```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


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()
```

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

```
df['Drug'].unique()
```

```
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
```

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

# Encode labels for sex - male, female
df['Sex']= label_encoder.fit_transform(df['Sex'])


label_encoder = preprocessing.LabelEncoder()

df['BP']= label_encoder.fit_transform(df['BP'])


label_encoder = preprocessing.LabelEncoder()

df['Cholesterol']= label_encoder.fit_transform(df['Cholesterol'])



df
```

Age Sex BP Cholesterol Na to K Drug

```
x = df.iloc[:,:5].values
x
```

```
array([[23.   ,  0.   ,  0.   ,  0.   , 25.355],
       [47.   ,  1.   ,  1.   ,  0.   , 13.093],
       [47.   ,  1.   ,  1.   ,  0.   , 10.114],
       [28.   ,  0.   ,  2.   ,  0.   ,  7.798],
       [61.   ,  0.   ,  1.   ,  0.   , 18.043],
       [22.   ,  0.   ,  2.   ,  0.   ,  8.607],
       [49.   ,  0.   ,  2.   ,  0.   , 16.275],
       [41.   ,  1.   ,  1.   ,  0.   , 11.037],
       [60.   ,  1.   ,  2.   ,  0.   , 15.171],
       [43.   ,  1.   ,  1.   ,  1.   , 19.368],
       [47.   ,  0.   ,  1.   ,  0.   , 11.767],
       [34.   ,  0.   ,  0.   ,  1.   , 19.199],
       [43.   ,  1.   ,  1.   ,  0.   , 15.376],
       [74.   ,  0.   ,  1.   ,  0.   , 20.942],
       [50.   ,  0.   ,  2.   ,  0.   , 12.703],
       [16.   ,  0.   ,  0.   ,  1.   , 15.516],
       [69.   ,  1.   ,  1.   ,  1.   , 11.455],
       [43.   ,  1.   ,  0.   ,  0.   , 13.972],
       [23.   ,  1.   ,  1.   ,  0.   ,  7.298],
       [32.   ,  0.   ,  0.   ,  1.   , 25.974],
       [57.   ,  1.   ,  1.   ,  1.   , 19.128],
       [63.   ,  1.   ,  2.   ,  0.   , 25.917],
       [47.   ,  1.   ,  1.   ,  1.   , 30.568],
       [48.   ,  0.   ,  1.   ,  0.   , 15.036],
       [33.   ,  0.   ,  1.   ,  0.   , 33.486],
       [28.   ,  0.   ,  0.   ,  1.   , 18.809],
       [31.   ,  1.   ,  0.   ,  0.   , 30.366],
       [49.   ,  0.   ,  2.   ,  1.   ,  9.381],
       [39.   ,  0.   ,  1.   ,  1.   , 22.697],
       [45.   ,  1.   ,  1.   ,  0.   , 17.951],
       [18.   ,  0.   ,  2.   ,  1.   ,  8.75 ],
       [74.   ,  1.   ,  0.   ,  0.   ,  9.567],
       [49.   ,  1.   ,  1.   ,  1.   , 11.014],
       [65.   ,  0.   ,  0.   ,  1.   , 31.876],
       [53.   ,  1.   ,  2.   ,  0.   , 14.133],
       [46.   ,  1.   ,  2.   ,  1.   ,  7.285],
       [32.   ,  1.   ,  0.   ,  1.   ,  9.445],
       [39.   ,  1.   ,  1.   ,  1.   , 13.938],
       [39.   ,  0.   ,  2.   ,  1.   ,  9.709],
       [15.   ,  1.   ,  2.   ,  0.   ,  9.084],
       [73.   ,  0.   ,  2.   ,  0.   , 19.221],
       [58.   ,  0.   ,  0.   ,  1.   , 14.239],
       [50.   ,  1.   ,  2.   ,  1.   , 15.79 ],
       [23.   ,  1.   ,  2.   ,  0.   , 12.26 ],
       [50.   ,  0.   ,  2.   ,  1.   , 12.295],
       [66.   ,  0.   ,  2.   ,  1.   ,  8.107],
       [37.   ,  0.   ,  0.   ,  0.   , 13.091],
       [68.   ,  1.   ,  1.   ,  0.   , 10.291],
       [23.   ,  1.   ,  2.   ,  0.   , 31.686],
       [28.   ,  0.   ,  1.   ,  0.   , 19.796],
       [58.   ,  0.   ,  0.   ,  0.   , 19.416],
       [67.   ,  1.   ,  2.   ,  1.   , 10.898],
       [62.   ,  1.   ,  1.   ,  1.   , 27.183],
       [24.   ,  0.   ,  0.   ,  1.   , 18.457],
       [68.   ,  0.   ,  0.   ,  1.   , 10.189],
       [26.   ,  0.   ,  1.   ,  0.   , 14.16 ],
       [65.   ,  1.   ,  0.   ,  1.   , 11.34 ],
       [40.   ,  1.   ,  0.   ,  0.   , 27.826],
```

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

```
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],
            [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]
```

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

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


model = Sequential()
model.add(Dense(8,input_dim=5,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(16,activation='relu'))
model.add(Dense(25,activation='relu'))
model.add(Dense(5,activation='softmax'))
```

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

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 8)                 48

 dense_6 (Dense)             (None, 32)                288

 dense_7 (Dense)             (None, 16)                528

 dense_8 (Dense)             (None, 25)                425

 dense_9 (Dense)             (None, 5)                 130

=================================================================
Total params: 1,419
Trainable params: 1,419
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(xtrain,ytrain,epochs=200,batch_size=15,validation_data=(xtest,ytest))
```

```
Epoch 1/200
11/11 [==============================] - 2s 28ms/step - loss: 1.7889 - accuracy: 0.1750 - val_loss: 1.2823 - val_accuracy: 0.5250
Epoch 2/200
11/11 [==============================] - 0s 5ms/step - loss: 1.3196 - accuracy: 0.4875 - val_loss: 1.2972 - val_accuracy: 0.5250
Epoch 3/200
11/11 [==============================] - 0s 6ms/step - loss: 1.2320 - accuracy: 0.4750 - val_loss: 1.1691 - val_accuracy: 0.5500
Epoch 4/200
11/11 [==============================] - 0s 7ms/step - loss: 1.1503 - accuracy: 0.5625 - val_loss: 1.1312 - val_accuracy: 0.5500
Epoch 5/200
11/11 [==============================] - 0s 6ms/step - loss: 1.1114 - accuracy: 0.5625 - val_loss: 1.0776 - val_accuracy: 0.5500
Epoch 6/200
11/11 [==============================] - 0s 6ms/step - loss: 1.0783 - accuracy: 0.5750 - val_loss: 1.1117 - val_accuracy: 0.5500
Epoch 7/200
```

```
11/11 [==============================] - 0s 7ms/step - loss: 1.0774 - accuracy: 0.5625 - val_loss: 1.0695 - val_accuracy: 0.5750
Epoch 8/200
11/11 [==============================] - 0s 6ms/step - loss: 1.0504 - accuracy: 0.5688 - val_loss: 1.0864 - val_accuracy: 0.5500
Epoch 9/200
11/11 [==============================] - 0s 6ms/step - loss: 1.0337 - accuracy: 0.5813 - val_loss: 1.0475 - val_accuracy: 0.6000
Epoch 10/200
11/11 [==============================] - 0s 7ms/step - loss: 1.0226 - accuracy: 0.5813 - val_loss: 1.0777 - val_accuracy: 0.5500
Epoch 11/200
11/11 [==============================] - 0s 6ms/step - loss: 1.0168 - accuracy: 0.5938 - val_loss: 1.0618 - val_accuracy: 0.5750
Epoch 12/200
11/11 [==============================] - 0s 7ms/step - loss: 0.9993 - accuracy: 0.6000 - val_loss: 1.0571 - val_accuracy: 0.5750
Epoch 13/200
11/11 [==============================] - 0s 7ms/step - loss: 0.9992 - accuracy: 0.5813 - val_loss: 1.0518 - val_accuracy: 0.5750
Epoch 14/200
11/11 [==============================] - 0s 5ms/step - loss: 1.0001 - accuracy: 0.5938 - val_loss: 1.0587 - val_accuracy: 0.5750
Epoch 15/200
11/11 [==============================] - 0s 7ms/step - loss: 0.9646 - accuracy: 0.6125 - val_loss: 0.9961 - val_accuracy: 0.6000
Epoch 16/200
11/11 [==============================] - 0s 8ms/step - loss: 0.9724 - accuracy: 0.5813 - val_loss: 0.9826 - val_accuracy: 0.6250
Epoch 17/200
11/11 [==============================] - 0s 6ms/step - loss: 0.9651 - accuracy: 0.5938 - val_loss: 1.0068 - val_accuracy: 0.6000
Epoch 18/200
11/11 [==============================] - 0s 7ms/step - loss: 0.9439 - accuracy: 0.6000 - val_loss: 0.9711 - val_accuracy: 0.6000
Epoch 19/200
11/11 [==============================] - 0s 7ms/step - loss: 0.9439 - accuracy: 0.6000 - val_loss: 0.9521 - val_accuracy: 0.6000
Epoch 20/200
11/11 [==============================] - 0s 6ms/step - loss: 0.9313 - accuracy: 0.5938 - val_loss: 0.9789 - val_accuracy: 0.5750
Epoch 21/200
11/11 [==============================] - 0s 7ms/step - loss: 0.9227 - accuracy: 0.6062 - val_loss: 0.9119 - val_accuracy: 0.6500
Epoch 22/200
11/11 [==============================] - 0s 9ms/step - loss: 0.9001 - accuracy: 0.6313 - val_loss: 0.9408 - val_accuracy: 0.5750
Epoch 23/200
11/11 [==============================] - 0s 9ms/step - loss: 0.8986 - accuracy: 0.6187 - val_loss: 0.9543 - val_accuracy: 0.5500
Epoch 24/200
11/11 [==============================] - 0s 8ms/step - loss: 0.8838 - accuracy: 0.6125 - val_loss: 0.8646 - val_accuracy: 0.6750
Epoch 25/200
11/11 [==============================] - 0s 8ms/step - loss: 0.8858 - accuracy: 0.6562 - val_loss: 1.0404 - val_accuracy: 0.5250
Epoch 26/200
11/11 [==============================] - 0s 8ms/step - loss: 0.8542 - accuracy: 0.6313 - val_loss: 0.8161 - val_accuracy: 0.7250
Epoch 27/200
11/11 [==============================] - 0s 8ms/step - loss: 0.8455 - accuracy: 0.6438 - val_loss: 0.9397 - val_accuracy: 0.5500
Epoch 28/200
11/11 [==============================] - 0s 9ms/step - loss: 0.8266 - accuracy: 0.6500 - val_loss: 0.8131 - val_accuracy: 0.6750
Epoch 29/200
11/11 [==============================] - 0s 11ms/step - loss: 0.8034 - accuracy: 0.6750 - val_loss: 0.8082 - val_accuracy: 0.6500
```

```
#testing with random data
pred = model.predict([[23,1.0,1.0,0.0,23.55]])
anss = np.argmax(pred)
print(anss)
```

```
1/1 [==============================] - 0s 124ms/step
0
```

```
#testing with random data
pred = model.predict([[50,0,2,1,7.285]])
anss = np.argmax(pred)
print(anss)
```

```
1/1 [==============================] - 0s 39ms/step
4
```