Task 1: Read the dataset and do data pre-processing

```
df = pd.read_csv('/content/drug200.csv')  # Reading the data
df.head()  # Visualizing the data
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

```
df.isnull().sum() # Checking for null values
```

```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

```
df['Drug'].unique()  # Finding unique category
```

```
array(['DrugY', 'drugC', 'drugX', 'drugA', 'drugB'], dtype=object)
```

```
df['Drug'].value_counts()  # Finding the count of observations based on unique value
```

```
DrugY    91
drugX    54
```

```
drugA    23
drugC    16
drugB    16
Name: Drug, dtype: int64
```

```
# Splitting the data
x = df.iloc[:,1:5].values
x
```

```
array([['F', 'HIGH', 'HIGH', 25.355],
       ['M', 'LOW', 'HIGH', 13.093],
       ['M', 'LOW', 'HIGH', 10.114],
       ['F', 'NORMAL', 'HIGH', 7.798],
       ['F', 'LOW', 'HIGH', 18.043],
       ['F', 'NORMAL', 'HIGH', 8.607],
       ['F', 'NORMAL', 'HIGH', 16.275],
       ['M', 'LOW', 'HIGH', 11.037],
       ['M', 'NORMAL', 'HIGH', 15.171],
       ['M', 'LOW', 'NORMAL', 19.368],
       ['F', 'LOW', 'HIGH', 11.767],
       ['F', 'HIGH', 'NORMAL', 19.199],
       ['M', 'LOW', 'HIGH', 15.376],
       ['F', 'LOW', 'HIGH', 20.942],
       ['F', 'NORMAL', 'HIGH', 12.703],
       ['F', 'HIGH', 'NORMAL', 15.516],
       ['M', 'LOW', 'NORMAL', 11.455],
       ['M', 'HIGH', 'HIGH', 13.972],
       ['M', 'LOW', 'HIGH', 7.298],
       ['F', 'HIGH', 'NORMAL', 25.974],
       ['M', 'LOW', 'NORMAL', 19.128],
       ['M', 'NORMAL', 'HIGH', 25.917],
       ['M', 'LOW', 'NORMAL', 30.568],
       ['F', 'LOW', 'HIGH', 15.036],
       ['F', 'LOW', 'HIGH', 33.486],
       ['F', 'HIGH', 'NORMAL', 18.809],
       ['M', 'HIGH', 'HIGH', 30.366],
       ['F', 'NORMAL', 'NORMAL', 9.381],
       ['F', 'LOW', 'NORMAL', 22.697],
       ['M', 'LOW', 'HIGH', 17.951],
       ['F', 'NORMAL', 'NORMAL', 8.75],
       ['M', 'HIGH', 'HIGH', 9.567],
       ['M', 'LOW', 'NORMAL', 11.014],
```

```
['F', 'HIGH', 'NORMAL', 31.876],
['M', 'NORMAL', 'HIGH', 14.133],
['M', 'NORMAL', 'NORMAL', 7.285],
['M', 'HIGH', 'NORMAL', 9.445],
['M', 'LOW', 'NORMAL', 13.938],
['F', 'NORMAL', 'NORMAL', 9.709],
['M', 'NORMAL', 'HIGH', 9.084],
['F', 'NORMAL', 'HIGH', 19.221],
['F', 'HIGH', 'NORMAL', 14.239],
['M', 'NORMAL', 'NORMAL', 15.79],
['M', 'NORMAL', 'HIGH', 12.26],
['F', 'NORMAL', 'NORMAL', 12.295],
['F', 'NORMAL', 'NORMAL', 8.107],
['F', 'HIGH', 'HIGH', 13.091],
['M', 'LOW', 'HIGH', 10.291],
['M', 'NORMAL', 'HIGH', 31.686],
['F', 'LOW', 'HIGH', 19.796],
['F', 'HIGH', 'HIGH', 19.416],
['M', 'NORMAL', 'NORMAL', 10.898],
['M', 'LOW', 'NORMAL', 27.183],
['F', 'HIGH', 'NORMAL', 18.457],
['F', 'HIGH', 'NORMAL', 10.189],
['F', 'LOW', 'HIGH', 14.16],
['M', 'HIGH', 'NORMAL', 11.34],
['M', 'HIGH', 'HIGH', 27.826],
```

```
y = pd.get_dummies(df.iloc[:,5:]).values
y
```

```
array([[1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
```

```
[0, 0, 0, 0, 1],
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 1],
[0, 1, 0, 0, 0],
[0, 0, 0, 1, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 1],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 1],
[0, 0, 1, 0, 0],
[0, 0, 0, 0, 1],
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 1],
[0, 0, 0, 0, 1],
[0, 1, 0, 0, 0],
[0, 0, 0, 0, 1],
[0, 0, 0, 0, 1],
[0, 0, 0, 0, 1],
[1, 0, 0, 0, 0],
[0, 0, 1, 0, 0],
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 1],
[0, 0, 0, 0, 1],
[0, 0, 0, 0, 1],
[0, 1, 0, 0, 0],
[0, 0, 0, 1, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 1],
[1, 0, 0, 0, 0],
[1, 0, 0, 0, 0],
[0, 0, 1, 0, 0],
[0, 0, 0, 1, 0],
[0, 0, 1, 0, 0],
```

```
[1. 0. 0. 0. 0.]
```

```
# Splitting the data into train and test
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=21)
xtrain.shape, xtest.shape, ytrain.shape, ytest.shape
```

```
((160, 4), (40, 4), (160, 5), (40, 5))
```

Task 2: Build the ANN model with (input layer, min 3 hidden layers & output layer)

```
# ANN Model
model = Sequential()
model.add(Dense(8,input_dim=4,activation='relu'))
#model.add(Dense(32,activation='relu'))
model.add(Dense(26,activation='relu'))
model.add(Dense(3,activation='softmax'))

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

model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_36 (Dense)            (None, 8)                 40

 dense_37 (Dense)            (None, 26)                234

 dense_38 (Dense)            (None, 3)                 81

=================================================================
Total params: 355
Trainable params: 355
Non-trainable params: 0
_________________________________________________________________
```

Task 3: Test the model with random data

```
# Splitting the data into train and test
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=21)
xtrain.shape, xtest.shape, ytrain.shape, ytest.shape

# Generate random data to test the model
X_new = np.random.rand(5,4)

# Predict the classes of the new data
y_pred = model.predict(X_new)

# Print the predicted classes
print(y_pred)
```

```
1/1 [==============================] - 0s 18ms/step
[[0.33307767 0.33042318 0.33649918]
 [0.32373506 0.36513993 0.311125  ]
 [0.33278942 0.3312323  0.33597827]
 [0.29992178 0.337207   0.36287123]
 [0.33994704 0.32332256 0.33673042]]
```