In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
```

In [2]:
```
dataset = load_breast_cancer()
```

In [3]:
```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df.head()
```

Out[3]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symmet
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.24
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.18
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.20
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.25
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.18

5 rows × 30 columns

In [4]:
```
df['label'] = dataset.target
```

In [5]:
```
# Checking Missing values
df.isnull().sum()
```

Out[5]:
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
label                      0
dtype: int64
```

In [6]:
```
x = df.drop(columns='label' ,axis=1)
y= df['label']
```

In [7]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_st
```

In [8]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [9]:
```
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

BUILD ANN MODEL - Artificial Neural Network

In [10]:
```
import tensorflow as tf
from tensorflow import keras
```

In [11]:

```
# Setting up the layers of Neural Network

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(30,)),
    keras.layers.Dense(20,activation='relu'),
    keras.layers.Dense(2,activation='sigmoid')
])
```

In [12]:

```
#Compiling the Neural Network

model.compile(optimizer='adam',
             loss='sparse_categorical_crossentropy',
             metrics=['accuracy'])
```

In [13]:

```
# Training the neural network

history = model.fit(x_train_scaled , y_train , validation_split=0.15 ,epochs
```

```
Epoch 1/10
13/13 [==============================] - 0s 11ms/step - loss: 1.5346 - acc
uracy: 0.2306 - val_loss: 0.9321 - val_accuracy: 0.3913
Epoch 2/10
13/13 [==============================] - 0s 3ms/step - loss: 1.0065 - accu
racy: 0.4326 - val_loss: 0.5966 - val_accuracy: 0.7101
Epoch 3/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6575 - accu
racy: 0.6736 - val_loss: 0.4274 - val_accuracy: 0.8261
Epoch 4/10
13/13 [==============================] - 0s 2ms/step - loss: 0.4702 - accu
racy: 0.8187 - val_loss: 0.3359 - val_accuracy: 0.8986
Epoch 5/10
13/13 [==============================] - 0s 2ms/step - loss: 0.3604 - accu
racy: 0.8938 - val_loss: 0.2799 - val_accuracy: 0.9275
Epoch 6/10
13/13 [==============================] - 0s 1ms/step - loss: 0.2907 - accu
racy: 0.9119 - val_loss: 0.2415 - val_accuracy: 0.9420
Epoch 7/10
13/13 [==============================] - 0s 3ms/step - loss: 0.2435 - accu
racy: 0.9249 - val_loss: 0.2161 - val_accuracy: 0.9420
Epoch 8/10
13/13 [==============================] - 0s 3ms/step - loss: 0.2134 - accu
racy: 0.9352 - val_loss: 0.1958 - val_accuracy: 0.9275
Epoch 9/10
13/13 [==============================] - 0s 1ms/step - loss: 0.1883 - accu
racy: 0.9378 - val_loss: 0.1808 - val_accuracy: 0.9275
Epoch 10/10
13/13 [==============================] - 0s 3ms/step - loss: 0.1710 - accu
racy: 0.9456 - val_loss: 0.1698 - val_accuracy: 0.9275
```

In [14]:

```
!jt -t oceans16
```

```
'jt' is not recognized as an internal or external command,
operable program or batch file.
```

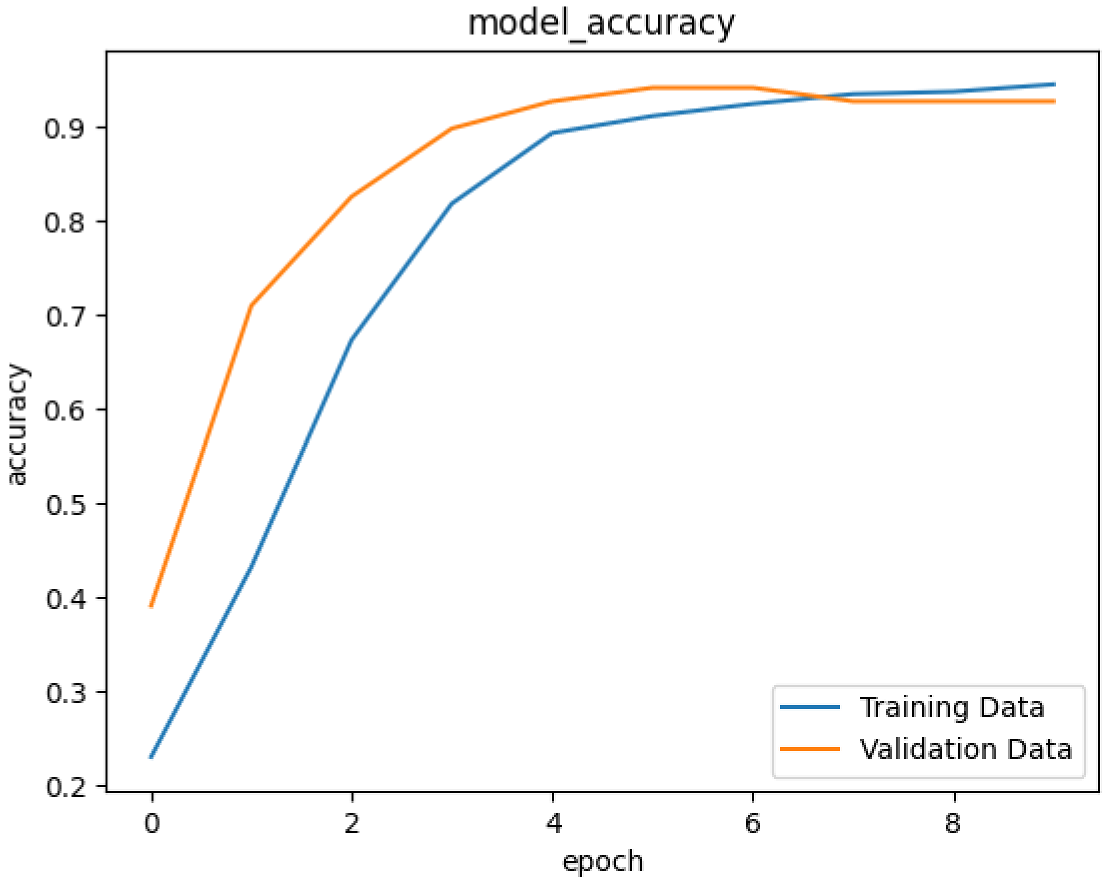
VISUALIZATION

In [15]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('model_accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

plt.legend(['Training Data','Validation Data'],loc='lower right')
```

Out[15]:

```
<matplotlib.legend.Legend at 0x1fe5c2e99d0>
```

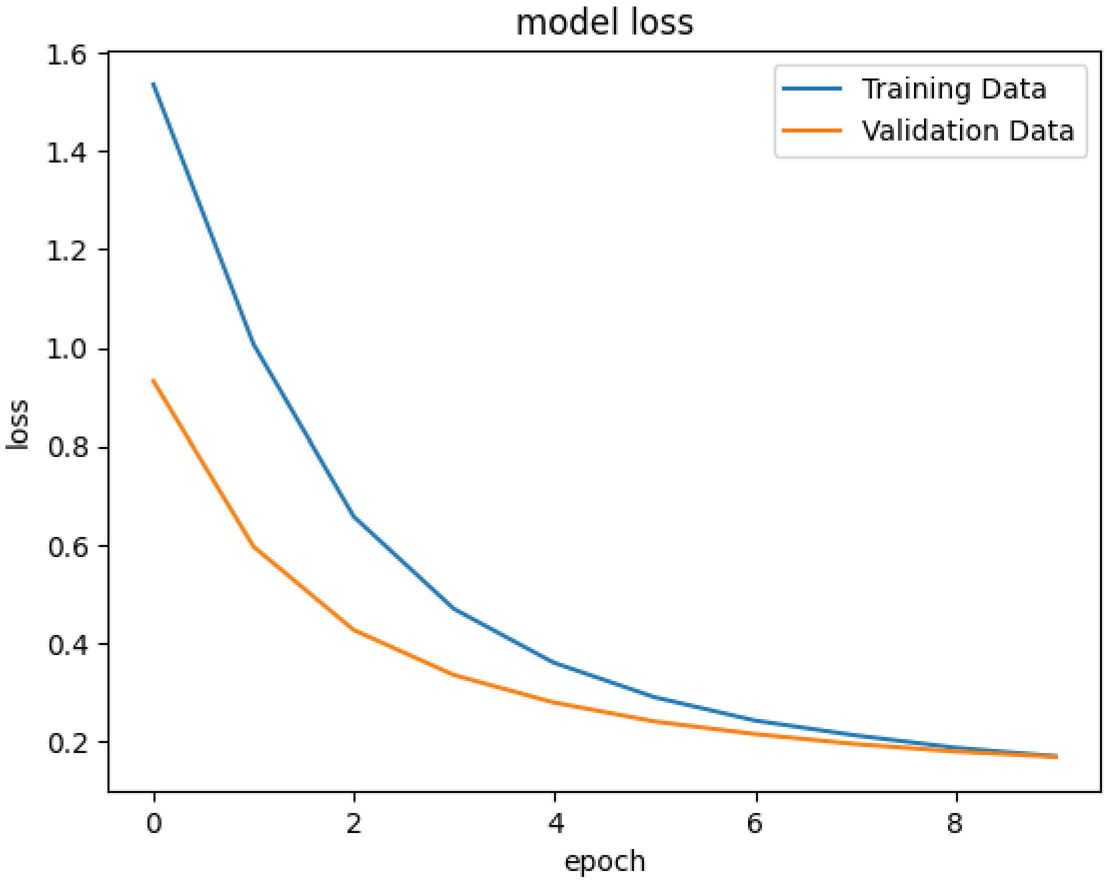


In [16]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')

plt.legend(['Training Data','Validation Data'],loc='upper right')
```

Out[16]:

```
<matplotlib.legend.Legend at 0x1fe7fbbac70>
```



Prediction

In [17]:

```
loss,accuracy = model.evaluate(x_test_scaled,y_test)
print(accuracy)
```

```
4/4 [==============================] - 0s 1ms/step - loss: 0.1871 - accura
cy: 0.9737
0.9736841917037964
```

In [18]:

```
print(loss)
```

```
0.18713927268981934
```

In [19]:

```
y_pred=model.predict(x_test_scaled)
```

```
4/4 [==============================] - 0s 721us/step
```

In [ ]:

In [79]:
```
!pip install numpy
!pip install matplotlib
!pip install pandas
!pip install scikit-learn
!pip install keras
!pip install tensorflow
```

```
Requirement already satisfied: numpy in c:\users\student\.conda\envs\mal
likarjuna\lib\site-packages (1.24.3)
Requirement already satisfied: matplotlib in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (3.7.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\student\.con
da\envs\mallikarjuna\lib\site-packages (from matplotlib) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\student\.co
nda\envs\mallikarjuna\lib\site-packages (from matplotlib) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\student\.co
nda\envs\mallikarjuna\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.20 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib) (10.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\student\.con
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [4]:
```
df=pd.read_csv("Churn_Modelling.csv")
```

In [5]:
```
X=df.iloc[:,3:13]
y=df.iloc[:,13]
X.head()
```

Out[5]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

In [6]:
```
geography=pd.get_dummies(X['Geography'],dtype=int)
geography
```

Out[6]:

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	1	0	0
3	1	0	0
4	0	0	1
...	...	...	...
9995	1	0	0
9996	1	0	0
9997	1	0	0
9998	0	1	0
9999	1	0	0

10000 rows × 3 columns

In [7]:
```
gender=pd.get_dummies(X['Gender'],drop_first=True,dtype=int)
gender
```

Out[7]:

	Male
0	0
1	0
2	0
3	0
4	0
...	...
9995	1
9996	1
9997	0
9998	1
9999	0

10000 rows × 1 columns

In [8]:
```
X=pd.concat([X,geography,gender],axis=1)
```

In [9]:

```
X=X.drop(['Geography','Gender'],axis=1)
X.head()
```

Out[9]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimat
0	619	42	2	0.00	1	1	1	1
1	608	41	1	83807.86	1	0	1	
2	502	42	8	159660.80	3	1	0	
3	699	39	1	0.00	2	0	0	
4	850	43	2	125510.82	1	1	1	

In [10]:

```
from sklearn.model_selection import train_test_split
```

In [11]:

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

In [12]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [13]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [14]:

```
classifier=Sequential()
```

In [15]:

```
classifier.add(Dense(6,kernel_initializer='he_uniform',activation='relu',inp
```

In [16]:

```
classifier.add(Dense(6,kernel_initializer='he_uniform',activation='relu'))
```

In [17]:

```
classifier.add(Dense(1,kernel_initializer='glorot_uniform',activation='relu'
```

In [18]:

```
classifier.compile(optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

In [19]:

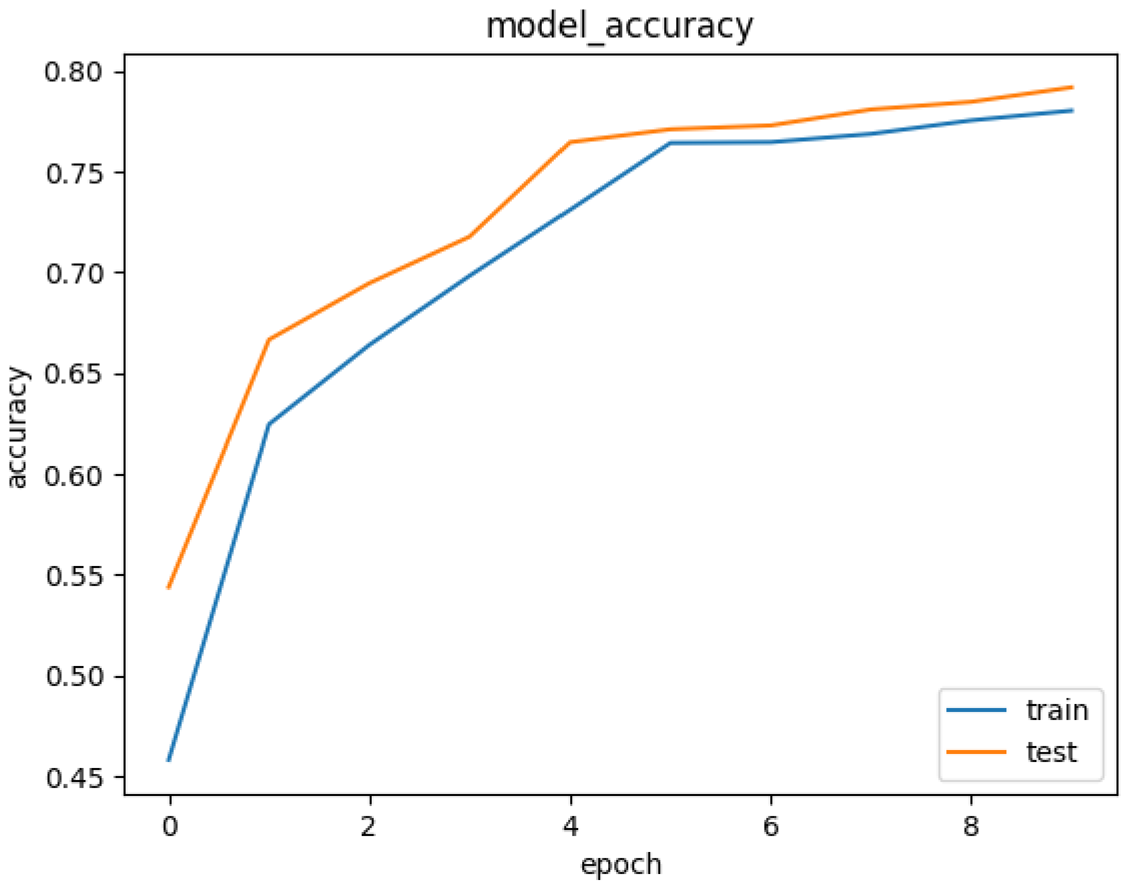
```
model_history=classifier.fit(X_train,y_train,validation_split=0.33,batch_siz
```

```
Epoch 1/10
536/536 [==============================] - 1s 1ms/step - loss: 4.2277 - ac
curacy: 0.4579 - val_loss: 2.6548 - val_accuracy: 0.5437
Epoch 2/10
536/536 [==============================] - 0s 865us/step - loss: 1.7962 -
accuracy: 0.6244 - val_loss: 1.3310 - val_accuracy: 0.6664
Epoch 3/10
536/536 [==============================] - 0s 856us/step - loss: 0.9973 -
accuracy: 0.6637 - val_loss: 0.7725 - val_accuracy: 0.6944
Epoch 4/10
536/536 [==============================] - 0s 878us/step - loss: 0.7581 -
accuracy: 0.6981 - val_loss: 0.6583 - val_accuracy: 0.7175
Epoch 5/10
536/536 [==============================] - 0s 849us/step - loss: 0.6353 -
accuracy: 0.7309 - val_loss: 0.5560 - val_accuracy: 0.7645
Epoch 6/10
536/536 [==============================] - 0s 856us/step - loss: 0.6106 -
accuracy: 0.7641 - val_loss: 0.5452 - val_accuracy: 0.7709
Epoch 7/10
536/536 [==============================] - 0s 852us/step - loss: 0.5614 -
accuracy: 0.7645 - val_loss: 0.5141 - val_accuracy: 0.7728
Epoch 8/10
536/536 [==============================] - 0s 874us/step - loss: 0.5317 -
accuracy: 0.7686 - val_loss: 0.5032 - val_accuracy: 0.7808
Epoch 9/10
536/536 [==============================] - 0s 888us/step - loss: 0.5214 -
accuracy: 0.7753 - val_loss: 0.4959 - val_accuracy: 0.7846
Epoch 10/10
536/536 [==============================] - 0s 872us/step - loss: 0.5097 -
accuracy: 0.7802 - val_loss: 0.4902 - val_accuracy: 0.7917
```

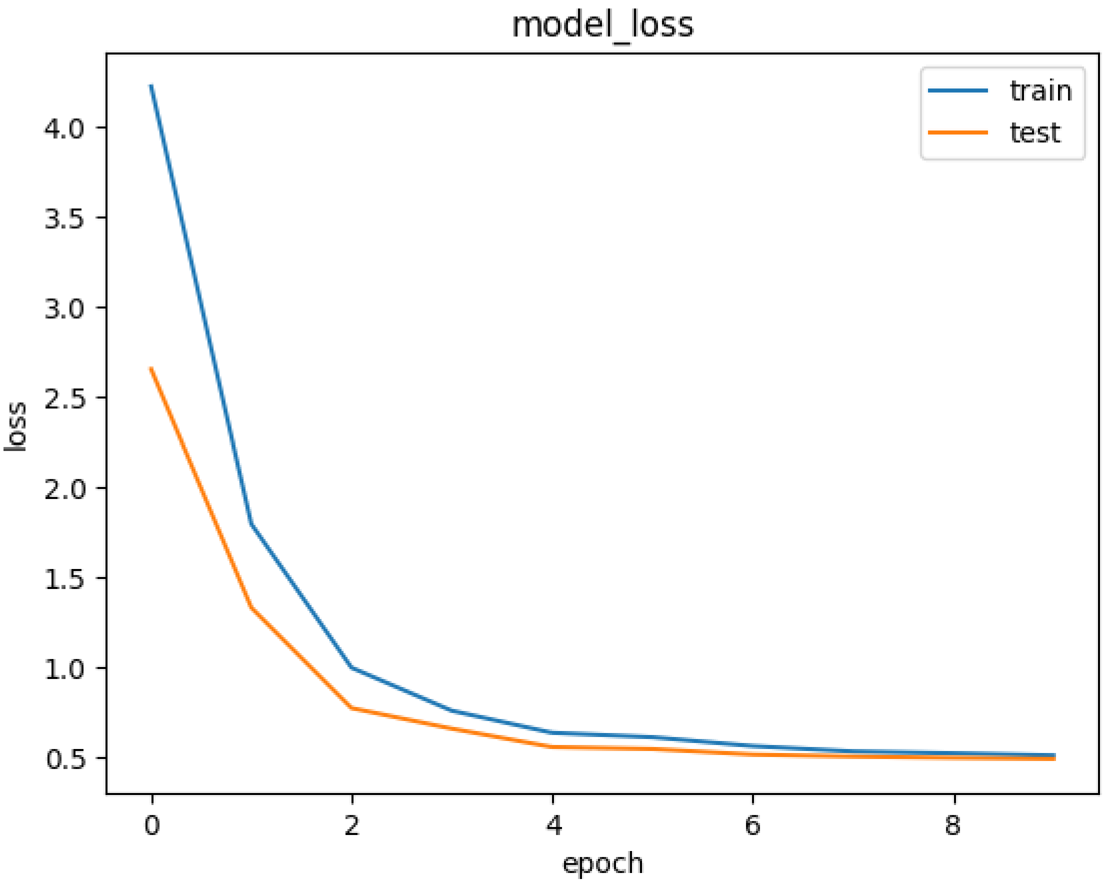
In [20]:

```
print(model_history.history.keys())
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('model_accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='lower right')
plt.show()
```

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



In [21]:
```
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('model_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='upper right')
plt.show()
```



In [ ]:
```
```

In [12]:
```
import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
from keras_tuner.tuners import RandomSearch
```

In [13]:
```
df = pd.read_csv('Real_Combine.csv')
df.head()
```

Out[13]:

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
0	7.4	9.8	4.8	1017.6	93.0	0.5	4.3	9.4	219.720833
1	7.8	12.7	4.4	1018.5	87.0	0.6	4.4	11.1	182.187500
2	6.7	13.4	2.4	1019.4	82.0	0.6	4.8	11.1	154.037500
3	8.6	15.5	3.3	1018.7	72.0	0.8	8.1	20.6	223.208333
4	12.4	20.9	4.4	1017.3	61.0	1.3	8.7	22.2	200.645833

In [14]:
```
df=df.dropna()
df.isnull().sum()
```

Out[14]:
```
T         0
TM        0
Tm        0
SLP       0
H         0
VV        0
V         0
VM        0
PM 2.5    0
dtype: int64
```

In [15]:
```
# Creation of feature set and target set
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [16]:
```
def hyper_tune(param):
    model = keras.Sequential()
    for i in range(param.Int('num_layers' ,2,20)): #hidden layers range
        model.add(layers.Dense(units=param.Int('units_'+str(i),
                                              min_value=32, #neurons
                                              max_value=512,
                                              step=32), #32+32=64,64+32=96,.
                               activation='tanh'))
    model.add(layers.Dense(1,activation='linear'))
    model.compile(
        optimizer=keras.optimizers.Adam(param.Choice('learning_rate' , [1e-2
        loss='mean_absolute_error',
        metrics=['mean_absolute_error'])
    return model
```

In [17]:

```
tuner = RandomSearch(
    hyper_tune,
    objective='val_mean_absolute_error',
    max_trials=5,
    executions_per_trial=3,
    directory='project',
    overwrite=True,
    project_name = 'Air Quality Index AQI'
)
```

In [18]:

```
tuner.search_space_summary()   #skip
```

```
Search space summary
Default search space size: 4
num_layers (Int)
{'default': None, 'conditions': [], 'min_value': 2, 'max_value': 20, 'ste
p': 1, 'sampling': 'linear'}
units_0 (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 512, 'st
ep': 32, 'sampling': 'linear'}
units_1 (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 512, 'st
ep': 32, 'sampling': 'linear'}
learning_rate (Choice)
{'default': 0.01, 'conditions': [], 'values': [0.01, 0.001, 0.0001], 'orde
red': True}
```

In [19]:

```
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test = tts(x,y,test_size=0.3,random_state=0)
```

In [20]:

```
tuner.search(x_train , y_train , epochs=5, validation_data=(x_test,y_test))
```

```
Trial 5 Complete [00h 00m 04s]
val_mean_absolute_error: 65.05960337320964

Best val_mean_absolute_error So Far: 64.90737406412761
Total elapsed time: 00h 00m 24s
INFO:tensorflow:Oracle triggered exit
```

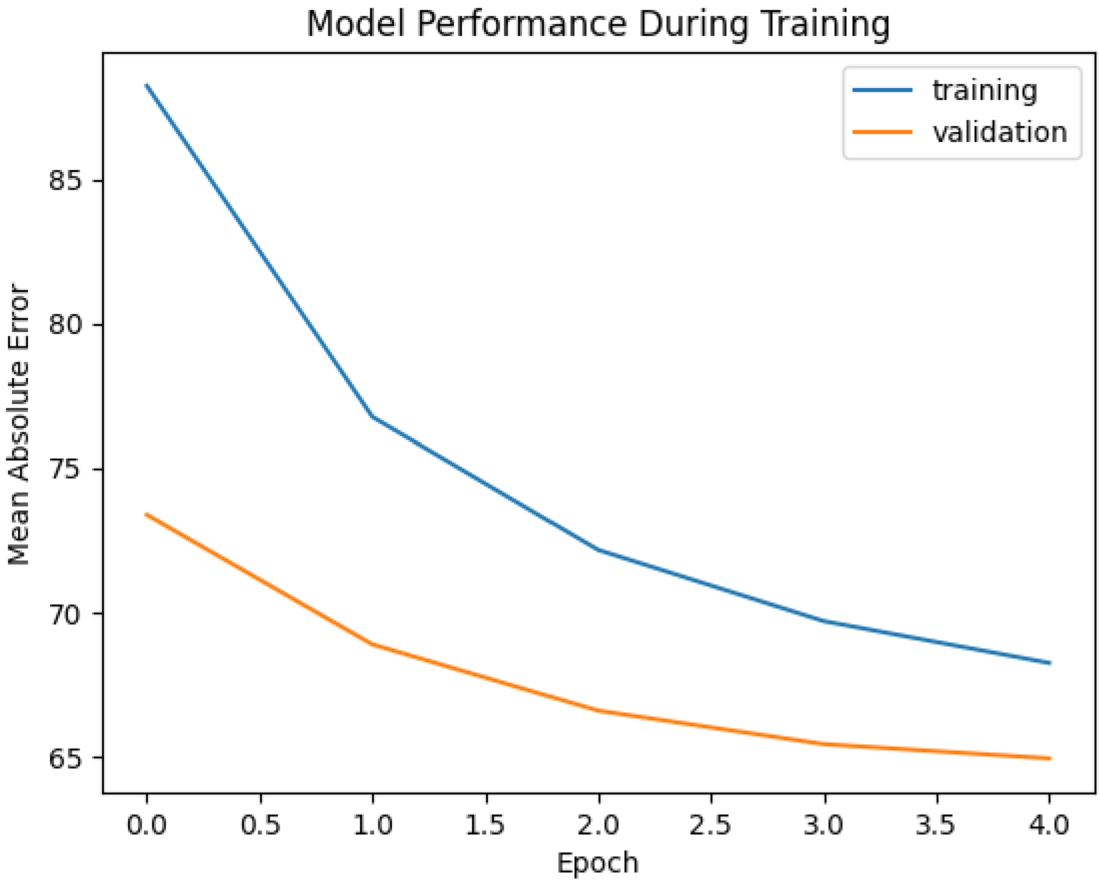
In [22]:

```
import matplotlib.pyplot as plt

#Get the best Hyperparameters found during the search
best_hps = tuner.get_best_hyperparameters(1)[0]

#Build the Model witht he best hyperparameters
model=hyper_tune(best_hps)

#Train the model with the best hyperparameters on the full training set
history = model.fit(x_train,y_train , epochs=5 ,validation_data = (x_test,y_
```

```
Epoch 1/5
24/24 [==============================] - 1s 7ms/step - loss: 88.2543 - mea
n_absolute_error: 88.2543 - val_loss: 73.3785 - val_mean_absolute_error: 7
3.3785
Epoch 2/5
24/24 [==============================] - 0s 4ms/step - loss: 76.7671 - mea
n_absolute_error: 76.7671 - val_loss: 68.8798 - val_mean_absolute_error: 6
8.8798
Epoch 3/5
24/24 [==============================] - 0s 4ms/step - loss: 72.1575 - mea
n_absolute_error: 72.1575 - val_loss: 66.5776 - val_mean_absolute_error: 6
6.5776
Epoch 4/5
24/24 [==============================] - 0s 5ms/step - loss: 69.6863 - mea
n_absolute_error: 69.6863 - val_loss: 65.4151 - val_mean_absolute_error: 6
5.4151
Epoch 5/5
24/24 [==============================] - 0s 4ms/step - loss: 68.2343 - mea
n_absolute_error: 68.2343 - val_loss: 64.9215 - val_mean_absolute_error: 6
4.9215
```

In [23]:

```
#Plot the Training and Validation Metrics for each Epoch
plt.plot(history.history['mean_absolute_error'] , label='training')
plt.plot(history.history['val_mean_absolute_error'] , label='validation')
plt.title('Model Performance During Training')
plt.xlabel('Epoch')
plt.ylabel('Mean Absolute Error')
plt.legend()
plt.show()
```



In [ ]:

In [1]:
```
import tensorflow as tf
from tensorflow import keras
```

In [2]:
```
# Loading the MNIST dataset -> 0 to 9 handwritten data
(x_train,y_train) , (x_test,y_test) = keras.datasets.mnist.load_data()
```

In [3]:
```
# Normalize the pixel values between 0 and 1
x_train = x_train / 255.0
x_test = x_test / 255.0
```

In [4]:
```
# Define the ANN Model Architecture

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),   # Convert the 28x28 Image i
    keras.layers.Dense(128,activation='relu'),   # Hidden Layer with 128 Uni
    keras.layers.Dense(10,activation='softmax')  #Output Layer with 10 units
])
```

In [5]:
```
#Compile the Model
model.compile(optimizer='adam' , loss='sparse_categorical_crossentropy' , me
```

In [6]:
```
history = model.fit(x_train,y_train,epochs=5,validation_data=(x_test,y_test)
```

```
Epoch 1/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2541 -
accuracy: 0.9270 - val_loss: 0.1301 - val_accuracy: 0.9601
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1116 -
accuracy: 0.9670 - val_loss: 0.0964 - val_accuracy: 0.9704
Epoch 3/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0777 -
accuracy: 0.9764 - val_loss: 0.0850 - val_accuracy: 0.9743
Epoch 4/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0586 -
accuracy: 0.9828 - val_loss: 0.0805 - val_accuracy: 0.9753
Epoch 5/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0451 -
accuracy: 0.9860 - val_loss: 0.0794 - val_accuracy: 0.9758
```

In [7]:
```
# Predict the Labels of the test Set
import numpy as np
y_pred = model.predict(x_test)
y_pred = np.argmax(y_pred, axis=1)
```

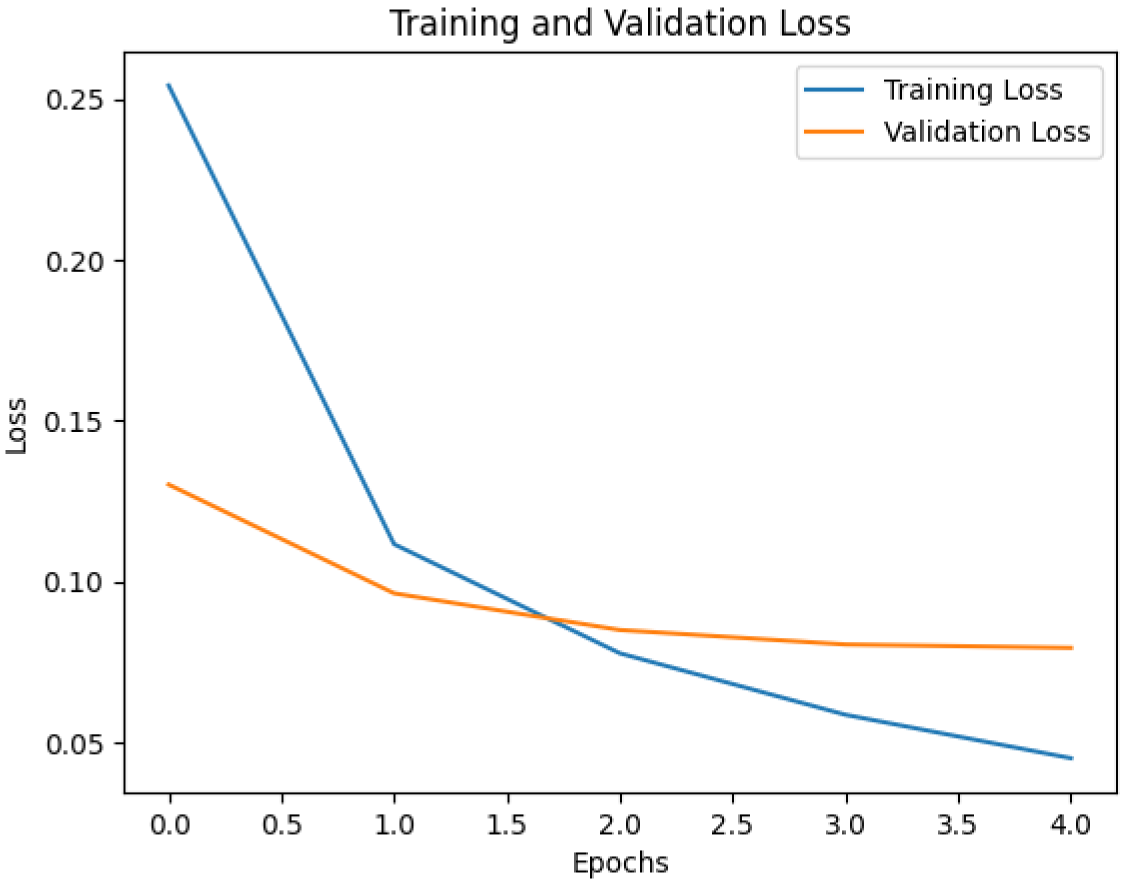
```
313/313 [==============================] - 0s 616us/step
```

In [8]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
# Print the Confusion Matrix
print('Confusion Matrix')
print(cm)

# Calculate the Accuracy
acc=accuracy_score(y_test,y_pred)

# Printing the Accuracy
print('Accuracy :',acc)
```

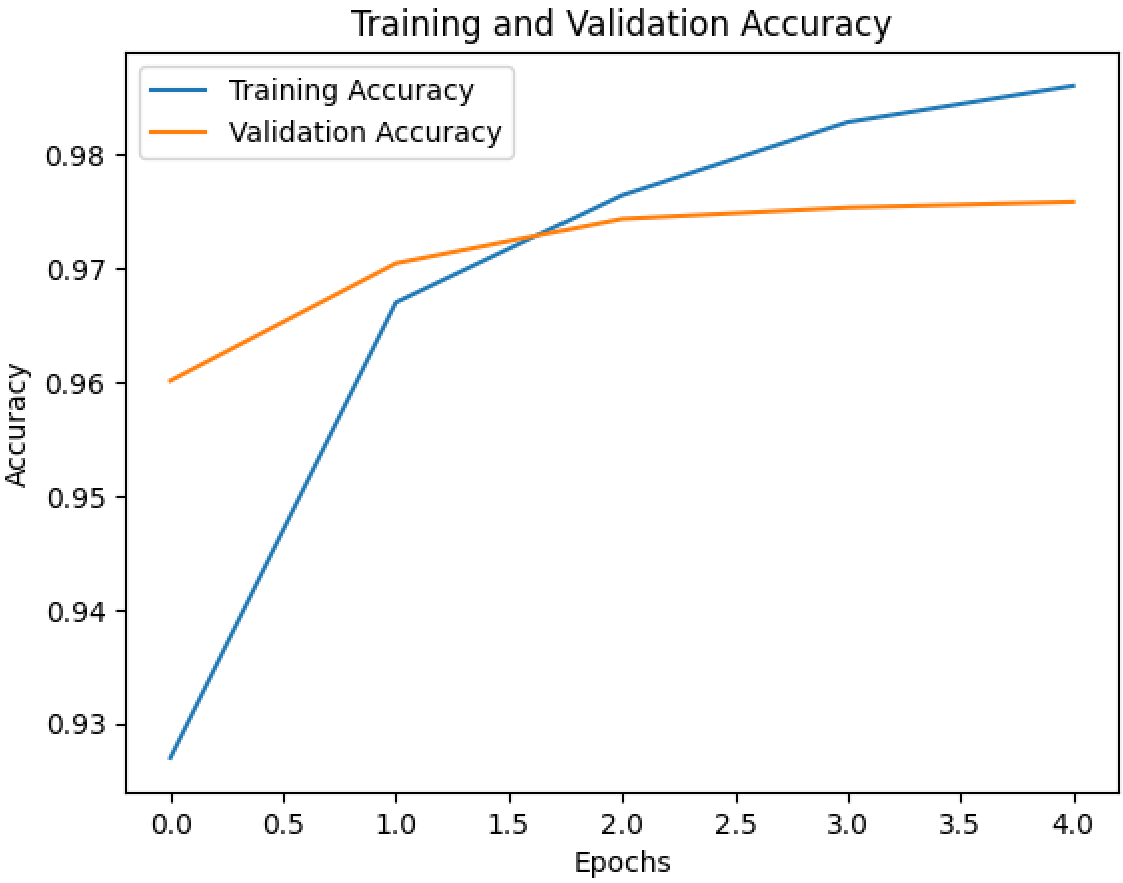
```
Confusion Matrix
[[ 973    0    1    2    0    0    1    1    2    0]
 [   0 1119    4    2    0    1    2    0    7    0]
 [   7    1  999    2    2    0    2    4   15    0]
 [   1    0    2  989    0    3    0    4    7    4]
 [   0    0    3    1  959    0    4    2    3   10]
 [   2    1    0    9    1  872    3    0    4    0]
 [   4    1    0    1    3    6  939    0    4    0]
 [   1    5    8    3    2    0    0  999    6    4]
 [   3    0    2    2    5    4    3    4  950    1]
 [   3    4    0    5    9    7    0   10   12  959]]
Accuracy : 0.9758
```

In [10]:
```
import matplotlib.pyplot as plt

# Plotting the Training and Validation Loss
plt.plot(history.history['loss'] , label='Training Loss')
plt.plot(history.history['val_loss'] , label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [11]:

```
# Plotting the Training and Validation Loss
plt.plot(history.history['accuracy'] , label='Training Accuracy')
plt.plot(history.history['val_accuracy'] , label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [ ]:

```

```

In [1]:

```
!pip install seaborn
!pip install mlxtend
```

```
Collecting seaborn
  Obtaining dependency information for seaborn from https://files.pythonho
sted.org/packages/7b/e5/83fcd7e9db036c179e0352bfcd20f81d728197a16f883e7b90
307a88e65e/seaborn-0.13.0-py3-none-any.whl.metadata (https://files.pythonh
osted.org/packages/7b/e5/83fcd7e9db036c179e0352bfcd20f81d728197a16f883e7b9
0307a88e65e/seaborn-0.13.0-py3-none-any.whl.metadata)
  Downloading seaborn-0.13.0-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\student\ap
pdata\roaming\python\python39\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=1.2 in c:\users\student\.conda\envs
\mallikarjuna\lib\site-packages (from seaborn) (2.1.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.3 in c:\users\student
\.conda\envs\mallikarjuna\lib\site-packages (from seaborn) (3.7.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seabor
n) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seabo
rn) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seabo
rn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seabor
n) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\student\.conda\en
vs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn)
(10.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seabor
n) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\student\.c
onda\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=3.3->se
aborn) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\stud
ent\.conda\envs\mallikarjuna\lib\site-packages (from matplotlib!=3.6.1,>=
3.3->seaborn) (6.0.1)
Requirement already satisfied: pytz>=2020.1 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from pandas>=1.2->seaborn) (2023.3.post
1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: zipp>=3.1.0 in c:\users\student\.conda\envs
\mallikarjuna\lib\site-packages (from importlib-resources>=3.2.0->matplotl
ib!=3.6.1,>=3.3->seaborn) (3.16.2)
Requirement already satisfied: six>=1.5 in c:\users\student\.conda\envs\ma
llikarjuna\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.
1,>=3.3->seaborn) (1.16.0)
Downloading seaborn-0.13.0-py3-none-any.whl (294 kB)
   ---------------------------------------- 0.0/294.6 kB ? eta -:--:--
   ---- ---------------------------------- 30.7/294.6 kB 660.6 kB/s eta 0:
00:01
   ---------- ---------------------------- 81.9/294.6 kB 919.0 kB/s eta 0:
00:01
   --------------- ---------------------- 122.9/294.6 kB 901.1 kB/s eta 0:
00:01
   ---------------------- --------------- 174.1/294.6 kB 958.1 kB/s eta 0:
00:01
```

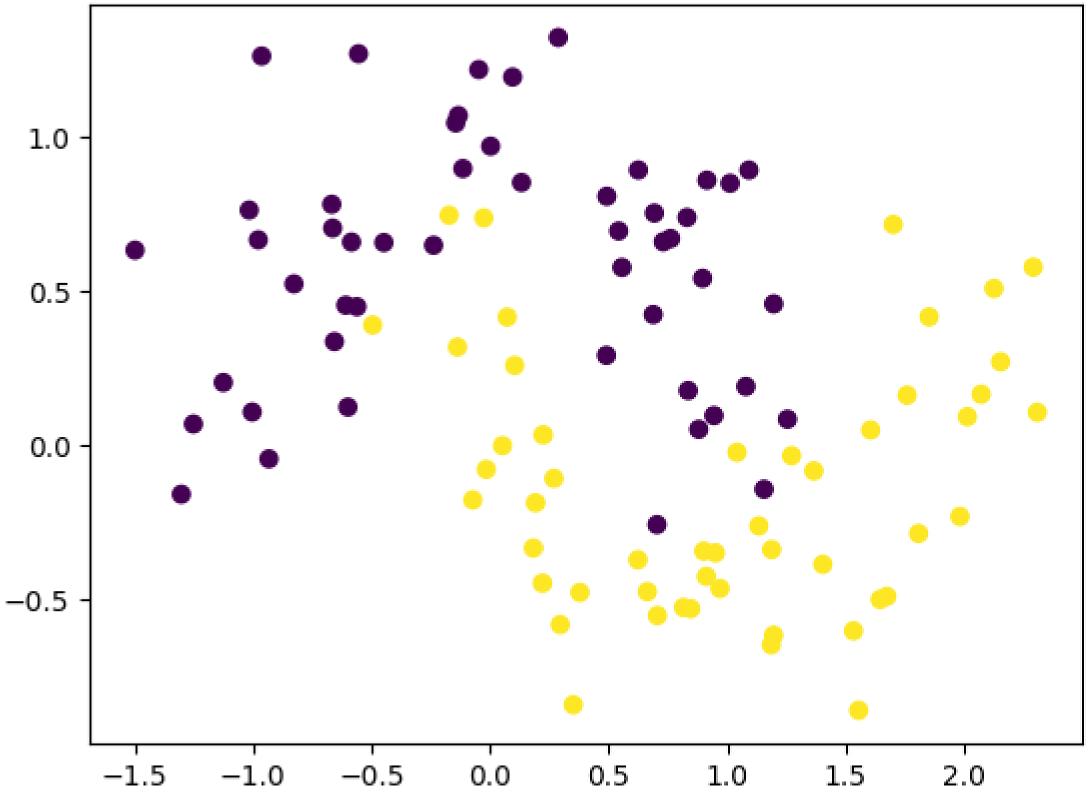
```
   ---------------------------- -------- 225.3/294.6 kB 986.4 kB/s eta 0:
00:01
   ---------------------------------- --- 266.2/294.6 kB 966.0 kB/s eta 0:
00:01
   -------------------------------------- 294.6/294.6 kB 960.6 kB/s eta 0:
00:00
Installing collected packages: seaborn
Successfully installed seaborn-0.13.0
Requirement already satisfied: mlxtend in c:\users\student\appdata\roaming
\python\python39\site-packages (0.23.0)
Requirement already satisfied: scipy>=1.2.1 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from mlxtend) (1.11.2)
Requirement already satisfied: numpy>=1.16.2 in c:\users\student\appdata\r
oaming\python\python39\site-packages (from mlxtend) (1.24.3)
Requirement already satisfied: pandas>=0.24.2 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from mlxtend) (2.1.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\student\.co
nda\envs\mallikarjuna\lib\site-packages (from mlxtend) (1.3.0)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from mlxtend) (3.7.3)
Requirement already satisfied: joblib>=0.13.2 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from mlxtend) (1.3.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.
1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.
0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2
3.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\student\.conda\en
vs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (10.0.
0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (3.
1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\student\.c
onda\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\stud
ent\.conda\envs\mallikarjuna\lib\site-packages (from matplotlib>=3.0.0->ml
xtend) (6.0.1)
Requirement already satisfied: pytz>=2020.1 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from pandas>=0.24.2->mlxtend) (2023.3.po
st1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\student\.c
onda\envs\mallikarjuna\lib\site-packages (from scikit-learn>=1.0.2->mlxten
d) (3.2.0)
Requirement already satisfied: zipp>=3.1.0 in c:\users\student\.conda\envs
\mallikarjuna\lib\site-packages (from importlib-resources>=3.2.0->matplotl
ib>=3.0.0->mlxtend) (3.16.2)
Requirement already satisfied: six>=1.5 in c:\users\student\.conda\envs\ma
```

```
llikarjuna\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.0.0
->mlxtend) (1.16.0)
```

In [50]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
import seaborn as sns
from mlxtend.plotting import plot_decision_regions
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
#from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam
```

In [51]:
```
X, y = make_moons(100, noise=0.25,random_state=2) # toy dataset with 2 featu
```

In [52]:
```
import matplotlib.pyplot as plt
plt.scatter(X[:,0], X[:,1], c=y) # to generates different colors with  binar
plt.show()
```

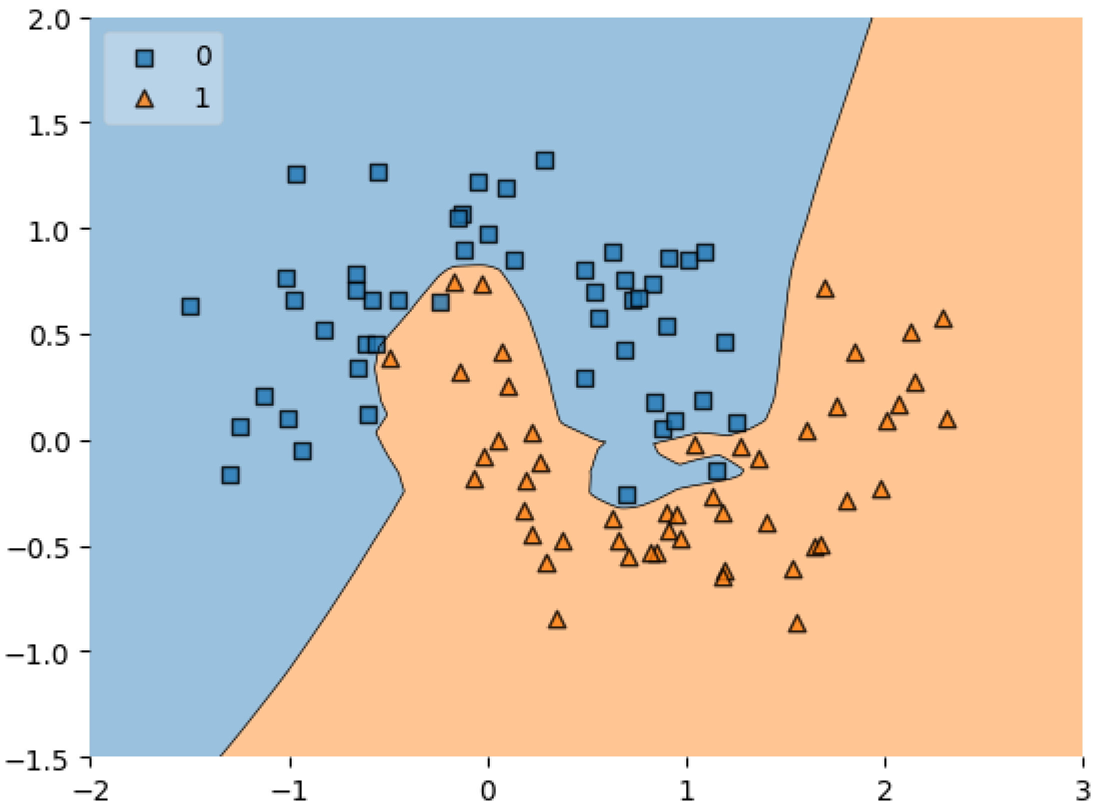


In [53]:
```
# Generate simple ANN network
model1 = Sequential()
model1.add(Dense(128,input_dim=2, activation="relu"))
model1.add(Dense(128, activation="relu"))
model1.add(Dense(1,activation='sigmoid'))
model1.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_27 (Dense)            (None, 128)               384       
                                                                 
 dense_28 (Dense)            (None, 128)               16512     
                                                                 
 dense_29 (Dense)            (None, 1)                 129       
                                                                 
=================================================================
Total params: 17025 (66.50 KB)
Trainable params: 17025 (66.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [54]:
```
adam = Adam(learning_rate=0.01)
model1.compile(loss='binary_crossentropy', optimizer=adam,
metrics=['accuracy'])
history1 = model1.fit(X, y, epochs=2000, validation_split =
0.2,verbose=0)
plot_decision_regions(X, y.astype('int'), clf=model1, legend=2) # X is for i
plt.xlim(-2,3) # sets the limits of the x-axis
plt.ylim(-1.5,2) # sets the limits of the y-axis
plt.show()
```

```
9600/9600 [==============================] - 4s 446us/step
```

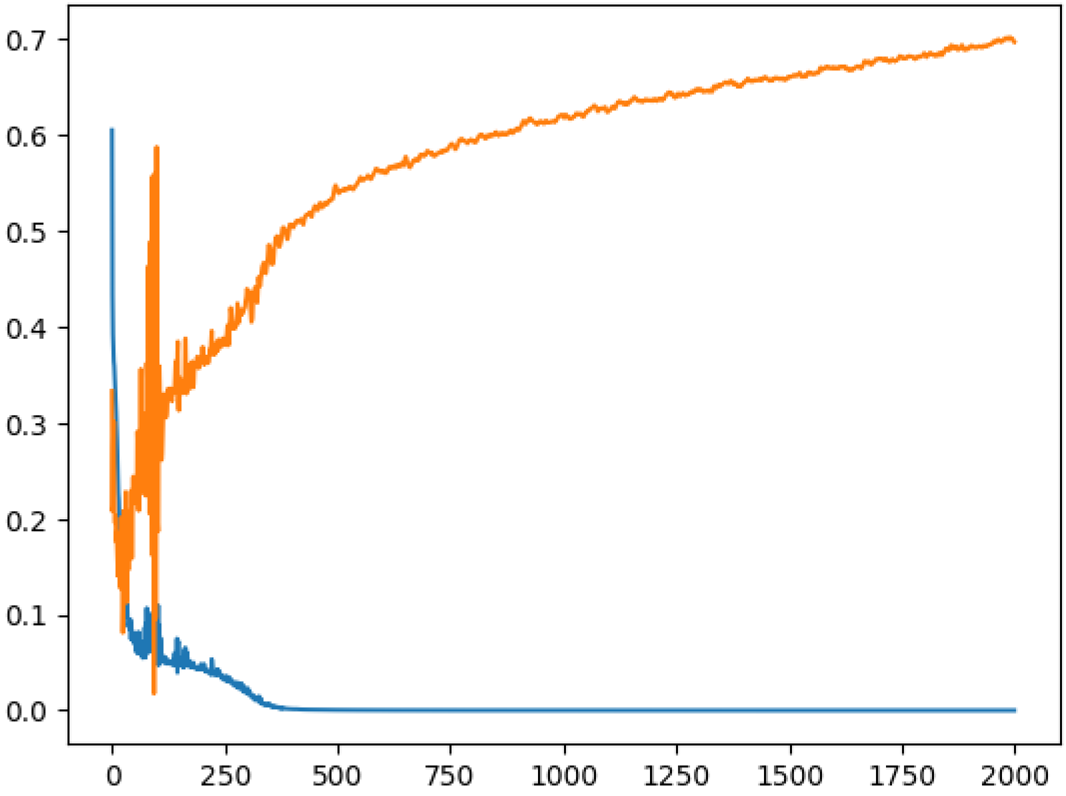


In [55]:

```
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
```

Out[55]:

```
[<matplotlib.lines.Line2D at 0x2b19abb8d60>]
```



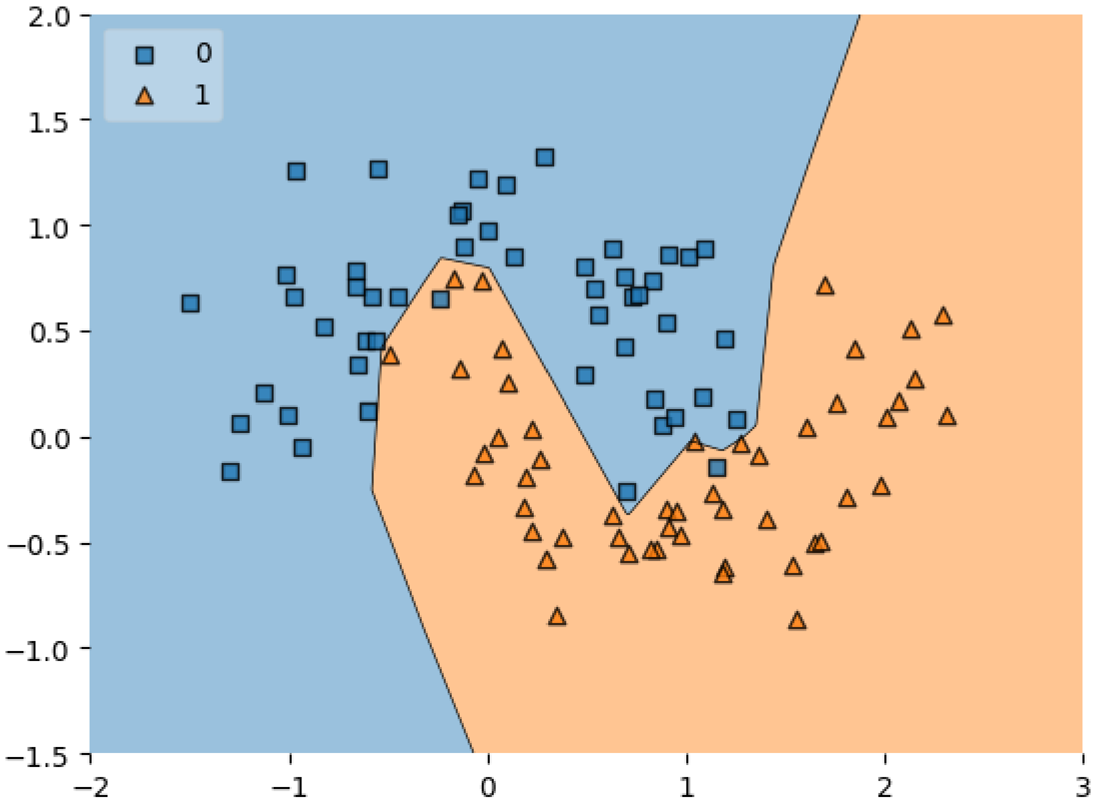
In [56]:

```
model2 = Sequential()
model2.add(Dense(128,input_dim=2, activation="relu",kernel_regularizer=tenso
model2.add(Dense(128, activation="relu",kernel_regularizer=tensorflow.keras.
model2.add(Dense(1,activation='sigmoid'))
model2.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_30 (Dense)            (None, 128)               384

 dense_31 (Dense)            (None, 128)               16512

 dense_32 (Dense)            (None, 1)                 129

=================================================================
Total params: 17025 (66.50 KB)
Trainable params: 17025 (66.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [57]:
```
adam = Adam(learning_rate=0.01)
model2.compile(loss='binary_crossentropy', optimizer=adam,
metrics=['accuracy'])
history2 = model2.fit(X, y, epochs=2000, validation_split =
0.2,verbose=0)
plot_decision_regions(X, y.astype('int'), clf=model2, legend=2)
plt.xlim(-2,3)
plt.ylim(-1.5,2)
plt.show()
```

```
9600/9600 [==============================] - 4s 446us/step
```

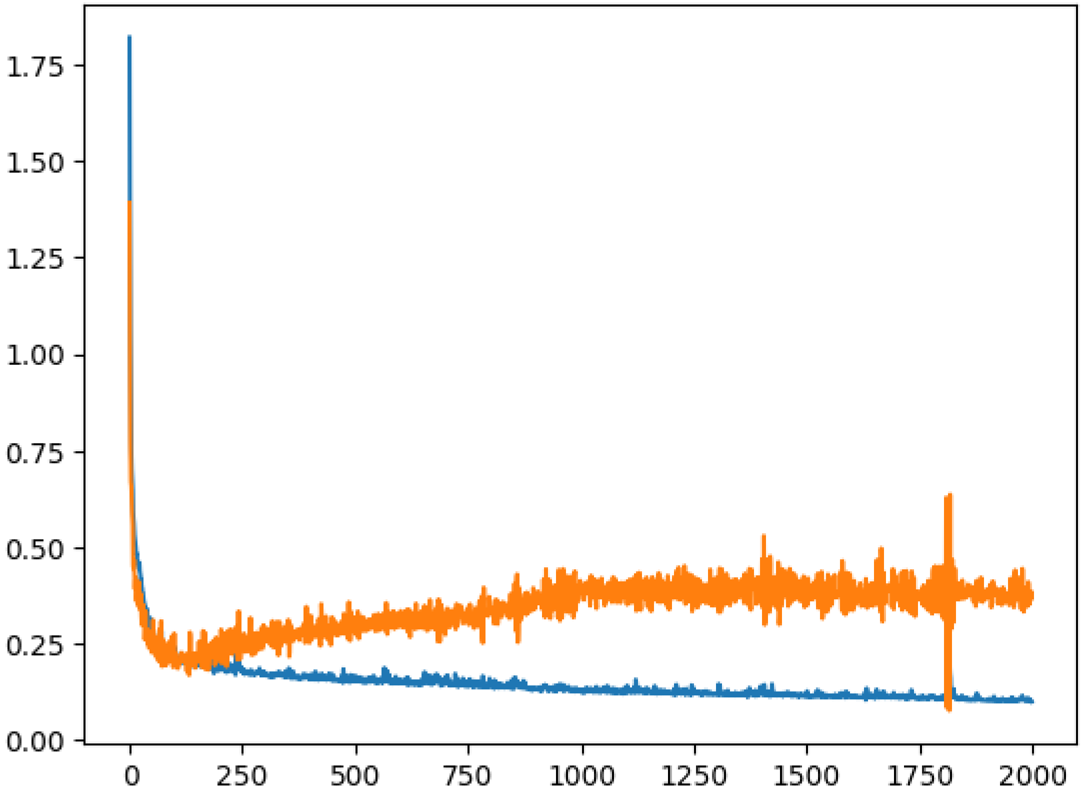


In [58]:

```
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
```

Out[58]:

```
[<matplotlib.lines.Line2D at 0x2b196004070>]
```



In [59]:

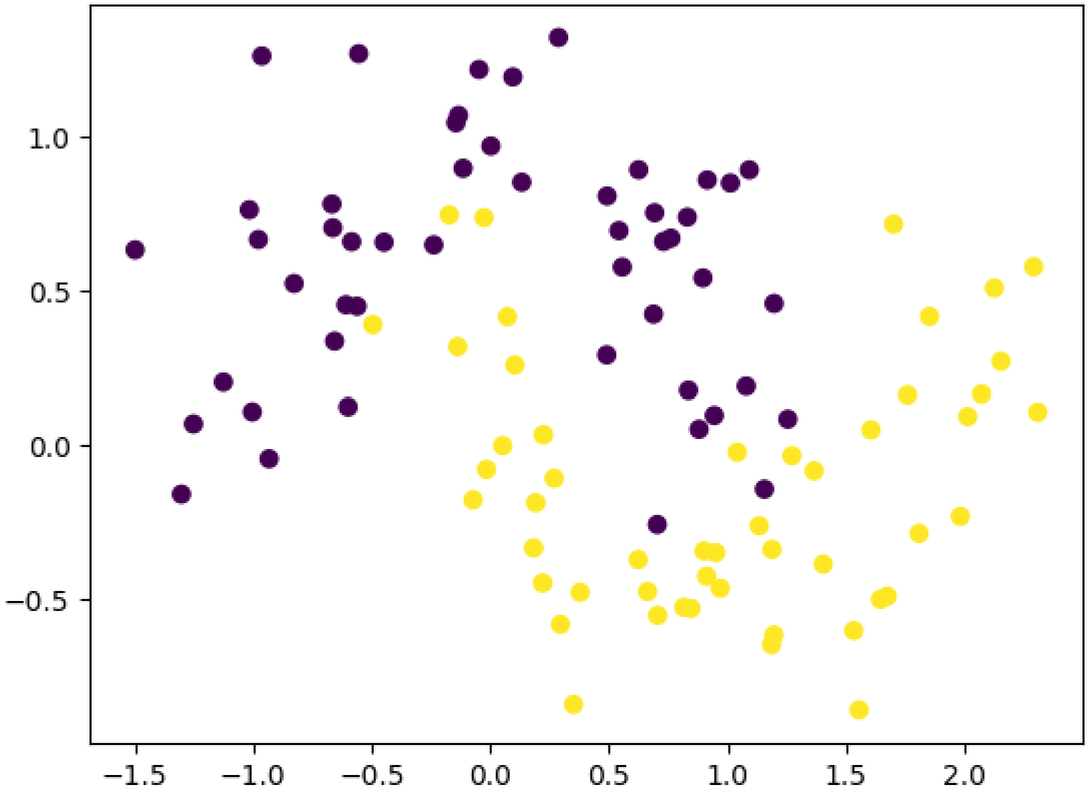
```
# Calculation of accuarcy of each model
# Calculate the accuracy for model1
acc_model1 = history1.history['accuracy'][-1] * 100
# Calculate the accuracy for model2
acc_model2 = history2.history['accuracy'][-1] * 100
print(f"Accuracy for Model 1: {acc_model1:.2f}%")
print(f"Accuracy for Model 2: {acc_model2:.2f}%")
```

```
Accuracy for Model 1: 100.00%
Accuracy for Model 2: 98.75%
```

In [ ]:

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
import seaborn as sns
from mlxtend.plotting import plot_decision_regions
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam
```

In [9]:
```
X, y = make_moons(100, noise=0.25, random_state=2)
# Visualize the data
plt.scatter(X[:,0], X[:,1], c=y)
plt.show()
```



In [10]:

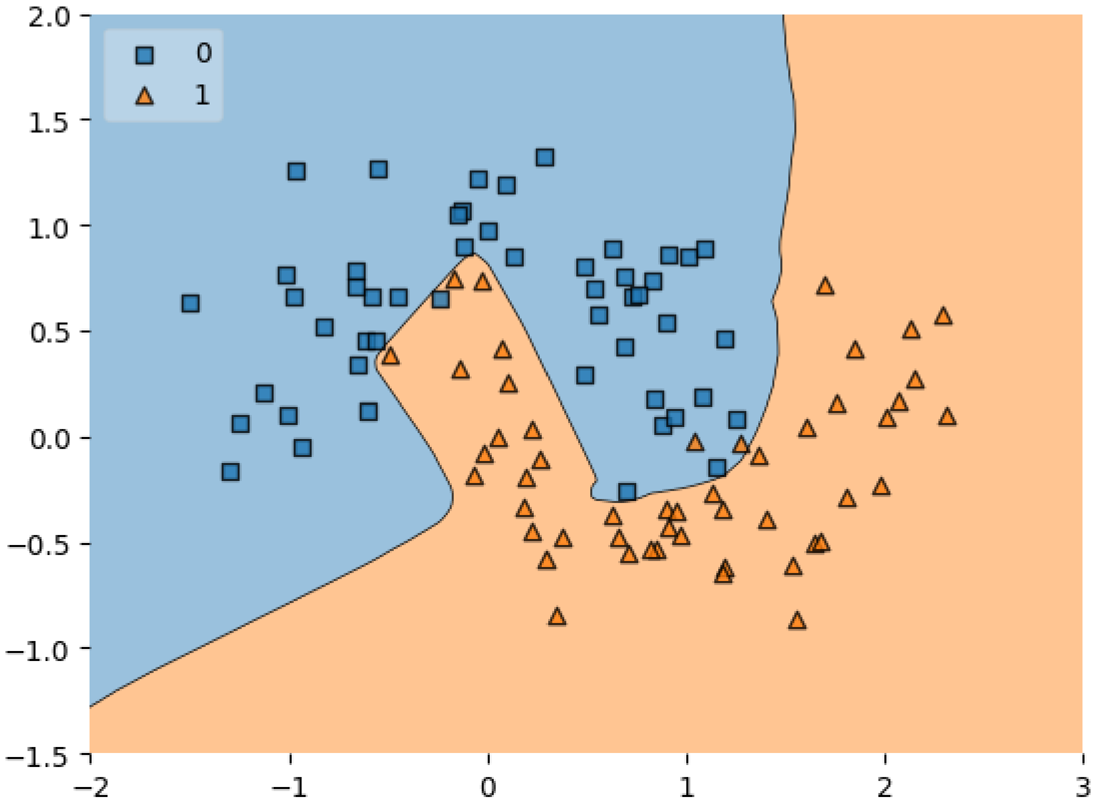
```
# Build the model with dropout layers
model = Sequential()
model.add(Dense(128, input_dim=2, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(128, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_3 (Dense)             (None, 128)               384       
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_4 (Dense)             (None, 128)               16512     
                                                                 
 dropout_3 (Dropout)         (None, 128)               0         
                                                                 
 dense_5 (Dense)             (None, 1)                 129       
                                                                 
=================================================================
Total params: 17025 (66.50 KB)
Trainable params: 17025 (66.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [13]:

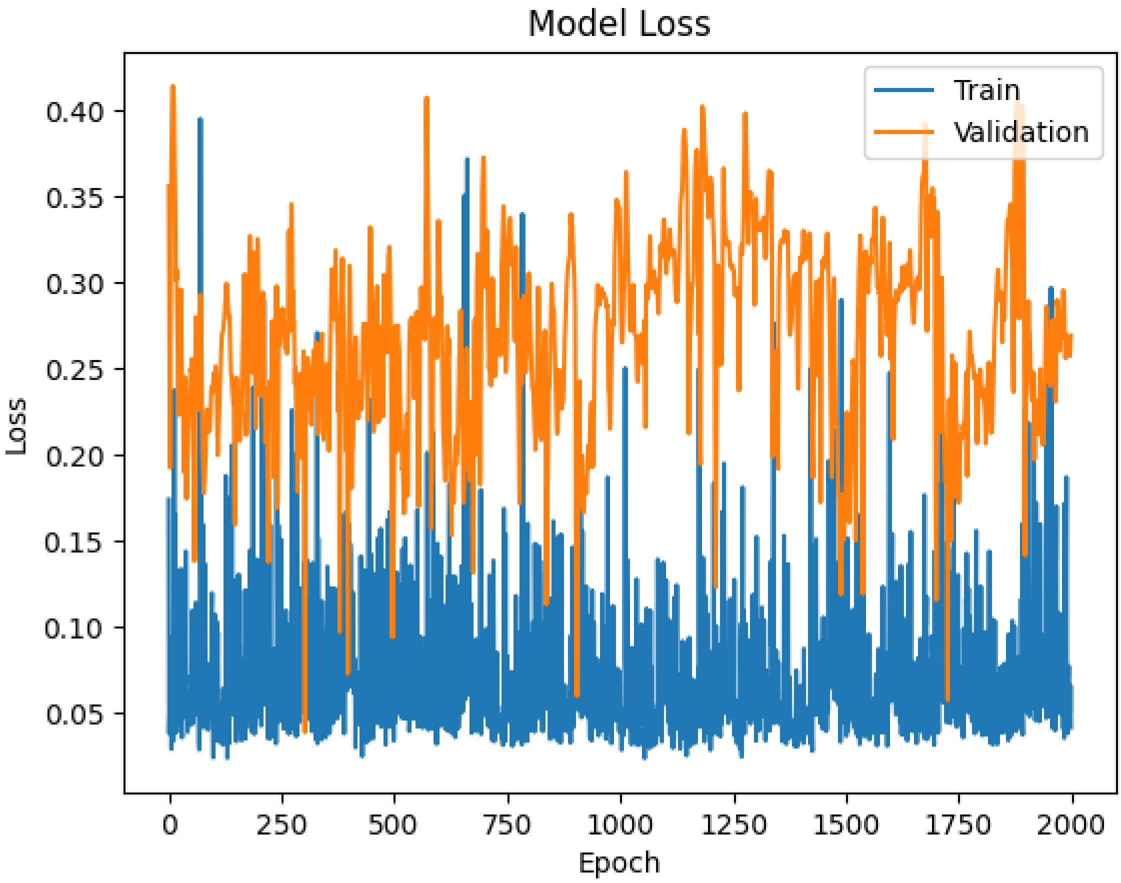
```
adam = Adam(learning_rate=0.01)
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy
history = model.fit(X, y, epochs=2000, validation_split=0.2, verbose=0)
# Visualize the decision boundary
plot_decision_regions(X, y.astype('int'), clf=model, legend=2)
plt.xlim(-2,3)
plt.ylim(-1.5,2)
plt.show()
```

```
9600/9600 [==============================] - 5s 486us/step
```



In [14]:

```
# Plot the loss curve
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



In [16]:

```
 # Calculation of accuarcy of each model
# Calculate the accuracy for model1
acc_model1 = history.history['accuracy'][-1] * 100
acc_model1
```

Out[16]:

```
97.50000238418579
```

In [ ]:

In [19]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [20]:
```
df=pd.read_csv("sentiment.csv")
df.head(20)
```

Out[20]:

	Index	message to examine	label (depression result)
0	106	just had a real good moment. i missssssssss hi...	0
1	217	is reading manga http://plurk.com/p/mzp1e	0
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0
3	288	@lapcat Need to send 'em to my accountant tomo...	0
4	540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0
5	624	so sleepy. good times tonight though	0
6	701	@SilkCharm re: #nbn as someone already said, d...	0
7	808	23 or 24ï¿½C possible today. Nice	0
8	1193	nite twitterville workout in the am -ciao	0
9	1324	@daNanner Night, darlin'! Sweet dreams to you	0
10	1332	Good morning everybody!	0
11	1368	Finally! I just created my WordPress Blog. The...	0
12	1578	kisha they cnt get over u til they get out frm...	0
13	1595	@nicolerichie Yes i remember that band, It was...	0
14	1861	I really love reflections and shadows	0
15	1889	@blueaero ooo it's fantasy? i like fantasy no...	0
16	1899	@rokchic28 no probs, I sell nothing other than...	0
17	1919	@shipovalov &quot;NOKLA connecting people&quot...	0
18	1992	Once again stayed up to late and have to start...	0
19	2097	@Kal_Penn I just read about your new job, CONG...	0

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
Index                        0
message to examine           0
label (depression result)    0
dtype: int64
```

In [22]:
```
tfidf_vectorizer=TfidfVectorizer(max_features=5000)
x=tfidf_vectorizer.fit_transform(df['message to examine'])
y=df['label (depression result)']
```

In [23]:
```
x=x.toarray()
```

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_stat
```

In [25]:
```
model=keras.Sequential([ keras.layers.Dense(128,activation='relu',input_shap
                        keras.layers.Dense(64,activation='relu'),
                        keras.layers.Dense(64,activation='relu'),
                        keras.layers.Dense(64,activation='relu'),
                        keras.layers.Dense(64,activation='relu'),
                        keras.layers.Dense(1,activation='sigmoid'),

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

In [29]:
```
history=model.fit(x_train,y_train,epochs=5,batch_size=32,validation_data=(x_
```

```
Epoch 1/5
258/258 [==============================] - 3s 8ms/step - loss: 0.2193 - ac
curacy: 0.9013 - val_loss: 0.0710 - val_accuracy: 0.9792
Epoch 2/5
258/258 [==============================] - 2s 7ms/step - loss: 0.0161 - ac
curacy: 0.9944 - val_loss: 0.0812 - val_accuracy: 0.9796
Epoch 3/5
258/258 [==============================] - 2s 7ms/step - loss: 0.0036 - ac
curacy: 0.9993 - val_loss: 0.0721 - val_accuracy: 0.9811
Epoch 4/5
258/258 [==============================] - 2s 7ms/step - loss: 0.0021 - ac
curacy: 0.9998 - val_loss: 0.0780 - val_accuracy: 0.9835
Epoch 5/5
258/258 [==============================] - 2s 7ms/step - loss: 0.0019 - ac
curacy: 0.9998 - val_loss: 0.1011 - val_accuracy: 0.9806
```

In [30]:
```
model.save("senti.keras")
```

In [32]:
```
loaded_model=keras.models.load_model('senti.keras')
loaded_model
```

Out[32]:
```
<keras.src.engine.sequential.Sequential at 0x27ac2f54910>
```

In [ ]:

In [ ]:

In [1]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense
from keras.callbacks import EarlyStopping,ModelCheckpoint
import matplotlib.pyplot as plt
```

In [2]:
```
data=datasets.load_digits()
x=data.images
y=data.target
```

In [3]:
```
x=x.reshape((x.shape[0],8,8,1))
x=x.astype('float32')/255
y=to_categorical(y)
```

In [4]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [5]:
```
model=Sequential()
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(8,8,1))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

In [6]:
```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['acc
```

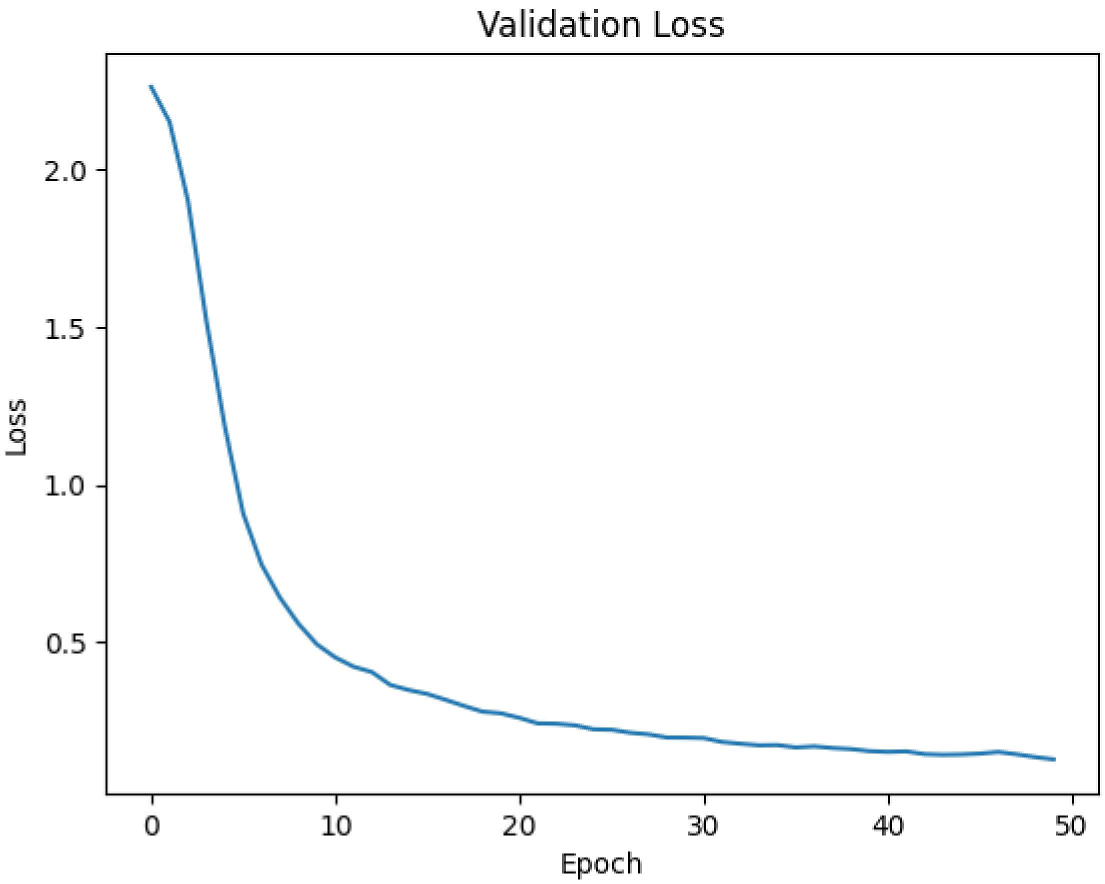
In [7]:
```
earlystop=EarlyStopping(monitor='val_loss',patience=10)
best_weights=ModelCheckpoint('best_weights.h5',save_best_only=True,monitor='
```

In [8]:
```
history=model.fit(x_train,y_train,epochs=50,batch_size=32,validation_data=(x
                  callbacks=[earlystop,best_weights])
```

```
Epoch 1/50
45/45 [==============================] - 1s 8ms/step - loss: 2.2863 - ac
curacy: 0.2394 - val_loss: 2.2626 - val_accuracy: 0.4306
Epoch 2/50
45/45 [==============================] - 0s 3ms/step - loss: 2.2159 - ac
curacy: 0.4934 - val_loss: 2.1515 - val_accuracy: 0.4972
Epoch 3/50

C:\Users\Student\AppData\Roaming\Python\Python39\site-packages\keras\src
\engine\training.py:3000: UserWarning: You are saving your model as an H
DF5 file via `model.save()`. This file format is considered legacy. We r
ecommend using instead the native Keras format, e.g. `model.save('my_mod
el.keras')`.
  saving_api.save_model(

45/45 [==============================] - 0s 4ms/step - loss: 2.0349 - ac
curacy: 0.6757 - val_loss: 1.9010 - val_accuracy: 0.5472
Epoch 4/50
45/45 [==============================] - 0s 4ms/step - loss: 1.7072 - ac
curacy: 0.6729 - val_loss: 1.5203 - val_accuracy: 0.7472
```

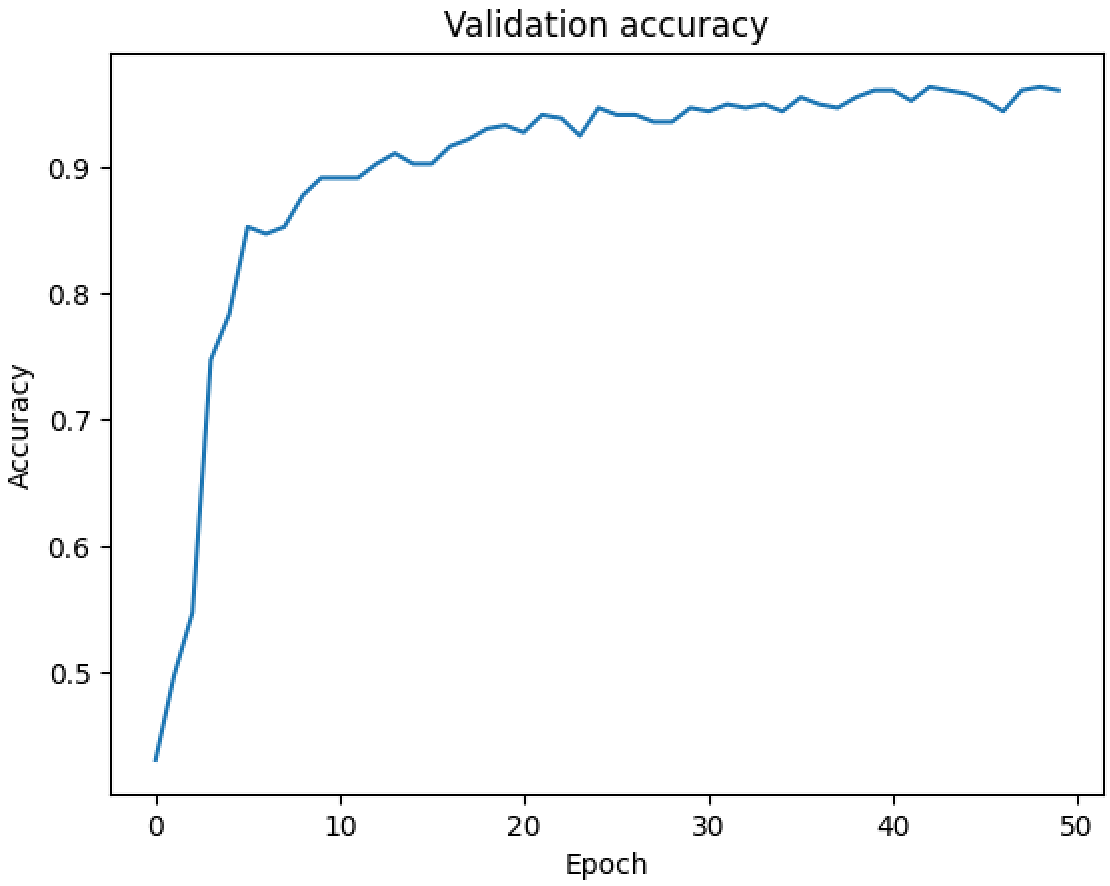
In [9]:

```
plt.plot(history.history['val_loss'])
plt.title('Validation Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```



In [10]:

```
plt.plot(history.history['val_accuracy'])
plt.title('Validation accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```



In [11]:

```
model.load_weights("best_weights.h5")
```

In [12]:

```
test_loss,test_acc=model.evaluate(x_test,y_test,verbose=0)
print("Test Loss:",test_loss)
print("Test Accuracy:",test_acc)
```

```
Test Loss: 0.1287679523229599
Test Accuracy: 0.9611111283302307
```

In [ ]:

In [3]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:

```
model=keras.Sequential(
[
    layers.Conv2D(32,(3,3),activation="relu",padding="same",input_shape=(28,
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dense(10,activation="softmax"),

])
```

In [4]:

```
(x_train,y_train),(x_test,y_test)=keras.datasets.mnist.load_data()

#Scale the images to [0,1] range
x_train = x_train.astype("float32")/255
x_test = x_test.astype("float32")/255

#Add a channel dimension to the images
x_train = np.expand_dims(x_train,-1)
x_test = np.expand_dims(x_test,-1)

#Split the training set into training and validationsets
x_train,x_val=x_train[:50000],x_train[50000:]
y_train,y_val=y_train[:50000],y_train[50000:]
```

In [5]:

```
datagen=ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    horizontal_flip=False,
    vertical_flip=False,
)
```

In [6]:

```
#without data augmentation
model.compile(loss="sparse_categorical_crossentropy",optimizer="adam",metric
history1=model.fit(x_train,y_train,batch_size=32,epochs=5,validation_data=(x
```

```
Epoch 1/5
1563/1563 [==============================] - 7s 4ms/step - loss: 0.2310 -
accuracy: 0.9345 - val_loss: 0.1003 - val_accuracy: 0.9731
Epoch 2/5
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0868 -
accuracy: 0.9748 - val_loss: 0.0810 - val_accuracy: 0.9780
Epoch 3/5
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0644 -
accuracy: 0.9802 - val_loss: 0.0709 - val_accuracy: 0.9785
Epoch 4/5
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0528 -
accuracy: 0.9838 - val_loss: 0.0748 - val_accuracy: 0.9803
Epoch 5/5
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0445 -
accuracy: 0.9865 - val_loss: 0.0716 - val_accuracy: 0.9788
```

In [7]:

```
#with data augmentation
model.compile(loss="sparse_categorical_crossentropy",optimizer="adam",metric
history2=model.fit(datagen.flow(x_train,y_train,batch_size=32),epochs=5,vali
```

```
Epoch 1/5
1563/1563 [==============================] - 9s 6ms/step - loss: 0.3067 -
accuracy: 0.9091 - val_loss: 0.0909 - val_accuracy: 0.9745
Epoch 2/5
1563/1563 [==============================] - 9s 6ms/step - loss: 0.2064 -
accuracy: 0.9369 - val_loss: 0.0780 - val_accuracy: 0.9773
Epoch 3/5
1563/1563 [==============================] - 9s 6ms/step - loss: 0.1842 -
accuracy: 0.9431 - val_loss: 0.1015 - val_accuracy: 0.9726
Epoch 4/5
1563/1563 [==============================] - 8s 5ms/step - loss: 0.1648 -
accuracy: 0.9504 - val_loss: 0.1111 - val_accuracy: 0.9679
Epoch 5/5
1563/1563 [==============================] - 8s 5ms/step - loss: 0.1531 -
accuracy: 0.9534 - val_loss: 0.1421 - val_accuracy: 0.9614
```

In [8]:

```
#without data augmentation
model.evaluate(x_test,y_test)

#with data augmentation
model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.1287 - ac
curacy: 0.9637
313/313 [==============================] - 1s 2ms/step - loss: 0.1287 - ac
curacy: 0.9637
```

Out[8]:

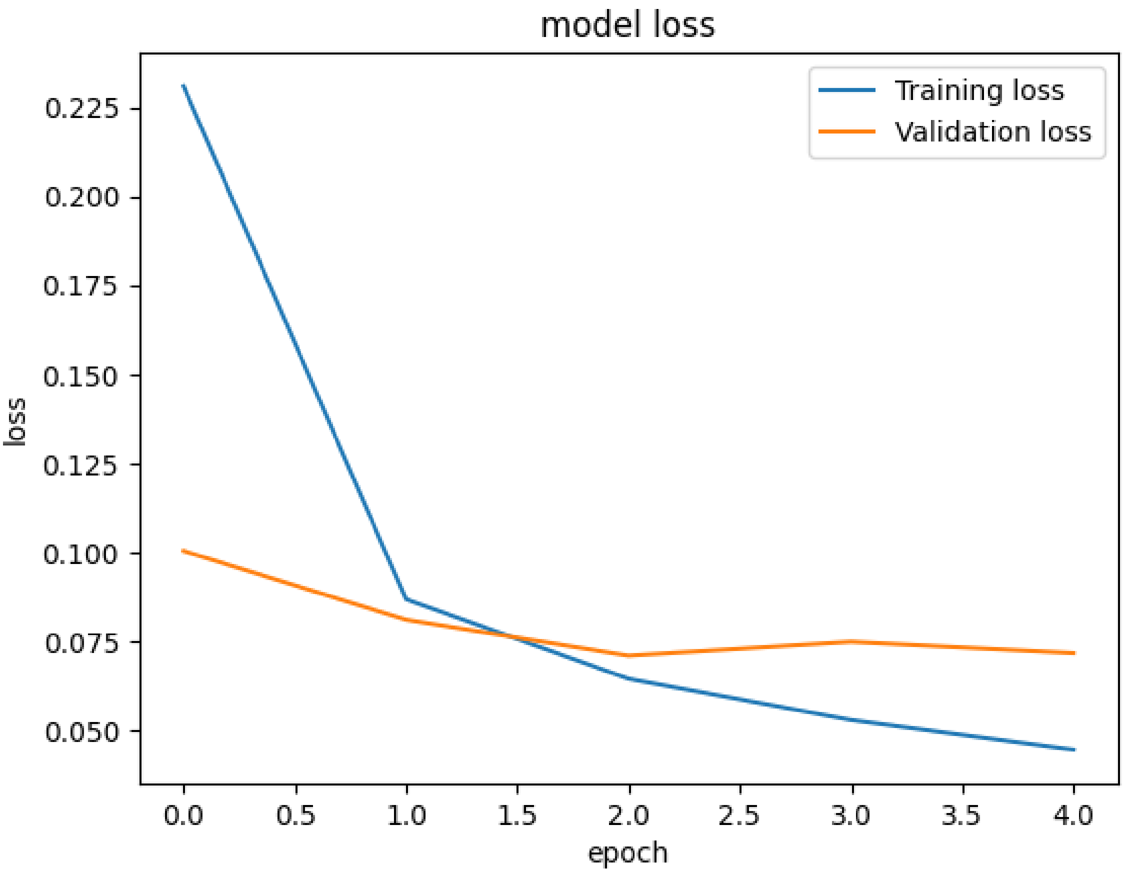
```
[0.1286708116531372, 0.963699996471405]
```

In [9]:

```
import matplotlib.pyplot as plt
plt.plot(history1.history['loss'],label="Training loss")
plt.plot(history1.history['val_loss'],label='Validation loss')
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show
```

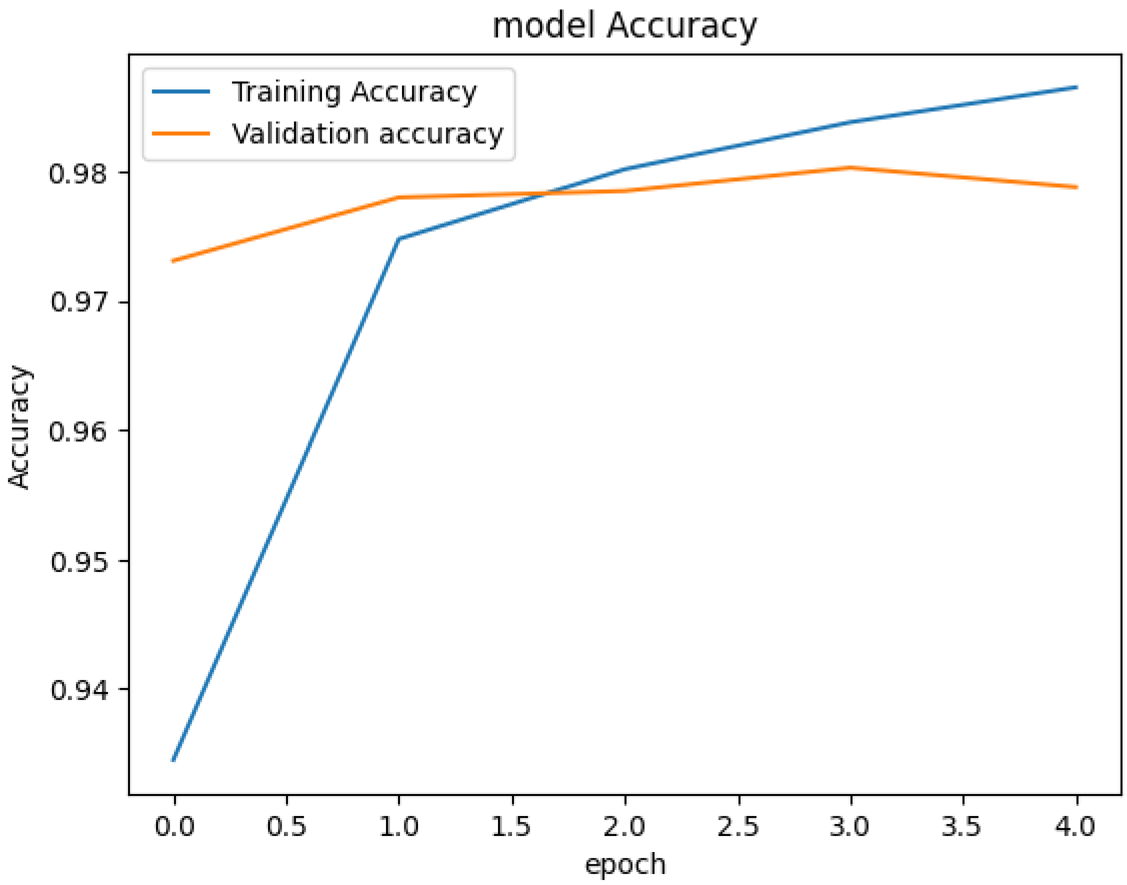
Out[9]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [10]:
```
plt.plot(history1.history['accuracy'],label="Training Accuracy")
plt.plot(history1.history['val_accuracy'],label='Validation accuracy')
plt.title('model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend()
plt.show
```

Out[10]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]:
```

```

In [39]:

```
import numpy as np
import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import VGG16,VGG19,ResNet50
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
```

In [40]:

```
(x_train,y_train),(x_test,y_test)=cifar10.load_data()
```

In [41]:

```
x_train=x_train.astype("float32")/255.0
x_test=x_test.astype("float32")/255.0
```

In [42]:

```
y_train=to_categorical(y_train,10)
y_test=to_categorical(y_test,10)
```

In [43]:

```
vgg16=VGG16(weights="imagenet",include_top=False,input_shape=(32,32,3))
vgg19=VGG19(weights="imagenet",include_top=False,input_shape=(32,32,3))
resnet=ResNet50(weights="imagenet",include_top=False,input_shape=(32,32,3))
```

In [44]:

```
vgg16_output=layers.GlobalAveragePooling2D()(vgg16.output)
vgg16_output=layers.Dense(10,activation="softmax")(vgg16_output)

vgg19_output=layers.GlobalAveragePooling2D()(vgg19.output)
vgg19_output=layers.Dense(10,activation="softmax")(vgg19_output)

resnet_output=layers.GlobalAveragePooling2D()(resnet.output)
resnet_output=layers.Dense(10,activation="softmax")(resnet_output)
```

In [45]:

```
vgg16_model=keras.Model(inputs=vgg16.input,outputs=vgg16_output)
vgg19_model=keras.Model(inputs=vgg19.input,outputs=vgg19_output)
resnet_model=keras.Model(inputs=resnet.input,outputs=resnet_output)
```

In [46]:

```
vgg16_model.compile(loss="categorical_crossentropy",optimizer="adam",metrics
vgg19_model.compile(loss="categorical_crossentropy",optimizer="adam",metrics
resnet_model.compile(loss="categorical_crossentropy",optimizer="adam",metric
```

In [47]:

```
vgg16_loss,vgg16_accuracy=vgg16_model.evaluate(x_test,y_test)
vgg19_loss,vgg19_accuracy=vgg19_model.evaluate(x_test,y_test)
resnet_loss,resnet_accuracy=resnet_model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 13s 41ms/step - loss: 2.5819 -
accuracy: 0.1063
313/313 [==============================] - 16s 52ms/step - loss: 2.6089 -
accuracy: 0.0972
313/313 [==============================] - 15s 44ms/step - loss: 3.9080 -
accuracy: 0.0997
```

In [48]:

```
print("VGG16 Test Accuracy: ",vgg16_accuracy)
print("VGG19 Test Accuracy: ",vgg19_accuracy)
print("Resnet Test Accuracy: ",resnet_accuracy)
```

```
VGG16 Test Accuracy:  0.1062999963760376
VGG19 Test Accuracy:  0.09719999879598618
Resnet Test Accuracy:  0.09969999641180038
```

In [50]:
```
import numpy as np
import pandas as pd
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Embedding,LSTM
from sklearn.model_selection import train_test_split
```

In [51]:
```
df=pd.read_csv("sentiment.csv")
df.head()
```

Out[51]:

	Index	message to examine	label (depression result)
0	106	just had a real good moment. i missssssssss hi...	0
1	217	is reading manga http://plurk.com/p/mzp1e	0
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0
3	288	@lapcat Need to send 'em to my accountant tomo...	0
4	540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0

In [52]:
```
text=df.iloc[:,1]
```

In [53]:
```
labels=df.iloc[:,-1]
```

In [54]:
```
tokenizer=Tokenizer()
tokenizer.fit_on_texts(text)
sequences=tokenizer.texts_to_sequences(text)
data=pad_sequences(sequences)
```

In [55]:
```
x_train,x_test,y_train,y_test=train_test_split(data,labels,test_size=0.2,rar
model=Sequential()
model.add(Embedding(len(tokenizer.word_index)+1,32,input_length=data.shape[1
model.add(LSTM(64))
model.add(Dense(1,activation='sigmoid'))
```

In [56]:
```
y_train=np.asarray(y_train,dtype=float)
y_test=np.asarray(y_test,dtype=float)
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy
```

In [59]:
```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=1)
```

```
258/258 [==============================] - 6s 21ms/step - loss: 0.2124 - a
ccuracy: 0.9169 - val_loss: 0.0234 - val_accuracy: 0.9947
```

Out[59]:
```
<keras.src.callbacks.History at 0x1b738c16370>
```

In [60]:
```
new_text="I had a terrible experience with the product and services"
new_sequence=tokenizer.texts_to_sequences([new_text])
new_data=pad_sequences(new_sequence,maxlen=data.shape[1])
predicted_sentiment=model.predict(new_data)
```

```
1/1 [==============================] - 0s 177ms/step
```

In [64]:

```
scores=model.evaluate(x_test,y_test)
print("Accuracy : ",scores[1])
```

```
65/65 [==============================] - 0s 6ms/step - loss: 0.0234 - accu
racy: 0.9947
Accuracy :  0.9946679472923279
```

In [ ]:

Using Basic RNN

In [1]:

```
import tensorflow as tf
import numpy as np
```

In [2]:

```
# Sample text data
text = "This is a sample text used to demonstrate predictive text with basic
```

In [3]:

```
# Preprocess the text and create a vocabulary
tokenizer = tf.keras.layers.TextVectorization()
tokenizer.adapt(text.split())
```

In [5]:

```
# Convert text to sequences of token indices
text_sequences = tokenizer(text)
text_sequences
```

Out[5]:

```
<tf.Tensor: shape=(24,), dtype=int64, numpy=
array([ 2, 17, 23, 12,  3,  8, 10, 20, 14,  3,  6, 21, 13, 18,  2, 19,  7,
       15, 11, 16,  5, 22,  4,  9], dtype=int64)>
```

In [6]:

```
# Create training data (X) and target data (y)
X = text_sequences[:-1]
y = text_sequences[1:]
```

In [7]:

```
# Build a basic RNN model using Keras
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.get_vocabulary()), out
    tf.keras.layers.SimpleRNN(128, return_sequences=True),
    tf.keras.layers.Dense(len(tokenizer.get_vocabulary()), activation='softm
])
```

In [8]:

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')
```

In [9]:
```
# Train the model
model.fit(X, y, epochs=50)
```

```
Epoch 1/50
1/1 [==============================] - 4s 4s/step - loss: 3.1775
Epoch 2/50
1/1 [==============================] - 0s 15ms/step - loss: 3.1636
Epoch 3/50
1/1 [==============================] - 0s 11ms/step - loss: 3.1497
Epoch 4/50
1/1 [==============================] - 0s 5ms/step - loss: 3.1358
Epoch 5/50
1/1 [==============================] - 0s 7ms/step - loss: 3.1219
Epoch 6/50
1/1 [==============================] - 0s 11ms/step - loss: 3.1078
Epoch 7/50
1/1 [==============================] - 0s 4ms/step - loss: 3.0936
Epoch 8/50
1/1 [==============================] - 0s 5ms/step - loss: 3.0793
Epoch 9/50
1/1 [==============================] - 0s 15ms/step - loss: 3.0647
Epoch 10/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0499
Epoch 11/50
1/1 [==============================] - 0s 1ms/step - loss: 3.0348
Epoch 12/50
1/1 [==============================] - 0s 6ms/step - loss: 3.0195
Epoch 13/50
1/1 [==============================] - 0s 16ms/step - loss: 3.0037
Epoch 14/50
1/1 [==============================] - 0s 11ms/step - loss: 2.9876
Epoch 15/50
1/1 [==============================] - 0s 7ms/step - loss: 2.9710
Epoch 16/50
1/1 [==============================] - 0s 3ms/step - loss: 2.9540
Epoch 17/50
1/1 [==============================] - 0s 18ms/step - loss: 2.9365
Epoch 18/50
1/1 [==============================] - 0s 11ms/step - loss: 2.9184
Epoch 19/50
1/1 [==============================] - 0s 5ms/step - loss: 2.8998
Epoch 20/50
1/1 [==============================] - 0s 9ms/step - loss: 2.8806
Epoch 21/50
1/1 [==============================] - 0s 5ms/step - loss: 2.8607
Epoch 22/50
1/1 [==============================] - 0s 18ms/step - loss: 2.8402
Epoch 23/50
1/1 [==============================] - 0s 15ms/step - loss: 2.8191
Epoch 24/50
1/1 [==============================] - 0s 0s/step - loss: 2.7971
Epoch 25/50
1/1 [==============================] - 0s 6ms/step - loss: 2.7745
Epoch 26/50
1/1 [==============================] - 0s 16ms/step - loss: 2.7511
Epoch 27/50
1/1 [==============================] - 0s 12ms/step - loss: 2.7268
Epoch 28/50
1/1 [==============================] - 0s 4ms/step - loss: 2.7017
Epoch 29/50
1/1 [==============================] - 0s 804us/step - loss: 2.6758
Epoch 30/50
1/1 [==============================] - 0s 17ms/step - loss: 2.6490
Epoch 31/50
```

```
1/1 [==============================] - 0s 15ms/step - loss: 2.6213
Epoch 32/50
1/1 [==============================] - 0s 11ms/step - loss: 2.5927
Epoch 33/50
1/1 [==============================] - 0s 5ms/step - loss: 2.5631
Epoch 34/50
1/1 [==============================] - 0s 5ms/step - loss: 2.5326
Epoch 35/50
1/1 [==============================] - 0s 17ms/step - loss: 2.5011
Epoch 36/50
1/1 [==============================] - 0s 11ms/step - loss: 2.4686
Epoch 37/50
1/1 [==============================] - 0s 9ms/step - loss: 2.4351
Epoch 38/50
1/1 [==============================] - 0s 7ms/step - loss: 2.4006
Epoch 39/50
1/1 [==============================] - 0s 9ms/step - loss: 2.3650
Epoch 40/50
1/1 [==============================] - 0s 10ms/step - loss: 2.3284
Epoch 41/50
1/1 [==============================] - 0s 8ms/step - loss: 2.2908
Epoch 42/50
1/1 [==============================] - 0s 7ms/step - loss: 2.2522
Epoch 43/50
1/1 [==============================] - 0s 15ms/step - loss: 2.2125
Epoch 44/50
1/1 [==============================] - 0s 5ms/step - loss: 2.1718
Epoch 45/50
1/1 [==============================] - 0s 14ms/step - loss: 2.1301
Epoch 46/50
1/1 [==============================] - 0s 11ms/step - loss: 2.0875
Epoch 47/50
1/1 [==============================] - 0s 10ms/step - loss: 2.0439
Epoch 48/50
1/1 [==============================] - 0s 9ms/step - loss: 1.9994
Epoch 49/50
1/1 [==============================] - 0s 10ms/step - loss: 1.9540
Epoch 50/50
1/1 [==============================] - 0s 9ms/step - loss: 1.9077
```

Out[9]:

```
<keras.src.callbacks.History at 0x1fd9b957fd0>
```

In [10]:

```python
# Function to generate the next word
def generate_next_word(seed_text):
    seed_sequence = tokenizer(seed_text)
    predicted_probabilities = model.predict(seed_sequence)
    predicted_index = np.argmax(predicted_probabilities)
    predicted_word = tokenizer.get_vocabulary()[predicted_index]
    return predicted_word
```

In [13]:

```python
# Test the predictive text system
input_text = "used"
predicted_word = generate_next_word(input_text)
print(f"Input: '{input_text}', Predicted: '{predicted_word}'")
```

```
1/1 [==============================] - 0s 23ms/step
Input: 'used', Predicted: 'to'
```

Using LSTM

In [15]:

```
import tensorflow as tf
import numpy as np

# Sample text data
text = "This is a sample text used to demonstrate predictive text with LSTM.

# Preprocess the text and create a vocabulary
tokenizer = tf.keras.layers.TextVectorization()
tokenizer.adapt(text.split())

# Convert text to sequences of token indices
text_sequences = tokenizer(text)

# Create training data (X) and target data (y)
X = text_sequences[:-1]
y = text_sequences[1:]

# Build an LSTM model using Keras
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.get_vocabulary()), out
    tf.keras.layers.LSTM(128, return_sequences=True),
    tf.keras.layers.Dense(len(tokenizer.get_vocabulary()), activation='softm
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')

# Train the model
model.fit(X, y, epochs=50)

# Function to generate the next word
def generate_next_word(seed_text):
    seed_sequence = tokenizer(seed_text)
    predicted_probabilities = model.predict(seed_sequence)
    predicted_index = np.argmax(predicted_probabilities)
    predicted_word = tokenizer.get_vocabulary()[predicted_index]
    return predicted_word

# Test the predictive text system
input_text = "This is"
predicted_word = generate_next_word(input_text)
print(f"Input: '{input_text}', Predicted: '{predicted_word}'")
```

```
Epoch 1/50
1/1 [==============================] - 4s 4s/step - loss: 3.1360
Epoch 2/50
1/1 [==============================] - 0s 15ms/step - loss: 3.1333
Epoch 3/50
1/1 [==============================] - 0s 11ms/step - loss: 3.1305
Epoch 4/50
1/1 [==============================] - 0s 14ms/step - loss: 3.1277
Epoch 5/50
1/1 [==============================] - 0s 11ms/step - loss: 3.1249
Epoch 6/50
1/1 [==============================] - 0s 15ms/step - loss: 3.1221
Epoch 7/50
1/1 [==============================] - 0s 14ms/step - loss: 3.1192
Epoch 8/50
1/1 [==============================] - 0s 14ms/step - loss: 3.1163
Epoch 9/50
1/1 [==============================] - 0s 15ms/step - loss: 3.1134
Epoch 10/50
1/1 [==============================] - 0s 14ms/step - loss: 3.1103
Epoch 11/50
1/1 [==============================] - 0s 14ms/step - loss: 3.1072
Epoch 12/50
1/1 [==============================] - 0s 13ms/step - loss: 3.1039
Epoch 13/50
1/1 [==============================] - 0s 12ms/step - loss: 3.1006
Epoch 14/50
1/1 [==============================] - 0s 14ms/step - loss: 3.0972
Epoch 15/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0936
Epoch 16/50
1/1 [==============================] - 0s 6ms/step - loss: 3.0899
Epoch 17/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0860
Epoch 18/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0820
Epoch 19/50
1/1 [==============================] - 0s 8ms/step - loss: 3.0779
Epoch 20/50
1/1 [==============================] - 0s 10ms/step - loss: 3.0735
Epoch 21/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0690
Epoch 22/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0642
Epoch 23/50
1/1 [==============================] - 0s 13ms/step - loss: 3.0592
Epoch 24/50
1/1 [==============================] - 0s 14ms/step - loss: 3.0540
Epoch 25/50
1/1 [==============================] - 0s 13ms/step - loss: 3.0486
Epoch 26/50
1/1 [==============================] - 0s 10ms/step - loss: 3.0429
Epoch 27/50
1/1 [==============================] - 0s 9ms/step - loss: 3.0370
Epoch 28/50
1/1 [==============================] - 0s 18ms/step - loss: 3.0308
Epoch 29/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0243
Epoch 30/50
1/1 [==============================] - 0s 10ms/step - loss: 3.0174
Epoch 31/50
```

```
1/1 [==============================] - 0s 22ms/step - loss: 3.0103
Epoch 32/50
1/1 [==============================] - 0s 12ms/step - loss: 3.0028
Epoch 33/50
1/1 [==============================] - 0s 10ms/step - loss: 2.9950
Epoch 34/50
1/1 [==============================] - 0s 12ms/step - loss: 2.9868
Epoch 35/50
1/1 [==============================] - 0s 11ms/step - loss: 2.9783
Epoch 36/50
1/1 [==============================] - 0s 12ms/step - loss: 2.9693
Epoch 37/50
1/1 [==============================] - 0s 10ms/step - loss: 2.9599
Epoch 38/50
1/1 [==============================] - 0s 10ms/step - loss: 2.9501
Epoch 39/50
1/1 [==============================] - 0s 11ms/step - loss: 2.9399
Epoch 40/50
1/1 [==============================] - 0s 12ms/step - loss: 2.9291
Epoch 41/50
1/1 [==============================] - 0s 10ms/step - loss: 2.9179
Epoch 42/50
1/1 [==============================] - 0s 11ms/step - loss: 2.9062
Epoch 43/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8940
Epoch 44/50
1/1 [==============================] - 0s 12ms/step - loss: 2.8812
Epoch 45/50
1/1 [==============================] - 0s 12ms/step - loss: 2.8679
Epoch 46/50
1/1 [==============================] - 0s 10ms/step - loss: 2.8540
Epoch 47/50
1/1 [==============================] - 0s 10ms/step - loss: 2.8395
Epoch 48/50
1/1 [==============================] - 0s 11ms/step - loss: 2.8244
Epoch 49/50
1/1 [==============================] - 0s 10ms/step - loss: 2.8086
Epoch 50/50
1/1 [==============================] - 0s 9ms/step - loss: 2.7922
1/1 [==============================] - 1s 873ms/step
Input: 'This is', Predicted: 'example'
```

In [ ]:

In [2]:

```
!pip install lime shap
```

```
Requirement already satisfied: lime in c:\users\student\appdata\roaming\py
thon\python39\site-packages (0.2.0.1)
Requirement already satisfied: shap in c:\users\student\appdata\roaming\py
thon\python39\site-packages (0.43.0)
Requirement already satisfied: matplotlib in c:\users\student\.conda\envs
\mallikarjuna\lib\site-packages (from lime) (3.7.3)
Requirement already satisfied: numpy in c:\users\student\.conda\envs\malli
karjuna\lib\site-packages (from lime) (1.24.3)
Requirement already satisfied: scipy in c:\users\student\.conda\envs\malli
karjuna\lib\site-packages (from lime) (1.11.2)
Requirement already satisfied: tqdm in c:\users\student\.conda\envs\mallik
arjuna\lib\site-packages (from lime) (4.66.1)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\student\.con
da\envs\mallikarjuna\lib\site-packages (from lime) (1.3.0)
Requirement already satisfied: scikit-image>=0.12 in c:\users\student\.con
da\envs\mallikarjuna\lib\site-packages (from lime) (0.22.0)
Requirement already satisfied: pandas in c:\users\student\.conda\envs\mall
ikarjuna\lib\site-packages (from shap) (2.1.0)
Requirement already satisfied: packaging>20.9 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from shap) (23.1)
Requirement already satisfied: slicer==0.0.7 in c:\users\student\appdata\r
oaming\python\python39\site-packages (from shap) (0.0.7)
Requirement already satisfied: numba in c:\users\student\.conda\envs\malli
karjuna\lib\site-packages (from shap) (0.58.1)
Requirement already satisfied: cloudpickle in c:\users\student\.conda\envs
\mallikarjuna\lib\site-packages (from shap) (3.0.0)
Requirement already satisfied: networkx>=2.8 in c:\users\student\.conda\en
vs\mallikarjuna\lib\site-packages (from scikit-image>=0.12->lime) (3.2.1)
Requirement already satisfied: pillow>=9.0.1 in c:\users\student\.conda\en
vs\mallikarjuna\lib\site-packages (from scikit-image>=0.12->lime) (10.0.0)
Requirement already satisfied: imageio>=2.27 in c:\users\student\.conda\en
vs\mallikarjuna\lib\site-packages (from scikit-image>=0.12->lime) (2.32.0)
Requirement already satisfied: tifffile>=2022.8.12 in c:\users\student\.co
nda\envs\mallikarjuna\lib\site-packages (from scikit-image>=0.12->lime) (2
023.9.26)
Requirement already satisfied: lazy_loader>=0.3 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from scikit-image>=0.12->lime) (0.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\student\.conda\en
vs\mallikarjuna\lib\site-packages (from scikit-learn>=0.18->lime) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\student\.c
onda\envs\mallikarjuna\lib\site-packages (from scikit-learn>=0.18->lime)
(3.2.0)
Requirement already satisfied: colorama in c:\users\student\.conda\envs\ma
llikarjuna\lib\site-packages (from tqdm->lime) (0.4.6)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib->lime) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from matplotlib->lime) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib->lime) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\student\.cond
a\envs\mallikarjuna\lib\site-packages (from matplotlib->lime) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from matplotlib->lime) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\student\.c
onda\envs\mallikarjuna\lib\site-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\stud
ent\.conda\envs\mallikarjuna\lib\site-packages (from matplotlib->lime) (6.
0.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in c:\users\stud
ent\.conda\envs\mallikarjuna\lib\site-packages (from numba->shap) (0.41.1)
```

```
Requirement already satisfied: pytz>=2020.1 in c:\users\student\.conda\env
s\mallikarjuna\lib\site-packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\student\.conda\e
nvs\mallikarjuna\lib\site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: zipp>=3.1.0 in c:\users\student\.conda\envs
\mallikarjuna\lib\site-packages (from importlib-resources>=3.2.0->matplotl
ib->lime) (3.16.2)
Requirement already satisfied: six>=1.5 in c:\users\student\.conda\envs\ma
llikarjuna\lib\site-packages (from python-dateutil>=2.7->matplotlib->lime)
(1.16.0)
```

In [4]:
```
!pip install ipywidgets
```

```
Collecting ipywidgets
  Obtaining dependency information for ipywidgets from https://files.pyt
honhosted.org/packages/4a/0e/57ed498fafbc60419a9332d872e929879ceba2d73cb
11d284d7112472b3e/ipywidgets-8.1.1-py3-none-any.whl.metadata (https://fi
les.pythonhosted.org/packages/4a/0e/57ed498fafbc60419a9332d872e929879ceb
a2d73cb11d284d7112472b3e/ipywidgets-8.1.1-py3-none-any.whl.metadata)
  Downloading ipywidgets-8.1.1-py3-none-any.whl.metadata (2.4 kB)
Collecting comm>=0.1.3 (from ipywidgets)
  Obtaining dependency information for comm>=0.1.3 from https://files.py
thonhosted.org/packages/7b/a6/5fd0242e974914b139451eea0a61ed9fd2e47157e3
3a67939043c50a94dd/comm-0.2.0-py3-none-any.whl.metadata (https://files.p
ythonhosted.org/packages/7b/a6/5fd0242e974914b139451eea0a61ed9fd2e47157e
33a67939043c50a94dd/comm-0.2.0-py3-none-any.whl.metadata)
  Downloading comm-0.2.0-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: ipython>=6.1.0 in c:\users\student\.conda
\envs\mallikarjuna\lib\site-packages (from ipywidgets) (8.15.0)
Requirement already satisfied: traitlets>=4.3.1 in c:\users\student\.con
da\envs\mallikarjuna\lib\site-packages (from ipywidgets) (5.7.1)
Collecting widgetsnbextension~=4.0.9 (from ipywidgets)
  Obtaining dependency information for widgetsnbextension~=4.0.9 from ht
```

In [6]:
```
import lime
from lime.lime_tabular import LimeTabularExplainer
import shap
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

In [7]:
```
iris=load_iris()
x=iris.data
y=iris.target

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_sta
```

In [8]:
```
model = keras.Sequential([
    keras.layers.Dense(8,input_dim=4, activation='relu'),
    keras.layers.Dense(3,activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [9]:
```
model.fit(x_train,y_train,epochs=50,batch_size=16,verbose=0)
```

Out[9]:
```
<keras.src.callbacks.History at 0x1dfb6ca3bb0>
```

In [10]:
```
explainer=LimeTabularExplainer(x_train,mode="classification")
```

In [11]:
```
# Explain the prediction using LIME
explanation=explainer.explain_instance(x_test[0],model.predict, num_features
explanation.show_in_notebook
```

```
157/157 [==============================] - 0s 994us/step
```

Out[11]:
```
<bound method Explanation.show_in_notebook of <lime.explanation.Explanatio
n object at 0x000001DFB7EBA970>>
```

SHAP

In [54]:
```
import shap
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

In [55]:
```
iris= load_iris()
x = iris.data
y=iris.target
```

In [56]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_st
```

In [57]:
```
#get the original feature names
feature_names_encoded = iris.feature_names
```

In [58]:
```
#create label encodrr
label_encoder= LabelEncoder()
feature_names_encoded = label_encoder.fit_transform(feature_names_encoded)
feature_names_decoded=label_encoder.inverse_transform(feature_names_encoded)
```

In [59]:
```
model= keras.Sequential([
    keras.layers.Dense(8, input_dim=4, activation='relu'),
    keras.layers.Dense(3,activation='softmax')

])
```

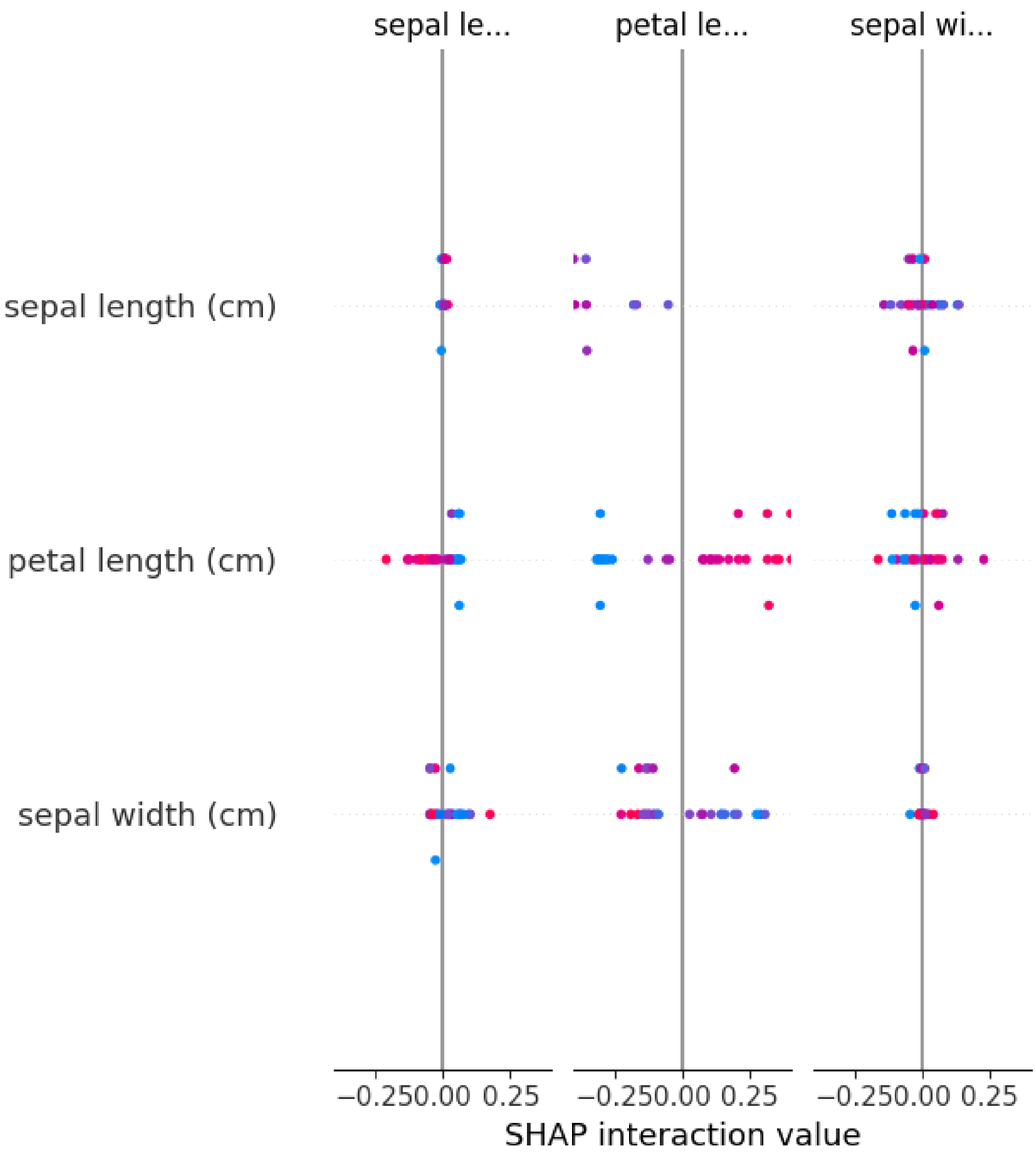
In [60]:
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metric
```

In [61]:
```
model.fit(x_train,y_train,epochs=50,batch_size=16,verbose=0)
```

Out[61]:
```
<keras.src.callbacks.History at 0x1dfbba5b3a0>
```

In [62]:

```
#create a SHAP explainer
shap_explainer = shap.Explainer(model,x_train)
#explain predictions for the test set
shap_values = shap_explainer(x_test)
#visualize the SHAP values with Label-decoded feature
shap.summary_plot(shap_values, x_test, feature_names=feature_names_decoded)
```



Developing Interactive Model using Tkinter

In [63]:

```
!pip install tk
```

```
Requirement already satisfied: tk in c:\users\student\appdata\roaming\pyth
on\python39\site-packages (0.1.0)
```

In [103]:
```
import tkinter as tk
from tkinter import messagebox
import joblib
import shap
import lime.lime_tabular
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
```

In [104]:
```
import tkinter as tk
from tkinter import messagebox
import joblib
import shap
import lime.lime_tabular
import numpy as np
```

In [105]:
```
iris= load_iris()
x = iris.data
y=iris.target
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_st
```

In [106]:
```
model= keras.Sequential([
    keras.layers.Dense(8, input_dim=4, activation='relu'),
    keras.layers.Dense(3,activation='softmax')

])
```

In [107]:
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metric
model.fit(x_train,y_train,epochs=50,batch_size=16,verbose=0)
```

Out[107]:
```
<keras.src.callbacks.History at 0x1dfbd09b490>
```

In [108]:
```
#save the trained model
joblib.dump(model, 'lime_shap.pk1')
```

Out[108]:
```
['lime_shap.pk1']
```

In [109]:
```
lime_explainer = lime.lime_tabular.LimeTabularExplainer(x_train,mode="classi
```

In [110]:
```
#create SHap explainer
shap_explainer= shap.Explainer(model,x_train)
```

In [111]:
```
#Function to run the model with LIME explanation
def run_with_lime():
    model = joblib.load('lime_shap.pk1')
    explanation=explainer.explain_instance(x_test[0],model.predict,num_featu
    explanation.show_in_notebook()
```

In [112]:
```
def run_with_shap():
    model = joblib.load('lime_shap.pk1')
    shap_values = shap_explainer(x_test)
    shap.summary_plot(shap_values, x_test, feature_names=feature_names_decod
```

In [113]:
```
root=tk.Tk()
root.title('model_explanation_tool')
```

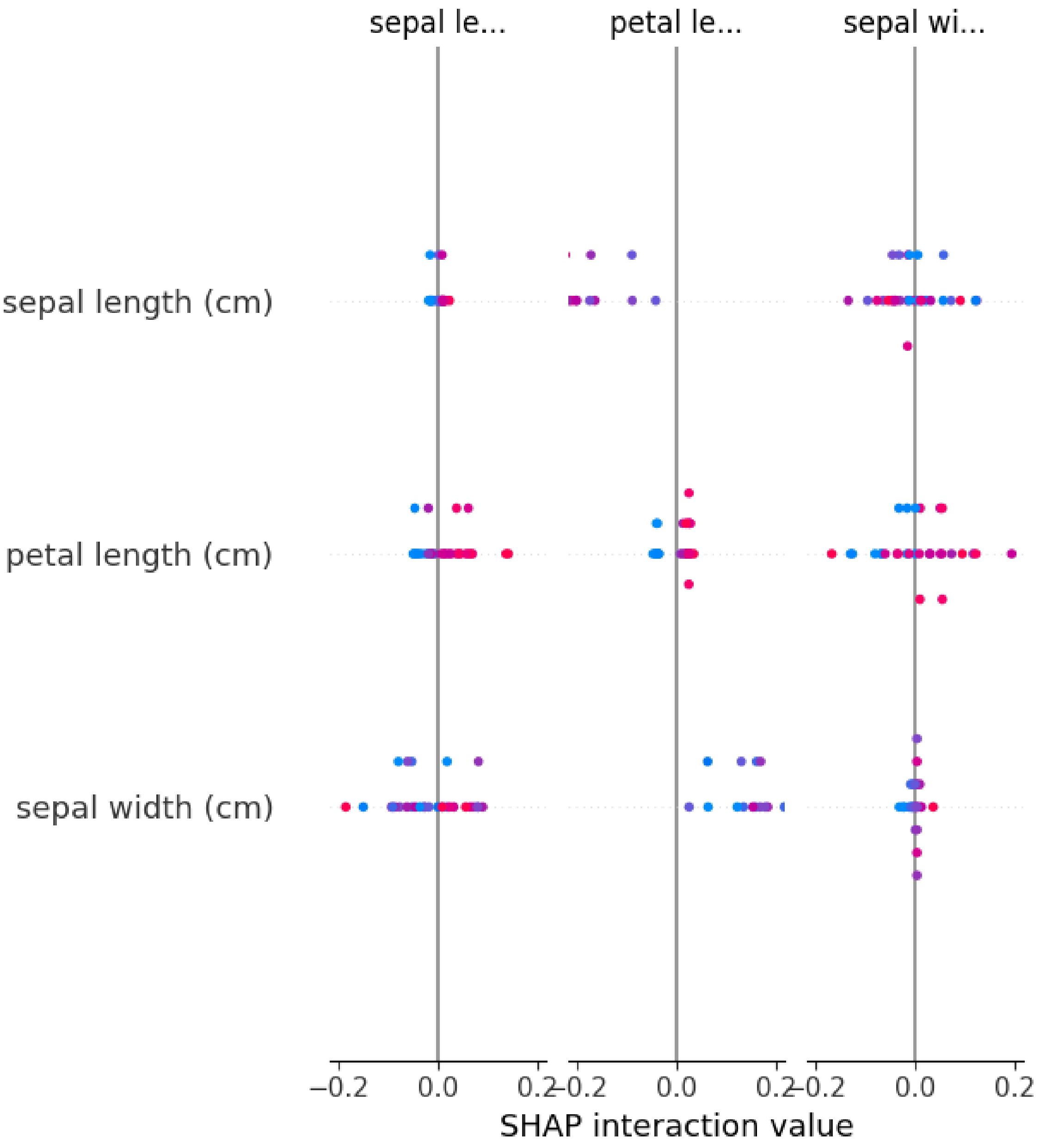
Out[113]:
```
''
```

In [114]:
```
#place the buttons in the window
lime_button = tk.Button(root,text="Explain with LIME ", command = run_with_l
shap_button = tk.Button(root,text="Explain with SHAP ", command = run_with_s
```

In [115]:
```
lime_button.pack(pady=10)
shap_button.pack(pady=10)
```

In [ ]:
```
# Start the GUI Loop
root.mainloop()
```

```
157/157 [==============================] - 0s 479us/step
```



In [ ]: