(7,8))
sns.scatterplot(x='PC1',y='PC2',data=segment,hue='phase_Stroke',palette='Set1')
plt.show()
```



In [115]:
```
log_model=LogisticRegression(max_iter=1000)
x_s=segment.drop(['phase_Stroke'],axis=1)
y_s=segment['phase_Stroke']
log_model.fit(x_s,y_s)
```

Out[115]:
```
LogisticRegression(max_iter=1000)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [116]:
```
log_pred=log_model.predict(x_s)
```

In [117]:
```
accuracy_score(y_s,log_pred)
```

Out[117]:
```
0.8100907029478458
```

In [118]:
```
dtt=DecisionTreeClassifier(max_depth=5)
dtt.fit(x_s,y_s)
```

Out[118]:
```
DecisionTreeClassifier(max_depth=5)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [119]:
```
dtt_pred=dtt.predict(x_s)
```

In [120]:
```
accuracy_score(y_s,dtt_pred)
```

Out[120]:
```
0.8747165532879818
```

In [121]:
```
rff=RandomForestClassifier(n_estimators=100,criterion='gini',max_depth=None)
rff.fit(x_s,y_s)
```

Out[121]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [122]:
```
rff_pred=rff.predict(x_s)
```

In [123]:
```
accuracy_score(y_s,rff_pred)
```

Out[123]:
```
1.0
```

Based on the scree plot analysis, I determined that retaining two principal components (PC1 and PC2) captures the most significant variance in the dataset.

Subsequently, I applied these components to three different models for classification: logistic regression, decision tree, and RandomForestClassifier.

The logistic regression model achieved an accuracy of 0.8100907029478458, indicating its ability to reasonably classify the data. The decision tree model, constrained to a maximum depth of 5, demonstrated improved accuracy with a score of 0.8747165532879818, suggesting its ability to capture more complex patterns in the data. Lastly, the RandomForestClassifier, with default parameters, achieved a perfect accuracy score of 1.0, indicating a robust ability to classify the data.

##Neural Networks

In [124]:
```
#ANN
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.
3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2
3.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.
2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.
6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.
0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
(1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.
0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.
0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflo
w) (4.9.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (fro
m tensorflow) (0.36.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
60.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
(2.15.1)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.
15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.
0->tensorflow) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<
2.16,>=2.15->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensor
board<2.16,>=2.15->tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=
2.15->tensorflow) (3.5.2)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.1
6,>=2.15->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (fr
om tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=
2.15->tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<
3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<
3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3
->tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-aut
h-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<
3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->t
ensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.2
1.0->tensorboard<2.16,>=2.15->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.2
1.0->tensorboard<2.16,>=2.15->tensorflow) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->
tensorboard<2.16,>=2.15->tensorflow) (2.1.5)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules
>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=
0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
```

In [125]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.regularizers import l1,l2
```

In [126]:
```
xn=resampled_data.drop(['phase_Stroke'],axis=1)
yn=resampled_data['phase_Stroke']
```

In [127]:
```
x_n_train,x_n_test,y_n_train,y_n_test=train_test_split(xn,yn,test_size=0.2,random_state=42)
```

In [128]:
```
x_n_train.shape
```

Out[128]: (1411, 22)

In [132]:
```
model=Sequential()
model.add(Dense(units=15,activation='relu',kernel_regularizer=l2(0.01),input_shape=(x_n_train.shape[1],)))
model.add(Dense(units=10,activation='relu',kernel_regularizer=l2(0.01)))
model.add(Dense(units=1,activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 15)                345

 dense_4 (Dense)             (None, 10)                160

 dense_5 (Dense)             (None, 1)                 11

=================================================================
Total params: 516 (2.02 KB)
Trainable params: 516 (2.02 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [133]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [134]:
```
from keras.callbacks import EarlyStopping
early_stopping=EarlyStopping(monitor='accuracy',patience=10,verbose=0)
```

In [135]:
```
model.fit(x_n_train,y_n_train,epochs=50,batch_size=32,validation_split=0.2,callbacks=[early_stopping])
```

```
Epoch 1/50
36/36 [==============================] - 2s 16ms/step - loss: 1.5440 - accuracy: 0.4468 - val_loss: 1.1389 - v
al_accuracy: 0.3922
Epoch 2/50
36/36 [==============================] - 0s 6ms/step - loss: 1.0651 - accuracy: 0.3067 - val_loss: 0.9958 - va
l_accuracy: 0.3039
Epoch 3/50
36/36 [==============================] - 0s 6ms/step - loss: 0.9738 - accuracy: 0.3582 - val_loss: 0.9259 - va
l_accuracy: 0.3710
Epoch 4/50
36/36 [==============================] - 0s 6ms/step - loss: 0.8975 - accuracy: 0.5018 - val_loss: 0.8524 - va
l_accuracy: 0.6961
Epoch 5/50
36/36 [==============================] - 0s 5ms/step - loss: 0.8300 - accuracy: 0.7048 - val_loss: 0.7905 - va
l_accuracy: 0.7527
Epoch 6/50
36/36 [==============================] - 0s 5ms/step - loss: 0.7693 - accuracy: 0.8085 - val_loss: 0.7398 - va
l_accuracy: 0.8587
Epoch 7/50
```

In [136]:
```
ann_pred=model.predict(x_n_test)
```

```
12/12 [==============================] - 0s 3ms/step
```

In [137]:
```
loss,accuracy=model.evaluate(x_n_test,y_n_test)
```

```
12/12 [==============================] - 0s 3ms/step - loss: 0.1864 - accuracy: 1.0000
```

In [ ]:

In [1]:
```
!pip install tensorflow
```

```
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\nivetha\anaconda3\lib\site-packages (from google-auth
<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\nivetha\anaconda3\lib\site-packages (from
google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\nivetha\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\nivetha\anaconda3\lib\site-packages (from requests<3,>
=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\nivetha\anaconda3\lib\site-packages (from reques
ts<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\nivetha\anaconda3\lib\site-packages (from reques
ts<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\nivetha\anaconda3\lib\site-packages (from werkzeu
g>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\nivetha\anaconda3\lib\site-packages (from pyas
n1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.4.
8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\nivetha\anaconda3\lib\site-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(3.2.2)
```

In [3]:
```
from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
```

In [4]:
```
train_image=ImageDataGenerator(rescale=1./255)
test_image=ImageDataGenerator(rescale=1./255)
```

In [5]:
```
train_data=train_image.flow_from_directory("C:\\Users\\Nivetha\\OneDrive\\Desktop\\ML fruits dataset",target_size=(6
test_data=test_image.flow_from_directory("C:\\Users\\Nivetha\\OneDrive\\Desktop\\ML fruits dataset",target_size=(64,
```

```
Found 391 images belonging to 2 classes.
Found 391 images belonging to 2 classes.
```

In [6]:
```
train_data.image_shape
```

Out[6]:
```
(64, 64, 3)
```

In [7]:
```
train_data.classes
```

Out[7]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [12]:
```
class_name=['apple','tomato']
```

In [ ]:
```
#Building the model
```

In [8]:
```
from keras.layers import BatchNormalization,Dropout
model=Sequential()
model.add(Conv2D(32,(3,3),padding='same',strides=1,activation='relu',input_shape=train_data.image_shape))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),padding='same',strides=1,activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),padding='same',strides=1,activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.summary()
```

```
WARNING:tensorflow:From C:\Users\Nivetha\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_defa
ult_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Nivetha\anaconda3\Lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161:
The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        896

 max_pooling2d (MaxPooling2  (None, 32, 32, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_2 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 flatten (Flatten)           (None, 4096)              0

=================================================================
Total params: 56320 (220.00 KB)
Trainable params: 56320 (220.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [9]:
```
model=Sequential()
model.add(Conv2D(32,(3,3),padding='same',strides=1,activation='relu',input_shape=train_data.image_shape))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),padding='same',strides=1,activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),padding='same',strides=1,activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(1,activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 64, 64, 32)        896

 max_pooling2d_3 (MaxPoolin  (None, 32, 32, 32)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 32, 32, 64)        18496

 max_pooling2d_4 (MaxPoolin  (None, 16, 16, 64)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_5 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 flatten_1 (Flatten)         (None, 4096)              0

 dense (Dense)               (None, 512)               2097664

 batch_normalization (Batch  (None, 512)               2048
 Normalization)

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 2156545 (8.23 MB)
Trainable params: 2155521 (8.22 MB)
Non-trainable params: 1024 (4.00 KB)
_________________________________________________________________
```

In [ ]:
```
#compiling the model
```

In [10]:
```
from keras.optimizers import Adam
model.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\Nivetha\anaconda3\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name
tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

In [11]:
```
model.fit(train_data,epochs=10,batch_size=32,validation_data=test_data)
```

```
Epoch 1/10
WARNING:tensorflow:From C:\Users\Nivetha\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.r
agged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Nivetha\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The
name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functi
ons instead.

13/13 [==============================] - 4s 154ms/step - loss: 0.8284 - accuracy: 0.5729 - val_loss: 0.9766 - val_a
ccuracy: 0.2558
Epoch 2/10
13/13 [==============================] - 2s 126ms/step - loss: 0.5739 - accuracy: 0.6854 - val_loss: 1.1102 - val_a
ccuracy: 0.2967
Epoch 3/10
13/13 [==============================] - 2s 118ms/step - loss: 0.5467 - accuracy: 0.7494 - val_loss: 0.5398 - val_a
ccuracy: 0.7801
Epoch 4/10
13/13 [==============================] - 2s 128ms/step - loss: 0.5189 - accuracy: 0.7647 - val_loss: 0.5446 - val_a
ccuracy: 0.7519
Epoch 5/10
13/13 [==============================] - 2s 127ms/step - loss: 0.5303 - accuracy: 0.7621 - val_loss: 0.8302 - val_a
ccuracy: 0.4297
Epoch 6/10
13/13 [==============================] - 2s 128ms/step - loss: 0.4993 - accuracy: 0.7724 - val_loss: 0.5134 - val_a
ccuracy: 0.7519
Epoch 7/10
13/13 [==============================] - 2s 132ms/step - loss: 0.4805 - accuracy: 0.7801 - val_loss: 0.4780 - val_a
ccuracy: 0.7596
Epoch 8/10
13/13 [==============================] - 2s 119ms/step - loss: 0.4600 - accuracy: 0.7749 - val_loss: 0.4685 - val_a
ccuracy: 0.7877
Epoch 9/10
13/13 [==============================] - 2s 118ms/step - loss: 0.4403 - accuracy: 0.7928 - val_loss: 0.5412 - val_a
ccuracy: 0.7519
Epoch 10/10
13/13 [==============================] - 2s 118ms/step - loss: 0.3806 - accuracy: 0.8338 - val_loss: 0.7286 - val_a
ccuracy: 0.7519
```

Out[11]:
```
<keras.src.callbacks.History at 0x1d67399a0d0>
```

In [15]:
```
loss,accuracy=model.evaluate(test_data)
```

```
13/13 [==============================] - 1s 59ms/step - loss: 0.7286 - accuracy: 0.7519
```

```
#Number of Steps: The evaluation was conducted over 13 steps or batches.
Time Taken: Each step took approximately 59 milliseconds to complete.
Loss: The average loss across all steps was 0.7286.
Accuracy: The average accuracy across all steps was 75.19%.
This summary suggests that the model achieved a moderate level of accuracy, but the loss value indicates that
there is still room for improvement, especially if the loss function needs to be minimized further for better
performance.
```

In [ ]:
```

```