Assignment : Neural Network Classifier

- **Iris Dataset**

Importing Libraries and Dataset

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras import regularizers
```

In [ ]:
```
df= pd.read_csv('./Dataset/iris.csv')
```

In [ ]:
```
df.head()
```

Out[ ]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [ ]:
```
df.shape
```

Out[ ]:
```
(150, 6)
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [ ]:
```
df = df.drop(columns=['Id'])
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	1.500000e+02	1.500000e+02	1.500000e+02	1.500000e+02
mean	-4.736952e-16	-6.631732e-16	3.315866e-16	-2.842171e-16
std	1.003350e+00	1.003350e+00	1.003350e+00	1.003350e+00
min	-1.870024e+00	-2.438987e+00	-1.568735e+00	-1.444450e+00
25%	-9.006812e-01	-5.877635e-01	-1.227541e+00	-1.181504e+00
50%	-5.250608e-02	-1.249576e-01	3.362659e-01	1.332259e-01
75%	6.745011e-01	5.692513e-01	7.627586e-01	7.905908e-01
max	2.492019e+00	3.114684e+00	1.786341e+00	1.710902e+00

Checking Null Values

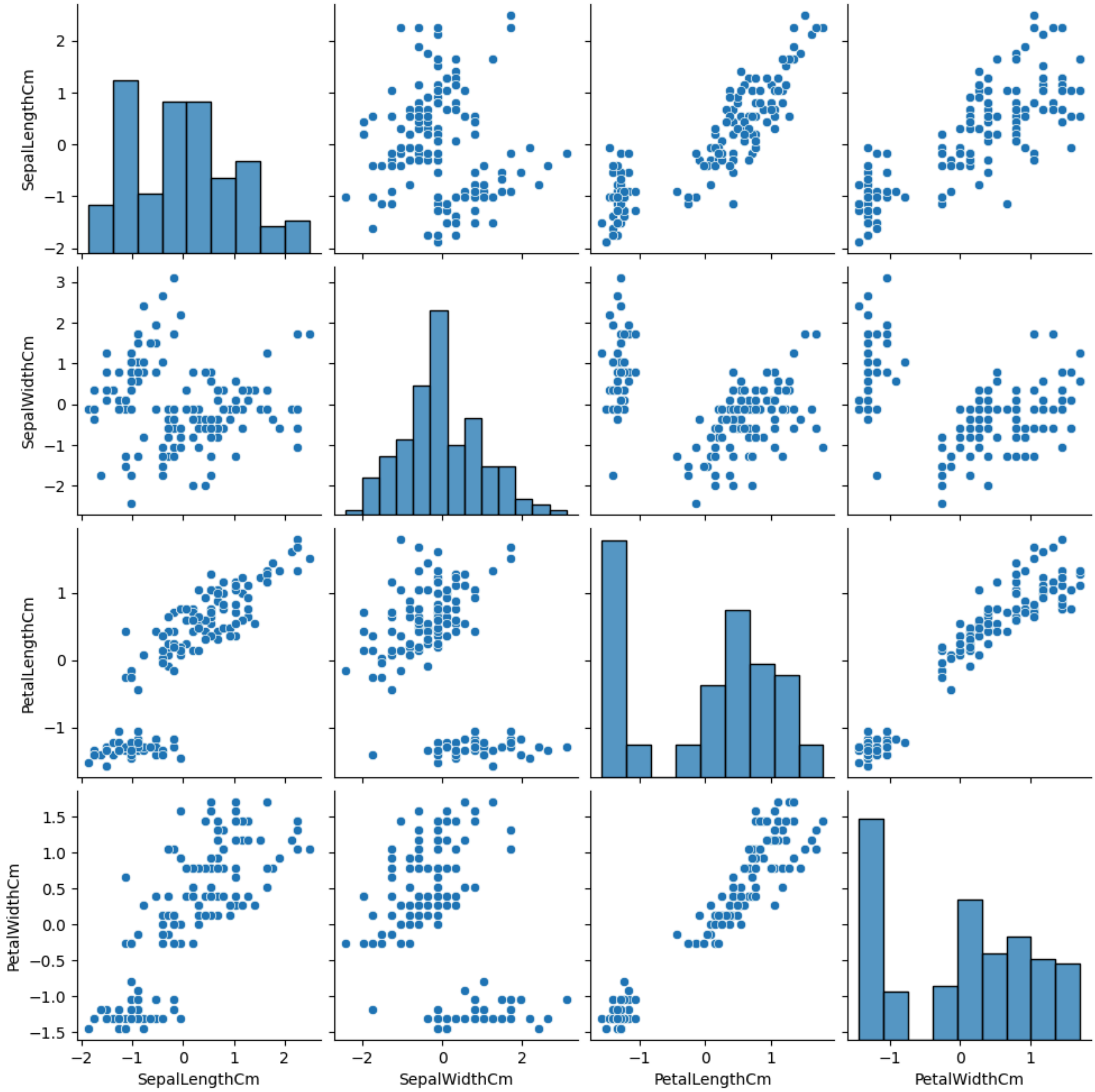
In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Visualizing Data

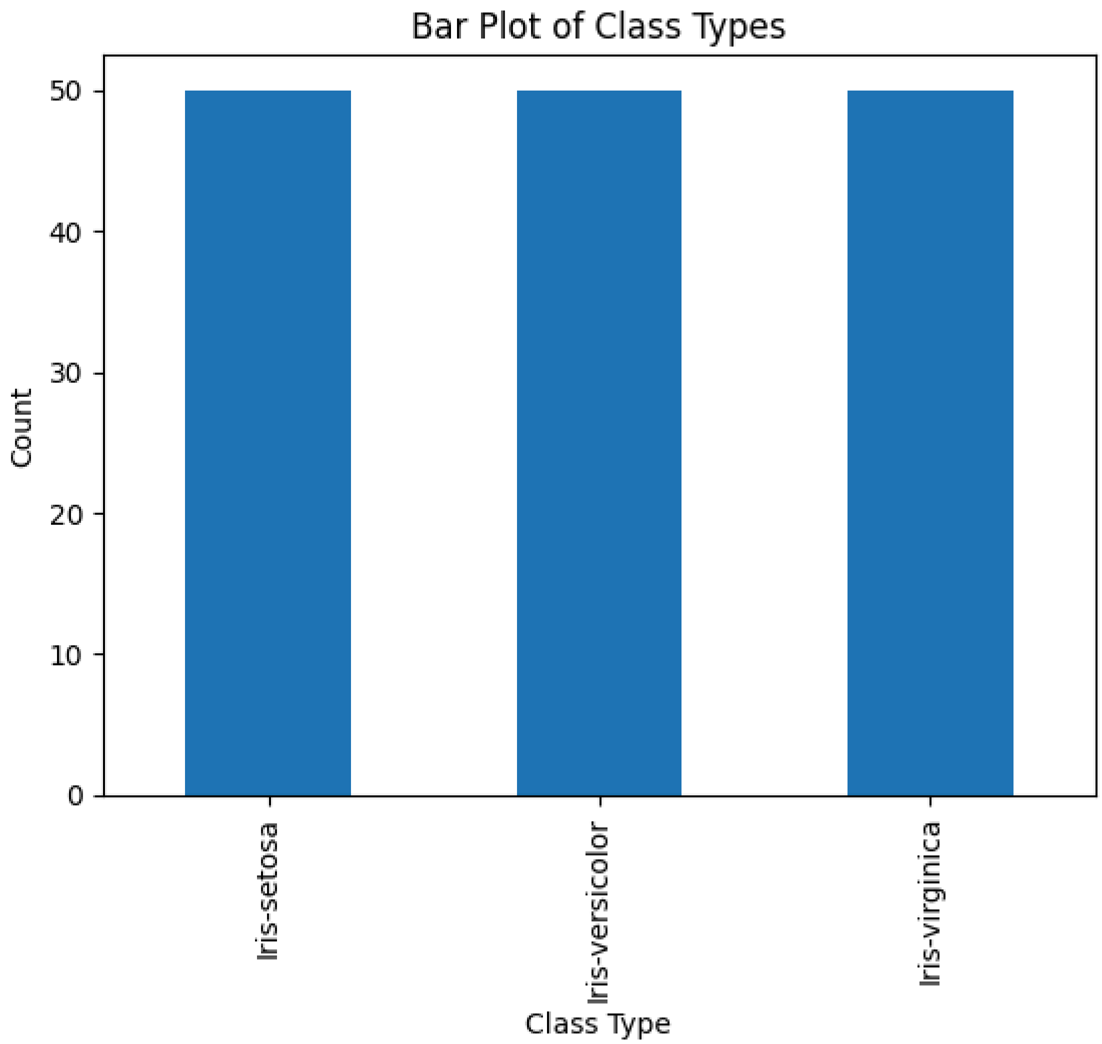
In [ ]:
```
sns.pairplot(df)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x17959498210>
```



In [ ]:

```
class_counts = df['Species'].value_counts()
class_counts.plot(kind='bar')
plt.title('Bar Plot of Class Types')
plt.xlabel('Class Type')
plt.ylabel('Count')
plt.show()
```



Scaling

In [ ]:
```
std_scaler = StandardScaler()
```

In [ ]:
```
features_to_scale = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
for feature in features_to_scale:
    df[feature] = std_scaler.fit_transform(df[feature].values.reshape(-1, 1))
```

In [ ]:
```
df.head()
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	-0.900681	1.032057	-1.341272	-1.312977	Iris-setosa
1	-1.143017	-0.124958	-1.341272	-1.312977	Iris-setosa
2	-1.385353	0.337848	-1.398138	-1.312977	Iris-setosa
3	-1.506521	0.106445	-1.284407	-1.312977	Iris-setosa
4	-1.021849	1.263460	-1.341272	-1.312977	Iris-setosa

In [ ]:
```
df.describe()
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	1.500000e+02	1.500000e+02	1.500000e+02	1.500000e+02
mean	4.736952e-17	-2.368476e-17	4.736952e-17	4.736952e-17
std	1.003350e+00	1.003350e+00	1.003350e+00	1.003350e+00
min	-1.870024e+00	-2.438987e+00	-1.568735e+00	-1.444450e+00
25%	-9.006812e-01	-5.877635e-01	-1.227541e+00	-1.181504e+00
50%	-5.250608e-02	-1.249576e-01	3.362659e-01	1.332259e-01
75%	6.745011e-01	5.692513e-01	7.627586e-01	7.905908e-01
max	2.492019e+00	3.114684e+00	1.786341e+00	1.710902e+00

Categorical to Numerical

In [ ]:
```
df['Species'].unique()
```

Out[ ]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [ ]:
```
le_object = LabelEncoder()
```

In [ ]:
```
df['Species'] = le_object.fit_transform(df['Species'])
df.head()
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	-0.900681	1.032057	-1.341272	-1.312977	0
1	-1.143017	-0.124958	-1.341272	-1.312977	0
2	-1.385353	0.337848	-1.398138	-1.312977	0
3	-1.506521	0.106445	-1.284407	-1.312977	0
4	-1.021849	1.263460	-1.341272	-1.312977	0

Separating dependent and independent variables

In [ ]:
```
X = df.drop('Species',axis=1)
y = df['Species']
```

In [ ]:
```
X
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977
...	...	...	...	...
145	1.038005	-0.124958	0.819624	1.447956
146	0.553333	-1.281972	0.705893	0.922064
147	0.795669	-0.124958	0.819624	1.053537
148	0.432165	0.800654	0.933356	1.447956
149	0.068662	-0.124958	0.762759	0.790591

150 rows × 4 columns

In [ ]:

```
y
```

Out[ ]:

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species, Length: 150, dtype: int32
```

Train-Test Split

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42
```

In [ ]:

```
X_train
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
4	-1.021849	1.263460	-1.341272	-1.312977
32	-0.779513	2.420475	-1.284407	-1.444450
142	-0.052506	-0.819166	0.762759	0.922064
85	0.189830	0.800654	0.421564	0.527645
86	1.038005	0.106445	0.535296	0.396172
...	...	...	...	...
71	0.310998	-0.587764	0.137236	0.133226
106	-1.143017	-1.281972	0.421564	0.659118
14	-0.052506	2.189072	-1.455004	-1.312977
92	-0.052506	-1.050569	0.137236	0.001753
102	1.522676	-0.124958	1.217684	1.185010

112 rows × 4 columns

In [ ]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(112, 4)
(112,)
(38, 4)
(38,)
```

Saving Cleaned Dfs to CSV files

In [ ]:
```
X_train.to_csv('./Assignment6_Dataset/X_train.csv')
y_train.to_csv('./Assignment6_Dataset/y_train.csv')
X_test.to_csv('./Assignment6_Dataset/X_test.csv')
y_test.to_csv('./Assignment6_Dataset/y_test.csv')
```

Modelling

In [ ]:
```
ann_model = Sequential()
ann_model.add(Dense(units=16, input_shape=(4,), activation='relu'))
```

```
c:\Users\Prasanna Pandhare\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src
\layers\core\dense.py:88: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a
layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer
in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [ ]:
```
ann_model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 16)	80

Total params: 80 (320.00 B)

Trainable params: 80 (320.00 B)

Non-trainable params: 0 (0.00 B)

In [ ]:
```
ann_model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=['accurac
```

In [ ]:
```
history = ann_model.fit(X_train,
        y_train,
        epochs = 100
        )
```

```
Epoch 1/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0000e+00 - loss: 9.2736
Epoch 2/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0000e+00 - loss: 9.1256
Epoch 3/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 9.6261
Epoch 4/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 9.4006
Epoch 5/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 9.0203
Epoch 6/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 9.4112
Epoch 7/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 8.5975
Epoch 8/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 8.5718
Epoch 9/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 9.0637
Epoch 10/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0000e+00 - loss: 8.8859
Epoch 11/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 8.7509
Epoch 12/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0000e+00 - loss: 8.6993
Epoch 13/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 8.4821
Epoch 14/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0000e+00 - loss: 8.3567
Epoch 15/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0000e+00 - loss: 8.5174
Epoch 16/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0057 - loss: 8.0587
Epoch 17/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0036 - loss: 8.5691
Epoch 18/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0057 - loss: 8.4501
Epoch 19/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0092 - loss: 8.5611
Epoch 20/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0301 - loss: 7.8346
Epoch 21/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0301 - loss: 8.3822
Epoch 22/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0238 - loss: 7.7357
Epoch 23/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0092 - loss: 7.9099
Epoch 24/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0326 - loss: 7.0503
Epoch 25/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0149 - loss: 6.9987
Epoch 26/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0387 - loss: 7.3272
Epoch 27/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0621 - loss: 6.7264
Epoch 28/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0631 - loss: 7.2022
Epoch 29/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0629 - loss: 6.4808
Epoch 30/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0661 - loss: 6.6547
Epoch 31/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.0775 - loss: 6.8226
Epoch 32/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0972 - loss: 6.5597
Epoch 33/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0997 - loss: 6.1699
```

```
Epoch 34/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1281 - loss: 5.9259
Epoch 35/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1685 - loss: 5.8126
Epoch 36/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1585 - loss: 5.7257
Epoch 37/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1652 - loss: 5.3635
Epoch 38/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2332 - loss: 5.4551
Epoch 39/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2235 - loss: 5.4468
Epoch 40/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2774 - loss: 5.6546
Epoch 41/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2543 - loss: 5.3415
Epoch 42/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3125 - loss: 5.4982
Epoch 43/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2926 - loss: 5.3124
Epoch 44/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3667 - loss: 4.8768
Epoch 45/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3781 - loss: 4.9628
Epoch 46/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3624 - loss: 5.1053
Epoch 47/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4037 - loss: 4.2918
Epoch 48/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4624 - loss: 3.5590
Epoch 49/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4451 - loss: 3.7804
Epoch 50/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4693 - loss: 3.1537
Epoch 51/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4900 - loss: 3.4972
Epoch 52/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5470 - loss: 3.4466
Epoch 53/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5815 - loss: 2.9274
Epoch 54/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5955 - loss: 2.8630
Epoch 55/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6137 - loss: 2.7916
Epoch 56/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6464 - loss: 2.4981
Epoch 57/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6454 - loss: 2.6175
Epoch 58/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5746 - loss: 2.8809
Epoch 59/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6302 - loss: 2.5447
Epoch 60/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6260 - loss: 2.6648
Epoch 61/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6271 - loss: 2.7621
Epoch 62/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6125 - loss: 2.8238
Epoch 63/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6396 - loss: 2.2649
Epoch 64/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6187 - loss: 2.6078
Epoch 65/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6365 - loss: 2.5872
Epoch 66/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6219 - loss: 2.2795
```

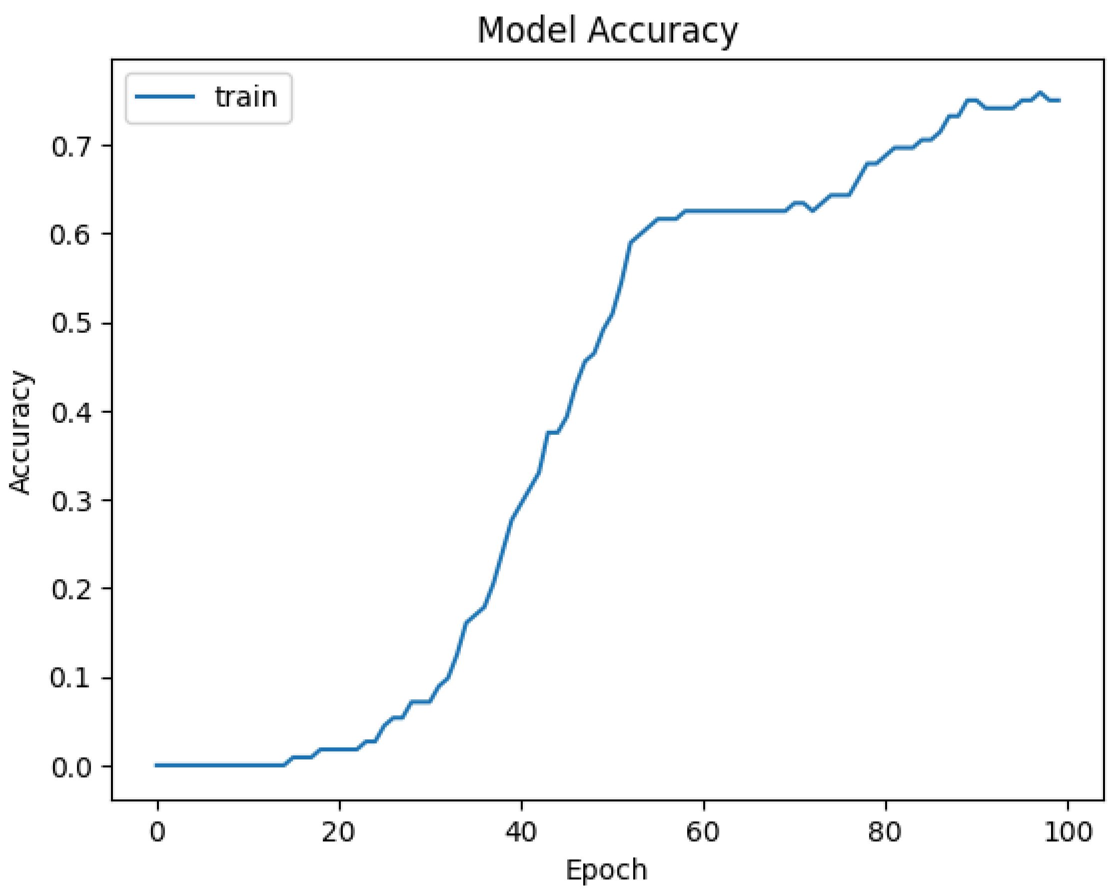
```
Epoch 67/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6115 - loss: 2.3791
Epoch 68/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6292 - loss: 2.6258
Epoch 69/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6198 - loss: 2.3910
Epoch 70/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5813 - loss: 2.3221
Epoch 71/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6557 - loss: 2.2596
Epoch 72/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6411 - loss: 1.9310
Epoch 73/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6375 - loss: 2.0386
Epoch 74/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6202 - loss: 2.2519
Epoch 75/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6113 - loss: 2.2735
Epoch 76/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6280 - loss: 2.2694
Epoch 77/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6457 - loss: 2.2491
Epoch 78/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6528 - loss: 2.0533
Epoch 79/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6704 - loss: 1.8613
Epoch 80/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6881 - loss: 2.1096
Epoch 81/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6885 - loss: 1.9978
Epoch 82/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7244 - loss: 1.9831
Epoch 83/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7192 - loss: 1.8419
Epoch 84/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7088 - loss: 1.9594
Epoch 85/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7280 - loss: 1.8343
Epoch 86/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6717 - loss: 1.8315
Epoch 87/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7128 - loss: 1.8672
Epoch 88/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7501 - loss: 1.5859
Epoch 89/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7283 - loss: 1.6355
Epoch 90/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7229 - loss: 1.6399
Epoch 91/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7563 - loss: 1.5349
Epoch 92/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7475 - loss: 1.6056
Epoch 93/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7454 - loss: 1.3607
Epoch 94/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7548 - loss: 1.4429
Epoch 95/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6829 - loss: 1.6841
Epoch 96/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7302 - loss: 1.3057
Epoch 97/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7417 - loss: 1.3400
Epoch 98/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7515 - loss: 1.4490
Epoch 99/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7365 - loss: 1.5199
```

```
Epoch 100/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6938 - loss: 1.5496
```

In [ ]:
```python
ann_model.layers[0].get_weights()
```

Out[ ]:
```
[array([[ 0.17535837,  0.5101146 ,  0.16120644,  0.10642558,  0.10033165,
          0.06057078,  0.2602838 ,  0.2787637 ,  0.19394796,  0.04441752,
          0.20228055,  0.14076908,  0.17942774,  0.22564845,  0.07398073,
          0.24094258],
        [-0.13264523, -0.28800833, -0.18984903, -0.15611121, -0.1198829 ,
         -0.5270991 , -0.08004733, -0.18823521, -0.18176961, -0.04904248,
         -0.08461405, -0.02305197,  0.07531834, -0.13541014, -0.02570221,
         -0.10873175],
        [ 0.27929708, -0.20344418,  0.12824382, -0.2488992 ,  0.06650526,
         -0.1911001 , -0.50332403, -0.20092003, -0.3431931 ,  0.03486646,
         -0.33007017, -0.22324774, -0.03233628, -0.01246677,  0.10821003,
         -0.10308858],
        [-0.5581435 , -0.27409038,  0.28306323,  0.21182123,  0.00958541,
         -0.10071055,  0.25220865, -0.18739675,  0.18569775,  0.06203302,
          0.09555542,  0.10753948,  0.05318423, -0.30282757, -0.22942944,
         -0.20955935]], dtype=float32),
 array([-0.0425511 ,  0.05186893, -0.16988632, -0.25825125, -0.32543644,
        -0.4154884 , -0.21298742, -0.2384223 , -0.25017217, -0.25588143,
        -0.1694253 , -0.18081217, -0.1585924 , -0.25080898, -0.1426096 ,
        -0.22499906], dtype=float32)]
```

In [ ]:
```python
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



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

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

```
Out[ ]:  array([[2.3847711e-01, 2.7041084e-01, 6.0979426e-02, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [3.0495193e-02, 2.8102187e-02, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [1.8205562e-01, 7.4174660e-01, 1.0311769e+00, 7.2670877e-03,
                  1.5890363e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  3.2932997e-02, 4.7670901e-02, 0.0000000e+00, 0.0000000e+00,
                  1.8518081e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 5.6986872e-02, 9.4574869e-02, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [2.5633258e-01, 6.1943740e-01, 2.7943370e-01, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.6769906e-02,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  5.5447221e-05, 2.8052032e-03, 0.0000000e+00, 1.6883165e-03],
                 [0.0000000e+00, 9.6360125e-02, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 1.2228264e-01, 5.2398634e-01, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 9.3015969e-02, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  1.3149711e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [1.9198468e-01, 6.4712775e-01, 4.4116232e-01, 7.5236827e-02,
                  0.0000000e+00, 5.3187191e-01, 0.0000000e+00, 9.5094562e-02,
                  1.2174535e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [7.8368798e-02, 2.4418026e-01, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 0.0000000e+00, 2.9027709e-01, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  4.0985629e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [1.8393525e-01, 1.1176239e-01, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [8.9934140e-02, 1.8673259e-01, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [1.9037101e-01, 9.5356956e-02, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 0.0000000e+00, 2.9248014e-02, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
                  0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
                 [0.0000000e+00, 0.0000000e+00, 5.0361943e-01, 0.0000000e+00,
```

```
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 7.2406381e-03, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[1.7064297e-01, 2.8988868e-01, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 2.4008468e-01, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[4.8340280e-02, 1.0275327e-02, 5.4770708e-03, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 5.5736423e-01, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[5.9682574e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 2.1099249e-01, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 2.7399058e-02, 5.2014911e-01, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 2.3156525e-01, 3.9495465e-01, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 4.2602652e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 5.3740602e-02, 5.3614461e-01, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 1.9841209e-02, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 6.8747655e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[1.9246937e-01, 4.9773389e-01, 6.1348295e-01, 0.0000000e+00,
 1.7173529e-02, 2.1657437e-02, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 5.1886308e-01, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 1.1247453e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[3.7173714e-02, 3.9691448e-02, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[1.1162487e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 3.0526355e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
[7.9471558e-02, 4.0396589e-01, 9.8935068e-02, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
```

```
        3.7956551e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [1.9541353e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [0.0000000e+00, 3.0701238e-01, 5.1422453e-01, 2.0384312e-02,
        0.0000000e+00, 6.5996140e-02, 0.0000000e+00, 0.0000000e+00,
        1.9136876e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [0.0000000e+00, 1.0428694e-01, 4.0912151e-02, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [5.7955649e-02, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00]],
      dtype=float32)
```

In [ ]:
```
y_pred_labels = np.argmax(y_pred, axis=1)
y_pred_labels
```

Out[ ]:
```
array([ 1,  0,  2,  2,  1,  1,  0,  2,  1,  1,  2,  0,  1,  0,  0,  2,  2,
        1,  0,  2,  0,  2,  0,  2, 12,  2,  2,  2,  1,  0,  0, 12,  1,  0,
        0,  2,  1,  0], dtype=int64)
```

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

Out[ ]:
```
0.7631578947368421
```

In [ ]:
```
precision_score(y_test, y_pred_labels, average = 'weighted')
```

Out[ ]:
```
0.8038461538461538
```

In [ ]:
```
recall_score(y_test, y_pred_labels, average = 'weighted')
```

```
c:\Users\Prasanna Pandhare\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\m
etrics\_classification.py:1469: UndefinedMetricWarning: Recall is ill-defined and being set to
0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[ ]:
```
0.7631578947368421
```

In [ ]:
```
f1_score(y_test, y_pred_labels, average = 'weighted')
```

Out[ ]:
```
0.7810275689223056
```

In [ ]:
```
target_names = ['Iris-setosa','Iris-versicolor','Iris-virginica','None']
```

In [ ]:
```
print(classification_report(y_true = y_test, y_pred = y_pred_labels, target_names = target_na
```

```
                 precision    recall  f1-score   support

    Iris-setosa       0.85      0.73      0.79        15
Iris-versicolor       0.70      0.64      0.67        11
 Iris-virginica       0.85      0.92      0.88        12
           None       0.00      0.00      0.00         0

       accuracy                           0.76        38
      macro avg       0.60      0.57      0.58        38
   weighted avg       0.80      0.76      0.78        38
```

```
c:\Users\Prasanna Pandhare\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Prasanna Pandhare\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Prasanna Pandhare\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

```
confusion_matrix(y_test, y_pred_labels)
```

Out[ ]:

```
array([[11,  3,  0,  1],
       [ 2,  7,  2,  0],
       [ 0,  0, 11,  1],
       [ 0,  0,  0,  0]], dtype=int64)
```